
THE HANDBOOK OF DATA MINING

Edited by

Nong Ye



THE HANDBOOK OF DATA MINING

Human Factors and Ergonomics
Gavriel Salvendy, Series Editor

Hendrick, H., and Kleiner, B. (Eds.): *Macroergonomics: Theory, Methods, and Applications*

Hollnagel, E. (Ed.): *Handbook of Cognitive Task Design*

Jacko, J.A., and Sears, A. (Eds.): *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*

Meister, D., and Enderwick, T. (Eds.): *Human Factors in System Design, Development, and Testing*

Stanney, Kay M. (Ed.): *Handbook of Virtual Environments: Design, Implementation, and Applications*

Stephanidis, C. (Ed.): *User Interfaces for All: Concepts, Methods, and Tools*

Ye, Nong (Ed.): *The Handbook of Data Mining*

Also in this Series

HCI 1999 Proceedings 2-Volume Set

- **Bullinger, H.-J., and Ziegler, J.** (Eds.): *Human-Computer Interaction: Ergonomics and User Interfaces*
- **Bullinger, H.-J., and Ziegler, J.** (Eds.): *Human-Computer Interaction: Communication, Cooperation, and Application Design*

HCI 2001 Proceedings 3-Volume Set

- **Smith, M.J., Salvendy, G., Harris, D., and Koubek, R.J.** (Eds.): *Usability Evaluation and Interface Design: Cognitive Engineering, Intelligent Agents, and Virtual Reality*
- **Smith, M.J., and Salvendy, G.** (Eds.): *Systems, Social, and Internationalization Design Aspects of Human-Computer Interaction*
- **Stephanidis, C.** (Ed.): *Universal Access in HCI: Towards an Information Society for All*

THE HANDBOOK OF DATA MINING

Edited by
Nong Ye
Arizona State University



Senior Acquisitions Editor: Debra Riegert
Editorial Assistant: Jason Planer
Cover Design: Kathryn Houghtaling Lacey
Textbook Production Manager: Paul Smolenski
Full-Service Compositor: TechBooks
Text and Cover Printer: Hamilton Printing Company

This book was typeset in 10/12 pt. Times, Italic, Bold, Bold Italic, and Courier. The heads were typeset in Americana Bold and Americana Bold Italic.

Copyright © 2003 by Lawrence Erlbaum Associates, Inc.
All rights reserved. No part of this book may be reproduced in any form, by photostat, microfilm, retrieval system, or any other means, without prior written permission of the publisher.

Lawrence Erlbaum Associates, Inc., Publishers
10 Industrial Avenue
Mahwah, New Jersey 07430

The editor, authors, and the publisher have made every effort to provide accurate and complete information in this handbook but the handbook is not intended to serve as a replacement for professional advice. Any use of this information is at the reader's discretion. The editor, authors, and the publisher specifically disclaim any and all liability arising directly or indirectly from the use or application of any information contained in this handbook. An appropriate professional should be consulted regarding your specific situation.

Library of Congress Cataloging-in-Publication Data

The handbook of data mining / edited by Nong Ye.
p. cm.—(Human factors and ergonomics)
Includes bibliographical references and index.
ISBN 0-8058-4081-8
1. Data mining. I. Ye, Nong. II. Series.

QA76.9.D343 H385 2003
006.3—dc21 2002156029

Books published by Lawrence Erlbaum Associates are printed on acid-free paper, and their bindings are chosen for strength and durability.

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

Contents

Foreword <i>Gavriel Salvendy</i>	xviii
Preface <i>Nong Ye</i>	xix
About the Editor	xxiii
Advisory Board	xxv
Contributors	xxvii

I: METHODOLOGIES OF DATA MINING

1	Decision Trees	3
	<i>Johannes Gehrke</i>	
	Introduction	3
	Problem Definition	4
	Classification Tree Construction	7
	Split Selection	7
	Data Access	8
	Tree Pruning	15
	Missing Values	17
	A Short Introduction to Regression Trees	20
	Problem Definition	20
	Split Selection	20
	Data Access	21
	Applications and Available Software	22
	Cataloging Sky Objects	22
	Decision Trees in Today's Data Mining Tools	22
	Summary	22
	References	23
2	Association Rules	25
	<i>Geoffrey I. Webb</i>	
	Introduction	26
	Market Basket Analysis	26
	Association Rule Discovery	27
	The Apriori Algorithm	28
	The Power of the Frequent Item Set Strategy	29
	Measures of Interestingness	31

Lift	31
Leverage	32
Item Set Discovery	32
Techniques for Frequent Item Set Discovery	33
Closed Item Set Strategies	33
Long Item Sets	35
Sampling	35
Techniques for Discovering Association Rules without Item Set Discovery	35
Associations with Numeric Values	36
Applications of Association Rule Discovery	36
Summary	37
References	38
3 Artificial Neural Network Models for Data Mining	41
<i>Jennie Si, Benjamin J. Nelson, and George C. Runger</i>	
Introduction to Multilayer Feedforward Networks	42
Gradient Based Training Methods for MFN	43
The Partial Derivatives	44
Nonlinear Least Squares Methods	45
Batch versus Incremental Learning	47
Comparison of MFN and Other Classification Methods	47
Decision Tree Methods	47
Discriminant Analysis Methods	48
Multiple Partition Decision Tree	49
A Growing MFN	50
Case Study 1—Classifying Surface Texture	52
Experimental Conditions	52
Quantitative Comparison Results of Classification Methods	53
Closing Discussions on Case 1	55
Introduction to SOM	55
The SOM Algorithm	56
SOM Building Blocks	57
Implementation of the SOM Algorithm	58
Case Study 2—Decoding Monkey’s Movement Directions from Its Cortical Activities	59
Trajectory Computation from Motor Cortical Discharge Rates	60
Using Data from Spiral Tasks to Train the SOM	62
Using Data from Spiral and Center→Out Tasks to Train the SOM	62
Average Testing Result Using the Leave-K-Out Method	63
Closing Discussions on Case 2	64
Final Conclusions and Discussions	65
References	65
4 Statistical Analysis of Normal and Abnormal Data	67
<i>Connie M. Borror</i>	
Introduction	67
Univariate Control Charts	68
Variables Control Charts	68
Attributes Control Charts	81

Cumulative Sum Control Charts	89
Exponentially Weighted Moving Average Control Charts	93
Choice of Control Charting Techniques	95
Average Run Length	96
Multivariate Control Charts	98
Data Description	98
Hotelling T^2 Control Chart	98
Multivariate EWMA Control Charts	101
Summary	102
References	102
5 Bayesian Data Analysis	103
<i>David Madigan and Greg Ridgeway</i>	
Introduction	104
Fundamentals of Bayesian Inference	104
A Simple Example	104
A More Complicated Example	106
Hierarchical Models and Exchangeability	109
Prior Distributions in Practice	111
Bayesian Model Selection and Model Averaging	113
Model Selection	113
Model Averaging	114
Model Assessment	114
Bayesian Computation	115
Importance Sampling	115
Markov Chain Monte Carlo (MCMC)	116
An Example	117
Application to Massive Data	118
Importance Sampling for Analysis of Massive Data Sets	118
Variational Methods	120
Bayesian Modeling	121
BUGS and Models of Realistic Complexity via MCMC	121
Bayesian Predictive Modeling	125
Bayesian Descriptive Modeling	127
Available Software	128
Discussion and Future Directions	128
Summary	128
Acknowledgments	129
References	129
6 Hidden Markov Processes and Sequential Pattern Mining	133
<i>Steven L. Scott</i>	
Introduction to Hidden Markov Models	134
Parameter Estimation in the Presence of Missing Data	136
The EM Algorithm	136
MCMC Data Augmentation	138
Missing Data Summary	140
Local Computation	140
The Likelihood Recursion	140

The Forward-Backward Recursions	141
The Viterbi Algorithm	142
Understanding the Recursions	143
A Numerical Example Illustrating the Recursions	143
Illustrative Examples and Applications	144
Fetal Lamb Movements	144
The Business Cycle	150
HMM Stationary and Predictive Distributions	153
Stationary Distribution of d_t	153
Predictive Distributions	154
Posterior Covariance of h	154
Available Software	154
Summary	154
References	155
 7 Strategies and Methods for Prediction	 159
<i>Greg Ridgeway</i>	
Introduction to the Prediction Problem	160
Guiding Examples	160
Prediction Model Components	161
Loss Functions—What We are Trying to Accomplish	162
Common Regression Loss Functions	162
Common Classification Loss Functions	163
Cox Loss Function for Survival Data	166
Linear Models	167
Linear Regression	168
Classification	169
Generalized Linear Model	172
Nonlinear Models	174
Nearest Neighbor and Kernel Methods	174
Tree Models	177
Smoothing, Basis Expansions, and Additive Models	179
Neural Networks	182
Support Vector Machines	183
Boosting	185
Availability of Software	188
Summary	189
References	190
 8 Principal Components and Factor Analysis	 193
<i>Daniel W. Apley</i>	
Introduction	194
Examples of Variation Patterns in Correlated Multivariate Data	194
Overview of Methods for Identifying Variation Patterns	197
Representation and Illustration of Variation Patterns in Multivariate Data	197
Principal Components Analysis	198
Definition of Principal Components	199
Using Principal Components as Estimates of the Variation Patterns	199

Factor Rotation	202
Capabilities and Limitations of PCA	202
Methods for Factor Rotation	203
Blind Source Separation	205
The Classic Blind Source Separation Problem	205
Blind Separation Principles	206
Fourth-Order Blind Separation Methods	208
Additional Manufacturing Applications	211
Available Software	211
Summary	212
References	212
 9 Psychometric Methods of Latent Variable Modeling	 215
<i>Edward Ip, Igor Cadez, and Padhraic Smyth</i>	
Introduction	216
Basic Latent Variable Models	217
The Basic Latent Class Model	217
The Basic Finite Mixture Model	221
The Basic Latent Trait Model	224
The Basic Factor Analytic Model	226
Common Structure	229
Extension for Data Mining	229
Extending the Basic Latent Class Model	229
Extending the Basic Mixture Model	232
Extending the Latent Trait Model	233
Extending the Factor Analytic Model	234
An Illustrative Example	236
Hierarchical Structure in Transaction Data	236
Individualized Mixture Models	237
Data Sets	238
Experimental Results	238
References and Tools	241
References	241
Tools	243
Summary	244
References	244
 10 Scalable Clustering	 247
<i>Joydeep Ghosh</i>	
Introduction	248
Clustering Techniques: A Brief Survey	249
Partitional Methods	250
Hierarchical Methods	255
Discriminative versus Generative Models	256
Assessment of Results	256
Visualization of Results	258
Clustering Challenges in Data Mining	259
Transactional Data Analysis	259

Next Generation Clickstream Clustering	260
Clustering Coupled Sequences	261
Large Scale Remote Sensing	261
Scalable Clustering for Data Mining	262
Scalability to Large Number of Records or Patterns, N	262
Scalability to Large Number of Attributes or Dimensions, d	264
Balanced Clustering	266
Sequence Clustering Techniques	266
Case Study: Similarity Based Clustering of Market Baskets and Web Logs	267
Case Study: Impact of Similarity Measures on Web Document Clustering	270
Similarity Measures: A Sampler	270
Clustering Algorithms and Text Data Sets	272
Comparative Results	273
Clustering Software	274
Summary	274
Acknowledgments	274
References	275
 11 Time Series Similarity and Indexing	 279
<i>Gautam Das and Dimitrios Gunopulos</i>	
Introduction	279
Time Series Similarity Measures	281
Euclidean Distances and L_p Norms	281
Normalization Transformations	282
General Transformations	282
Dynamic Time Warping	283
Longest Common Subsequence Similarity	284
Piecewise Linear Representations	287
Probabilistic Methods	288
Other Similarity Measures	288
Indexing Techniques for Time Series	289
Indexing Time Series When the Distance Function Is a Metric	290
A Survey of Dimensionality Reduction Techniques	292
Similar Time-Series Retrieval When the Distance Function Is Not a Metric	299
Subsequence Retrieval	301
Summary	302
References	302
 12 Nonlinear Time Series Analysis	 305
<i>Ying-Cheng Lai, Zonghua Liu, Nong Ye, and Tolga Yalcinkaya</i>	
Introduction	305
Embedding Method for Chaotic Time Series Analysis	307
Reconstruction of Phase Space	307
Computation of Dimension	309
Detection of Unstable Periodic Orbits	311
Computing Lyapunov Exponents from Time Series	317
Time-Frequency Analysis of Time Series	323
Analytic Signals and Hilbert Transform	324
Method of EMD	331

Summary	338
Acknowledgment	338
References	338
13 Distributed Data Mining	341
<i>Byung-Hoon Park and Hillol Kargupta</i>	
Introduction	342
Related Research	343
Data Distribution and Preprocessing	344
Homogeneous/Heterogeneous Data Scenarios	345
Data Preprocessing	345
Distributed Data Mining Algorithms	346
Distributed Classifier Learning	346
Collective Data Mining	349
Distributed Association Rule Mining	350
Distributed Clustering	351
Privacy Preserving Distributed Data Mining	352
Other DDM Algorithms	353
Distributed Data Mining Systems	353
Architectural Issues	354
Communication Models in DDM Systems	356
Components Maintenance	356
Future Directions	357
References	358
II: MANAGEMENT OF DATA MINING	
14 Data Collection, Preparation, Quality, and Visualization	365
<i>Dorian Pyle</i>	
Introduction	366
How Data Relates to Data Mining	366
The “10 Commandments” of Data Mining	368
What You Need to Know about Algorithms Before Preparing Data	369
Why Data Needs to be Prepared Before Mining It	370
Data Collection	370
Choosing the Right Data	370
Assembling the Data Set	371
Assaying the Data Set	372
Assessing the Effect of Missing Values	373
Data Preparation	374
Why Data Needs Preparing: The Business Case	374
Missing Values	375
Representing Time: Absolute, Relative, and Cyclic	376
Outliers and Distribution Normalization	377
Ranges and Normalization	378
Numbers and Categories	379
Data Quality	380
What Is Quality?	382
Enforcing Quality: Advantages and Disadvantages	384
Data Quality and Model Quality	384

Data Visualization	384
Seeing Is Believing	385
Absolute Versus Relative Visualization	388
Visualizing Multiple Interactions	391
Summary	391
15 Data Storage and Management	393
<i>Tong (Teresa) Wu and Xiangyang (Sean) Li</i>	
Introduction	393
Text Files and Spreadsheets	395
Text Files for Data	395
Spreadsheet Files	395
Database Systems	397
Historical Databases	397
Relational Database	398
Object-Oriented Database	399
Advanced Topics in Data Storage and Management	402
OLAP	402
Data Warehouse	403
Distributed Databases	404
Available Software	406
Summary	406
Acknowledgments	407
References	407
16 Feature Extraction, Selection, and Construction	409
<i>Huan Liu, Lei Yu, and Hiroshi Motoda</i>	
Introduction	410
Feature Extraction	411
Concepts	411
Algorithms	412
An Example	413
Summary	413
Feature Selection	414
Concepts	414
Algorithm	415
An Example	416
Summary	417
Feature Construction	417
Concepts	417
Algorithms and Examples	418
Summary	419
Some Applications	420
Summary	421
References	422
17 Performance Analysis and Evaluation	425
<i>Sholom M. Weiss and Tong Zhang</i>	
Overview of Evaluation	426
Training versus Testing	426

Measuring Error	427
Error Measurement	427
Error from Regression	428
Error from Classification	429
Error from Conditional Density Estimation	429
Accuracy	430
False Positives and Negatives	430
Precision, Recall, and the <i>F</i> Measure	430
Sensitivity and Specificity	431
Confusion Tables	431
ROC Curves	432
Lift Curves	432
Clustering Performance: Unlabeled Data	432
Estimating Error	433
Independent Test Cases	433
Significance Testing	433
Resampling and Cross-Validation	435
Bootstrap	436
Time Series	437
Estimating Cost and Risk	437
Other Attributes of Performance	438
Training Time	438
Application Time	438
Interpretability	438
Expert Evaluation	439
Field Testing	439
Cost of Obtaining Labeled Data	439
References	439
 18 Security and Privacy	 441
<i>Chris Clifton</i>	
Introduction: Why There Are Security and Privacy Issues with Data Mining	441
Detailed Problem Analysis, Solutions, and Ongoing Research	442
Privacy of Individual Data	442
Fear of What Others May Find in Otherwise Releasable Data	448
Summary	451
References	451
 19 Emerging Standards and Interfaces	 453
<i>Robert Grossman, Mark Hornick, and Gregor Meyer</i>	
Introduction	453
XML Standards	454
XML for Data Mining Models	454
XML for Data Mining Metadata	456
APIs	456
SQL APIs	456
Java APIs	457
OLE DB APIs	457

Web Standards	457
Semantic Web	457
Data Web	458
Other Web Services	458
Process Standards	458
Relationships	458
Summary	459
References	459

III: APPLICATIONS OF DATA MINING

20	Mining Human Performance Data	463
	<i>David A. Nembhard</i>	
	Introduction and Overview	463
	Mining for Organizational Learning	464
	Methods	464
	Individual Learning	467
	Data on Individual Learning	468
	Methods	468
	Individual Forgetting	474
	Distributions and Patterns of Individual Performance	474
	Other Areas	476
	Privacy Issues for Human Performance Data	477
	References	477
21	Mining Text Data	481
	<i>Ronen Feldman</i>	
	Introduction	482
	Architecture of Text Mining Systems	483
	Statistical Tagging	485
	Text Categorization	485
	Term Extraction	489
	Semantic Tagging	489
	DIAL	491
	Development of IE Rules	493
	Auditing Environment	499
	Structural Tagging	500
	Given	500
	Find	500
	Taxonomy Construction	501
	Implementation Issues of Text Mining	505
	Soft Matching	505
	Temporal Resolution	506
	Anaphora Resolution	506
	To Parse or Not to Parse?	507
	Database Connectivity	507
	Visualizations and Analytics for Text Mining	508
	Definitions and Notations	508
	Category Connection Maps	509

Relationship Maps	510
Trend Graphs	516
Summary	516
References	517
22 Mining Geospatial Data	519
<i>Shashi Shekhar and Ranga Raju Vatsavai</i>	
Introduction	520
Spatial Outlier Detection Techniques	521
Illustrative Examples and Application Domains	521
Tests for Detecting Spatial Outliers	522
Solution Procedures	525
Spatial Colocation Rules	525
Illustrative Application Domains	526
Colocation Rule Approaches	527
Solution Procedures	530
Location Prediction	530
An Illustrative Application Domain	530
Problem Formulation	532
Modeling Spatial Dependencies Using the SAR and MRF Models	533
Logistic SAR	534
MRF Based Bayesian Classifiers	535
Clustering	537
Categories of Clustering Algorithms	539
K-Medoid: An Algorithm for Clustering	540
Clustering, Mixture Analysis, and the EM Algorithm	541
Summary	544
Acknowledgments	545
References	545
23 Mining Science and Engineering Data	549
<i>Chandrika Kamath</i>	
Introduction	550
Motivation for Mining Scientific Data	551
Data Mining Examples in Science and Engineering	552
Data Mining in Astronomy	552
Data Mining in Earth Sciences	555
Data Mining in Medical Imaging	557
Data Mining in Nondestructive Testing	557
Data Mining in Security and Surveillance	558
Data Mining in Simulation Data	558
Other Applications of Scientific Data Mining	561
Common Challenges in Mining Scientific Data	561
Potential Solutions to Some Common Problems	562
Data Registration	564
De-Noising Data	565
Object Identification	566
Dimensionality Reduction	567
Generating a Good Training Set	568
Software for Scientific Data Mining	568

Summary	569	
References	569	
24	Mining Data in Bioinformatics	573
<i>Mohammed J. Zaki</i>		
Introduction	574	
Background	574	
Basic Molecular Biology	574	
Mining Methods in Protein Structure Prediction	575	
Mining Protein Contact Maps	577	
Classifying Contacts Versus Noncontacts	578	
Mining Methodology	578	
How Much Information Is There in Amino Acids Alone?	581	
Using Local Structures for Contact Prediction	582	
Characterizing Physical, Protein-Like Contact Maps	587	
Generating a Database of Protein-Like Structures	588	
Mining Dense Patterns in Contact Maps	589	
Pruning and Integration	590	
Experimental Results	591	
Future Directions for Contact Map Mining	593	
Heuristic Rules for “Physicality”	593	
Rules for Pathways in Contact Map Space	594	
Summary	595	
References	596	
25	Mining Customer Relationship Management (CRM) Data	597
<i>Robert Cooley</i>		
Introduction	597	
Data Sources	599	
Data Types	599	
E-Commerce Data	601	
Data Preparation	604	
Data Aggregation	605	
Feature Preparation	607	
Pattern Discovery	608	
Pattern Analysis and Deployment	610	
Robustness	610	
Interestingness	611	
Deployment	611	
Sample Business Problems	612	
Strategic Questions	612	
Operational Questions	613	
Summary	615	
References	616	
26	Mining Computer and Network Security Data	617
<i>Nong Ye</i>		
Introduction	618	
Intrusive Activities and System Activity Data	618	

Phases of Intrusions	619
Data of System Activities	620
Extraction and Representation of Activity Features for Intrusion Detection	623
Features of System Activities	624
Feature Representation	625
Existing Intrusion Detection Techniques	628
Application of Statistical Anomaly Detection Techniques to Intrusion Detection	629
Hotelling's T^2 Test and Chi-Square Distance Test	629
Data Source and Representation	631
Application of Hotelling's T^2 Test and Chi-Square Distance Test	633
Testing Performance	633
Summary	634
References	635
27 Mining Image Data	637
<i>Chabane Djeraba and Gregory Fernandez</i>	
Introduction	637
Related Works	639
Method	641
How to Discover the Number of Clusters: k	641
K -Automatic Discovery Algorithm	644
Clustering Algorithm	646
Experimental Results	646
Data Sets	647
Data Item Representation	648
Evaluation Method	649
Results and Analysis	650
Summary	654
References	655
28 Mining Manufacturing Quality Data	657
<i>Murat C. Testik and George C. Runger</i>	
Introduction	657
Multivariate Control Charts	658
Hotelling T^2 Control Charts	658
MEWMA Charts	660
Nonparametric Properties of the MEWMA Control Charts	663
Summary	667
References	668
Author Index	669
Subject Index	681

Foreword

With the rapid introduction of highly sophisticated computers, (tele)communication, service, and manufacturing systems, a major shift has occurred in the way people use technology and work with it. The objective of this book series on Human Factors and Ergonomics is to provide researchers and practitioners a platform where important issues related to these changes can be discussed, and methods and recommendations can be presented for ensuring that emerging technologies provide increased productivity, quality, satisfaction, safety, and health in the new workplace and the Information Society.

This Handbook was created with the input of a distinguished International Board of 15 who represented some of the foremost authorities in Data Mining from academia and industry. Nong Ye very astutely established this Board to ensure that a balanced depth and breadth coverage occurs in the Handbook. The 28 chapters of the Handbook were authored by 45 of some of the leading international authorities, representing four continents.

The 245 figures and 72 tables of the Handbook illustrate, in a most effective way, the concept, methods, and tools of data mining. The 1,164 references of the Handbook provide a road map for further in-depth study of any of the data mining concepts and methods. Thirteen of the chapters deal with the methodologies of data mining; six chapters deal with how to manage data mining for maximizing outcome utility. Finally, nine chapters illustrate the methods and processes utilized for diversified data mining applications.

The Handbook should be of use to both developers of data mining methods and tools and to those who want to use data mining in order to derive scientific inferences relating to specific domains where extensive data is available in scattered reports and publications.

—Gavriel Salvendy
Series Editor
West Lafayette, Indiana
July 2002

Preface

Advanced technologies have enabled us to collect large amounts of data on a continuous or periodic basis in many fields. On one hand, these data present the potential for us to discover useful information and knowledge that we could not see before. On the other hand, we are limited in our ability to manually process large amounts of data to discover useful information and knowledge. This limitation demands automatic tools for data mining to mine useful information and knowledge from large amounts of data. Data mining has become an active area of research and development. This book presents comprehensive coverage of data mining concepts, algorithms, methodologies, management issues, and tools, which are all illustrated through simple examples and real-world applications for an easy understanding and mastering of those materials. Necessary materials for data mining are presented and organized coherently in one volume. This enables one to quickly start and conduct research work or practical applications of data mining without spending precious time searching for those materials from many different sources. Advanced topics in the area of data mining are also introduced in this handbook with extensive references for further readings.

Materials in this handbook are organized into three parts on methodologies, management, and applications of data mining, respectively. Part I includes chapters 1–13, which present various data mining methodologies, including concepts, algorithms, and available software tools for each methodology. Chapter 1 describes decision trees—a widely used data mining methodology in practice for learning prediction and classification models, data patterns, and so on. Chapter 2 introduces association rules—a data mining methodology that is usually used to discover frequently cooccurring data items, for example, items that are commonly purchased together by customers at grocery stores. Chapter 3 presents artificial neural networks with some models, which support the supervised learning of prediction and classification models, and other models, which support unsupervised learning of data structures. Chapter 4 describes statistical process control techniques that can be used to reveal data similarities and differences, especially when dealing with two categories of data: data produced in normal conditions and data produced in abnormal conditions, for anomaly detection. Chapter 5 introduces the Bayesian approach to various data mining problems. Chapter 6 presents hidden Markov models that can be used to mine sequential data patterns. Chapter 7 provides a coherent framework for a systematic understanding of existing prediction and classification models for data mining. Chapter 8 describes principal components analysis—a statistical method that is often used to reduce data dimensions and reveal the underlying structure of data. Chapter 9 introduces psychometric methods of latent variable modeling that can be used to reveal unobserved latent variables and their relationships with observed data variables. Chapter 10 presents both traditional clustering methods and advanced methods for scalable clustering of data to generate homogeneous groups of data and reveal data patterns. Chapter 11 presents methods of determining similarity of time-series data and generating the index of time-series data for data retrieval based on time-series similarity. Chapter 12 introduces nonlinear time-series analysis methods, including wavelet analysis, which may not be well known to the data mining

community but can be useful in revealing nonlinear data patterns and their evolution over time. Chapter 13 presents methods of performing data mining in a distributed manner.

Part II of this handbook includes chapters 14–19, which address various issues we typically face in the life cycle of data mining projects. Before we use a data mining methodology in part I of this handbook, we often need to understand what information and knowledge we want to get, which data contain such information and knowledge for us to collect, how we prepare the collected data in an appropriate form for data mining, and how we preprocess the data to improve the data quality for better data mining outcomes. These issues are addressed in chapter 14 on data collection, preparation, quality, and visualization. The data usually need to be stored on computers and retrieved for data mining. Chapter 15 describes data storage and management. Often the collected raw data are not directly useful for revealing the information and knowledge that we look for in data mining. Features need to be extracted, selected, or constructed from the raw data to reveal useful information and knowledge. Chapter 16 presents methods of feature extraction, selection, and construction. Chapter 17 introduces various measures and methods to evaluate the performance of a data mining methodology and its outcome. Giving access to data for data mining may jeopardize data security and privacy that aims to protect confidential and private information, which may be revealed from data mining. Concerns about data security and privacy may prevent the release of data for data mining, and thus hinder data mining projects. Chapter 18 presents issues of data security and privacy and methods of overcoming problems caused by concerns about data security and privacy. Because data may come from different sources and be collected in different formats, it is important to establish standards and interfaces for facilitating the use of various kinds of data. Chapter 19 introduces emerging standards and interfaces that will greatly benefit data mining projects.

Part III of this handbook includes chapters 20–28, which illustrate the applications of data mining to various kinds of data, including human performance data in chapter 20, text data in chapter 21, geospatial data in chapter 22, science and engineering data in chapter 23, data in bioinformatics in chapter 24, online and offline customer transaction and activity data in chapter 25, security related computer audit data and network traffic data in chapter 26, image data in chapter 27, and manufacturing quality data in chapter 28. From these chapters in part III, we can see how various data mining methodologies are applied and various management issues of data mining projects are addressed in real-world applications in many fields. Hence, these chapters illustrate the integrated applications of materials in parts I and II in various application contexts.

The type of available data and the nature of a data mining problem typically determines which data mining methodology is appropriate. A data mining problem may aim at predicting and classifying data, uncovering data patterns to gain insights into certain phenomena, recognizing data similarities and differences, and so on. Data may come without the attachment of any prior knowledge (unlabelled data) or with knowledge about data categories or groups (labeled data, e.g., data of organization shoppers and individual shoppers in purchasing data at grocery stores) in the form of data records with or without time-stamps to indicate the time sequence of the data records (separate data records or time series data), and so on. Table 1 gives a guide to which data mining methodologies are typically used for which kinds of data and data mining problems. However, this table does not cover all kinds of data and data mining problems that one may have, and it shows only the typical use of the various data mining methodologies described in this handbook. The use of data mining methodologies may go beyond those included in the table. Someone who is interested in applying data mining to a problem in a specific application domain may start with part III of this handbook, to find the chapter(s) dealing with the application data that are most similar to the data they have, and learn which data mining methodologies can be applied and how they are applied. For those

TABLE 1
A Guide for the Typical Use of Data Mining Methodologies for Various Data Mining Problems and Data Types

Data Mining Methodology	Data Type				Data Mining Problem		
	Labeled Data	Unlabeled Data	Separate Data Records	Time Series Data	Predication and Classification	Discovery of Data Patterns, Associations, and Structure	Recognition of Data Similarities and Differences
Decision trees (chapter 1)	X		X		X	X	X
Association rules (chapter 2)		X	X			X	X
Artificial neural networks (chapter 3)	X	X	X	X	X		X
Statistical analysis of normal and abnormal data (chapter 4)		X	X	X	X	X	X
Bayesian data analysis (chapter 5)	X	X	X	X	X	X	X
Hidden Markov processes and sequential pattern mining (chapter 6)	X	X		X	X		X
Prediction and classification models (chapter 7)	X		X	X	X	X	X
Principal components analysis (chapter 8)		X	X			X	X
Psychometric methods of latent variable modeling (chapter 9)	X	X	X		X	X	X
Scalable clustering (chapter 10)		X	X			X	X
Time series similarity and indexing (chapter 11)	X	X		X	X		X
Nonlinear time series analysis (chapter 12)	X	X	X	X	X	X	X

data mining methodologies used in the chapter(s) in part III, one may get a detailed description of the methodologies by reading the corresponding chapters in part I of this handbook. After understanding the data mining methodologies and their application to specific application data, one may read the chapters in part II of this handbook to get familiar with issues and methods of carrying out a data mining project. Someone who is interested in research on a data mining methodology, management issue, or application may read the corresponding chapter in this handbook to learn the current state of the art and active research problems and areas.

Many people contributed greatly to this handbook. First of all, I would like to thank Gavriel Salvendy who put his trust in me and invited me to edit this handbook. He is my mentor and friend, who is always helpful, supportive, and inspirational, for which I appreciate him most sincerely. I am grateful to the members of the advisory board, who have helped me put together a list of possible authors for each topic area. Many members of the advisory board have also taken time to review the chapters in this handbook and make helpful comments. I thank all the contributing authors who have done such an excellent job on their chapters. Without their contribution and cooperation, this handbook would not be possible. I would like to thank Toni Farley, who helped edit some chapters and organize the final set of the submitted materials; and Joshua Brown, who went over most chapters to make editing comments. My thanks also go to Senior Editor Debra Riegert, her editorial assistant, Jason Planer, Production Manager Paul Smolenski at Lawrence Erlbaum Associates, and Joanne Bowser at TechBooks, who worked with me on this project in a truly pleasant, helpful way. It was my great pleasure to work with all of these wonderful people.

I would like to thank my family, whose love and support made it all possible and delightful.

—Nong Ye
Tempe, Arizona
June 2002

About the Editor

Dr. Nong Ye is a Professor of Industrial Engineering and an Affiliated Professor of Computer Science and Engineering at Arizona State University (ASU). She is the founding director of the Information and Systems Assurance Laboratory at ASU. Dr. Ye holds a Ph.D. degree in Industrial Engineering from Purdue University, West Lafayette, Indiana, and a M.S. degree in Computer Science from the Chinese Academy of Sciences, Beijing, China, and a B.S. degree in Computer Science from Peking University, Beijing, China. Her research work focuses on establishing the scientific and engineering foundation for assuring quality/reliability of information systems and industrial systems. Her research in this area has advanced theories and techniques in

- Statistical process control and multivariate statistical analysis,
- Data mining and management,
- Optimization and decision making,
- System modeling and simulation, especially for complex adaptive systems with interacting, autonomous agents and emergent phenomena.

Dr. Ye's research accomplishments include over \$6.6M external research funding and over ninety journal and conference publications. Dr. Ye has served on the Editorial Board of the *International Journal of Human-Computer Interaction* and the *International Journal of Cognitive Ergonomics*, and has been the guest editor of a special journal issue for *IEEE Transactions on Reliability, Quality and Reliability Engineering International*, and *International Journal of Human-Computer Interaction*. She is a senior member of the Institute of Industrial Engineers (IIE) and a senior member of IEEE.

Editor

Nong Ye
Arizona State University
Tempe, Arizona

Series Editor

Gavriel Salvendy
Purdue University
West Lafayette, Indiana

Advisory Board

Christopher W. Clifton
Associate Professor of Computer
Science
Purdue University
West Lafayette, Indiana

Hiroshi Motoda
Department of Advance Reasoning
Division of Intelligent Systems Science
The Institute of Scientific and Industrial
Research
Osaka University
Osaka, Japan

Joydeep Ghosh
Professor
Department of Electrical & Computer
Engineering
University of Texas at Austin
Austin, Texas

Ross Quinlan
Director
RuleQuest Research Pty Ltd
St Ives, Australia

Richard J. Koubek
Professor and Head
The Harold and Inge Marcus Department of
Industrial and Manufacturing Engineering
Penn State University
University Park, Pennsylvania

Greg Ridgeway
Associate Statistician
Economics and Statistics Group
RAND Corporation
Santa Monica, California

Huan Liu
Associate Professor
Department of Computer Science and
Engineering
Arizona State University
Tempe, Arizona

Benjamin Van Roy
Assistant Professor
Management Science and Engineering
Electrical Engineering
Computer Science (by courtesy)
Stanford University
Stanford, California

Douglas C. Montgomery
Professor
Department of Industrial Engineering
Arizona State University
Tempe, Arizona

Gavriel Salvendy
Professor
Purdue University
West Lafayette, Indiana

Michael J. Smith
Professor
Industrial Engineering
Engineering Professional Development
University of Wisconsin
Madison, Wisconsin

Geoffrey I. Webb
Chair of Information Technology Research
School of Computer Science and Software
Engineering
Monash University
Clayton, Victoria, Australia

Sholom M. Weiss
Research Staff Member
IBM Thomas J. Watson Research Center
Yorktown Heights, New York

Fuqing Yang
Professor
Department of Computer Science and
Technology
Peking University
Beijing, P. R. China

Philip S. Yu
Manager of Software Tools and Techniques
Group
IBM Thomas J. Watson Research Center
Yorktown Heights, New York

Contributors

Daniel W. Apley

Assistant Professor
Department of Industrial Engineering
Texas A&M University
College Station, Texas
apley@tamu.edu

Connie M. Borror

Lecturer
Department of Industrial Engineering
Arizona State University
Tempe, Arizona
conni@asu.edu

Igor V. Cadez

Graduate Student
Department of Information and
Computer Science
University of California Irvine
Irvine, California
icadez@ics.uci.edu

Christopher W. Clifton

Associate Professor
of Computer Science
Purdue University
West Lafayette, Indiana
clifton@cs.purdue.edu

Robert Cooley

U.S. Technical Director
KXEN
San Francisco, California
rob.cooley@kxen.com

Gautam Das

Researcher
Data Management, Exploration and
Mining Group
Microsoft Research
Redmond, Washington
gautamd@microsoft.com

Chabane Djeraba

Associate Professor
Polytechnic School of Nantes University
Nantes Cedex, France
djeraba@irin.univ-nantes.fr

Ronen Feldman

Senior Lecturer
Mathematics and Computer Science
Department
Director of the Data Mining Laboratory
Bar-Ilan University
Ramat Gan, Israel
ronen@clearforest.com

Gregory Fernandez

Ph.D. Student
Wayne State University
Detroit, Michigan
gfernand@ifrance.com

Johannes Gehrke

Assistant Professor
Department of Computer Science
Cornell University
Ithaca, New York
johannes@cs.cornell.edu

Joydeep Ghosh

Professor
Department of Electrical & Computer
Engineering
University of Texas at Austin
Austin, Texas
ghosh@ece.utexas.edu

Robert Grossman

Director of the National Center for Data
Mining
University of Illinois at Chicago
Chicago, Illinois
grossman@uic.edu

Dimitrios Gunopulos

Associate Professor
Department of Computer Science and
Engineering
University of California Riverside
Riverside, California
dg@cs.ucr.edu

Mark F. Hornick

Senior Manager
Oracle Corporation
Burlington Massachusetts
mark.hornick@oracle.com

Edward Ip

Assistant Professor
Information and Operations
Management
Marshall School of Business
University of Southern California
Los Angeles, California
eddie.ip@marshall.usc.edu

Chandrika Kamath

Computer Scientist
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, California
kamath2@llnl.gov

Hillol Kargupta

Assistant Professor
Department of Computer Science and Electrical
Engineering
University of Maryland Baltimore County
Baltimore, Maryland
hillol@cs.umbc.edu

Ying-Cheng Lai

Professor
Department of Mathematics
Department of Electrical Engineering
Arizona State University
Tempe, Arizona
yclai@chaos1.la.asu.edu

Xiangyang Li

Post-Doctorate
Rensselaer Polytechnic Institute
Troy, New York
xyl199@hotmail.com

Huan Liu

Associate Professor
Department of Computer Science and
Engineering
Arizona State University
Tempe, Arizona
hliu@asu.edu

Zonghua Liu

Post-Doctoral Fellow
Department of Mathematics
Arizona State University
Tempe, Arizona
zhliu@chaos4.la.asu.edu

David Madigan

Professor of Statistics
Rutgers University
Piscataway, New Jersey
dmadigan@rci.rutgers.edu

Gregor Meyer

Senior Software Engineer
Business Intelligence
IBM
San Jose, California
gregorm@us.ibm.com

Hiroshi Motoda

Professor
Department of Advance Reasoning
Division of Intelligent Systems Science
The Institute of Scientific and Industrial
Research
Osaka University
Osaka, Japan
motoda@sanken.osaka-u.ac.jp

Benjamin Nelson

Senior Statistician
Weyerhaeuser Company
Tacoma, Washington
ben.nelson@weyerhaeuser.com

David A. Nembhard

Assistant Professor
Department of Industrial Engineering
The University of Wisconsin-Madison
Madison, Wisconsin
nembhard@engr.wisc.edu

Byung-Hoon Park

Research Associate
Computer Science & Mathematics
Division
Oak Ridge National Laboratory
Oak Ridge, TN
parklbh@ornl.gov

Dorian Pyle

Consultant
Data Miners
Leominster, Massachusetts
dpyle@modelandmine.com

Greg Ridgeway

Associate Statistician
Economics and Statistics Group
RAND Corporation
Santa Monica, California
gregr@rand.org

George C. Rungier

Professor
Department of Industrial Engineering
Arizona State University
Tempe, Arizona
George.Runger@asu.edu

Steven L. Scott

Assistant Professor of Statistics
Information and Operations Management
Department
Marshall School of Business
University of Southern California
Los Angeles, California
sls@usc.edu

Shashi Shekhar

Professor
Department of Computer Science
and Engineering
University of Minnesota
Minneapolis, Minnesota
shekhar@cs.umn.edu

Jennie Si

Professor
Department of Electrical Engineering
Arizona State University
Tempe, Arizona
si@asu.edu

Padhraic Smyth

Associate Professor
Department of Information
and Computer Science
University of California Irvine
Irvine, California
smyth@ics.uci.edu

Murat Testik

Ph.D. Candidate
Department of Industrial Engineering
Arizona State University
Tempe, Arizona
Murat.Testik@asu.edu

Ranga R. Vatsavai

Research Fellow
Remote Sensing and Geospatial Analysis
Laboratory
Department of Computer Science and
Engineering
University of Minnesota
Minneapolis, Minnesota
vatsavai@cs.umn.edu

Geoffrey I. Webb

Chair of Information Technology
Research
School of Computer Science and Software
Engineering
Monash University
Clayton, Victoria, Australia
webb@giwebb.com

Sholom M. Weiss

Research Staff Member
IBM Thomas J. Watson Research Center
Yorktown Heights, New York
weiss@us.ibm.com

Tong (Teresa) Wu

Assistant Professor
Department of Industrial Engineering
Arizona State University
Tempe, Arizona
teresa.wu@asu.edu

Tolga Yalcinkaya

Department of Physics and Astronomy
University of Kansas
Lawrence, Kansas 66045
info@allesta.com

Nong Ye

Professor

Department of Industrial Engineering

Affiliated Professor

Department of Computer Science and

Engineering

Arizona State University

Tempe, Arizona

nongye@asu.edu

Mohammed J. Zaki

Assistant Professor

Computer Science Department

Rensselaer Polytechnic

Institute

Troy, New York

zaki@cs.rpi.edu

Lei Yu

Ph.D. Student

Department of Computer Science and

Engineering

Arizona State University

Tempe, Arizona

leiyu@asu.edu

Tong Zhang

Research Staff Member

Knowledge Management Department

IBM Thomas J. Watson Research

Center

Yorktown Heights, New York

tzhang@watson.ibm.com

I

Methodologies of Data Mining

1

Decision Trees

Johannes Gehrke
Cornell University

Introduction	3
Problem Definition	4
Classification Tree Construction	7
Split Selection	7
Data Access	8
Tree Pruning	15
Missing Values	17
A Short Introduction to Regression Trees	20
Problem Definition	20
Split Selection	20
Data Access	21
Applications and Available Software	22
Cataloging Sky Objects	22
Decision Trees in Today's Data Mining Tools	22
Summary	22
References	23

INTRODUCTION

The overarching goal of classification and regression is to build a model that can be used for prediction. The following terminology will help make this notion precise. In a classification or regression problem, we are given a data set of *training records* (also called the *training*

database). Each record has several attributes. Attributes whose domain is numerical are called *numerical attributes*, whereas attributes whose domain is not numerical are called *categorical attributes*. (A *categorical* attribute takes values from a set of categories. Some authors distinguish between categorical attributes that take values in an unordered set [*nominal* attributes] and categorical attributes having ordered domains [*ordinal* attributes].)

There is one distinguished attribute called the *dependent attribute*. The remaining attributes are called *predictor attributes*; they are either numerical or categorical in nature. If the dependent attribute is categorical, the problem is referred to as a *classification problem*, and we call the dependent attribute the *class label*. (We will denote the elements of the domain of the class label attribute as *class labels*; the meaning of the term *class label* will be clear from the context.) If the dependent attribute is numerical, the problem is called a *regression problem*. In this chapter we concentrate on classification problems, although similar techniques can be applied to regression problems.

The goal of classification is to build a concise model of the distribution of the dependent attribute in terms of the predictor attributes. The resulting model is used to assign values to a database in which the values of the predictor attributes are known but the value of the dependent attribute is unknown. Classification has a wide range of applications, including scientific experiments, medical diagnosis, fraud detection, credit approval, and target marketing (Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996). Many classification models have been proposed in the literature: neural networks (Bishop, 1995; Ripley, 1996), genetic algorithms (Goldberg, 1989), Bayesian methods (Cheeseman & Stutz, 1996), log-linear models and other statistical methods (Agresti, 1990; Christensen, 1997; James, 1985), decision tables (Kohavi, 1995), and tree-structured models, so-called *classification trees* (Breiman, Friedman, Olshen, & Stone, 1984; Quinlan, 1986). There exist excellent overviews of classification methods (Hand, 1997; Michie, Spiegelhalter, & Taylor, 1994; Weiss & Kulikowski, 1991).

Classification trees are especially attractive in a data mining environment for several reasons. First, due to their intuitive representation, the resulting classification model is easy to understand (Breiman et al., 1984). Second, classification trees are nonparametric and thus especially suited for exploratory knowledge discovery. Third, classification trees can be constructed relatively fast compared to other methods (Lim, Loh, & Shih, 2000). And last, the accuracy of classification trees is comparable to other classification models (Hand, 1997; Lim et al., 2000). Every major data mining tool includes some form of classification tree model construction component (Goebel & Gruenwald, 1999).

In this chapter we introduce the reader to classification tree construction; an excellent survey of research papers about classification tree construction can be found in Murthy (1998). The remainder of this chapter is structured as follows: In the next section we state the problem of classification tree construction formally and introduce some terminology and notation that we use throughout this chapter. We then show how classification trees are constructed and discuss an important algorithmic component of classification tree construction, the split selection method, in detail. We also introduce the related problem of regression tree construction.

PROBLEM DEFINITION

Let X_1, \dots, X_m, C be random variables where X_i has domain $\text{dom}(X_i)$; we assume without loss of generality that $\text{dom}(C) = \{1, 2, \dots, J\}$. A *classifier* is a function

$$d : \text{dom}(X_1) \times \dots \times \text{dom}(X_m) \mapsto \text{dom}(C).$$

Let $P(X', C')$ be a probability distribution on $\text{dom}(X_1) \times \dots \times \text{dom}(X_m) \times \text{dom}(C)$ and let

Record Id	Car	Age	Children	Subscription
1	sedan	23	0	yes
2	sports	31	1	no
3	sedan	36	1	no
4	truck	25	2	no
5	sports	30	0	no
6	sedan	36	0	no
7	sedan	25	0	yes
8	truck	36	1	no
9	sedan	30	2	yes
10	sedan	31	1	yes
11	sports	25	0	no
12	sedan	45	1	yes
13	sports	23	2	no
14	truck	45	0	yes

FIG. 1.1. An example training database for magazine subscription.

$t = \langle t.X_1, \dots, t.X_m, t.C \rangle$ be a record randomly drawn from P ; i.e., t has probability $P(X', C')$ that $\langle t.X_1, \dots, t.X_m \rangle \in X'$ and $t.C \in C'$. We define the *misclassification rate* R_d of classifier d to be $P(d\langle t.X_1, \dots, t.X_m \rangle \neq t.C)$. In terms of the informal introduction to this chapter, the training database D is a random sample from P , the X_i correspond to the predictor attributes, and C is the class label attribute.

A *decision tree* is a special type of classifier. It is a directed, acyclic graph T in the form of a tree. We focus on binary decision trees, although the techniques we describe can be generalized to nonbinary decision trees. (See the survey by Murthy [1998] for more relevant citations.)

Thus, we assume in the remainder of this chapter that each node has either zero or two outgoing edges. If a node has no outgoing edges, it is called a *leaf node*; otherwise it is called an *internal node*. Each leaf node is labeled with one class label; each internal node n is labeled with one predictor attribute X_n called the *splitting attribute*. Each internal node n has a predicate q_n , called the *splitting predicate* associated with it. If X_n is a numerical attribute, q_n is of the form $X_n \leq x_n$, where $x_n \in \text{dom}(X_n)$; x_n is called the *split point* at node n . If X_n is a categorical attribute, q_n is of the form $X_n \in Y_n$ where $Y_n \subset \text{dom}(X_n)$; Y_n is called the *splitting subset* at node n . The combined information of splitting attribute and splitting predicates at node n is called the *splitting criterion* of n . An example training database is shown in Fig. 1.1, and a sample classification tree is shown in Fig. 1.2.

We associate with each node $n \in T$ a predicate

$$f_n : \text{dom}(X_1) \times \dots \times \text{dom}(X_m) \mapsto \{\text{true}, \text{false}\},$$

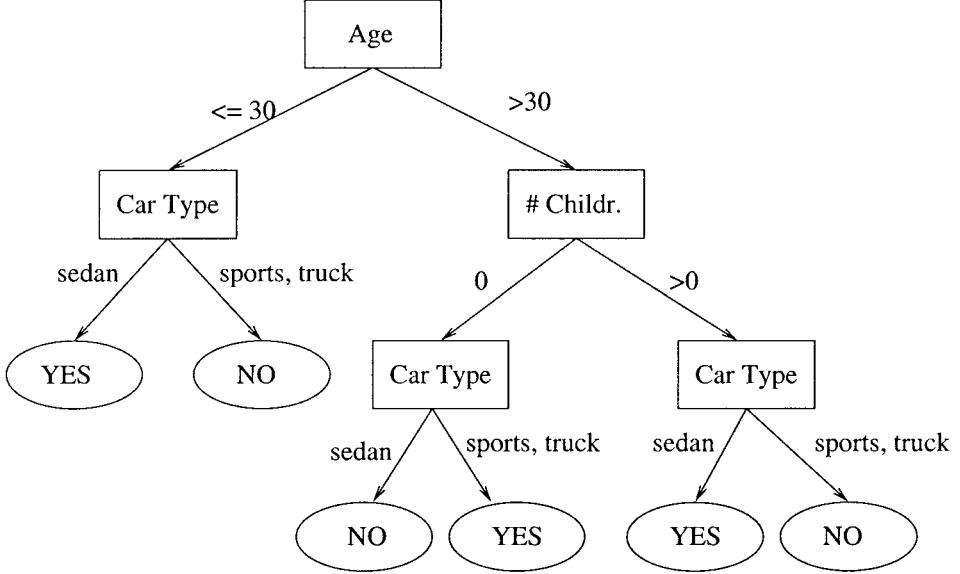


FIG. 1.2. Magazine subscription example classification tree.

called its *node predicate* as follows: For the root node n , $f_n \stackrel{\text{def}}{=} \text{true}$. Let n be a nonroot node with parent p whose splitting predicate is q_p . If n is the left child of p , define $f_n \stackrel{\text{def}}{=} f_p \wedge q_p$; if n is the right child of p , define $f_n \stackrel{\text{def}}{=} f_p \wedge \neg q_p$. Informally, f_n is the conjunction of all splitting predicates on the internal nodes on the path from the root node to n . Since each leaf node $n \in T$ is labeled with a class label, n encodes the classification rule $f_n \rightarrow c$, where c is the label of n . Thus, the tree T encodes a function $T : \text{dom}(X_1) \times \dots \times \text{dom}(X_m) \mapsto \text{dom}(C)$ and is therefore a classifier, called a *decision tree classifier*. (We will denote both the tree and the induced classifier by T ; the semantics will be clear from the context.) Let us define the notion of the *family of tuples* of a node in a decision tree T with respect to a database D . (We will drop the dependency on D from the notation because it is clear from the context.) For a node $n \in T$ with parent p , F_n is the set of records in D that follows the path from the root to n when being processed by the tree, formally

$$F_n \stackrel{\text{def}}{=} \{t \in D : f_n(t)\}.$$

We also define F_n^i for $i \in \{1, \dots, J\}$ as the set of records in F_n with class label i , formally

$$F_n^i \stackrel{\text{def}}{=} \{t \in D : f_n(t) \wedge t.C = i\}.$$

We can now formally state the problem of classification tree construction: Given a data set $D = \{t_1, \dots, t_n\}$ where the t_i are independent random samples from an unknown probability distribution P , find a decision tree classifier T such that the misclassification rate $R_T(P)$ is minimal.

A classification tree is usually constructed in two phases. In phase one, the *growth phase*, an overly large decision tree is constructed from the training data. In phase two, the *pruning phase*, the final size of the tree T is determined with the goal to minimize R_T . (It is possible to interleave growth and pruning phase for performance reasons as in the PUBLIC pruning method described later.) All decision tree construction algorithms grow the tree top-down in

Input: Node n , partition D , split selection method \mathcal{SS}

Output: Decision tree for D rooted at node n

Top-Down Decision Tree Induction Schema:

BuildTree(Node n , dataset D , split selection method \mathcal{SS})

- (1) Apply \mathcal{SS} to D to find the splitting criterion
- (2) **if** n splits
- (3) Use best split to partition D into D_1 and D_2
- (4) BuildTree(n_1, D_1, \mathcal{SS})
- (5) BuildTree(n_2, D_2, \mathcal{SS})
- (6) **endif**

FIG. 1.3. Classification tree construction.

the following greedy way: At the root node n , the training database is examined, and a splitting criterion for n is selected. Recursively, at a nonroot node n , the family of n is examined and from it a splitting criterion is selected. This schema is depicted in Fig. 1.3.

During the tree growth phase, two different algorithmic issues need to be addressed. The first issue is to devise an algorithm such that the resulting tree T minimizes R_T ; we will call this part of the overall decision tree construction algorithm the *split selection method*. The second issue is to devise an algorithm for data management for when the training database is very large; we will call this part of the overall decision tree construction algorithm the *data access method*. During the pruning phase a third issue arises, namely how to find a good estimator \widehat{R}_T of R_T and how to efficiently calculate \widehat{R}_T .

CLASSIFICATION TREE CONSTRUCTION

Split Selection

In the remainder of this section, we introduce a special class of split selection methods called *impurity-based* split selection methods. We concentrate on impurity-based split selection methods because this class of split selection methods is widely used (Breiman et al., 1984; Quinlan, 1986). We restrict our attention in this chapter to binary classification trees.

Impurity-based split selection methods find the splitting criterion by minimizing a concave *impurity function* imp_θ such as the entropy (Quinlan, 1986), the gini-index (Breiman et al., 1984) or the index of correlation χ^2 (Morimoto, Fukuda, Matsuzawa, Tokuyama, & Yoda, 1998). (Arguments for the concavity of the impurity function can be found in Breiman et al. [1984].) The most popular split selection methods such as CART (Breiman et al., 1984) and C4.5 (Quinlan, 1986) fall into this group. At each node, all predictor attributes X are examined and the impurity imp_θ of the best split on X is calculated. The final split is chosen such that the combination of splitting attribute and splitting predicates minimizes the value of imp_θ .

In this chapter we can only give the basics of impurity-based split point selection methods; a more formal treatment can be found in Breiman et al. (1984). Consider the example training database shown in Fig. 1.1. The training database has two numerical predictor attributes (*age* and *number of children*) and the class label has two values. Note that the splitting predicate $age \leq a$ for each attribute value a partitions D into D_1 and D_2 , where D_1 contains all records with $t.age \leq a$ and D_2 contains all remaining records. Our goal is to select a “good” attribute value a in the domain of *age* as splitting point. How can the quality of an attribute value as split point be quantified? Impurity-based split selection methods assess the quality of an attribute value a as potential split point value by calculating the value of an *impurity function*. Because the goal is to separate the two classes through the structure imposed by the decision tree, the arguments to the impurity function are the relative fractions of each class label in the partitions D_1 and D_2 induced by a , that is, the percentage of records with class label i in D_1 and D_2 for $i \in \{1, \dots, J\}$. The impurity function returns a value that measures how “pure” the training database would be if it were split into two parts at the split point candidate a .

As an example, consider the *gini*-index as impurity function. Let p_Y be the relative frequency of class label “Yes” and p_N be the relative frequency of class label “No” in the data set D . Then the *gini*-index of a binary split of a data set D into D_1 and D_2 is defined as follows:

$$\text{gini}(D_1, D_2) \stackrel{\text{def}}{=} \frac{|D_1|}{|D|} \text{gini}(D_1) + \frac{|D_2|}{|D|} \text{gini}(D_2),$$

where

$$\text{gini}(D) \stackrel{\text{def}}{=} 1 - p_Y^2 - p_N^2,$$

and $|S|$ denotes the number of elements in set S . Consider the training database D from Fig. 1.1 shown in Fig. 1.4 sorted on the predictor attribute *age*. The splitting predicate $age \leq 30$ induces a partitioning of D into D_1 and D_2 , where D_1 contains the set of records with record identifiers $\{1, 4, 5, 7, 9, 11, 13\}$ and D_2 contains the remaining records in D . Simple calculations show that $\text{gini}(D_1, D_2) = 0.4898$. Figure 1.5 shows the value of the *gini*-index for all split point candidates in the domain of the predictor attribute *age*; note that the value of the *gini*-index only changes at attribute values that occur in the training database. Also note that a sort of the training database D is required to test all possible splits of the form $X \leq c$. Thus, impurity-based split selection methods require (conceptually) a sort of the training database D for each numerical attribute at each node of the tree.

As another example, consider the entropy *ent* as impurity function. The entropy of a data set D is defined as follows:

$$\text{ent}(D) = -p_Y \cdot \log(p_Y) - p_N \cdot \log(p_N).$$

Note that split selection for a categorical attribute X is slightly different as splits are of the form $X \in Y$, where Y is a subset of the domain of X . For example, assume that we have a categorical attribute *car* with four different values: sports (SP), sedan (SE), truck (T), and minivan (M). Then split selection would investigate $Y = \{\text{SP}\}$, $Y = \{\text{SE}\}$, $Y = \{\text{T}\}$, $Y = \{\text{M}\}$, $Y = \{\text{SP,SE}\}$, $Y = \{\text{SP,T}\}$, $Y = \{\text{SP,M}\}$, $Y = \{\text{SE,T}\}$, $Y = \{\text{SE,M}\}$, and $Y = \{\text{T,M}\}$.

Data Access

Over the last few years, there has been a surge in the development of scalable data access methods for classification tree construction. Due to space limitation, we will only cover SPRINT (Shafer, Agrawal, & Mehta, 1996), RainForest (Gehrke, Ramakrishnan, & Ganti, 2000), and

Record Id	Car	Age	Children	Subscription
1	sedan	23	0	yes
13	sports	23	2	no
4	truck	25	2	no
7	sedan	25	0	yes
11	sports	25	0	no
5	sports	30	0	no
9	sedan	30	2	yes
2	sports	31	1	no
10	sedan	31	1	yes
3	sedan	36	1	no
6	sedan	36	0	no
8	truck	36	1	no
12	sedan	45	1	yes
14	truck	45	0	yes

FIG. 1.4. Example training database sorted on age.

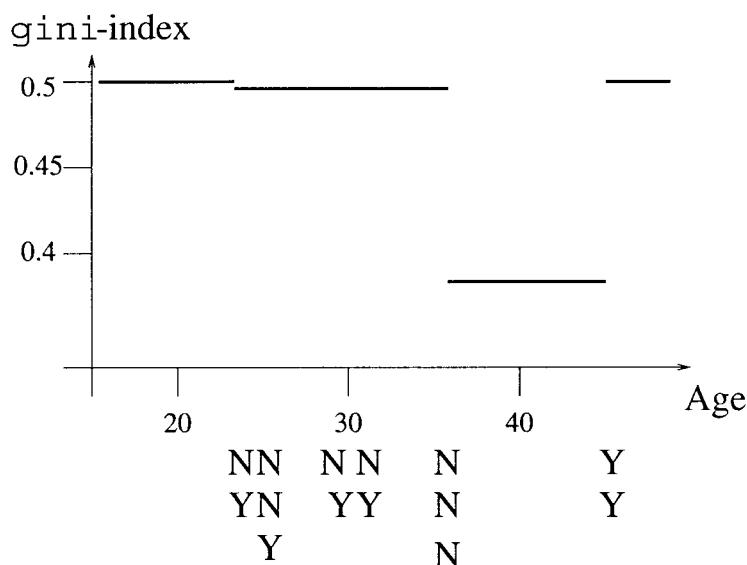


FIG. 1.5. Impurity function for age.

BOAT (Gehrke, Ganti, Ramakrishnan, & Loh, 1999). Other scalable classification tree construction algorithms include SLIQ (Mehta, Agrawal, & Rissanen, 1996) and CLOUDS (Alsabti, Ranka, & Singh, 1998).

Sprint

SPRINT works for very large data sets and removes all relationships between main memory and size of the training database. SPRINT scales any impurity-based split selection method such as the gini-index (Breiman et al., 1984). Recall that to decide on the splitting attribute at a node n , the algorithm requires access to F_n for each *numerical* attribute in sorted order. So conceptually, for each node n of the decision tree, a sort of F_n for each numerical attribute is required. The domains of categorical attributes are usually small enough to keep the required information for split selection in main memory.

SPRINT avoids sorting for the numerical attributes at each node through the creation of *attribute lists*. The attribute list L_X for attribute X is a vertical partition of the training database D : For each record $t \in D$ the entry of t into L_X consists of the projection of t onto X , the class label, and t 's record identifier. The attribute lists are created at the beginning of the algorithm, and the attribute lists for numerical attributes are sorted once as a preprocessing step.

The attribute lists created by SPRINT for the example training database from Fig. 1.1 are shown in Fig. 1.6.

Attr. List for Car			Attr. List for Age		
Car	Subser.	Rid	Age	Subser.	Rid
sedan	yes	1	23	yes	1
sports	no	2	23	no	13
sedan	no	3	25	no	4
truck	no	4	25	yes	7
sports	no	5	25	no	11
sedan	no	6	30	no	5
sedan	yes	7	30	yes	9
truck	no	8	31	no	2
sedan	yes	9	31	yes	10
sedan	yes	10	36	no	3
sports	no	11	36	no	6
sedan	yes	12	36	no	8
sports	no	13	45	yes	12
truck	yes	14	45	yes	14

FIG. 1.6. SPRINT attribute lists.

Attr. List for Car			Attr. List for Age		
Car	Subscr.	Rid	Age	Subscr.	Rid
sedan	yes	1	23	yes	1
truck	no	4	23	no	13
sports	no	5	25	no	4
sedan	yes	7	25	yes	7
sedan	yes	9	25	no	11
sports	no	11	30	no	5
sports	no	13	30	yes	9

FIG. 1.7. SPRINT attribute lists of the left child of the root node.

During the tree growth phase, whenever an internal node n splits, F_n is distributed among n 's children. Because every record is vertically partitioned over all attribute lists, each attribute list needs to be distributed across the children separately. The distribution of an attribute list is performed through a hash-join with the attribute list of the splitting attribute (Ramakrishnan & Gehrke, 1999). The record identifier, which is duplicated into each attribute list, establishes the connection between the vertical parts of the record. Because during the hash-join each attribute list is read and distributed sequentially, the initial sort order of the attribute list is preserved. In our example, assume that the root node splits as shown in Fig. 1.2, that is, the splitting attribute of the root node is age , the splitting predicate on the left outgoing edge is $age \leq 30$, and the splitting predicate on the right outgoing edge is $age > 30$. The attribute lists of the left child node of the root node are shown in Fig. 1.7.

RainForest

An examination of the split selection methods in the literature reveals that the greedy schema can be refined to the generic *RainForest Tree Induction Schema* shown in Fig. 1.8. A broad class of split selection methods, namely those that generate splitting criteria involving a single

RainForest refinement to the schema in Figure 1.3:

- (1a) for each predictor attribute X
- (1b) Construct the AVC-set of X
- (1c) Call $\mathcal{SS}.\text{find_best_partitioning}(\text{AVC-set of } X)$
- (1d) endfor
- (2a) $\mathcal{SS}.\text{decide_splitting_criterion}();$
- (2b) if n splits ...

FIG. 1.8. RainForest refinement.

Car	Subscription		Age	Subscription	
	Yes	No		Yes	No
sedan	5	2	23	1	1
sports	0	4	25	1	2
truck	1	2	30	1	1
			31	1	1
			36	0	3
			45	2	0

FIG. 1.9. RainForest AVC-sets.

splitting attribute, proceed according to this generic schema, and we do not know of any method in this class that does not adhere to it.¹ Consider a node n of the decision tree. The split selection method has to make two decisions while examining the family of n : (a) It has to select the splitting attribute X , and (b) it has to select the splitting predicates on X . Once the splitting criterion is determined, the algorithm is recursively applied to each of the children of n . Let us denote by \mathcal{SS} a representative split selection method.

Note that at a node n , the utility of a predictor attribute X as a possible splitting attribute is examined independent of the other predictor attributes: The *sufficient statistics* are the class label distributions for each distinct attribute value of X . We define the *AVC-set* of a predictor attribute X at node n to be the projection of F_n onto X and the class label where counts of the individual class labels are aggregated. (The acronym AVC stands for Attribute-Value, Classlabel.) We will denote the AVC-set of predictor attribute X at node n by $\text{AVC}_n(X)$. To give a formal definition, recall that we assumed without loss of generality that the domain of the class label is the set $\{1, \dots, J\}$ (see Problem Definition). Let $a_{n,X,x,i}$ be the number of records t in F_n with attribute value $t.X = x$ and class label $t.C = i$. Formally,

$$a_{n,X,x,i} \stackrel{\text{def}}{=} |\{t \in F_n : t.X = x \wedge t.C = i\}|.$$

For a predictor attribute X , let $S \stackrel{\text{def}}{=} \text{dom}(X) \times N^J$, where N denotes the set of natural numbers. Then

$$\begin{aligned} \text{AVC}_n(X) &\stackrel{\text{def}}{=} \{(x, a_1, \dots, a_J) \in S : \exists t \in F_n : \\ &(t.X = x \wedge \forall i \in \{1, \dots, J\} : a_i = a_{n,X,x,i})\}. \end{aligned}$$

We define the *AVC-group* of a node n to be the set of the AVC-sets of all predictor attributes at node n . Note that the size of the AVC-set of a predictor attribute X at node n depends only on the number of distinct attribute values of X and the number of class labels in F_n .

As an example, consider the training database shown in Fig. 1.1. The AVC group of the root node is depicted in Fig. 1.9. Assume that the root node splits as shown in Fig. 1.2. The AVC-group of the left child node of the root node is shown in Fig. 1.9 and the AVC-group of the left child node of the root node is shown in Fig. 1.10.

¹Split selection methods that generate linear combination splits cannot be captured by RainForest.

Car	Subscription	
	Yes	No
sedan	3	0
sports	0	3
truck	0	1

Age	Subscription	
	Yes	No
23	1	1
25	1	2
30	1	1

FIG. 1.10. RainForest AVC-sets of the left child of the root node.

If the training database is stored inside a database system, the AVC-set of a node n for predictor attribute X can be retrieved through a simple SQL query:

```
SELECT      D.X,  D.C,  COUNT(*)
FROM        D
WHERE       fn
GROUP BY   D.X,  D.C
```

To construct the AVC-sets of all predictor attributes at a node n , a UNION query would be necessary. (In this case, the SELECT clause also needs to retrieve some identifier of the attribute to distinguish individual AVC-sets.) Graefe, Fayyad, & Chaudhuri (1998) observe that most database systems evaluate the UNION-query through several scans and introduce a new operator that allows gathering of sufficient statistics in one database scan.

Given this observation, we can devise several algorithms that construct as many AVC-sets as possible in main memory while minimizing the number of scans over the training database. As an example of the simplest such algorithm, assume that the complete AVC-group of the root node fits into main memory. Then we can construct the tree according to the following simple schema: Read the training database D and construct the AVC-group of the root node n in memory. Then determine the splitting criterion from the AVC-sets through an in-memory computation. Then make a second pass over D and partition D into children partitions D_1 and D_2 . This simple algorithm reads the complete training database twice and writes the training database once per level of the tree; more sophisticated algorithms are possible (Gehrke et al., 2000). Experiments show that RainForest outperforms SPRINT on the average by a factor of three. Note that RainForest has a large minimum memory requirement: RainForest is only applicable if the AVC-group of the root node fits in memory. (This requirement can be relaxed through more sophisticated memory management [Gehrke et al., 2000].)

Boat

In this section we introduce BOAT, another scalable algorithm for decision tree construction. Like the RainForest framework, BOAT is applicable to a wide range of split selection methods. BOAT is the only decision tree construction algorithm that constructs several levels of the tree in a single scan over the database; all other algorithms make one scan over the database for each level of the tree. We give only an informal overview here; details can be found in the full article (Gehrke et al., 1999). We assume in the remainder of this section that the split selection method \mathcal{SS} is impurity based and produces trees with binary splits.

Let T be the optimal tree constructed using split selection method \mathcal{SS} on training database D . The complete training database D does not fit in memory, so we obtain a large sample D'

from D such that D' fits in memory. We can now use a traditional main-memory decision tree construction algorithm to compute a *sample* tree T' from D' . Each node $n \in T'$ has a *sample splitting criterion* consisting of a *sample splitting attribute* and a *sample split point* (or a *sample splitting subset*, for categorical attributes). Intuitively, T' will be quite “similar” to the final tree T constructed from the complete training database D because T' was constructed using a large, representative sample $D' \subset D$. But how similar is T' to T ? Can we use the knowledge about T' to help or guide us in the construction of T , our final goal? Ideally, we would like to say that the final splitting criteria of T are very “close” to the sample splitting criteria of T' . But without any qualification and quantification of “similarity” or “closeness,” information about T' is useless in the construction of T .

Consider a node n in the sample tree T' with numerical sample splitting attribute X_n and sample splitting predicate $X_n \leq x$. By T' being close to T we mean that the final splitting attribute at node n is X and that the final split point is inside a confidence interval around x . Our goal is to calculate the confidence interval using only the in-memory sample D' , so that during construction of T , we can restrict our search for the final split point to the attribute values inside the confidence interval around x . If the splitting attribute at node n in the tree is categorical, we say that T' is close to T if the sample splitting criterion and the final splitting criterion are identical. Thus, the notion of closeness for categorical attributes is more stringent because splitting attribute as well as splitting subset have to agree.

We can use bootstrapping (Efron & Tibshirani, 1993) to the in-memory sample D' to construct many different trees to obtain confidence intervals that contain the final split points for nodes with numerical splitting attributes and the complete final splitting criterion for nodes with categorical splitting attributes. We call the information at a node n obtained through bootstrapping the *coarse splitting criterion* at node n . Through computation of the coarse splitting criterion, we restrict our search when building T to trees that are close to T' . We call this first part of the algorithm the *sampling phase*.

Let us assume for the moment that the information obtained through bootstrapping is always correct. Using the sample $D' \subset D$ and the bootstrapping method, the coarse splitting criteria for all nodes n of the tree have been calculated and are correct. Now the search space of possible splitting criteria at each node of the tree is greatly reduced. We know the splitting attribute at each node n of the tree; for numerical splitting attributes we know a confidence interval of attribute values that contains the final split point; for categorical splitting attributes we know the final splitting subset exactly. Consider a node n of the tree with numerical splitting attribute. To decide on the final split point, we need to examine the value of the impurity function only at the attribute values inside the confidence interval. If we had all the tuples that fall inside the confidence interval of n in memory, then we could calculate the final split point exactly by calculating the value of the impurity function at all possible split points inside the confidence interval. To bring these tuples in memory, we make one scan over D and keep all tuples that fall inside a confidence interval at any node in memory. Then we postprocess each node with a numerical splitting attribute to find the exact value of the split point using the tuples collected during the database scan.

The coarse splitting criterion at a node n obtained from the sample D' through bootstrapping is correct only with high probability. This means that occasionally the final splitting attribute is different from the sample splitting attribute. In addition, even if the sample splitting attribute is equal to the final splitting attribute, it could happen that the final split point is outside the confidence interval, or that the final splitting subset is different from the sample splitting subset. Because we want to guarantee that our method generates exactly the same tree as if the complete training dataset were used, we have to be able to check whether the coarse splitting criteria actually are correct. We can prove a necessary condition that holds in two of the three

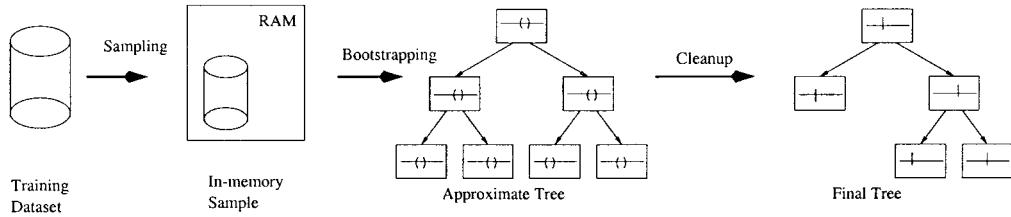


FIG. 1.11. Steps of the algorithm.

cases of incorrectness of the coarse splitting criterion: (a) the case that the splitting attribute is different from the sample splitting attribute, and (b) the case that the correct numerical splitting attribute was computed, but the split point is outside the confidence interval. The third condition, when the correct categorical splitting attribute was computed but the final splitting subset is different from the sample splitting subset, can be detected trivially during the cleanup phase. Thus, whenever the coarse splitting criterion at a node n is not correct, we will detect it during the cleanup phase and can take necessary measures. Therefore, we can guarantee that BOAT *always* finds *exactly the same tree* as if a traditional main-memory algorithm were run on the complete training dataset. The steps of the BOAT algorithm are illustrated in Fig. 1.11.

Tree Pruning

The pruning phase of classification tree construction decides on the tree of the right size in order to prevent overfitting and to minimize the misclassification error $R_T(P)$. In bottom-up pruning in the tree growth phase, the tree is grown until the size of the family of each leaf node n falls below a user-defined threshold c ; the pruning phase follows the growth phase. Examples of bottom-up pruning strategies are cost-complexity pruning, pruning with an additional set of records called a test set (Breiman et al., 1984), and pruning based on the minimum description length (MDL) principle (Mehta, Rissanen, & Agrawal, 1995; Quinlan & Rivest, 1989). In top-down pruning during the growth phase, a statistic s_n is computed at each node n , and based on the value of s_n , tree growth at node n is continued or stopped (Quinlan, 1993). Bottom-up pruning results usually in trees of higher quality (Breiman et al., 1984; Lim et al., 2000; Quinlan, 1993), but top-down pruning is computationally more efficient because no parts of the tree are first constructed and later discarded.

In this section we describe the PUBLIC pruning algorithm, an algorithm that integrates bottom-up pruning into the tree growth phase; thus, PUBLIC preserves the computational advantages of top-down pruning while preserving the good properties of top-down pruning (Rastogi & Shim, 1998). We first introduce MDL pruning, and then we discuss the PUBLIC algorithm.

MDL Pruning

Pruning based on the MDL principle (Rissanen, 1989) sees the classification tree as a means to encode the values of the class label attribute given the predictor attributes X_1, \dots, X_m . The MDL principle states that the “best” classification tree is the tree that can be encoded with the least number of bits. Thus, we first need to define an encoding schema that allows us to encode any binary decision tree. Given an encoding schema, we can then prune a given classification tree by selecting the subtree with minimum code length.

We will describe the MDL encoding schema for binary splitting predicates from Mehta et al. (1995). First, each node requires one bit to encode its type (leaf or intermediate node). For an intermediate node n , we need to encode its splitting criterion, consisting of the splitting

Input: Binary classification tree T rooted at node n
Output: Pruned tree T' rooted at n

MDLPrune(Node n)

- (1) **if** n is a leaf node **return** $C_{leaf}(F_n)$
- (2) $leftCost = \text{MDLPrune}(\text{left child node})$
- (3) $rightCost = \text{MDLPrune}(\text{right child node})$
- (4) **if** $(C_{leaf}(n) + 1 \leq C_{split}(n) + 1 + leftCost + rightCost)$
- (5) Prune child nodes
- (6) **return** $C_{leaf}(n) + 1$
- (6) **endif**
- (7) **return** $C_{split}(n) + 1 + leftCost + rightCost$

FIG. 1.12. MDL pruning schema.

attribute X ($\log m$ bits because there are m predictor attributes) and splitting predicate. Let X be the splitting attribute at node n and assume that X has v different attribute values. If X is a numerical attribute, the split will be of the form $X \leq c$. Because c can take $v - 1$ different values, encoding of the split point c requires $\log(v - 1)$ bits. If X is a categorical attribute, the split will be of the form $X \in Y$. Because Y can take $2^v - 2$ different values, the encoding requires $\log(2^v - 2)$ bits. We denote the cost of a split at a node n by $C_{Split}(n)$. For a leaf node n , we need to encode the class labels for the $|F_n|$ records. Metha et al. (1995) show that this cost can be computed using the following equation:

$$C_{leaf}(n) = \sum_i n_i \log \frac{|F_n|}{|F_n^i|} + \frac{k-1}{2} \log \frac{|F_n|}{2} + \log \frac{\pi^{k/2}}{\Gamma(k/2)}.$$

Given this encoding schema, a fully grown tree can be pruned bottom-up according to the recursive procedure shown in Fig. 1.12.

PUBLIC

The PUBLIC algorithm integrates the building and pruning phase by computing a lower bound $L(n)$ on the MDL cost of any subtree rooted at a node n . A trivial lower bound is $L(n) = 1$ (for the encoding of n). Rastogi & Shim (1998) give in their PUBLIC algorithms more sophisticated lower bounds including the following.

PUBLIC Lower Bound. Consider a (sub-)tree T with $s > 1$ nodes rooted at node n . Then the cost $C(T)$ of encoding T with the encoding schema described in the previous section

Input: Root r , node n , partition D ,
 split selection method \mathcal{SS}
Output: Decision tree for D rooted at node n

PUBLIC Integrated Construction and Pruning:

PUBLIC(Node r , node n , dataset D , split method \mathcal{SS})

- (1) Mark n inProgress
- (2) PUBLICPrune(r)
- (3) if node n not marked as pruned or leaf
- (4) Apply \mathcal{SS} to D to find the splitting criterion
- (5) if node n splits
- (6) Use best split to partition D into D_1 and D_2
- (7) PUBLIC($r, n_1, D_1, \mathcal{SS}$)
- (8) PUBLIC($r, n_2, D_2, \mathcal{SS}$)
- (9) endif
- (10) endif

FIG. 1.13. PUBLIC recursive pruning algorithm.

has the following lower bound:

$$C(n) \geq 2 \cdot (s - 1) + 1 + (s - 1) \cdot \log m + \sum_{i=s+1}^k |F_n^i|$$

With this lower bound, which we denote by $L(n)$, we can now refine the MDL pruning schema from Fig. 1.12 to the PUBLIC Pruning Schema shown in Fig. 1.13 and 1.14. PUBLIC distinguishes two different types of leaf nodes: “true” leaf nodes that are the result of pruning or that cannot be expanded any further, and “intermediate” leaf nodes n , where the subtree rooted at n might be grown further. During the growth phase the PUBLIC pruning schema is executed from the root node of the tree. Rastogi & Shim (1998) show experimentally that integration of pruning with the growth phase of the tree results in significant savings in overall tree construction. More sophisticated ways of integrating tree growth with pruning in addition to tighter lower bounds are possible.

Missing Values

It is often the case that real-life training databases have missing values; that is, in some records one or several predictor attributes have NULL values. (Note that during tree construction we cannot use a record from the training database that has a missing value for the class label, that

Input: Node n

PUBLICPrune(Node n)

- (1) **if** n is a leaf **return** $C(F_n)$ **endif**
- (2) **if** n marked as inProgress
- (3) **return** $L(n)$
- (4) **endif**
- (5) leftCost = PUBLICPrune(left child node)
- (6) rightCost = PUBLICPrune(right child node)
- (7) **if** $(C_{leaf}(n) + 1 \leq C_{split}(n) + 1 + \text{leftCost} + \text{rightCost})$
- (8) Recursively mark child nodes of n as pruned
- (9) Mark n as leaf node
- (10) **return** $C_{leaf}(n) + 1$
- (11)**endif**
- (12)**return** $C_{split}(n) + 1 + \text{leftCost} + \text{rightCost}$

FIG. 1.14. PUBLIC pruning step.

is, a record t with $t.C = \text{NULL}$.) Consider a record t with a missing value for predictor attribute X , i.e., $t.X = \text{NULL}$, and let n be the root node of the tree. Note that the attribute values $t.X'$ for predictor attributes $X' \neq X$ can be utilized for the computation of the splitting criterion $\text{crit}(n)$. This utilization is possible since for construction of the AVC-set of predictor attribute X' , the projection $\langle t.X', t.C \rangle$ of a record t suffices. Whether t has missing values for predictor attribute X does not influence t 's utility for the AVC-sets of predictor attributes $X' \neq X$. But after $\text{crit}(n)$ has been computed, F_n has to be partitioned among n 's children. Assume that X is the splitting attribute at node n . If for record t , the value of predictor attribute X is missing, it is not clear to which child of n the record t should be sent. Naively, t cannot continue to participate in further construction of the subtree rooted at n . Algorithms for missing values address this problem.

Friedman (1997) suggested that all records with missing attribute values should be ignored during tree construction whereas other authors argue that it is beneficial to make maximum use of all records (Breiman et al., 1984). In the remainder of this section we describe two methods, estimation (Loh & Shih, 1997) and surrogate splits (Breiman et al., 1984), and show how they can be incorporated into our framework. The goal of these methods is to maximize the use of the training database as well as to be able to classify future records with missing values.

Estimation

In *estimation*, missing values are estimated from the data whenever necessary (Loh & Shih, 1997). Assume that at node n we want to estimate the attribute value of predictor attribute X in record t with class label i , that is, $t.X = \text{NULL}$ and $t.C = i$. (Recall that the value of the

class label is known for all records in the training dataset.) One simple estimator of $t.X$ is the node class mean, if X is numerical, or the node class mode, if X is categorical. Formally, let $\hat{t}.X$ be the estimated value of $t.X$. Then $\hat{t}.X$ is defined as follows ($i \in \{1, \dots, J\}$):

$$\begin{aligned}\hat{t}.X &\stackrel{\text{def}}{=} \frac{\sum_{t \in F_n \wedge t.X \neq \text{NULL} \wedge t.C = i} t.X}{|\{t \in F_n : t.X \neq \text{NULL} \wedge t.C = i\}|}, \text{ if } X \text{ is numerical, and} \\ \hat{t}.X &\stackrel{\text{def}}{=} \operatorname{argmax}_{j \in \text{dom}(X)} |\{t \in F_n \wedge t.X = j \wedge t.C = i\}|, \text{ if } X \text{ is categorical.}\end{aligned}$$

It can be easily seen that the AVC-set of predictor attribute X contains sufficient statistics to calculate $\hat{t}.X$.

Surrogate Splits

Another method for computing missing values, which is called *surrogate splits*, was introduced by Breiman et al. (1984) in CART. Let X be the splitting attribute at node n and denote by s_X the splitting criterion at node n . The algorithm selects a splitting criterion $s_{X'}$ for each predictor attribute $X' \neq X$ such that $s_{X'}$ is most similar to s_X in terms of sending records to the same child node. Assume that the best split s_X is on predictor attribute X ; s_X partitions F_n into $F_{n_1}, F_{n_2}, \dots, F_{n_k}$. Consider a split $s_{X'}$ on another predictor attribute X' ; $s_{X'}$ partitions F_n into $F'_{n'_1}, F'_{n'_2}, \dots, F'_{n'_k}$.² Then the probability $p(s_X, s_{X'})$ that $s_{X'}$ results in the same prediction as s_X is estimated by the proportion of records in F_n that $s_{X'}$ and s_X send to the same child. Formally,

$$p(s_X, s_{X'}) = \frac{|F_{n_1} \cap F'_{n'_1}|}{|F_n|} + \frac{|F_{n_2} \cap F'_{n'_2}|}{|F_n|} + \dots + \frac{|F_{n_k} \cap F'_{n'_k}|}{|F_n|}.$$

The best surrogate split for predictor attribute X' with respect to splitting criterion s_X is the split $s_{X'}$ that maximizes $p(s_X, s_{X'})$. (Only surrogate splits that are better than the naive predictor are used. The naive predictor sends a record to the child with the largest family.) Using $p(s_X, s_{X'})$, the predictor attributes $X' \neq X$ can be ranked according to the quality of their surrogate splits as quantified by $p(s_X, s_{X'})$. For a record t with $t.X = \text{NULL}$, the prediction of the highest ranked surrogate split $s_{X'}$ such that $t.X' \neq \text{NULL}$ is used to determine the child partition of t .

The best surrogate split for a predictor attribute X' is found through an exhaustive search over all possible candidate splits. To calculate the quality of a candidate surrogate split $s_{X'}$, we need the count of records on which the prediction of s_X and $s_{X'}$ agree. For any candidate split $s_{X'}$, $p(s_X, s_{X'})$ can be calculated from the following data structure: For each attribute value $x \in X'$, we store the count of records that split s_X sends to the i th child of n for $i \in \{1, \dots, k\}$. These sufficient statistics, which we call the *surrogate-count set*, are as compact as the AVC-set of predictor attribute X' : The size of the surrogate-count set depends only on the number of different attribute values of predictor attribute X' and the number of partitions induced by s_X . The surrogate-count set can be created in one scan over the family of n as follows: For each record $t \in F_n$ with $t.X \neq \text{NULL}$, we apply s_X . If record t is sent to child number i , the counter for attribute-value $t.X'$ and child number i in the surrogate-count set of predictor attribute X' is increased. Thus, surrogate splits can be easily incorporated into our framework. In case the training database is stored in a database system, the surrogate-count set for each predictor attribute $X' \neq X$ can be retrieved through a simple SQL query.

As an example, consider the training database shown in Fig. 1.1. Assume that the split s at the root node is on predictor attribute age and that the splitting predicates on the outgoing edges are $age \leq 30$ and $age > 30$, respectively, as in the example decision tree shown in Fig. 1.2.

²Breiman et al. only discuss surrogate splits for binary trees. We present here the (natural) generalization to k -ary trees for $k \geq 2$.

Let us call this split s . The family of tuples of the left child node of the root node consists of the records with record identifiers $\{1, 4, 5, 7, 9, 11, 13\}$. The family of tuples of the right child node of the root node consists of the records with record identifiers $\{2, 3, 6, 8, 10, 12, 14\}$. We show now how to calculate the best surrogate split. We first consider the predictor attribute *number of children*. The first possible split s' is $\text{number of children} \leq 0$. The family of tuples of the left child of the root node under split s' consists of the records with the record identifiers $\{1, 5, 6, 7, 11, 14\}$, the family of tuples of the right child of the root node consists of the remaining records, and we calculate $p(s, s') = \frac{8}{14}$. If s' is *number of children* ≤ 1 , it is easy to check that $p(s, s') = \frac{4}{14}$. Further calculations using the predictor attribute *car type* show that the best surrogate split s' actually involves the attribute *car type* with the predicate $\text{car type} \in \{\text{sports}\}$ on the left outgoing edge and the predicate $\text{car type} \in \{\text{sedan}, \text{truck}\}$ on the right outgoing edge. Here $p(s, s') = \frac{9}{14}$. Thus, a record whose value for the predictor attribute *age* is missing uses the predictor attribute *car type* to decide which branch to take.

A SHORT INTRODUCTION TO REGRESSION TREES

In this section we introduce the problem of regression tree construction according to the definition of CART regression trees given by Breiman et al. (1984). We first introduce the problem formally and then discuss split selection and data access.

Problem Definition

Recall that in a regression problem the dependent variable Y is numerical. A *prediction rule* is defined analogous to a classifier. It is a function:

$$d : \text{dom}(X_1) \times \cdots \times \text{dom}(X_m) \mapsto \text{dom}(Y).$$

CART regression trees fit a constant regression model at each leaf node of the tree. We define the *mean squared error* $R_d^{\text{Reg}}(P)$ of prediction rule d to be the expected square error using $d(\langle t.X_1, \dots, t.X_m \rangle)$ as the predictor of $t.Y$, formally

$$R_d^{\text{Reg}}(P) \stackrel{\text{def}}{=} E(t.Y - d(\langle t.X_1, \dots, t.X_m \rangle))^2.$$

A *regression tree* is defined analogous to a decision tree. (The predicted value at each leaf node n is a constant $c_n \in \text{dom}(Y)$.) The *regression tree construction problem* can be formally stated as follows.

Regression Tree Construction Problem. Given a dataset $D = \{t_1, \dots, t_N\}$ where the t_i are independent random samples from a probability distribution P , find a regression tree T such that the mean squared error $R_T^{\text{Reg}}(P)$ is minimal (Breiman et al., 1984).

Split Selection

Let \bar{Y}_n be the mean of the dependent variable in node n and R_n be the sum of squares of Y within node n if \bar{Y}_n is used as predictor of Y . Formally,

$$\bar{Y}_n \stackrel{\text{def}}{=} \frac{\sum_{t \in F_n} t.Y}{|F_n|} \quad \text{and} \quad R_n \stackrel{\text{def}}{=} \sum_{t \in F_n} (t.Y - \bar{Y}_n)^2.$$

At node n , let s be a splitting criterion that partitions F_n into F_{n_1}, \dots, F_{n_k} . Then the quality $q(s)$ of splitting criterion s is measured in CART by the weighted decrease in variability. (The function $q(s)$ plays the role of the impurity function discussed under the subhead Split Selection.) Formally,

$$q(s) \stackrel{\text{def}}{=} R_n - \sum_{i=1}^k \frac{|F_{n_i}|}{|F_n|} R_{n_i}.$$

Given this quality assessment of a split, the CART regression tree split selection method proceeds exactly like the corresponding CART split selection method for classification trees. At a node n , for a numerical predictor attribute X , all splits of the form $X \leq x$ with $x \in \text{dom}(X)$ are evaluated. (In practice, we chose x as the midpoint of two consecutive attribute values occurring in the training database.) For a categorical predictor attribute X , all splits of the form $X \subseteq Y$ with $Y \subseteq \text{dom}(X)$ are evaluated. The splitting criterion that results in the highest decrease in variability is chosen greedily. How to prune the constructed regression tree is an orthogonal issue as in classification tree construction.

Data Access

Recall from preceding material that the splitting criterion $\text{crit}(n)$ of the CART piecewise constant regression trees involves exactly one predictor attribute. We will show that the sufficient statistics to calculate the quality of a candidate split s on predictor attribute X contain the following information: For each attribute value $x \in \text{dom}(X)$ we store the number of tuples t in F_n with $t.X = x$, the sum of the values of the dependent attribute and the sum of squares of the dependent attribute. Formally, let $F_n(X = x) \stackrel{\text{def}}{=} \{t \in F_n : t.X = x\}$. Thus, for all $x \in \text{dom}(X)$ such that $F_n(X = x) \neq \emptyset$, we store the following information:

$$\vec{x}_n \stackrel{\text{def}}{=} \left[x, |F_n(X = x)|, \sum_{t \in F_n(X = x)} t.Y, \sum_{t \in F_n(X = x)} (t.Y)^2 \right]$$

We define the *AVS-set* of a predictor attribute X at node n as the collection of the values \vec{x}_n . (The acronym AVS stands for attribute-value, squares). Recall that the central measure for the calculation of the quality $q(s)$ of splitting criterion s was the sum of squares of Y within node n :

$$q(s) \stackrel{\text{def}}{=} R_n - \sum_{i=1}^k \frac{|F_{n_i}|}{|F_n|} R_{n_i},$$

where R_n was defined as follows:

$$R_n \stackrel{\text{def}}{=} \sum_{t \in F_n} (t.Y - \bar{Y}_n)^2.$$

But we can rewrite R_n as

$$R_n = \sum_{t \in F_n} (t.Y)^2 - |F_n| \cdot \bar{Y}_n^2.$$

Now it is easy to see from the preceding equation that for any candidate splitting criterion s involving predictor attribute X , its quality $q(s)$ can be calculated from the statistics in the AVS-set. Thus, by replacing the AVC-set with the AVS-set, the framework results in a scalable algorithm of the CART regression tree split selection method. In case the training database is stored in a database system, the AVS-set for each predictor attribute X can be retrieved

through a simple SQL query analogous to the query described earlier under the subhead RainForest.

APPLICATIONS AND AVAILABLE SOFTWARE

Decision trees are one of the most popular data mining models, and they have been applied to a plethora of applications. We describe one application here in some detail; more pointers to applications can be found in the excellent survey by Murthy (1998).

Cataloging Sky Objects

In the Second Palomar Observatory Sky Survey, astronomers collected 3 terabytes of image data over 6 years, containing data about 2 billion sky objects. The goal of the project was to determine the class of each object, star versus galaxy. The data was first preprocessed, and individual sky objects were isolated using image segmentation techniques. The astronomers then defined a set of 40 different predictor attributes and created a training database of sky objects with their classification using visual inspection by experts. Decision tree construction was used to train a classifier that could predict the class of an object for unclassified sky objects. The resulting classifier had over 94% accuracy on unseen data and resulted in a threefold increase in the number of available classified sky objects. This data was later used by astronomers to discover 16 new red-shift quasars in about 10% of the usual time to find such objects.

Decision Trees in Today's Data Mining Tools

At the writing of this chapter (spring 2002), all major data mining vendors provide decision tree algorithms in their data mining suites. An up-to-date list of tools can be found at: <http://www.kdnuggets.com>. Sample programs include the following:

- SPSS AnswerTree. Commercial decision tree tool from SPSS (SPS, 2002).
- SAS Decision Trees. Available as part of the SAS Enterprise Miner Suite (SAS, 2002).
- IBM Intelligent Miner (IBM, 2002). Implements the SPRINT Algorithm discussed in this chapter.
- CART. Commercial version of one of the first decision tree programs. Available through Salford Systems (Sal, 2002).
- QUEST. Freeware classification tree construction program (Loh & Shih, 1997; Shih, 2002).
- C4.5. Both a free version (Quinlan, 2002) and a commercial version (Rul, 2002) are available.

SUMMARY

We have surveyed decision tree construction through a discussion of several recent results from the data mining literature. Decision tree construction is still a very active research area with many open problems, such as bias in split selection, scalability, scalable regression tree construction, and construction of decision trees for structured data such as sequences or tree-structured data (Murthy, 1998).

REFERENCES

- Agresti, A. (1990). *Categorical Data Analysis*. New York: Wiley.
- Alsabti, K., Ranka, S., & Singh, V. (1998). Clouds: A decision tree classifier for large datasets, In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)* (pp. 2–8). Menlo Park, CA: AAAI Press.
- Bishop, C. (1995). *Neural networks for pattern recognition*. New York: Oxford University Press.
- Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Boca Raton, FL: CRC Press.
- Cheeseman, P., & Stutz, J. (1996). Bayesian classification (autoClass): Theory and results, In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy, (Eds.) *Advances in knowledge discovery and data mining* (pp. 153–180). Boston: AAAI/MIT Press.
- Christensen, R. (1997). *Log-linear models and logistic regression*, (2nd ed.). New York: Springer Verlag.
- Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. London: Chapman & Hall.
- Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (Eds.) (1996). *Advances in knowledge discovery and data mining*. Boston: AAAI/MIT Press.
- Friedman, J. H. (1977). A recursive partitioning decision rule for nonparametric classifiers. *IEEE Transactions on Computers*, 26, 404–408.
- Gehrke, J., Ganti, V., Ramakrishnan, R., & Loh, W.-Y. (1999). BOAT—optimistic decision tree construction. In A. Delis, C. Faloutsos, & S. Ghandeharizadeh (Eds.) *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data* (pp. 169–180). New York: ACM Press.
- Gehrke, J., Ramakrishnan, R., & Ganti, V. (2000). Rainforest—a framework for fast decision tree construction of large datasets. *Data Mining and Knowledge Discovery*, 4(2/3), 127–162.
- Goebel, M., & Gruenwald, L. (1999). A survey of data mining software tools. *SIGKDD Explorations*, 1(1), 20–33.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. San Francisco: Morgan Kaufmann.
- Graefe, G., Fayyad, U., & Chaudhuri, S. (1998). On the efficient gathering of sufficient statistics for classification from large SQL databases. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 204–208). Menlo Park, CA: AAAI Press.
- Hand, D. (1997). *Construction and assessment of classification rules*. Chichester, UK: John Wiley & Sons.
- IBM (2002). IBM intelligent miner [online]. Retrieved from <http://www.ibm.com/software/data/iminer/>
- James, M. (1985). *Classification algorithms*. New York: Wiley.
- Kohavi, R. (1995). The power of decision tables. In N. Lavrac & S. Wrobel, (Eds.), *Proceedings of the Eighth European Conference on Machine Learning (ECML)* (Vol. 912 of *Lecture Notes in Computer Science*). Berlin: Springer-Verlag.
- Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 48, 203–228.
- Loh, W.-Y., & Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, 7(4), 815–840.
- Mehta, M., Agrawal, R., & Rissanan, J. (1996). SLIQ: A fast scalable classifier for data mining. In *Proceedings of the Fifth International Conference on Extending Database Technology (EDBT)*. (pp. 18–32). Berlin: Springer-Verlag.
- Mehta, M., Rissanan, J., & Agrawal, R. (1995). MDL-based decision tree pruning. In *Proceedings of the First International Conference on Knowledge Discovery in Databases and Data Mining*.
- Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification*. London: Ellis Horwood.
- Morimoto, Y., Fukuda, T., Matsuzawa, H., Tokuyama, T., & Yoda, K. (1998). Algorithms for mining association rules for binary segmentations of huge categorical databases. In *Proceedings of the 24th International Conference on Very Large Databases (VLDB)*. (pp. 380–391). San Francisco: Morgan Kaufmann.
- Murthy, S. K. (1998). Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4), 345–389.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco: Morgan Kaufmann.
- Quinlan, J. R., & Rivest, R. L. (1989). Inferring decision trees using minimum description length principle. *Information and Computation*, 80, 227–248.
- Quinlan, R. (2002). C4.5 release 8. Retrieved from <http://www.cse.unsw.edu.au/quinlan/>
- Ramakrishnan, R., & Gehrke, J. (1999). *Database Management Systems* (2nd ed.). New York: McGrawHill.

- Rastogi, R., & Shim, K. (1998). Public: A decision tree classifier that integrates building and pruning. In A. Gupta, O. Shmueli, & J. Wisdom (Eds.) *Proceedings of the 24th International Conference on Very Large Data Bases*. (pp. 404–415). San Francisco: Morgan Kaufmann.
- Ripley, B. (1996). *Pattern recognition and neural networks*. Cambridge, UK: Cambridge University Press.
- Rissanen, J. (1989). *Stochastic Complexity in Statistical Inquiry*. Singapore: World Scientific Publishing Co.
- Rui. (2002). *C5.0* [online]. Retrieved from <http://www.rulequest.com>
- Sal. (2002). *Cart decision tree software* [online]. Retrieved from <http://www.salford-systems.com>
- SAS. (2002). *Sas enterprise miner* [online]. Retrieved from <http://www.sas.com/products/miner>
- Shafer, J., Agrawal, R., & Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. In *Proceeding of the 22nd International Conference on Very Large Databases*. San Francisco: Morgan Kaufmann.
- Shih, Y.-S. (2002). *Quest user manual* [online]. Retrieved from <http://www.cs.wisc.edu/loh/quest.html>
- SPS. (2002). *Spss answertree* [online]. Retrieved from <http://www.spss.com/answertree>
- Weiss, S. M., & Kulikowski, C. A. (1991). *Computer systems that learn: Classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. San Francisco: Morgan Kaufmann.

2

Association Rules

Geoffrey I. Webb
Monash University

Introduction	26
Market Basket Analysis	26
Association Rule Discovery	27
The Apriori Algorithm	28
The Power of the Frequent Item Set Strategy	29
Measures of Interestingness	31
Lift	31
Leverage	32
Item Set Discovery	32
Techniques for Frequent Item Set Discovery	33
Closed Item Set Strategies	33
Long Item Sets	35
Sampling	35
Techniques for Discovering Association Rules without Item Set Discovery	35
Associations with Numeric Values	36
Applications of Association Rule Discovery	36
Summary	37
References	38

INTRODUCTION

Association rule discovery shares many features of classification rule induction (discussed in chap. 1). Both use rules to characterize regularities within a set of data. However, these two rule discovery paradigms differ substantially in intent. Whereas classification rule induction focuses on acquiring a capacity to make predictions, association rule discovery focuses on providing insight to the user. Specifically, it focuses on detecting and characterizing unexpected interrelationships between data elements.

These different foci have led to substantial differences in methods and techniques within the two paradigms. Classification rule induction typically uses heuristic search to find a small number of rules that jointly cover the majority of the training data. Association rule discovery typically uses complete search to find large numbers of rules with no regard for covering all training data.

Because of the focus on discovering small rule sets, classification rule induction systems often make choices between alternative rules with similar performance. Although the machine learning system may have no basis on which to distinguish between such rules, their value to the user may differ substantially due to factors extraneous to those represented in the data. Examples of such factors include the practicality of formulating business rules based on the discovered rule and the degree to which the discovered rule integrates with the user's wider understanding of the domain. In contrast, association rule discovery systems return all rules that satisfy user specified constraints, allowing the user to identify which specific rules have the greatest value.

To understand these differences between classification rule induction and association rule discovery it is valuable to start with an examination of the initial application for which association rule discovery was developed: market basket analysis. This background is covered in following paragraphs. The section headed Association Rule Discovery provides a description of the association rule discovery task, introducing the terminology that will be required through the rest of the chapter and the Apriori algorithm on which most association rule discovery algorithms are based. For some data mining applications the objective is to discover which data elements interact within the data. In such a context the formation of a rule that distinguishes some elements as belonging to an antecedent and some as belonging to a consequent may be inappropriate. The fourth section, Measures of Interestingness, explores metrics for identifying the relative level of interest of an association. Item Set Discovery examines the discovery of associations without developing rules. A section headed Techniques for Frequent Item Set Discovery describes extensions to the Apriori algorithm and is followed by a section discussing rule-search approaches to association rule discovery. The next two sections examine techniques for discovering associations with numeric data and discuss key applications for association rule discovery. The final section is a summary.

MARKET BASKET ANALYSIS

Modern cash registers are valuable data capture tools. Stores typically capture and store all of the information contained on the docket with which the customer is presented. This includes the “basket” of goods that were bought in the transaction. *Market basket analysis* is the analysis of this *basket data* to identify combinations of items with affinities to one another. That is, market basket analysis seeks to identify combinations of items, the presence of which in a transaction affects the likelihood of the presence of another specific item or combination of items.

TABLE 2.1
Hypothetical Market Basket Data

plums, lettuce, tomatoes
celery, confectionery
confectionery
apples, carrots, tomatoes, potatoes, confectionery
apples, oranges, lettuce, tomatoes, confectionery
peaches, oranges, celery, potatoes
beans, lettuce, tomatoes
oranges, lettuce, carrots, tomatoes, confectionery
apples, bananas, plums, carrots, tomatoes, onions, confectionery
apples, potatoes

The apocryphal example is a chain of convenience stores that performs a market basket analysis and discovers that disposable diapers and beer are frequently bought together. This unexpected knowledge is potentially valuable as it provides insight into the purchasing behavior of both disposable diaper and beer purchasing customers for the convenience store chain.

Table 2.1 provides a simple example of hypothetical market basket data. Each line lists a combination of products bought in a single transaction. Although real applications often examine many millions of transactions with 10 to hundreds of thousands of different products, this example is restricted to 10 transactions involving 14 products. One would not normally generalize from such a small sample. We use the small sample only to illustrate the principles of association rules discovery.

Visual inspection of the example data might reveal the regularity that all four transactions involving lettuce also involved tomatoes and that four of the six transactions that involved tomatoes also involved lettuce. Lettuce and tomatoes appear to go together. Association rule discovery seeks to identify and characterize such affinities.

ASSOCIATION RULE DISCOVERY

Association rule discovery finds relationships or affinities between *item sets*. An item set is a set of items. Each transaction is an item set. Example item sets from Table 2.1 include {plums, lettuce, tomatoes} and {lettuce, tomatoes}. Any set of items is an item set, even combinations that do not occur in the data, such as {plums, celery}.

An association rule is composed of two item sets called an *antecedent* and a *consequent*. The consequent often is restricted to containing a single item. The rules typically are displayed with an arrow leading from the antecedent to the consequent, for example {tomatoes} → {lettuce}. The antecedent and consequent frequently are called, respectively, the left-hand side (or LHS) and the right-hand side (or RHS). An association rule indicates an affinity between the antecedent item set and the consequent item set. An association rule is accompanied by frequency-based statistics that describe that relationship.

The two statistics that were used initially to describe these relationships were *support* and *confidence* (Agrawal, Imielinski, & Swami, 1993). These are numeric values. To describe them we need to define some numeric terms. Let D be the database of transactions and N be the number of transactions in D . Each transaction D_i is an item set. Let $support(X)$ be the

proportion of transactions that contain item set X :

$$\text{support}(X) = |\{I | I \in D \wedge I \supseteq X\}|/N$$

where I is an item set and $|\cdot|$ denotes the cardinality of a set.

The support of an association rule is the proportion of transactions that contain both the antecedent and the consequent. The confidence of an association rule is the proportion of transactions containing the antecedent that also contain the consequent. For an association $A \rightarrow C$,

$$\text{support}(A \rightarrow C) = \text{support}(A \cup C)$$

$$\text{confidence}(A \rightarrow C) = \text{support}(A \cup C)/\text{support}(A).$$

If support is sufficiently high (and the transactions represent a random sample from the same data distribution as future transactions), then confidence is a reasonable estimate of the probability that any given future transaction that contains the antecedent will also contain the consequent.

Association rule discovery may seek all association rules that exceed minimum bounds (specified by the user) on support and confidence. A naïve algorithm for performing association rule discovery would simply consider every possible combination of antecedent and consequent, evaluate the support and confidence, and discard all associations that did not satisfy the specified constraints. However, the number of possible combinations is typically very large. If there are 1,000 items, then the number of item sets that can be formed from the various combinations of those items will be $2^{1,000}$ (that is approximately 10^{300}). This is the number of antecedents. For each antecedent there is the potential to form a rule with any item set as a consequent that does not contain any member of the antecedent. It should be apparent that the search space of possible association rules is extremely large and that a naive approach will not deliver computationally tractable association rule discovery.

The Apriori Algorithm

The Apriori algorithm (Agrawal & Srikant, 1994) tackles this problem by reducing the number of item sets that are considered. The user specifies a minimum support, *min-support*. It follows from the definition of support for an association rule that $\text{support}(A \cup C) \leq \text{min-support}$ for any association rule $A \rightarrow C$ such that $\text{support}(A \rightarrow C) \leq \text{min-support}$. Apriori first generates all item sets that satisfy *min-support*. Such item sets are called *frequent* item sets. For each frequent item set X it then generates all association rules $A \rightarrow C$ such that $A \cup C = X$ and $A \cap C = \emptyset$. Any rules that do not satisfy further user specified constraints, such as minimum confidence, are discarded and the rest output. As $\text{support}(A) \geq \text{support}(A \rightarrow C)$ and $\text{support}(C) \geq \text{support}(A \rightarrow C)$, if $A \cup C$ is a frequent item set then both A and C must also be frequent item sets. Support and confidence as well as other metrics by which an association rule $A \rightarrow C$ is commonly evaluated can all be derived from the $\text{support}(A)$, $\text{support}(C)$, and $\text{support}(A \cup C)$. Hence, recording all frequent item sets and their support records all information required to generate and evaluate all association rules that satisfy *min-support*.

Table 2.2 presents the Apriori algorithm for generating frequent item sets. Line 3 generates new candidates. This step relies on a prior ordering of the items. An item set can be frequent only if all of its subsets are also frequent item sets. Rather than the computationally expensive exercise of testing that all subsets of a potential candidate are frequent, the candidate generation process generates candidates for which two subsets are both frequent, the two subsets formed

TABLE 2.2
The Apriori Algorithm for Generating Frequent Item Sets

```

1.    $L_1 = \{\text{frequent one-item-item sets}\}$ 
2.   for  $k = 2; L_{k-1} \neq \emptyset; k++$  do begin
3.        $C_k = \{(x_1, x_2, \dots, x_{k-2}, x_{k-1}, x_k) \mid \{x_1, x_2, \dots, x_{k-2}, x_{k-1}\} \in L_{k-1} \wedge$ 
            $\{x_1, x_2, \dots, x_{k-2}, x_k\} \in L_{k-1}\}$ 
4.       for all_transactions  $t \in D$  do begin
5.           for all candidates  $c \in C_k \wedge c \subseteq t$  do
6.                $c.\text{count}++;$ 
7.           end
8.            $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$ 
9.       end
10.      return  $\bigcup_k L_k;$ 

```

by removing one each of the two highest ranked elements of the new candidate. Very efficient procedures can be developed for generating such candidates from the frequent item sets of size $k - 1$ (Agrawal & Srikant, 1994).

The Power of the Frequent Item Set Strategy

In most market basket applications, each individual product appears in only a small proportion of transactions. Hence, the numbers of frequent item sets will be relatively low, even when *min-support* is set to a relatively small value. Thus, frequent item set approaches enable the search space to be reduced to a manageable size for the sparse data utilized in market basket analysis.

To illustrate the principles involved, Tables 2.3 and 2.4 present the frequent item sets and association rules that result from the application of this method to the example data. The first step is to select values for *min-support* and *min-confidence*. I use *min-support* = 0.25 and *min-confidence* = 0.5. Frequent item set generation results in the 15 frequent item sets listed in Table 2.3. From these, 54 association rules with confidence greater or equal to 0.5 can be generated. These are listed in Table 2.4. The antecedent of each rule is listed before the arrow. The consequent is listed after the arrow. The consequent is followed by the support and

TABLE 2.3
Frequent Item Sets

```

apples [Support=0.4]
carrots [Support=0.3]
confectionery [Support=0.6]
lettuce [Support=0.4]
oranges [Support=0.3]
potatoes [Support=0.3]
tomatoes [Support=0.6]
apples, confectionery [Support=0.3]
apples, tomatoes [Support=0.3]
carrots, confectionery [Support=0.3]
carrots, tomatoes [Support=0.3]
confectionery, tomatoes [Support=0.4]
lettuce, tomatoes [Support=0.4]
apples, confectionery, tomatoes [Support=0.3]
carrots, confectionery, tomatoes [Support=0.3]

```

TABLE 2.4
Association Rules

→ apples, confectionery, tomatoes [Support=0.3, Confidence=0.30]
→ apples, confectionery [Support=0.3, Confidence=0.30]
→ apples, tomatoes [Support=0.3, Confidence=0.30]
→ apples [Support=0.4, Confidence=0.40]
→ carrots, confectionery, tomatoes [Support=0.3, Confidence=0.30]
→ carrots, confectionery [Support=0.3, Confidence=0.30]
→ carrots, tomatoes [Support=0.3, Confidence=0.30]
→ carrots [Support=0.3, Confidence=0.30]
→ confectionery, tomatoes [Support=0.4, Confidence=0.40]
→ confectionery [Support=0.6, Confidence=0.60]
→ lettuce, tomatoes [Support=0.4, Confidence=0.40]
→ lettuce [Support=0.4, Confidence=0.40]
→ oranges [Support=0.3, Confidence=0.30]
→ potatoes [Support=0.3, Confidence=0.30]
→ tomatoes [Support=0.6, Confidence=0.60]
apples → [Support=0.4, Confidence=1.00]
apples → confectionery [Support=0.3, Confidence=0.75]
apples → confectionery, tomatoes [Support=0.3, Confidence=0.75]
apples → tomatoes [Support=0.3, Confidence=0.75]
apples, confectionery → [Support=0.3, Confidence=1.00]
apples, confectionery, tomatoes → [Support=0.3, Confidence=1.00]
apples, tomatoes → [Support=0.3, Confidence=1.00]
apples, confectionery → tomatoes [Support=0.3, Confidence=1.00]
apples, tomatoes → confectionery [Support=0.3, Confidence=1.00]
carrots → [Support=0.3, Confidence=1.00]
carrots → confectionery [Support=0.3, Confidence=1.00]
carrots → tomatoes, confectionery [Support=0.3, Confidence=1.00]
carrots → tomatoes [Support=0.3, Confidence=1.00]
carrots, confectionery, tomatoes → [Support=0.3, Confidence=1.00]
carrots, confectionery → [Support=0.3, Confidence=1.00]
carrots, tomatoes → [Support=0.3, Confidence=1.00]
carrots, confectionery → tomatoes [Support=0.3, Confidence=1.00]
confectionery → [Support=0.6, Confidence=1.00]
confectionery → apples, tomatoes [Support=0.3, Confidence=0.50]
confectionery → apples [Support=0.3, Confidence=0.50]
confectionery → carrots [Support=0.3, Confidence=0.50]
confectionery → tomatoes, carrots [Support=0.3, Confidence=0.50]
confectionery → tomatoes [Support=0.4, Confidence=0.67]
confectionery, tomatoes → [Support=0.4, Confidence=1.00]
lettuce → [Support=0.4, Confidence=1.00]
lettuce → tomatoes [Support=0.4, Confidence=1.00]
lettuce, tomatoes → [Support=0.4, Confidence=1.00]
oranges → [Support=0.3, Confidence=1.00]
potatoes → [Support=0.3, Confidence=1.00]
tomatoes → [Support=0.6, Confidence=1.00]
tomatoes → apples, confectionery [Support=0.3, Confidence=0.50]
tomatoes → apples [Support=0.3, Confidence=0.50]
tomatoes → carrots, confectionery [Support=0.3, Confidence=0.50]
tomatoes → carrots [Support=0.3, Confidence=0.50]
tomatoes → confectionery [Support=0.4, Confidence=0.67]
tomatoes → lettuce [Support=0.4, Confidence=0.67]
tomatoes, confectionery → apples [Support=0.3, Confidence=0.75]
tomatoes, confectionery → carrots [Support=0.3, Confidence=0.75]
tomatoes, carrots → confectionery [Support=0.3, Confidence=1.00]

confidence in brackets. Note that the 15 associations with an empty consequent provide very little information and many association rule discovery systems do not generate them.

MEASURES OF INTERESTINGNESS

Initial association rule discovery techniques found all associations that satisfied user specified constraints on support and confidence. This often resulted in the identification of many thousands of association rules. It is not possible to manually assess such large numbers of rules. Clearly, it is desirable to reduce the numbers of rules so that only the most interesting rules are identified. Often, the interestingness of a rule will relate to the difference between the support of the rule and the product of the support for the antecedent and the support for the consequent. If the antecedent and consequent are independent of one another, then the support for the rules should approximately equal the product of the support for the antecedent and the support for the consequent. If the antecedent and consequent are independent, then the rule is unlikely to be of interest no matter how high the confidence (Piatetsky-Shapiro, 1991).

For example, a retail data mining project might seek to understand the relationship between purchases of lingerie and other products. A simple association analysis (treating as a single basket all of a customer's purchases over a 12-month period) finds, with extraordinary high support and confidence, that lingerie is associated with confectionery. This initially causes some excitement, as it was not expected and is felt to be an affinity that could be used for marketing purposes. However, subsequent analyses find that all major product categories are associated with confectionery with high confidence. In short, most customers buy confectionery over a 12-month period. An association with confectionery as the consequent with confidence of 0.87 turns out to be of little interest because 87% of all customers purchase confectionery. To be of interest, an association with confectionery as the consequent would have to have greater confidence than 0.87.

Such realizations have led to the development of numerous alternative measures of interestingness (Brin, Motwani, Ullman, & Tsur, 1997; Freitas, 1999; Klemettinen, Mannila, Ronkainen, Toivonen, & Verkamo, 1999; Kamber & Shinghal, 1995; Piatetsky-Shapiro, 1991; Sahar, 1999). It is useful to distinguish between objective and subjective measures of interestingness (Liu, Hsu, Chen, & Ma, 2000; Silberschatz & Tuzhilin, 1996). Objective measures include measures such as support and confidence that can be applied independently of a particular application. Subjective measures relate the interestingness of a rule to the specific information needs of a user in a specific context.

Lift

The most popular objective measure of interestingness is *lift*.

$$\text{Lift}(A \rightarrow C) = \text{confidence}(A \rightarrow C) / \text{support}(C).$$

This is the ratio of the frequency of the consequent in the transactions that contain the antecedent over the frequency of the consequent in the data as a whole. Lift values greater than 1 indicate that the consequent is more frequent in transactions containing the antecedent than in transactions that do not.

For example, consider the association $\{\text{tomatoes}\} \rightarrow \{\text{lettuce}\}$. $\text{Support}(\{\text{lettuce}\}) = 0.4$. $\text{Confidence}(\{\text{tomatoes}\} \rightarrow \{\text{lettuce}\}) = 0.67$. Hence, $\text{lift}(\{\text{tomatoes}\} \rightarrow \{\text{lettuce}\}) = 0.67/0.4 = 1.675$. As a contrast, consider another association with the same confidence,

$\{\text{tomatoes}\} \rightarrow \{\text{confectionery}\}$. $\text{Support}(\{\text{confectionery}\}) = 0.6$. $\text{Confidence}(\{\text{tomatoes}\} \rightarrow \{\text{confectionery}\}) = 0.67$. Hence, $\text{lift}(\{\text{tomatoes}\} \rightarrow \{\text{confectionery}\}) = 0.67/0.6 = 1.117$. These relative values of lift indicate that tomatoes have a greater impact on the frequency of lettuce than they do on the frequency of confectionery.

Leverage

Although lift is widely used, it is not always a good measure of how interesting a rule will be. An association with lower frequency and higher lift may be of less interest than an alternative rule with higher frequency and lower lift, because the latter applies to more individuals and hence the total increase in the number of individuals resulting from the interaction between the antecedent and consequent is greater. A measure (proposed by Piatetsky-Shapiro, 1991) that captures both the volume and the strength of the effect in a single value is *leverage*.

$$\text{Leverage}(A \rightarrow C) = \text{support}(A \rightarrow C) - \text{support}(A) \times \text{support}(C)$$

This is the difference between the observed frequency with which the antecedent and consequent co-occur and the frequency that would be expected if the two were independent.

For example, consider the associations $\{\text{carrots}\} \rightarrow \{\text{tomatoes}\}$ and $\{\text{lettuce}\} \rightarrow \{\text{tomatoes}\}$. Both have confidence = 1.0 and lift = 1.667. However, the latter might be of greater interest because it applies to more customers. $\text{Support}(\{\text{carrots}\} \rightarrow \{\text{tomatoes}\}) = 0.3$. $\text{Support}(\{\text{carrots}\}) = 0.3$. $\text{Support}(\{\text{tomatoes}\}) = 0.6$. Hence, $\text{leverage}(\{\text{carrots}\} \rightarrow \{\text{tomatoes}\}) = 0.3 - 0.3 \times 0.6 = 0.3 - 0.18 = 0.12$. $\text{Support}(\{\text{lettuce}\} \rightarrow \{\text{tomatoes}\}) = 0.4$. $\text{Support}(\{\text{lettuce}\}) = 0.4$. $\text{Support}(\{\text{tomatoes}\}) = 0.6$. Hence, $\text{leverage}(\{\text{carrots}\} \rightarrow \{\text{tomatoes}\}) = 0.4 - 0.4 \times 0.6 = 0.4 - 0.24 = 0.16$. The absolute impact of the latter association is greater than that of the former.

Measures such as lift or leverage can be used to further constrain the set of associations discovered by setting a minimum value below which associations are discarded.

ITEM SET DISCOVERY

Close inspection of Table 2.3 shows that many variants of similar association rules are generated. For example, the last two association rules listed contain the same items but with different combinations thereof as the antecedent and consequent. For any item set, all associations between partitions of the item set will have the same support, but may have different confidence, lift, and leverage. For example, in Table 2.4 the rules

$$\text{apples} \rightarrow \text{confectionery}, \text{tomatoes}$$

and

$$\text{apples, tomatoes} \rightarrow \text{confectionery}$$

both have support of 0.3 but have differing confidence. Both have antecedents and consequents that are different partitions of the frequent item set {apples, confectionery, tomatoes}.

For some applications it is the affinity of a set of products to each other that is of interest, and the division of the products into an antecedent and consequent provides no benefit. For

example, if you are considering which items to locate close to one another in a supermarket, then it is logical groupings that are relevant rather than the form of causal relationship that might be implied by a rule structure. The item set identified previously identifies the affinity among the items without implying an implicit ordering among them.

Measures of interestingness are required if interesting item sets are to be discovered automatically. Counterparts to association rule lift and leverage can be defined as follows.

$$\text{item set-lift}(I) = \text{support}(I) / \prod_{i \in I} \text{support}(\{i\})$$

$$\text{item set-leverage}(I) = \text{support}(I) - \prod_{i \in I} \text{support}(\{i\})$$

Item set lift is the ratio of the observed support to the support that would be expected if there were no correlation between the items. The item set lift for {apples, confectionery, tomatoes} equals the support for this item set (0.3) divided by the product of the support for each item in the item set ($0.4 \times 0.6 \times 0.6 = 0.144$), which equals 2.08. Item set leverage is the difference between the observed support and the support that would be expected if there were no correlation between the items (0.156 in the above example).

TECHNIQUES FOR FREQUENT ITEM SET DISCOVERY

Most research in the area of association rule discovery has focused on the subproblem of efficient frequent item set discovery (for example, Han, Pei, & Yin, 2000; Park, Chen, & Yu, 1995; Pei, Han, Lu, Nishio, Tang, & Yang, 2001; Savasere, Omiecinski, & Navathe, 1995). When seeking all associations that satisfy constraints on support and confidence, once frequent item sets have been identified, generating the association rules is trivial.

The Apriori algorithm, described previously, is very efficient when the data is sparse and there are few frequent item sets containing large numbers of items. Data is considered *sparse* when each item is relatively infrequent. Sparse data gives rise to relatively few frequent item sets. When applied to sparse data, the Apriori algorithm is relatively fast, as the number of potential item sets is small, and hence relatively little processing is required. Whether data can be considered sparse depends to some extent on the value of *min-support*. The higher the value of *min-support* the fewer candidate frequent item sets.

Apriori is less efficient, however, when data is dense (Bayardo, 1997). For dense data the number of frequent item sets can greatly outnumber the number of transactions, requiring a lot of computation for the management of the candidate frequent item sets and discovered frequent item sets.

Closed Item Set Strategies

One approach to reducing these overheads is to reduce the number of frequent item sets that must be processed. For example, frequent closed item sets approaches record only frequent item sets that are closed (Pasquier, Bastide, Taouil, & Lakhal, 1999; Pei, Han, & Mao, 2000). An item set I is closed if and only if there is no superset S of I such that $\text{support}(S) = \text{support}(I)$. Note that if the support of the superset S equals the support of I , then the item sets I and S must both appear in the same transactions. It is possible from the set of closed frequent item

TABLE 2.5
Closed Frequent Item Sets

apples [Support=0.4]
confectionery [Support=0.6]
oranges [Support=0.3]
potatoes [Support=0.3]
tomatoes [Support=0.6]
confectionery, tomatoes [Support=0.4]
lettuce, tomatoes [Support=0.4]
apples, confectionery, tomatoes [Support=0.3]
carrots, confectionery, tomatoes [Support=0.3]

sets and their support to determine the set of all frequent item sets and their support. Table 2.5 lists the closed frequent item sets for our running example. There are 9 closed frequent item sets as opposed to 15 frequent item sets.

In addition to the computational advantages arising from reducing the number of frequent item sets that must be processed, proponents of the closed frequent item set approach also advocate restricting the association rules that are generated to rules $A \rightarrow C$ such that A is a closed frequent item set and $A \cup C$ is a closed frequent item set. The resulting rules are held to be the most interesting rules. Table 2.6 lists the rules so generated. The 54 rules in Table 2.4 have been compressed to just 26.

TABLE 2.6
Association Rules Generated from Closed Frequent Item Sets

→ apples, confectionery, tomatoes [Support=0.3, Confidence=0.30]
→ apples [Support=0.4, Confidence=0.40]
→ confectionery, tomatoes [Support=0.4, Confidence=0.40]
→ confectionery [Support=0.6, Confidence=0.60]
→ lettuce, tomatoes [Support=0.4, Confidence=0.40]
→ oranges [Support=0.3, Confidence=0.30]
→ potatoes [Support=0.3, Confidence=0.30]
→ tomatoes [Support=0.6, Confidence=0.60]
apples → [Support=0.4, Confidence=1.00]
apples → confectionery, tomatoes [Support=0.3, Confidence=0.75]
apples, confectionery, tomatoes → [Support=0.3, Confidence=1.00]
carrots, confectionery, tomatoes → [Support=0.3, Confidence=1.00]
confectionery → [Support=0.6, Confidence=1.00]
confectionery → apples, tomatoes [Support=0.3, Confidence=0.50]
confectionery → tomatoes, carrots [Support=0.3, Confidence=0.50]
confectionery → tomatoes [Support=0.4, Confidence=0.67]
confectionery, tomatoes → [Support=0.4, Confidence=1.00]
lettuce, tomatoes → [Support=0.4, Confidence=1.00]
oranges → [Support=0.3, Confidence=1.00]
potatoes → [Support=0.3, Confidence=1.00]
tomatoes → [Support=0.6, Confidence=1.00]
tomatoes → apples, confectionery [Support=0.3, Confidence=0.50]
tomatoes → carrots, confectionery [Support=0.3, Confidence=0.50]
tomatoes → confectionery [Support=0.4, Confidence=0.67]
tomatoes → lettuce [Support=0.4, Confidence=0.67]
tomatoes, confectionery → apples [Support=0.3, Confidence=0.75]
tomatoes, confectionery → carrots [Support=0.3, Confidence=0.75]

Long Item Sets

Another problem for the Apriori algorithm occurs if some frequent item sets are very long. Apriori requires one scan of the database for each item set length. When frequent item sets can contain many items, this approach can become prohibitively slow. One approach to overcoming this problem is to seek only maximal frequent item sets (Bayardo, 1998; Gouda & Zaki, 2001). A frequent item set is maximal if it is not a subset of any other frequent item set. The set of maximal frequent item sets is a subset of the set of closed frequent item sets, allowing even greater computational efficiency than the closed frequent item set approach. The set of maximal frequent item sets compactly describes the set of all frequent item sets, as any frequent item set must be a subset of a maximal frequent item set. However, unlike the case with closed frequent item sets, it is not possible to determine the support of an arbitrary frequent item set just from the maximal frequent item sets and their support values.

Sampling

Another way to speed up Apriori is to use a sample from the database to identify all candidate frequent item sets (Toivonen, 1996). The sample can be chosen so that it can be manipulated in the computer's memory to allow fast processing. Simple statistical techniques can identify all candidate item sets within the sample, those that exceed a minimum probability of being frequent item sets in the full database. The support for each candidate can then be confirmed with a single scan through the database.

TECHNIQUES FOR DISCOVERING ASSOCIATION RULES WITHOUT ITEM SET DISCOVERY

One limitation of the frequent item set approach is that it relies on there being a meaningful minimum support level that is sufficiently strong to reduce the number of frequent item sets to a manageable level. However, in some data mining applications relatively infrequent associations are likely to be of great interest because they relate to high-value transactions. For example, some retailers have a relatively small number of transactions that each has abnormally large value. These correspond to an individual purchasing on behalf of an organization, such as a group leader purchasing the supplies for a camping group. The behavior of the purchaser is often quite atypical in respects other than the volume of items purchased. For instance, the purchases may be less price-sensitive than normal. Understanding such transactions is clearly of potential value to the retailer, but the infrequency with which they occur means that frequent item set analysis is unable to detect them. This problem is sometimes called the *vodka and caviar problem*, drawing on an example presented by Cohen et al. (2000) showing that associations between expensive items such as Ketel Vodka and Beluga caviar are likely to be of interest due to their high value but will be infrequent.

A number of algorithms have been developed that allow rules to be discovered without the use of a minimum support constraint. Cohen et al.'s (2000) algorithm finds associations with very high confidence.

Several algorithms for rule-space search directly explore the space of potential rules (Bayardo & Agrawal, 1999; Bayardo, Agrawal, & Gunopulos, 2000; Webb, 2000). This framework allows a wide range of criteria to be applied to identify interesting rules other than the minimum support criterion at the heart of the Apriori approach. A key requirement, however,

is that those criteria enable areas of the search space that may contain interesting rules to be efficiently identified.

ASSOCIATIONS WITH NUMERIC VALUES

Although association rules initially were developed in the context of market-basket analysis, they have proved useful in a wide range of applications. However, association rules are not directly applicable to numeric data. The standard approach to incorporating numeric data is *discretization*, a process by which the values of a numeric field in a database are divided into subranges. Each such subrange is treated as an item in the association rule analysis. Each record in the database is treated as containing the item corresponding to the subrange within which the numeric value for the record falls.

Most systems prediscretize the data. That is, intervals are identified before association rule analysis is performed. However, such an approach will use a single set of intervals for all association rules, even though different sets of intervals might be more appropriate in the context of associations between the numeric field and different alternative fields. Srikant and Agrawal (1996) address this problem by dynamically adjusting the discretization intervals during the association rule discovery process.

Another type of association-rule-like analysis for numeric values is provided by bump hunting (Friedman & Fisher, 1999) and impact rules (Aumann & Lindell, 1999; Webb, 2001). These techniques find antecedents that are associated with specific types of impact on a numeric variable. For example, the techniques might be used to find the groups of customers for whom the sum of the profitability is greatest (the most profitable customer segments) or for whom the average cost to service is least (the least expensive categories of customers).

APPLICATIONS OF ASSOCIATION RULE DISCOVERY

Although association rule discovery grew out of market basket analysis, it has been applied subsequently in a wide variety of contexts. One simple extension to market basket analysis is to augment the database records with information about the customers purchasing the items. Where loyalty cards or other mechanisms allow customers to be identified, customer information can readily be added. This information need not be limited to in-house data. Numerous data-enrichment services can add socioeconomic and other information to fully identified customers. This information can add greater richness to rules that are generated and can be particularly valuable for marketing applications. However, augmenting data in this way often results in dense data. For example, in contrast to the relative infrequency with which any one item is likely to be purchased at a store, if the gender of the purchaser is added to a record then one of the two values male and female must occur in at least 50% of records. The addition of just a small number of frequent values such as these can dramatically increase the number of frequent item sets, substantially degrading processing time when using frequent item set strategies.

Another variation on market basket analysis is to aggregate market baskets over a period of time. For example, analysis might be performed on all the products bought by a customer over a 12-month period. Such analysis can also be very useful for marketing purposes. However, such aggregation is also likely to result in dense data.

Applications of association rule analysis are not restricted to such analyses of purchasing behavior, however. The manner in which association rule approaches identify all rules that satisfy some set of measures of interestingness make them useful for a wide variety of applications in which the user seeks insight into a set of data. The user can specify the quantifiable measures of what makes a rule interesting. Further factors may apply that are difficult to quantify, but the user can apply these manually to select the most useful of the rules discovered by the automated system.

One interesting example of these types of application is text analysis (Feldman, 1999; Mladenic, 2000). In this analysis, each document is treated as a record. The association analysis identifies combinations of words that occur together in a document. Such analysis can be useful for document retrieval and classification purposes.

Another application of association rules has been for classification. Rather than the classical approach of learning classification rules, a large set of high-confidence association rules are discovered. To classify a new object, all rules that cover the object are identified, and then the relative support and confidence of each rule is used to identify the most probable class for the object (Liu, Hsu, & Ma, 1998).

SUMMARY

Association rule discovery identifies rules that describe affinities between combinations of data elements. This rule-mining paradigm developed in the context of market basket analysis, which seeks to identify products that are bought together. However, the techniques are more broadly applicable. In its most general conception, association rule discovery seeks to discover all rules that satisfy an arbitrary set of constraints with respect to a set of reference data.

Early approaches to association rule discovery sought all rules that satisfy constraints on support and confidence. More recent techniques utilize additional measures of how interesting a rule is, including lift and leverage.

In some data mining contexts there is no need to distinguish between the antecedent and consequent of an association. All that is required is information about the groups of data elements that have affinities together. In this context it may be desirable to discover item sets rather than association rules.

Most association rule discovery systems use the frequent item set strategy as exemplified by the Apriori algorithm (Agrawal & Srikant, 1994). This approach requires that a minimum support be specified. First all item sets that equal or exceed the support threshold are discovered. Then association rules are generated therefrom. This strategy is very efficient for the sparse data that typifies market basket analysis. Many techniques have been developed for speeding up the frequent item set discovery process (Pasquier, Bastide, Taouil, & Lakhal, 1999; Bayardo, 1998; Gouda & Zaki, 2001; Pei, Han, & Mao, 2000; Toivonen, 1996; Zaki, Parthasarathy, Ogihara, & Li, 1997).

For dense data, or when it is not possible to specify a minimum value for support, other approaches are desirable. This has led to the development of a number of algorithms for discovering interesting associations without the need to discover frequent item sets (Bayardo & Agrawal, 1999; Bayardo, Agrawal, & Gunopulos, 2000; Cohen et al., 2000; Webb, 2000).

Another direction in which association rule techniques have been extended is to accommodate numeric data. One approach is to discretize the numeric values (Srikant & Agrawal, 1996). Another family of approaches finds rules with numeric distributions as the antecedents (Friedman & Fisher, 1999; Aumann & Lindell, 1999; Webb, 2001).

REFERENCES

- Agrawal, R., Imielinski, T., & Swami, A. (1993). Mining associations between sets of items in massive databases. In *Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data* (pp. 207–216). New York: ACM Press.
- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases* (pp. 487–499). San Francisco: Morgan Kaufmann.
- Aumann, A., & Lindell, Y. (1999). A statistical theory for quantitative association rules. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD '99)* (pp. 261–270). New York: ACM Press.
- Bayardo, R. J., Jr. (1997). Brute-force mining of high-confidence classification rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD '97)* (pp. 123–126). Menlo Park, CA: AAAI Press.
- Bayardo, R. J., Jr. (1998). Efficiently mining long patterns from databases. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (pp. 85–93). New York: ACM Press.
- Bayardo, R. J., Jr., & Agrawal, R. (1999). Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 145–154). New York: ACM Press.
- Bayardo, R. J., Jr., Agrawal, R., & Gunopulos, D. (2000). Constraint-based rule mining in large, dense databases. *Data Mining and Knowledge Discovery*, 4, 217–240.
- Brin, S., Motwani, R., Ullman, J. D., & Tsur, S. (1997). Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 1997)* (pp. 255–264). AZ: New York: ACM Press.
- Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J., & Yang, C. (2000). Finding interesting associations without support pruning. In *Proceedings of the International Conference on Data Engineering* (pp. 489–499). New York: IEEE Press.
- Feldman, R. (Ed.). (1999). *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI '99) Workshop on Text Mining*. Somerset, NJ: IJCAI Inc.
- Freitas, A. A. (1999). On rule interestingness measures. *Knowledge-Based Systems*, 12, 309–315.
- Friedman, J. H., & Fisher, N. I. (1999). Bump hunting in high-dimensional data. *Statistics and Computing*, 9, 123–143.
- Gouda, K., & Zaki, M. J. (2001). Efficiently mining maximal frequent itemsets. In *Proceedings of the First IEEE International Conference on Data Mining* (pp. 163–170). New York: IEEE Press.
- Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2000)* (pp. 1–12). New York: ACM Press.
- Kamber, M., & Shinghal, R. (1995). Evaluating the interestingness of characteristic rules. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 263–266). Menlo Park, CA: AAAI Press.
- Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., & Verkamo, A. (1999). Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third International Conference on Information and Knowledge Management* (pp. 401–407). New York: ACM Press.
- Liu, B., Hsu, W., Chen, S., & Ma, Y. (2000). Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems*, 15, 47–55.
- Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98)* (pp. 80–86). Menlo Park, CA: AAAI Press.
- Mladenic, D. (Ed.). (2000). *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD '2000) Workshop on Text Mining*. New York: ACM Press.
- Park, J. S., Chen, M.-S., & Yu, P. S. (1995). An effective hash based algorithm for mining association rules. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 175–186). New York: ACM Press.
- Pasquier, N., Bastide, Y., Taouil, R., & Lakhal, L. (1999). Discovering frequent closed itemsets for association rules. In *Proceedings of the Seventh International Conference on Database Theory (ICDT '99)* (pp. 398–416). New York: Springer.
- Pei, J., Han, J., Lu, H., Nishio, S., Tang, S., & Yang, D. (2001). H-Mine: Hyper-structure mining of frequent patterns in large databases. In *Proceedings of the First IEEE International Conference on Data Mining* (pp. 441–448). New York: IEEE Press.
- Pei, J., Han, J., & Mao, R. (2000). CLOSET: An efficient algorithm for mining frequent closed itemsets. In *Proceedings of the 2000 ACM-SIGMOD International Workshop on Data Mining and Knowledge Discovery (DMKD '00)* (pp. 21–30). New York: ACM Press.

- Piatetsky-Shapiro, G. (1991). Discovery, analysis, and presentation of strong rules. In *Knowledge Discovery in Databases* (pp. 229–248). Menlo Park, CA: AAAI/MIT Press.
- Sahar, S. (1999) Interestingness via what is not interesting. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD-99)* (pp. 332–336). New York: ACM Press.
- Savasere, A., Omiecinski, E., & Navathe, S. (1995). An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Data Bases* (pp. 432–444). San Francisco: Morgan Kaufmann.
- Silberschatz, A., & Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8, 970–974.
- Srikant, R., & Agrawal, R. (1996). Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (pp. 1–12). New York: ACM Press.
- Toivonen, H. (1996). Sampling large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Databases* (pp. 134–145). San Francisco: Morgan Kaufmann.
- Webb, G. I. (2000). Efficient search for association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 99–107). New York: ACM Press.
- Webb, G. I. (2001). Discovering associations with numeric variables. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 383–388). New York: ACM Press.
- Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). New algorithms for fast discovery of association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 283–286). Menlo Park, CA: AAAI Press.

3

Artificial Neural Network Models for Data Mining

Jennie Si

*Department of Electrical Engineering, Arizona State
University, Tempe*

Benjamin J. Nelson and George C. Runger
*Department of Industrial Engineering, Arizona State
University, Tempe*

Introduction to Multilayer Feedforward Networks	42
Gradient Based Training Methods for MFN	43
The Partial Derivatives	44
Nonlinear Least Squares Methods	45
Batch versus Incremental Learning	47
Comparison of MFN and Other Classification Methods	47
Decision Tree Methods	47
Discriminant Analysis Methods	48
Multiple Partition Decision Tree	49
A Growing MFN	50
Case Study 1—Classifying Surface Texture	52
Experimental Conditions	52
Quantitative Comparison Results of Classification Methods	53
Closing Discussions on Case 1	55
Introduction to SOM	55
The SOM Algorithm	56
SOM Building Blocks	57
Implementation of the SOM Algorithm	58
Case Study 2—Decoding Monkey’s Movement Directions from Its	
Cortical Activities	59
Trajectory Computation from Motor Cortical Discharge Rates	60

The first author’s research is supported by the NSF under grants ECS-9553202 and ECS-0002098, and in part by DARPA under grant MDA 972-00-1-0027. The second and third authors’ research is supported in part by NSF under grant DMII-9713518.

Using Data from Spiral Tasks to Train the SOM	62
Using Data from Spiral and Center→Out Tasks to Train the SOM	62
Average Testing Result Using the Leave-K-Out Method	63
Closing Discussions on Case 2	64
Final Conclusions and Discussions	65
References	65

INTRODUCTION TO MULTILAYER FEEDFORWARD NETWORKS

Multilayer feedforward networks (MFN) have been applied successfully to solve some difficult and diverse problems, including nonlinear system identification and control, financial market analysis, signal modeling, power load forecasting, and so forth, by providing training in a supervised manner (Haykin, 1999; Hertz, Krogh, & Palmer, 1991; Narendra & Parthasarathy, 1990; White & Sofge, 1992). In supervised learning one starts with a training set and uses certain numerical procedures, which usually solve the nonlinear optimization problem deduced by supervised learning, to compute the network parameters (weights) by loading training data pairs into the network. The hope is that the network so designed will generalize, meaning that the input-output relationship computed by the network is within certain expectation as measured by testing error, for some input-output data pairs that were never used in training the network. A typical bad generalization may occur due to overfitting. Generalization usually is affected by three factors: the size and efficiency of the training data set, the complexity of the network, and the complexity of the physical problem at hand. In most neural network application problems the size of the training set is given, and herein it is assumed to be representative of the system. The issue to be addressed then is how to determine a set of network parameters for achieving good generalization. This may include choosing the right network size, an efficient algorithm for determining the weights for the network to achieve desired accuracy for training and testing.

The following aspects in the overall picture of network design play important roles just as the training mechanism to be addressed later. However, they are not the focus of the present chapter and interested readers may find discussions on these issues elsewhere (Haykin, 1999).

Among various feedforward network structures, static and recurrent networks are used widely in static function approximation and dynamic system modeling. A recurrent network is a network in which neurons may input the outputs from any other neurons (or from themselves) through feedback. Therefore, such a network constitutes a dynamic system. However, the term “static mapping” is often used to describe a memoryless system. In most data mining applications an MFN is used as a static mapping between the explanatory variables and their class labels.

Common examples of feedforward neural network implementations include multilayer perceptrons and radial basis function networks, as well as others (Haykin, 1999). These have been proved to be universal approximators (Cybenko, 1989; Funahashi, 1989; Hornik, Stinchcombe, & White, 1989; Park & Sandberg, 1991). Other than the apparent global versus local tuning characteristics for sigmoid and radial basis networks, respectively, and the faster learning capability by radial basis networks, another interesting aspect regarding the two networks is that when the input dimension is low, radial basis function networks are more efficient than sigmoid. However, when the input dimension is high, this order is reversed (Liu & Si, 1992; chap. 5 in White & Sofge, 1992). Therefore, a careful selection of the centers and widths of

the radial basis functions becomes critical, which is in general a nontrivial task. The training mechanism discussed in this chapter is applicable to either static or dynamic, sigmoid or radial basis networks, as will be shown in the following sections.

GRADIENT BASED TRAINING METHODS FOR MFN

Training a neural network is considered in this chapter to be an optimization problem. In particular, the objective of training a neural network is to associate input-output training pairs $\{(x^1, t^1), \dots, (x^\xi, t^\xi), \dots, (x^m, t^m)\}$ by properly adjusting the weights \mathbf{w} in the network such that an error measure $E(\mathbf{w})$ is minimized. A sum of squared error function is a commonly used measure.

$$E(\mathbf{w}) = \frac{1}{2} \sum_{\xi,i} (e_i^\xi)^2 = \frac{1}{2} \sum_{\xi,i} (t_i^\xi - o_i^\xi)^2, \quad (1)$$

where o^ξ represents the network output when the input is x^ξ .

Various training algorithms have been developed to adapt the weights \mathbf{w} in the network recursively to reduce the error defined in Equation 1 (Hertz, Krogh, & Palmer, 1991; Werbos, 1993, 1994). Two major types of algorithms have been widely used. On the one hand, there are several ad hoc techniques such as varying the learning rates in the back propagation algorithm, adding a momentum term to the basic back-propagation algorithm (Rumelhart, Hinton, & Williams, 1986). On the other hand, others use standard numerical optimization techniques like the conjugate gradient method or quasi-Newton (secant) methods (Dennis & Schnabel, 1983). Some nonlinear least squares methods have been tested and practically provide better performance than the previous two types of approaches (Hagen & Menhaj, 1994).

Back-propagation is a commonly used training technique in which parameters are moved in the opposite direction to the error gradient. Each step down the gradient results in smaller errors until an error minimum is reached. The computational complexity of back-propagation is contributed mostly by calculating the partial derivatives ($O(n)$ only). However, this gradient descent based method is linearly convergent only, and is usually very slow. Another heuristic technique of similar complexity to the back-propagation algorithm is the gradient algorithm with momentum. It makes changes proportional to a running average of the gradient. It may be considered as an approximation of the conjugate gradient method. In general, this technique can decrease the probability that the network gets stuck in a shallow minimum on the error surface and thus can decrease training times. However, the learning rate and the momentum parameter have to be chosen, carefully which is a highly heuristic process.

Newton's method provides good local convergence properties. It is q-quadratically convergent when a minimum is approached. Modifications are needed before it can be used at points that are remote from the solution. Calculation of the Hessian should be avoided. The Levenberg-Marquardt algorithm is a combination of gradient descent and Gauss-Newton methods that combines the advantages of the local convergence properties of the Gauss-Newton method and the global properties of gradient descent. A comparison study is reported in Hagen & Menhaj (1994) where the Levenberg-Marquardt method significantly outperforms the conjugate gradient and the back-propagation methods with a variable learning rate (Kollias & Anastassiou, 1989) in terms of training time and accuracy.

No matter which of these algorithm is used, the gradient information is always needed first.

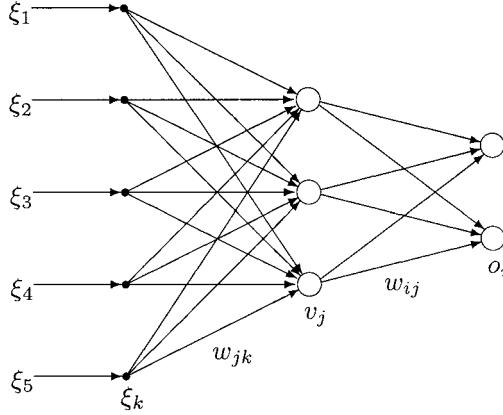


FIG. 3.1. A two-layer neural network.

The Partial Derivatives

The gradient descent method for adjusting the weights \mathbf{w} in the network to reduce the error in Equation 1 is a good starting point for deriving most of the training algorithms that have been discussed. Referring to Fig. 3.1, the partial derivatives with respect to the second layer of weights $\frac{\partial E}{\partial w_{ij}^2}$ are given by

$$\frac{\partial E}{\partial w_{ij}^2} = - \sum_{\xi} e_i^{\xi} \phi'(s_i^{\xi}) v_j^{\xi} = - \sum_{\xi} \delta_i^{\xi} v_j^{\xi} \quad (2)$$

where $i = 1, \dots, n_3$, $j = 1, \dots, n_2$, $\delta_i^{\xi} = \phi'(s_i^{\xi}) e_i^{\xi}$, $\phi(\cdot)$ is the nonlinearity at hidden and output nodes, and s_i^{ξ} is the weighted sum of inputs from previous layer to node i .

Similarly, $\frac{\partial E}{\partial w_{jk}^1}$ can be obtained by applying the chain rule,

$$\frac{\partial E}{\partial w_{jk}^1} = \frac{\partial E}{\partial v_j^{\xi}} \frac{\partial v_j^{\xi}}{\partial w_{jk}^1} = - \sum_{\xi, i} \delta_i^{\xi} w_{ij}^2 \phi'(s_j^{\xi}) x_k^{\xi} = - \sum_{\xi} \delta_i^{\xi} x_k^{\xi}, \quad (3)$$

where $j = 1, \dots, n_2$, $k = 1, \dots, n_1$, $\delta_j^{\xi} = \phi'(s_j^{\xi}) \sum_i w_{ij}^2 \delta_i^{\xi}$.

The basic back-propagation rule can thus be given as

$$\Delta w_{ij}^2 = -\eta \frac{\partial E}{\partial w_{ij}^2}, \quad (4)$$

$$\Delta w_{jk}^1 = -\eta \frac{\partial E}{\partial w_{jk}^1}, \quad (5)$$

where $\eta > 0$ is the learning rate.

As a general rule, for networks with any number of layers,

$$\frac{\partial E}{\partial w_{pq}^l} = - \sum_{\xi} \delta_p^{\xi} v_q^{\xi}. \quad (6)$$

The meaning of δ_p depends on the layer l . If l is the output layer, δ_p is given in Equation 2. For all other layers, δ_p is the back-propagated error from its upper layer through the weights between node p and its upper layer.

As an extension, the back-propagation with momentum rule is simply

$$\Delta w_{pq}^l(k+1) = -\eta \frac{\partial E}{\partial w_{pq}^l} + \alpha \Delta w_{pq}^l(k), \quad (7)$$

where η and α are parameters to be chosen properly.

Nonlinear Least Squares Methods

Let $\mathbf{w}^{(k)} \in R^n$ denote the network weight vector at the k th iteration. A new weight vector $\mathbf{w}^{(k+1)}$ is obtained by the following update rule,

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \Delta \mathbf{w}^{(k)} \quad (8)$$

Referring to Fig. 3.1, the weight vector \mathbf{w} may be formulated as

$$\mathbf{w} = (w_{11}^1, w_{12}^1, \dots, w_{35}^1, w_{11}^2, w_{12}^2, \dots, w_{23}^2)^T.$$

To simplify the notation further, let

$$E(\mathbf{w}) = \frac{1}{2} \sum_{\xi} (e^{\xi})^2 = \frac{1}{2} \sum_{\xi} (t^{\xi} - o^{\xi})^2 \quad (9)$$

which implies that the network has a single output. For multiple outputs the only added complexity is that the summation term in Equation 9 has to be increased from m to $m \times n_3$. Then it can be shown that (Dennis & Schnabel, 1983)

$$\nabla E(\mathbf{w}) = J^T(\mathbf{w})\epsilon(\mathbf{w}), \quad (10)$$

$$\nabla^2 E(\mathbf{w}) = J^T(\mathbf{w})J(\mathbf{w}) + S(\mathbf{w}), \quad (11)$$

where $\epsilon(\mathbf{w}) = (e^1, e^2, \dots, e^m)^T$, $S(\mathbf{w}) = \sum e^{\xi}(\mathbf{w})\nabla^2 e^{\xi}(\mathbf{w})$, and $J(\mathbf{w})$ is the Jacobian matrix given by

$$J = \begin{bmatrix} \frac{\partial e^1}{\partial w_1} & \frac{\partial e^1}{\partial w_2} & \dots & \frac{\partial e^1}{\partial w_n} \\ \frac{\partial e^2}{\partial w_1} & \frac{\partial e^2}{\partial w_2} & \dots & \frac{\partial e^2}{\partial w_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e^m}{\partial w_1} & \frac{\partial e^m}{\partial w_2} & \dots & \frac{\partial e^m}{\partial w_n} \end{bmatrix}. \quad (12)$$

Most of the nonlinear least squares methods are based on the Newton's method, which is

$$\Delta \mathbf{w} = -[\nabla^2 E(\mathbf{w})]^{-1} \nabla E(\mathbf{w}). \quad (13)$$

If $S(\mathbf{w}) \approx 0$, which is usually the case when a solution is approached, we obtain the Gauss-Newton update rule,

$$\Delta \mathbf{w} = -[J^T(\mathbf{w})J(\mathbf{w})]^{-1} J^T(\mathbf{w})\epsilon(\mathbf{w}). \quad (14)$$

The Levenberg-Marquardt algorithm is a modified version of Gauss-Newton, which is of the form

$$\Delta \mathbf{w} = -[J^T(\mathbf{w})J(\mathbf{w}) + \mu I]^{-1} J^T(\mathbf{w})\epsilon(\mathbf{w}), \quad (15)$$

where μ is a positive constant and I is the identity matrix.

It can be seen from Equation 15 that if μ is very large, the Levenberg-Marquardt algorithm approximates the gradient descent, whereas if μ is zero it becomes the Gauss-Newton method. But the Levenberg-Marquardt algorithm is faster than the gradient descent because more information is used in this quasi-second order approximation. It is better than the Gauss-Newton method in the sense that $(J^T J + \mu I_1)$ is always positive definite and the solution of Equation 15 exists, whereas for the Gauss-Newton method, singularity of $(J^T J)$ is a potential problem. In practical implementations μ is decreased after each successful step and increased only when a step increases the error.

As is evidenced in Equation 15, the weight updates require an inversion of an $n \times n$ matrix (n is the total number of weights in the network) for each iteration. The computational complexity of the Levenberg-Marquardt algorithm is about $O(n^3)$. If the number of network weights n is very large, the requirements for computation and memory become significant. However, the overall performance may be made up for by increased efficiency, especially when high precision is required (Battiti, 1992; Hagen & Menhaj, 1994).

As for the Jacobian matrix J , it can be obtained by simple modifications to the back-propagation algorithm.

Referring to Figure 3.1, for the weight w_{ij}^2 between the output node i (which is always 1 in this section) and the hidden node j , it is first labeled as the ξ th element in the weight vector \mathbf{w} , then from the partial derivatives of Equation 2, for the ξ th training pattern we obtain

$$J_{\xi\xi} = \frac{\partial e^\xi}{\partial w_\xi} = -\phi'(s_i^\xi)v_j^\xi. \quad (16)$$

For the weight w_{jk}^1 between the hidden node j and the input node k , find the labeling number ζ in the weight vector \mathbf{w} first, then use the idea similar to the back-propagated partial derivatives in Equation 3 to obtain

$$J_{\xi\zeta} = \frac{\partial e^\xi}{\partial w_\zeta} = -\phi'(s_i^\xi)w_{ij}^2\phi'(s_j^\xi)x_k^\xi. \quad (17)$$

The following is an example of how supervised learning can be implemented in computer simulation code. The Levenberg-Marquardt algorithm is used to illustrate the weight updating and converging process of a supervised learning algorithm. The basic gradient descent or gradient descent with momentum algorithms can be implemented the same way, except that the corresponding weight updating equations are replaced properly.

1. Specify a training error tolerance ϵ and two multipliers γ_1 and γ_2 . Initialize the weight vector $\mathbf{w}^{(0)}$, and set $k = 0$ and $\mu = \mu_0$.
2. Compute the network output and the error vector $\epsilon(\mathbf{w}^{(k)})$.
3. Compute the Jacobian matrix J of the error vector with respect to the network weights by Equations 16 and 17.
4. Compute $\Delta \mathbf{w}^{(k)}$, $\mathbf{w}^{(k+1)}$, $E(\mathbf{w}^{(k+1)})$ by Equations 15, 8, and 9, respectively.

5. If $E(\mathbf{w}^{(k+1)}) < E(\mathbf{w}^{(k)})$, then go to step 7; otherwise, do not update the weights at this time. Let $\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)}$ and continue.
6. If $\|(J^{(k+1)})^T \mathbf{e}^{(k+1)}\| \leq \varepsilon$, then we have reached a local optimal solution $\mathbf{w}^* = \mathbf{w}^{(k)}$ and stop; otherwise, let $\mu = \mu \cdot \gamma_1$, and go to step 4.
7. If $\|(J^{(k+1)})^T \mathbf{e}^{(k+1)}\| \leq \varepsilon$, then we have reached a local optimal solution $\mathbf{w}^* = \mathbf{w}^{(k+1)}$ and stop; otherwise, let $k = k + 1$, $\mu = \mu/\gamma_2$, and go to step 2.

Batch versus Incremental Learning

Note that the update rules given by Equations 2, 3, and 13 through 15 have summations over all patterns ξ . In neural networks literature they are referred to as batch training techniques, in which weights in the network are updated after a complete sweep through the training data set. On the other hand, it is a common practice to use the update rules in Equations 2, 3, and 13 through 15 incrementally; a pattern ξ is presented at the input and then all the weights are updated before the next pattern is considered. Incremental learning is also called online learning or real-time learning.

There is no clear advantage of one approach over the other in general. However for very high precision mappings, batch processing may be preferable (Battiti, 1992; Hagen & Menhaj, 1994). The fact that many training patterns possess redundant information has been used as an argument in favor of incremental learning, because many of the patterns contribute to similar gradient information, so that waiting for all patterns before one updating can be wasteful. In other words, averaging over more examples to obtain a better estimate of the gradient does not improve the learning speed sufficiently to compensate for the additional computational cost of taking these patterns into account. Nonetheless, the redundancy also can be limited by using batch training, provided that learning is started with a subset or a window of “relevant” patterns and continued after convergence by progressively increasing the example set.

COMPARISON OF MFN AND OTHER CLASSIFICATION METHODS

The classification of objects into groups or supervised learning is a common decision-making requirement. In general, there are two primary approaches to this process of classification. The traditional statistical approach (discriminant analysis) and the computer science approach (machine learning) seem to be quite different in their fundamental assumptions. With either of these approaches the problem is one of developing a model from a training data set that properly assigns future objects into predetermined classes. The reliability of the methods is evaluated either by cross-validation or by applying the model to a new test set of similar objects and evaluating the error rates (proportion of misclassified objects).

The classification problem tends to occur in a wide variety of areas. This widespread need to classify objects has led to the development of many different classification techniques. Some of these techniques are linear and quadratic discriminant analysis, decision trees, neural networks, mixed-integer programming, and expert rule-based systems.

Decision Tree Methods

Most decision tree methods learn the decision trees by a top-down approach, beginning with the question “which attribute should be used to define a partition?” To answer this question, each attribute is evaluated using the information gained to determine how well it alone classifies the

training examples. The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to partition with at that point in the tree. This forms a greedy search for a decision tree, in which the algorithm never backtracks to reconsider earlier choices. Decision tree is a discrete-value function method that is robust to noisy data and capable of learning “or” expressions. The most common decision tree algorithms are ID3, discussed by Forsyth and Rada (1986), and C4.5, covered by Quinlan (1983). The method used in the comparison is ID3. Some advantages are as follows.

1. Human understandable representation: The decision tree can be transferred into a set of “if-then” rules to improve human readability.
2. Decision tree learning methods are robust to errors in classification of the training set. The testing set and pruning algorithms are used to minimize the tree size and the misclassification.
3. Some decision tree methods can be used even when some training examples have unknown values. ID3 does not provide a mechanism to handle the missing attribute values; however, C4.5 does.

After the decision tree is generated, an attempt is made to prune the decision tree by eliminating some of the smaller branches. The basic algorithm of reduced error pruning is as follows.

1. Prepare a separate set of examples distinct from the training examples, called a “validating set,” to evaluate the effect of pruning.
2. Build the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
3. Nodes are pruned iteratively from the leaf (bottom) based on the validating set.
4. Pruning of nodes continues until further pruning is not required.

Discriminant Analysis Methods

The basic concept of discriminant analysis is to estimate the distance from the object to the center of the class and then assign the object to the nearest group. Thus, the primary question becomes one of determining the appropriate distance measure. Usually the Mahalanobis distance, based on the covariance matrix of the observations, is used to estimate these distances. Depending on the assumption of an equal covariance matrix for all classes, either linear (equal covariance matrix) or quadratic (unequal covariance matrix) discriminant analysis is used. When the covariance matrices are not equal, the Euclidean distance from the class centroids to the discrimination boundary is different for different classes.

Suppose a data set has n_p predictors and g classes. The following notation is used: \bar{x} is the column vector of length n_p containing the values of the predictors for this observation; \bar{m}_i is a column vector of length n_p containing the means of the predictors calculated from the data in class i ; S_i is the covariance matrix calculated from the data in class i ; $|S_i|$ is the determinant of S_i ; S is the pooled covariance matrix; p_i is the a priori probability that an observation is in group i .

For the equal covariance case the squared distance (Mahalanobis distance) of observation \bar{x} to the center (mean) of group i is given by

$$d_i^2(\bar{x}) = (\bar{x} - \bar{m}_i)^T S^{-1} (\bar{x} - \bar{m}_i). \quad (18)$$

An observation \bar{x} is classified into group i if the squared distance for \bar{x} to group i is the smallest. The equation for the squared distance can be expanded into

$$d_i^2(\bar{x}) = -2[\bar{m}_i^T S^{-1} \bar{x} - 0.5\bar{m}_i^T S^{-1} \bar{m}_i] + \bar{x}^T S^{-1} \bar{x}. \quad (19)$$

The term in square brackets is a linear function of \bar{x} , and it is called the linear discriminant function for group i . For a given \bar{x} , the largest linear discriminant function value occurs for the group with the smallest squared distance.

In quadratic discriminant analysis the groups are not assumed to have equal covariance matrices. Again, an observation is classified into the group from which it has the smallest squared distance. The squared distance is given by

$$d_i^2(\bar{x}) = (\bar{x} - \bar{m}_i)^T S_i^{-1} (\bar{x} - \bar{m}_i) - \ln|S_i|. \quad (20)$$

This does not simplify into a linear function. In quadratic discriminant analysis the multivariate distribution of the explanatory variables is assumed to be different in each group.

One expects linear discriminant analysis to perform well when the classes can be approximately described by multivariate normal distributions with identical covariance matrices. This assumption of multivariate normal distributions may seem reasonable for continuous data, but clearly not for categorical explanatory variables. The assumption of equal covariance matrices also seems to be fairly drastic, but this can always be resolved by using quadratic discriminant analysis.

The following are limitations and dangers of this approach.

1. Although categorical variables can be converted to indicator variables similar to what is done in regression, discriminant analysis is primarily effective for continuous explanatory variables. Furthermore, these indicator variables are not multivariate normally distributed. Consequently, linear discriminant analysis is not expected to excel in this case.
2. The use of the Mahalanobis distance to the centroid of a class can result in problems if unusual shaped classes were encountered. A half-torus is such an example where the centroids of the two classes are actually very close to each other. These types of unusual shaped classes will lead to high error rates for classification.
3. Discriminant analysis is dependent on the estimate of the covariance matrix. Classes with small membership will have poorly estimated covariance matrices. The error rates for testing data will tend to be much worse than for the training data, because the covariance matrix estimates may change significantly due to the random sampling of only a few members from that class.

Multiple Partition Decision Tree

The decision tree is constructed by using a single variable to partition the data set into one or more subsets. After each step each of the partitions is partitioned as if each were a separate data set. Each subset is partitioned without any regard for the other subsets. This process is repeated until the stopping criteria are met. This recursive partitioning creates a tree structure. The root of the tree is the entire data set. The subsets and sub-subsets form the branches. When a subset meets the stopping criteria and is not repartitioned, it is a leaf. Each subset in the final tree is called a node.

The major difference between a decision tree and a multiple partition decision tree is the number of variables used for partitioning the data set. In a multiple partitioning the number

of subsets for a variable can range from two to the number of unique values of the predictor variable.

The predictor variable used to form a partition is chosen to be the variable that is most significantly associated with the dependent variable. A chi-square test of independence for a contingency table is used as the measure of association. The stopping criterion is the p value for the chi-square test. If a predictor has more than two categories, there may be a large number of ways to partition the data. A combinatorial search algorithm is used to find a partition that has a small p value for the chi-square test.

This algorithm does not specifically support continuous explanatory variables. Continuous variables are treated as ordinal. Only values of a variable that are adjacent to each other can be combined. With a large number of continuous variables and a large number of objects, this can have a large impact on execution time.

The following are limitations and dangers for decision trees.

1. The limitation is to ordinal or nominal data, and execution speed is an issue when continuous variables are used.
2. This method splits the data by single variables at a time. This can cause high error rates if classes are shaped like two circles but with overlapping projections on some axes. This problem can be reduced by adding linear combinations of variables to the list of explanatory variables or by using principal components for the classification. Of course, if linear combinations must be determined from the data, the execution time might greatly increase.
3. One of the major dangers with continuous data is overfitting. When we are dealing with measured data with random variation, it is not uncommon for the measurement variability to be large. This creates a large number of unique values to split the training set on, which may not match up well with the test data set. What fits well for a training set may have much higher error rates for the test set.
4. One other issue with this method is the size of the decision tree. A traditional advantage of decision trees is the ease of interpretation and visualization. However, it is not uncommon for the number of nodes generated to exceed 150 using this method. This number of nodes greatly decreases the interpretability of the decision tree.

A Growing MFN

In most application problems the number of nodes in the hidden layer of an MFN needs to be determined. This requires dividing the data set into training and testing sets and training the network with a range of nodes in the hidden layer. By graphing the mean square error (MSE) for the training and testing sets in what are called learning curves, the number of hidden nodes can be estimated (Haykin, 1999). The number of hidden nodes is based on having a low MSE for the training set and a minimum difference between the training and testing set MSEs.

When a large number of statistical models are needed for systematic studies, this approach is unreasonable. To compensate, a method called cascading correlations (Fahlmann & Lebiere, 1990) can be used to generate the network. Cascading correlations is a very efficient algorithm for growing a network. The cascade network still consists of inputs, outputs, hidden units, and weights connecting the units between the layers (see Fig. 3.2). One major difference from the “fixed structure” feedforward network as shown in Fig. 3.1 is the construction of hidden notes. In a cascade network the hidden units are added one by one. The network starts with no hidden units. The weights from inputs leading to outputs are first trained by a regular supervised

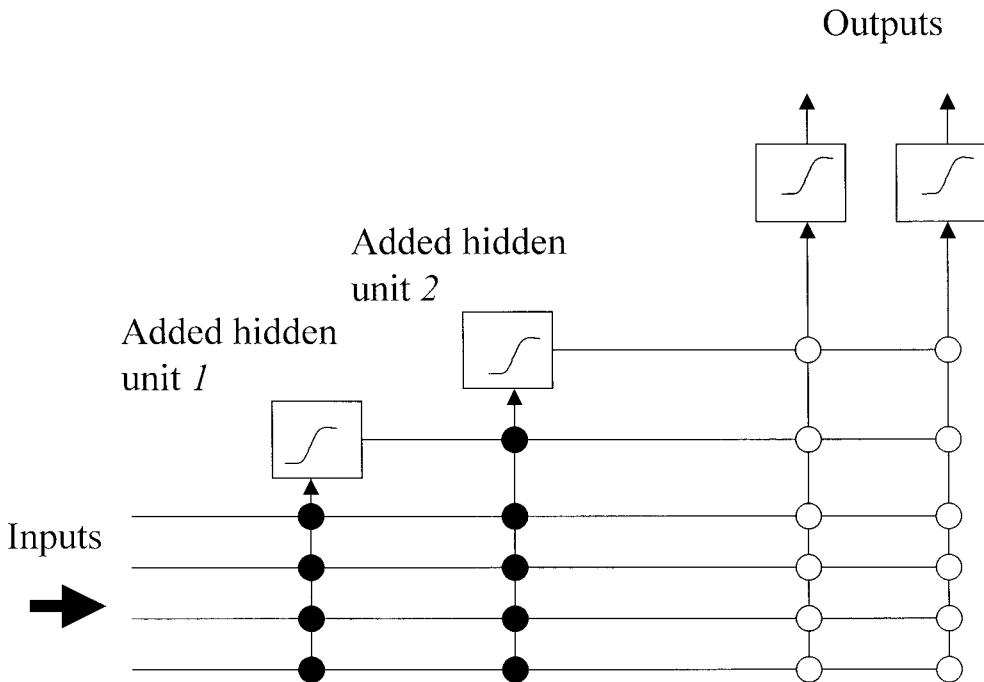


FIG. 3.2. Illustration of the Cascade network architecture. The vertical lines sum all incoming activation. Connections depicted with solid circles are frozen, while connections with hollow circles are trained repeatedly.

learning method, such as Equation 6 or Equation 7, without the need of back-propagating the errors. If the network performance is not satisfactory, then a first hidden node is added to improve the network approximation performance or to reduce the approximation error. When this is done, the weights leading up to outputs directly from inputs will remain unchanged; only the weight between the added hidden node and the outputs are updated by a supervised learning algorithm. The nodes can be added one at a time until a final error performance has been achieved.

Neural network models can handle both categorical and continuous variables. This is an advantage over discriminant analysis. However, decision tree works very well with categorical variables. In dealing with complex joint regions such as a half-torus, the neural network model is the best because it can form any complex shape due to its nonlinear function approximation capability. However, neural network models, for example those in available software packages, usually do not have functions to handle missing values in the program. Users must perform preprocessing on their own. Decision trees are easier when it comes to interpreting the classification results. MFN or discriminant analysis are analytical tools and, therefore, not best suited for visually interpreting the results. However, the self-organizing map (SOM), which will be introduced later in this chapter, does have strong visualization power, especially when the data is topologically related. The same characteristics will show in the output as well, and can be visualized. Neural networks can usually be trained continually using incremental learning algorithms. This is a clear advantage over other classification methods.

TABLE 3.1
Classes in the Texture Data Set

Code	Class
2	Grass lawn
3	Pressed calf leather
4	Handmade paper
6	Raffia looped to a high pile
7	Cotton canvas
8	Pigskin
9	Beach sand
10	Beach sand
12	Oriental straw cloth
13	Oriental straw cloth
14	Oriental grass fiber cloth

CASE STUDY 1—CLASSIFYING SURFACE TEXTURE

For this study the objects to be classified are all represented by continuous variables. The data were collected to classify textures of surfaces based on the estimation of fourth-order modified moments at four angles. The moments are estimated from a surface roughness scan at each of the angles, namely at 0, 45, 90, and 135 degree angles on the surface of the 11 textures. Table 3.1 contains descriptions of each of the texture classes. Note that Classes 9 and 10 are the same and Classes 12 and 13 are the same, but all 14 classes are included and used in the study. These moments are used as the explanatory variables for each of the methods. Specifically, the fourth-order moments are calculated using Equation 21:

$$m_4(i_1, i_2, i_3) = \frac{1}{M} \sum_{i=0}^{M-1-i_1} x_i x_{i+i_1} x_{i+i_2} x_{i+i_3}, \quad (21)$$

where $0 \leq i_1 \leq i_2 \leq i_3 \leq M - 1$ and $M = 3$.

There are 10 fourth-order moments at each angle due to the ordering requirement ($i_1 \leq i_2 \leq i_3$) in forming the moments, which creates a total of 40 explanatory variables. The original database consisted of 5,500 objects along with the associated class of each object. There were 11 classes of objects and 500 objects in each class in the original database. This particular database was specifically selected because of the large number of observations, explanatory variables, and classes.

Experimental Conditions

Classification methods were compared with 24 data sets that differed in certain parameters. The parameters studied were (a) number of objects in the training set, (b) number of classes for the dependent variable, and (c) number of explanatory variables. The levels selected for each of these parameters are shown in Table 3.2. A full-factorial experiment with three replicates was used. The data were selected for each condition in the following manner.

1. The records in the data set were sorted by class. Depending on the experimental conditions, either the first 5 or the first 10 classes were selected from the data set.
2. The resulting data set was randomized, and the number of objects to be used for the training was selected from the data set.

TABLE 3.2
Summary of Experimental Conditions

Factor	Levels
Analysis method	Linear discriminant analysis
	Quadratic discriminant analysis
	Neural networks
	Multiple-decision tree (TREEDISC)
	Single-decision tree
Number of objects	200, 500, 1,000, 1,500
Number of classes	5, 10
Number of explanatory variables	10, 20, 30

3. From the remaining objects a random selection of exactly half the size of the training set was selected as the test set. Thus, the training set ranged from 200 to 1,500 and the testing set ranged from 100 to 750. The rest of the data was ignored for that experimental condition.

4. The explanatory variables were not selected at random. The first 10, 20, or 30 variables were used in the analysis depending on the experimental conditions. A random selection would lead to different variables in each replicate. If certain variables had greater discriminating power, the variation in the replicates would increase just based on the selection (or elimination) of that variable.

5. Each of these 72 training sets was run through each of the methods, and the classification error rate was recorded on the training and testing data sets. The same settings were used for all sets of data. No attempt was made to optimize each training set by adjusting the parameters of the algorithm. For example, the neural network algorithm supported several different methods for variable transformation, noise estimation, and other algorithm parameters. These default values were used for all analysis, rather than searching for the best combination for each experimental run. Thus, we are comparing error rates under similar conditions rather than optimum error rate conditions.

6. The test data for each experimental condition were classified using the trained model, and the error rates were recorded.

Quantitative Comparison Results of Classification Methods

The test set error rates were analyzed in the following.

Figure 3.3 contains the graphical summaries of different error rates. There are two rows for each comparison. The three graphs in the first row summarize 5 classes and the second row summarizes 10 classes. The *x*-axis represents the number of data points and the *y*-axis represents the error rates. There are five series on each graph, one for each of the different methods.

Comparison of the test set error rates evaluates the classifier with new data from the same population. This is our ultimate interest. Based on Fig. 3.3, the largest effect appears to be the number of categories being classified. As the categories increase, the error rates increase. The results are summarized in Table 3.3.

In general, the decision trees performed the worst, neural networks were in the middle, and discriminant performed the best. The results are summarized in Table 3.3. The single decision tree algorithm used in our study was designed for categorical variables. A tree for continuous variables might have been more efficient.

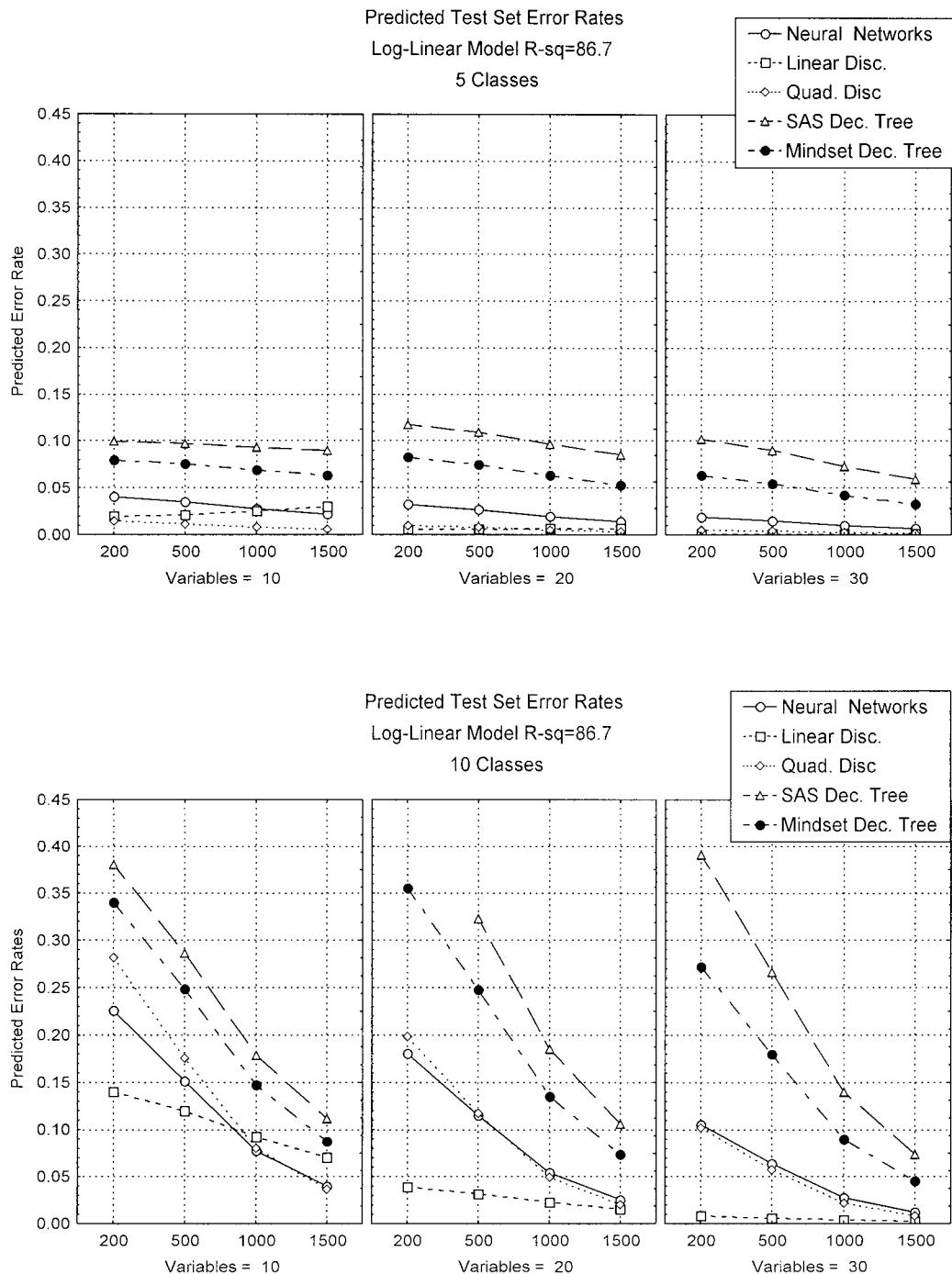


FIG. 3.3. Summary charts for test set MSE.

TABLE 3.3
Summary of Test Set Errors

Method	Comments
Neural network	As seen in Fig. 3.3, neural networks were comparable to and sometime better than discriminant analysis when the model was trained on 1,000 to 1,500 cases.
Linear discriminant	Linear discriminant performed well for all the conditions.
Quadratic discriminant	Quadratic discriminant analysis in general performed well for all the conditions, but had very high error rates when 10 categories were classified and the number of records was 200.
Mind-set multiple decision tree	Performed the worst at all the conditions.
SAS single decision tree	Was slightly better than multiple decision trees but not nearly as well a discriminant analysis.

Closing Discussions on Case 1

It is important to note that any conclusions drawn from this case study apply only to situations with continuous explanatory variables. For the specific data set used in comparisons, discriminant analysis worked best for almost all conditions considered here. Both of the decision tree methods performed much worse than all of the other methods. Most likely this was caused by the decision tree method of splitting the dimensions perpendicular to an axis.

An interesting result is that the neural networks worked nearly as well as, and in some cases outperformed, other methods. Neural networks require none of the multivariate distribution assumptions of discriminant analysis and generalize better for unusually shaped regions. Based on Fig. 3.3, under the worst condition of high numbers of classes and low number of variables, the neural network approach performed better than the other methods. However, it did require a high number (1,000) of data points before the advantage was seen. Even though the data distribution studied here seems to fit the conditions of discriminant analysis, the lack of assumptions and the ability to handle highly nonlinear classes makes neural networks a robust method that should perform well under a wide variety of conditions.

INTRODUCTION TO SOM

The idea of SOM in artificial neural networks may be traced back to the early work of von der Malsburg (von der Malsburg, 1973; Willshaw & von der Malsburg, 1976) in the 1970s on the self-organization of orientation sensitive nerve cells in the striate cortex. In 1976, Willshaw and von der Malsburg published the first paper on the formation of SOMs on biological grounds to explain retinotopic mapping from the retina to the visual cortex (in higher vertebrates). But it was not until the publication of Kohonen's (1982) paper on the SOM in 1982 that the SOM emerged as an attractive tool to model biological systems. Kohonen's model uses a computational shortcut to mimic basic functions similar to biological neural networks. The SOM was used to create an "ordered" map of input signals based on (a) the internal structure of the input signals themselves and (b) the coordination of the unit activities through the lateral connections between the units. Many implementation details of biological systems were "ignored" in Kohonen's SOM model. Instead, the SOM was an attempt to develop a model based on heuristically conceived, although biologically inspired, functional structure.

The SOM learning algorithm is a statistical data modeling tool with two distinct properties: (a) clustering of multidimensional input data and (b) spatial ordering of the output map so

that similar input patterns tend to produce a response in units that are close to each other in the output map. Although it performs the statistical data analysis function, the SOM is also a convenient tool for visualizing results from multidimensional data analysis.

Today, the SOM has found widespread applications in various areas (Chen & Liu, 2000; Davis & Si, 2001; Douzono, Hara, & Noguchi, 2000; Garavaglia, 2000; Hoglund, Hatonen, & Sorvari, 2000; Kohonen et al., 2000; Lin, Si, & Schwartz, 1997; Yanez-Suarez & Azimi-Sadjadi, 1999; Zhang, Fu, Yan, & Jabri, 1999; Zhu, Ramsey, & Chen, 2000). This section uses the SOM technique to decode neural spike trains recorded from a multichannel electrode array implanted in a monkey's motor cortex. The remainder of the chapter focuses explicitly on applying the SOM to simultaneous multichannel recording—from preparing the data to selecting the appropriate algorithm parameters.

The SOM Algorithm

To begin with the introduction of SOM, the concept of an “undirected graph” is needed. An undirected graph G is a pair $G = (R_g, E_g)$, where $R_g = \{r_1, r_2, \dots, r_L\}$ is a finite set of nodes or vertices in an m dimensional space with $r_i \in \mathcal{R}^m, i = 1, 2, \dots, L$. A node in G can be represented by its index i or its position r_i . E_g is a subset of the undirected edges, $E_g \subset \{[r_i, r_j] \mid r_i, r_j \in R_g \text{ and } i \neq j\}$. If $e_{ij} = [r_i, r_j] \in E_g$, then we say that node r_i is adjacent to node r_j (and vice versa) in G .

SOM is intended to learn the topological mapping $f : \mathcal{X} \subset \mathfrak{N}^n \rightarrow G \subset \mathfrak{N}^m$ by means of self-organization driven by samples X in \mathcal{X} , where G is a graph that determines the output map structure containing a set of nodes, each representing an element in the m -dimensional Euclidean space. Let $X = [x_1, x_2, \dots, x_n]^T \in \mathcal{X}$ be the input vector. It is assumed to be connected in parallel to every node in the output map. The weight vector of node i is denoted by $W_i = [w_{i1}, w_{i2}, \dots, w_{in}]^T \in \mathfrak{N}^n$ (see Fig. 3.4).

In Kohonen's SOM algorithm the graph (output map) is usually prespecified as a one-dimensional chain or two-dimensional lattice. A node in G can be represented by its index i or its position r_i . In this chapter we assume that only a one-dimensional chain with L nodes or a two-dimensional lattice with $L = L_x \times L_y$ nodes is used as an output map. Therefore, if the chain is used as the output map, we let $r_i = i, i = 1, 2, \dots, L$. If the lattice is used as an output map, we let $r_i = (i_x, i_y)$.

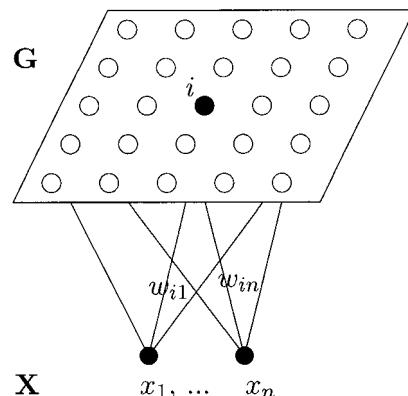


FIG. 3.4. The self-organizing map architecture.

SOM Building Blocks

There are two important steps in the self-organization algorithm:

1. Selecting neighborhood: find the activity bubble.
2. Adaptive process: update the weight vectors using the Hebbian learning law.

A computationally efficient procedure to accommodate the two steps can be elaborated as follows.

Find the Winner and the Neighborhood of the Winner. Find the output node with the largest response to the input vector. This can be done by simply comparing the inner products $W_i^T X$ for $i = 1, 2, \dots, L$ and selecting the node with the largest inner product. This process has the same effect in determining the location where the activity bubble is to be formed. If the weight vectors W_i are normalized, the inner product criterion is equivalent to the minimum Euclidean distance measure. Specially, if we use the index $c(X)$ to indicate the output node at which the weight vector “matches” the input vector X the best, we may then determine $c(X)$ by applying the following condition.

$$c(X) = \arg \min_i \|X - W_i\|, \quad i = 1, 2, \dots, L \quad (22)$$

where $\|\cdot\|$ denotes the Euclidean norm of the argument. In this chapter we use the Euclidean norm as the metric unless a specification is made. In most cases the Euclidean metric can be replaced by others with different characteristics.

The activity bubble is implemented by a topological neighborhood N_c of the node c . The neighborhood N_c depends on the output map structure used in the SOM.

Adaptive Process. The weight vectors inside the neighborhood of the winner are usually updated by Hebbian type learning law. Oja’s rule states that (Oja, 1982):

$$\frac{dW_i(t)}{dt} = \alpha(t)y_i(t)\{X(t) - y_i(t)W_i(t)\}, \quad i = 1, 2, \dots, L \quad (23)$$

where the negative component is a nonlinear *forgetting term*. We may further simplify Equation 23 by assigning the output y_i a binary value. $y_i(t) = 1$ if the node i is inside the activity bubble, that is, inside the neighborhood of the winner c ; $y_i(t) = 0$ otherwise. Therefore, we can rewrite Equation 23 as follows:

$$\frac{dW_i(t)}{dt} = \begin{cases} \alpha(t)(X(t) - W_i(t)) & \text{for } i \in N_c \\ 0 & \text{otherwise.} \end{cases} \quad (24)$$

The most convenient and widely accepted SOM algorithm was obtained in discrete-time format by Kohonen (1982, 1995) as follows:

$$W_i(k+1) = \begin{cases} W_i(k) + \alpha(k)(X(k) - W_i(k)) & \text{for } i \in N_c \\ W_i(k) & \text{otherwise.} \end{cases} \quad (25)$$

Implementation of the SOM Algorithm

The essence of Kohonen's SOM algorithm is to take a computational “shortcut” to achieve the effect accomplished by the typical “Mexican hat” lateral interactions. What follows is a step-by-step learning procedure for Kohonen's SOM algorithm.

Step 1. Initialize the weights by random numbers,

$$W_i(0) = (w_{i1}(0), w_{i2}(0), \dots, w_{in}(0)) \in \Re^n, \quad i = 1, 2, \dots, L.$$

Step 2. Draw a sample X from the input distribution.

Step 3. Find the winner W_c (best-matching node) using the minimum Euclidean distance criterion:

$$c(X) = \arg \min_i \|X(k) - W_i\|, \quad i = 1, 2, \dots, L. \quad (26)$$

Step 4. Update the winner and its neighbors by

$$W_i(k+1) = W_i(k) + \alpha(k) \Lambda(i, c)[X(k) - W_i(k)], \quad (27)$$

where $k = 0, 1, \dots$ denotes the discrete time steps, $\alpha(k)$ is the learning rate, and $\Lambda(i, c)$ is the neighborhood function of the winner.

Step 5. Compute $E_k = \sum_i \|W_i(k+1) - W_i(k)\|$, if $E_k \leq \epsilon$ stop; else repeat from step 1.

In this procedure $\Lambda(i, c)$ is a neighborhood function. $\Lambda(i, c)$ equals 1 for $i = c$ and falls off with the distance $\|r_c - r_i\|$ between node i and the winner c in the output layer, where r_c and r_i denote the coordinate positions of the winner c and node i in the output layer, respectively. Thus, those nodes close to the winner, as well as the winner c itself, will have their weights changed appreciably, whereas those farther away, where $\Lambda(i, c)$ is small, will experience little effect. It is here that the topological information is supplied: nearby nodes receive similar updates and thus end up responding to nearby input patterns. In the original learning algorithm proposed by Kohonen (1988) the neighborhood function $\Lambda(i, c)$ is defined as

$$\Lambda(i, c) = \begin{cases} 1 & \text{for } \|r_i - r_c\| \leq N_c(k) \\ 0 & \text{otherwise} \end{cases} \quad (28)$$

where $N_c(k)$ is some decreasing function of time. The value of $N_c(k)$ is usually large at the beginning of learning and then shrinks during training. We call this neighborhood function a square neighborhood function.

The bell-shaped neighborhood function (Kohonen, 1990) also is used frequently in practice:

$$\Lambda(i, c) = \exp(-\|r_i - r_c\|^2 / 2\sigma^2(k)) \quad (29)$$

where $\sigma(k)$ is the width parameter that affects the topology order in the output map and is gradually decreasing during training.

The learning rate $\alpha(k)$ in the learning algorithm is essential for convergence. The learning rate $\alpha(k)$ should be large enough so that the network could adapt quickly to the new training

patterns. On the other hand, $\alpha(k)$ should be small enough so that the network would not forget the experience from the past training patterns. For analytical purposes $\alpha(k)$ could be chosen to satisfy conditions in the Robbins-Monro algorithm (Lin & Si, 1998; Robbins & Monro, 1951). In the update scheme shown above, the winner and its neighborhood is simply leaning toward the current input pattern by moving along the vector $(X(k) - W_i)$ that pushes W_i toward X . The amount of adjustment of W_i depends on the value of the learning rate parameter $\alpha(k)$, which varies from 0 to 1. If $\alpha(k) = 0$, there is no update; when $\alpha(k) = 1$, W_c becomes X .

For effective global ordering of the output map, it has been observed experimentally to be advantageous to let the width parameter σ be very large in the beginning and gradually decrease during the learning process. As a matter of fact, it was shown in Lin & Si (1998) that if the range of the neighborhood function covers the entire output map, then each weight vector converges to the same stationary state, which is the mass center of the training data set. This implies that if we want to eliminate the effect of the initial conditions, we should use a neighborhood function covering a large range of the output map. On the other hand, if the range of the neighborhood function becomes 0, that is, $N_c(k) = 0$, the final iterations of the SOM algorithm may be viewed as a sequential updating process of vector quantization. It is interesting to note that even though the convergence arguments are made based on the Robbins-Monro algorithm, there have been no claims about topological ordering of the weight vectors. It remains a well-observed practice in many applications.

CASE STUDY 2—DECODING MONKEY'S MOVEMENT DIRECTIONS FROM ITS CORTICAL ACTIVITIES

A rhesus monkey was trained to trace with its index finger on a touch-sensitive computer monitor. In the spiral tracing task the monkey was trained to make a natural, smooth, drawing movement within approximately a 1-cm band around the presented figure. As the figure appeared, the target circle of about 1-cm radius moved a small increment along the spiral figure, and the monkey moved its finger along the screen surface to the target. The spirals we study in this chapter are outside→in spirals. A more detailed description of this process can be found in Schwartz (1994).

While the well-trained monkey was performing the previously learned routine, the occurrence of each spike signal in an isolated cell in the motor cortex was recorded. The result of these recordings is a spike train at the following typical time instances,

$$\tau_1, \tau_2, \dots, \tau_L.$$

This spike train can be ideally modeled by a series of δ functions at the appropriate times:

$$S_c(t) = \sum_{l=1}^L \delta(t - \tau_l) \quad (30)$$

Let Ω be the collection of the monkey's motor cortex cells the activities of which contribute to the monkey's arm movement. Realistically, we can record only a set of sample cells $S = \{S_i \in \Omega, i = 1, 2, \dots, n\}$ to analyze the relation between discharge rate patterns and the monkey's arm movements. We assume that the cells included in the set S constitute a good representation of the entire population. The firing activity of each cell S_i defines a point process $f_{s_i}(t)$. Let d_{ij} denote the average discharge rate for the i th cell at the j th interval $(t_j, t_j + dt)$. Taking d_{ij} of all

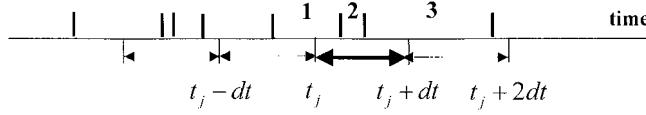


FIG. 3.5. Calculation of d_{ij} : the number of spike intervals overlapping with the j th time interval is determined to be 3. As shown, $2/3$ of the first interval, 100% of the second interval, and $1/4$ of the third interval are located in the j th time window. Therefore, $d_{ij} = \frac{\frac{2}{3} + 1 + \frac{1}{4}}{dt}$.

the cells $\{S_i \in \Omega, i = 1, 2, \dots, n\}$ in $(t_j, t_j + dt)$, we obtain a vector $X_j = [d_{1j}, d_{2j}, \dots, d_{nj}]$, which will later be used as the feature input vector relating to the monkey's arm movement during the time interval $(t_j, t_j + dt)$.

The average discharge rate $d_{ij} = \frac{d_{ij}}{dt}$ is calculated from the raw spike train as follows. For the j th time interval $(t_j, t_j + dt)$, one has to first determine all the intervals between any two neighboring spikes that overlap with the j th time interval. Each of the spike intervals overlapping completely with $(t_j, t_j + dt)$ contributes a 1 to D_{ij} . Each of those only overlapping partially with $(t_j, t_j + dt)$ will contribute the percentage of the overlapping to D_{ij} . An illustrative example is shown in Fig. 3.5.

Due to limitations in experimental realization, the firing activities of the n cells in S could not be recorded simultaneously. Consequently, the average discharge rate vector X_j could not be calculated simultaneously at time t_j . However, special care was given to ensure that the monkey repeated the same experiment n times to obtain the firing signal for all the n cells under almost the same experimental condition. We thus assume that the experiments of recording every single cell's spike signal were identically independent events, so we could use spike signals of the n cells as if they were recorded simultaneously.

In our simulations, spike signals from 81 cells in the motor cortex were used. The average discharge rates were calculated every 20 milliseconds as previously described. The entire time spans of the trials may vary. Cubic interpolations were used to normalize the recorded data (average discharge rates and movement directions in degrees) to 100 time bins. At a given bin, normalized directions (in degrees) were averaged across 81 cells, giving rise to a finger trajectory sequence: $\{\angle_1, \angle_2, \dots, \angle_{100}\}$. Correspondingly, at every time bin j , we have a normalized discharge rate vector $X_j = [d_{1j}, d_{2j}, \dots, d_{81j}]^T$, $j = 1, 2, \dots, 100$. Discharge rate vectors and moving directions were used to train and label the SOM.

Trajectory Computation from Motor Cortical Discharge Rates

The major objective of this study was to investigate the relationship between the monkey's arm movements and firing patterns in the motor cortex. The topology preserving property of the SOM is used to analyze real recorded data of a behaving monkey.

To train the network, we select n cells (in this study, $n = 81$) in the motor cortex. The average discharge rates of these n cells constitute a discharge rate vector $X_j = [d_{1j}, d_{2j}, \dots, d_{nj}]^T$, where $d_{ij}, i = 1, \dots, n$, is the average discharge rate of the i th cell at time bin t_j . A set of vectors $\{X_k | k = 1, 2, \dots, NP\}$ recorded from experiments are the training patterns of the network, where NP is the number of training vectors.

During training, a vector $X \in \{X_k | k = 1, 2, \dots, NP\}$ is selected from the data set at random. The discharge rate vectors are not labeled or segmented in any way in the training phase: all the features present in the original discharge rate patterns will contribute to the self-organization of the map. Once the training is done, as described previously, the network has learned the classification and topology relations in the input data space. Such a "learned"

network is then calibrated using the discharge rate vectors of which classifications are known. For instance, if a node i in the output layer wins most for discharge rate patterns from a certain movement direction, then the node will be labeled with this direction. In this part of the simulation we used numbers 1 ~ 16 to label the nodes in the output layer of the network, representing the 16 quantized directions equally distributed around a circle. For instance, a node with label “4” codes the movement direction of 90 degrees. If a node in the output layer of the network never wins in the training phase, we label the node with number “−1.” In the testing phase a “−1” node can become a “winner” just as those marked with positive numbers. Then the movement direction is read out from its nearest node with a positive direction number.

Figure 3.6 gives the learning results using discharge rates recorded from the 81 cells in the motor cortex in one trial of the spiral task. From the output map we can clearly identify three circular patterns representing the spiral trajectory. These results show that monkey’s arm movement directions were clearly encoded in firing patterns of the motor cortex. By the property of topology preserving, we can distinguish a clear pattern from the locations of the nodes in the map. Note that nodes with the same numbers (i.e., directions) on different spiral cycles are adjacent. This is the same for the actual trajectory and is what we mean by topology preserving, an attribute of the SOM. Different cycles of the spiral are located in correspondingly different locations of the feature map, suggesting that parameters in addition to directions influence cortical discharge rates.

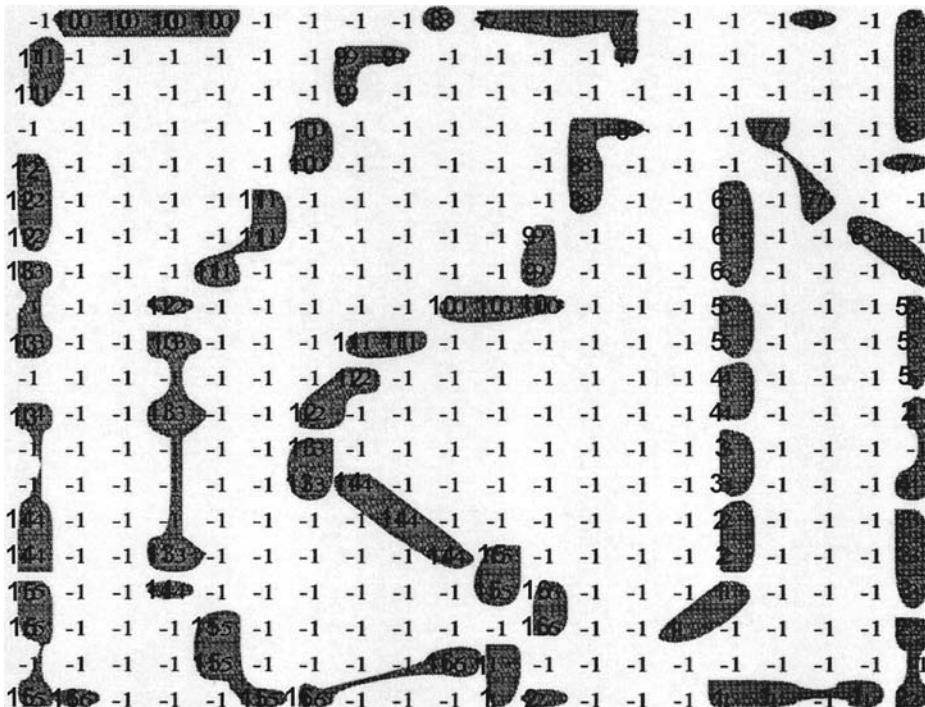


FIG. 3.6. The SOM for the discharge rates of the 81 neurons in the motor cortex in the spiral task. The size of the output map of the SOM is 20 by 20. The nodes marked with positive numbers are cluster “centers” representing the directions as labeled, while the nodes labeled with “−1” are the “neighbors” of the “centers.” The closer it is to the “center” the more possible it represents the coded direction of the “center.”

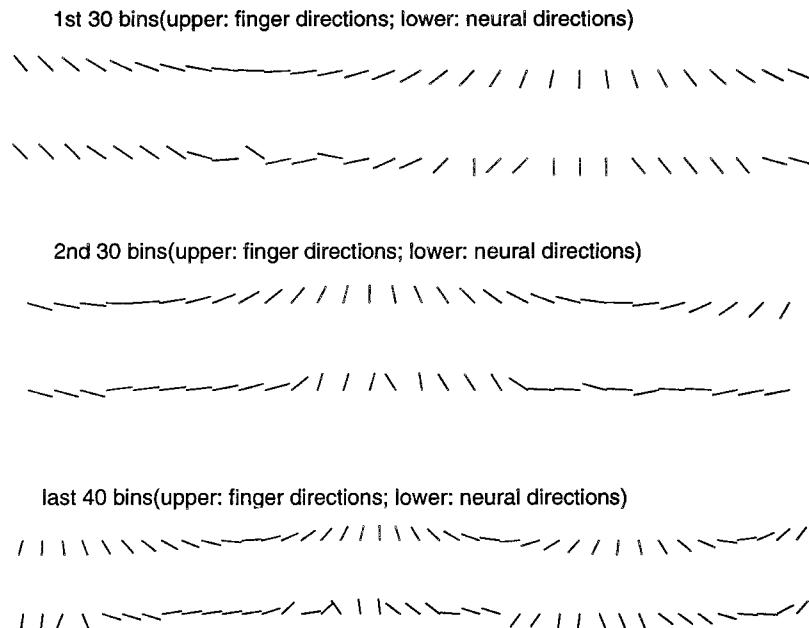


FIG. 3.7. The “neural directions” and finger directions in 100 bin series in the inward spiral tracing task.

Using Data from Spiral Tasks to Train the SOM

In this experiment we used data recorded from five trials in spiral tracing tasks. The first trial was used for testing and the rest for training. After the training and labeling procedure, every node in the output map of the SOM was associated with one specific direction. The direction of a winner was called “neural direction” when discharge rate vectors were presented as the input of the network bin by bin in the testing phase. Figure 3.7 gives “neural directions” against corresponding monkey’s finger movement directions.

After postfiltering the predicted “neural directions” with a moving average filter of three-bin width, we combined the neural directions tip-to-tail bin by bin. The resulting trace (“neural trajectory”) is similar to the finger trajectory. Note that the “neural directions” are unit vectors. To use vectors as a basis for trajectory construction, the vector magnitude must correspond to movement speed. During drawing, there is a “ $2/3$ power law” (Schwartz, 1994) relation between curvature and speed. Therefore, the speed remains almost constant during the spiral drawing. This is the reason we can use unit vectors in this construction and still get a shape that resembles the drawn figure. Figure 3.8 demonstrates the monkey’s complete continuous finger movement and the predicted neural trajectory.

Using Data from Spiral and Center→Out Tasks to Train the SOM

In addition to the spiral data used above, the data recorded in the center→out tasks were added to the training data set. Figure 3.9 shows that the overall performance of the SOM has been improved. This result suggests that the coding of arm movement direction is consistent across the two different tasks.

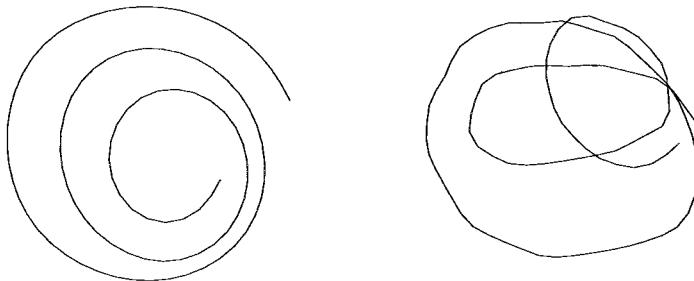


FIG. 3.8. Using data from spiral task for training. Left: the monkey's finger movement trajectory calculated with constant speed. Right: the SOM neural trajectory.

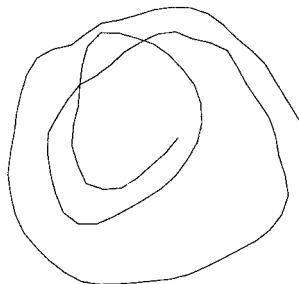


FIG. 3.9. Neural trajectory: using data from spiral and center→out tasks for training.

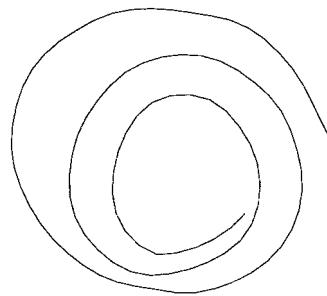


FIG. 3.10. Neural trajectory: average testing result using leave-k-out method.

Average Testing Result Using the Leave-K-Out Method

The data recorded from five trials in the spiral tasks and the center→out tasks were used in this experiment. Every time we chose one of the trials in spiral tasks for testing and the rest for training. We have a total of five different sets of experiments for training and testing the SOM. Note that the training and testing data were disjoint in each set of experiments. The neural trajectories from the five tests were averaged; Fig. 3.10 shows the averaged result. Due to the nonlinear nature of the SOM network a different training data set may result in a different outcome in the map. Therefore, the leave-k-out method provides us with a realistic view of the overall performance of the SOM. Figure 3.11 shows the bin-by-bin difference between the

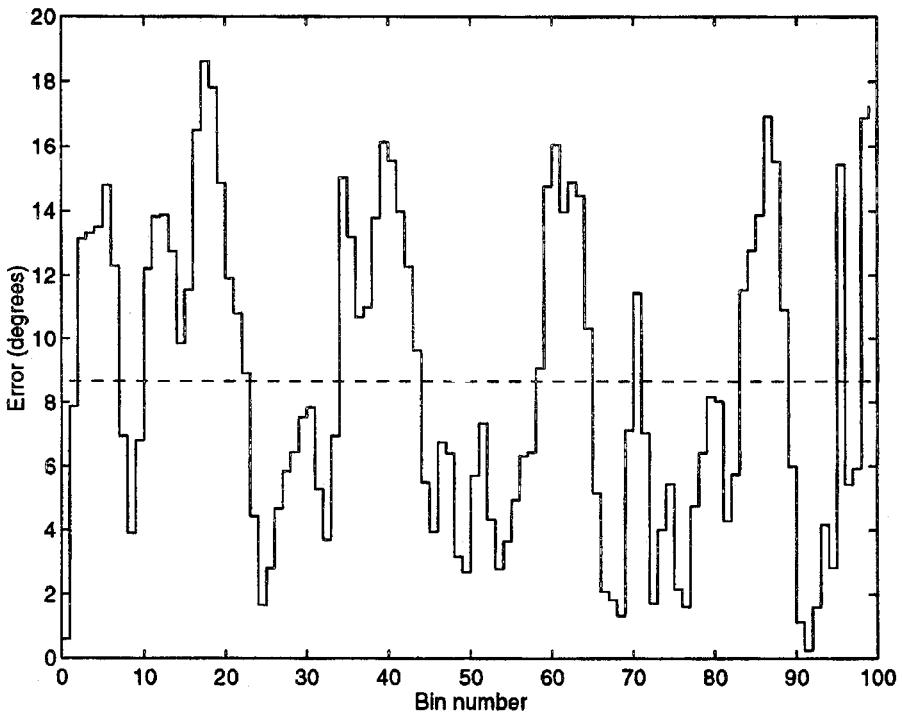


FIG. 3.11. The binwise difference between the finger directions and the neural directions in 100 bins. The dashed line is the average error in 100 bins. The neural trajectory is the average testing result of the SOM using the leave-k-out method.

neural directions and the finger directions in 100 bins. The dashed line in the figure represents the average error in 100 bins.

Closing Discussions on Case 2

Our results based on raw data recorded from a monkey's motor cortex have revealed that the monkey's arm trajectory was encoded in the averaged discharge rates of the motor cortex. Note from Fig. 3.6 that when the monkey's finger was moving along the same direction on different cycles of the spiral, there are distinct cycles on the output map as well. By the topology preserving property of the SOM this suggests that the output nodes decode not only directions but also probably speed, curvature, and position. Although the population vector method has previously shown this information to be present in motor cortex, the SOM model is able to capture regularities about speed and curvature without assuming any linear functional relationship to discharge rates as the population vector and OLE method do in Equations 2 and 3, respectively. Although we did not explicitly examine speed in this study, it has been shown that the population vector magnitude is highly correlated with speed (Schwartz, 1994), showing that this parameter is represented in motor cortical activity. For those drawing movements with larger changes of speed we may quantize the speed as we did with directions. As we have discussed earlier, the direction and the curvature or speed may be encoded in discharge rates as a feature unit, and the nodes in the output map can be labeled using directions and

speed together. Consequently, the same process of direction recognition using the SOM could be applied to speed recognition based on discharge rates of the motor cortex. Other types of networks, such as the sigmoid feedforward and radial basis function networks, could have been used to associate discharge rates with movement directions. However, as discussed previously, the SOM offers a unique combination of both data association (which some other networks can achieve) and topology information as represented by the two-dimensional output map (which is hard or impossible for other networks to implement).

FINAL CONCLUSIONS AND DISCUSSIONS

Two fundamentally important neural network models were introduced. To demonstrate how to use these models and how useful these models are, two real-life, sophisticated case studies were carried out with all the pertinent details provided, from data preparation to results interpretation. Both case studies essentially demonstrate capabilities of artificial neural network models as classification tools. They have shown the flexibility of how neural networks can be used. More importantly, from Case 2 it is interesting to note that the SOM model is also a good visualization tool.

In general, these results indicate that to be successful in classification analysis, one should know and understand all potential methods. The selection of the appropriate method should be based on the domain knowledge of the user. It is critical to understand the assumptions, advantages, and disadvantages of each of these methods. If the assumptions behind a particular algorithm are met, the specific method is likely to work best. Neural network models usually work under fewer assumptions about the data distribution and work well when the data set has complex, highly nonlinear distributions.

Commercial software packages have been improved tremendously over the past decade, which has made the use of neural network modeling tools easier. Matlab produced by Mathworks is one such example that has almost all the important neural network models covered. There is a freeware from Finland (<http://www.cis.hut.fi/research/som-research/>) exclusively for the SOM model. Some variants of the SOM model are also available.

REFERENCES

- Battiti, R. (1992). First- and second-order methods for learning: Between steepest descent and Newton's method. *Neural Computation*, 4, 141–166.
- Chen, F.-L., & Liu, S.-F. (2000). A neural-network approach to recognize defect spatial pattern in semiconductor fabrication. *IEEE Transactions on Semiconductor Manufacturing*, 13, 366–373.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2, 303–314.
- Davis, R. G., & Si, J. (2001). Knowledge discovery from supplier change control data for purchasing management. In *International Conferences on Info-tech and Info-net* (pp. 67–72). New York: IEEE press.
- Dennis, J. E., & Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*. Englewood Cliffs, NJ: Prentice Hall.
- Douzono, H., Hara, S., & Noguchi, Y. (2000). A clustering method of chromosome fluorescence profiles using modified self organizing map controlled by simulated annealing. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 4, 103–106.
- Fahlmann, S. E., & Lebiere, C. (1990). *The cascade-correlation learning architecture*, (Tech. Rep. No. CMU-CS-90-100). Carnegie-Mellon University, Pittsburgh, PA.
- Forsyth, R., & Rada, R. (1986). *Machine learning: Applications in expert systems and information retrieval*. West Sussex: Ellis Horwood.

- Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2, 183–192.
- Garavaglia, S. B. (2000). Health care customer satisfaction survey analysis using self-organizing maps and “exponentially smeared” data vectors. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 41, 119–124.
- Hagen, M. T., & Menhaj, M. B. (1994). Training feedforward networks with the Levenberg-Marquardt algorithm. *IEEE Transaction on Neural Networks*, 5, 989–993.
- Haykin, S. (1999). *Neural networks: A comprehensive foundation* (2nd ed.). Englewood Cliffs, NJ: Prentice Hall.
- Hertz, J., Krogh, A., & Palmer, R. G. (1991). *Introduction to the theory of neural computation*. Redwood City, CA: Addison-Wesley.
- Hoglund, A. J., Hatonen, K., & Sorvari, A. S. (2000). A computer host-based user anomaly detection system using the self-organizing map. *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 5, 411–416.
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multi-layer feedforward network are universal approximators. *Neural Networks*, 2, 359–366.
- Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59–69.
- Kohonen, T. (1988). *Self-organization and associative memory* (2nd ed.). Berlin: Springer-Verlag.
- Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78, 1464–1480.
- Kohonen, T. (1995). *Self-organizing map*. Heidelberg, Germany: Springer-Verlag.
- Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V., & Saarela, A. (2000). Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11, 574–585.
- Kolllias, S., & Anastassiou, D. (1989). An adaptive least squares algorithm for the effective training of artificial neural networks. *IEEE Transactions on Circuits and Systems*, 36, 1092–1101.
- Lin, S., & Si, J. (1998). Weight value convergence of the SOM algorithm. *Neural Computation*, 10, 807–814.
- Lin, S., Si, J., & Schwartz, A. B. (1997). Self-organization of firing activities in monkey’s motor cortex: Trajectory computation from spike signals. *Neural Computation*, 9, 607–621.
- Liu, B., & Si, J. (1994). The best approximation to C^2 functions and its error bounds using regular-center Gaussian networks. *IEEE Transactions on Neural Networks*, 5, 845–847.
- Narendra, K. S., & Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1, 4–27.
- Oja, E. (1982). A simplified neuron model as a principle component analyzer. *Journal of Mathematical Biology*, 15, 267–273.
- Park, J., & Sandberg, I. W. (1991). Universal approximation using radial basis function networks. *Neural Computation*, 3, 246–257.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufman. San Francisco.
- Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Annals Mathematical Statistics*, 22, 400–407.
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by backpropagating errors. *Nature*, 323, 533–536.
- Schwartz, A. B. (1994). Direct cortical representation of drawing. *Science*, 265, 540–542.
- von der Malsburg, C. J. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kebernetik*, 14, 85–100.
- Werbos, P. J. (1993). Supervised learning: Can it escape its local minimum? *Proceedings of the 1993 World Congress on Neural Networks III*, 358–363. New York: IEEE.
- Werbos, P. J. (1994). How can we cut prediction error in half by using a different training method? *Proceedings of the 1994 World Congress on Neural Networks II*, 225–230. New York: IEEE.
- White, D. A., & Sofge, D. A. (Eds.). (1992). *Handbook on intelligent control: Neural, fuzzy, and adaptive approaches*. New York: Van Nostrand-Reinhold.
- Willshaw, D. J., & von der Malsburg, C. (1976). How patterned neural connections can be set up by self-organization. *Proceeding of the Royal Society of London, Series B*, 194, 431–445.
- Yanez-Suarez, O., & Azimi-Sadjadi, M. R. (1999). Unsupervised clustering in Hough space for identification of partially occluded objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21, 946–950.
- Zhang, B., Fu, M., Yan, H., & Jabri, M. A. (1999). Handwritten digit recognition by adaptive-subspace self-organizing map (ASSOM). *IEEE Transactions on Neural Networks*, 10, 939–945.
- Zhu, B., Ramsey, M., & Chen H. (2000). Creating a large-scale content-based airphoto image digital library. *IEEE Transactions on Image Processing*, 9, 163–167.

4

Statistical Analysis of Normal and Abnormal Data

Connie M. Borror
Arizona State University

Introduction	67
Univariate Control Charts	68
Variables Control Charts	68
Attributes Control Charts	81
Cumulative Sum Control Charts	89
Exponentially Weighted Moving Average Control Charts	93
Choice of Control Charting Techniques	95
Average Run Length	96
Multivariate Control Charts	98
Data Description	98
Hotelling T^2 Control Chart	98
Multivariate EWMA Control Charts	101
Summary	102
References	102

INTRODUCTION

In many fields there is a need for distinguishing normal patterns and abnormal patterns of data, such as quality data for distinguishing in-control and out-of-control processes and computer security data for distinguishing normal and intrusive activities. Statistical process control (SPC) techniques are a set of well-established tools for mining data to uncover statistical patterns of normal and abnormal data. In this chapter SPC techniques for defining and mining normal and abnormal data patterns are described.

Statistical process control consists of several statistical methods that are useful in measuring, monitoring, and improving quality in a product or process. Control charts are the most common tools for monitoring quality and were first introduced by Walter A. Shewhart in the 1920s. Because of his pioneering work several univariate control charts are referred to as Shewhart control charts. The foundation of his work relied on the nature of variability present in a process.

Variability will always be present in a production process, regardless of how well maintained the process may be. The natural variability is often referred to as *inherent variability* and can be thought of as occurring by chance. If in a production process only *chance causes of variability* are present, the process is said to be operating in a state of statistical control. There are some situations in which variability is the result of some unusual occurrence or disturbance in the process. When *assignable causes of variability* are present in the process, we say that the process is out of control. Some examples would include machine wear out in a manufacturing process, operator errors, or defective materials. In computer systems, intrusion or anomaly detection can be thought of as an *assignable cause* of variability. An attack or intrusion would disrupt the normal pattern of the data, causing the variability in the system to increase. Control charts would be appropriate system-monitoring techniques in this situation.

In this chapter several univariate (monitoring a single quality characteristic of interest) control charts and multivariate (monitoring several quality characteristics simultaneously) control charts will be presented. Examples and recommendations are provided for the methods discussed. The reader is encouraged to examine Montgomery (2001) for more details, developments, and examples of each technique discussed in this chapter.

UNIVARIATE CONTROL CHARTS

Univariate control charts are used to monitor a single variable or attribute of interest. In this section univariate control charts for variables data and attributes data are presented. The control charts to be discussed are

- \bar{x} and R control charts,
- \bar{x} and S control charts,
- individuals control charts,
- fraction nonconforming control charts, and
- control charts for nonconformities.

For each Shewhart control chart presented in this section, the underlying assumption that must be satisfied is that the data being monitored follow a normal distribution (i.e., a bell-shaped curve). The Shewhart control charts are very sensitive to this assumption. If the normality assumption is violated, the overall performance of these charts can be very poor and result in incorrect signals. Abnormal patterns may not be detected quickly, if at all, while normal patterns may signal as abnormal.

Variables Control Charts

The most commonly used control charts for measurement or variables data are the Shewhart \bar{x} and R control charts and the \bar{x} and S control charts. The \bar{x} chart monitors the mean of the process, whereas the R chart and S chart monitor the process variability. The \bar{x} is used in conjunction with either the R chart or S chart for simultaneous monitoring of the process mean and process variability.

\bar{x} and R Control Charts

When a numerical measurement can be taken for a particular quality characteristic, the quality characteristic is referred to as a *variable*. The \bar{x} and R control charts are valuable statistical monitoring techniques for variables data and are often referred to as variables control charts.

Consider a variable that is known to be normally distributed with mean μ and standard deviation, σ (both of which are often unknown and must be estimated). Let x_1, x_2, \dots, x_n represent a sample of size n measurements taken on the variable of interest. The sample mean, \bar{x} , is then

$$\begin{aligned}\bar{x} &= \frac{x_1 + x_2 + \dots + x_n}{n} \\ &= \frac{\sum_{i=1}^n x_i}{n}\end{aligned}$$

where \bar{x} is also normally distributed with mean μ and standard error σ/\sqrt{n} (see Devore [2000] or Montgomery & Runger [2002]).

In practice, samples of size n may be taken at various time intervals. For example, in semiconductor manufacturing, wafer thickness is an important characteristic that must be monitored. The operator randomly selects a subgroup of wafers every 2 hours, for example, and measures the thickness. The sample average thickness can then be calculated for each sample, and the wafer thickness average can be monitored. If a sample mean were unusually large or small, this could indicate a problem with the process.

Suppose there are m subgroups each of size n chosen at random to monitor a particular process. General data is presented in Table 4.1. The sample mean and sample range are displayed in columns three and four, respectively. The sample range, R_i , is the difference between the maximum and the minimum of each sample and is often used to obtain an estimate of the sample variability.

The statistic $\bar{\bar{x}}$ is referred to as the grand average and is the best overall estimate of the true process mean, μ . \bar{R} is the average range and will be used to estimate the process variance and construct the necessary control charts.

Once the sample averages, sample ranges, grand average, and average range are calculated, the Shewhart control charts can be developed. The upper control limit (UCL), centerline (CL),

TABLE 4.1
General Notation for Subgroup Data

Subgroup i	Measurements	\bar{x}_i	R_i
1	$x_{11}, x_{21}, \dots, x_{n1}$	\bar{x}_1	R_1
2	$x_{12}, x_{22}, \dots, x_{n2}$	\bar{x}_2	R_2
3	$x_{13}, x_{23}, \dots, x_{n3}$	\bar{x}_3	R_3
\vdots	\vdots	\vdots	\vdots
m	$x_{1m}, x_{2m}, \dots, x_{nm}$	\bar{x}_m	R_m
$\bar{\bar{x}} = \frac{\sum_{i=1}^m \bar{x}_i}{m}$		$\bar{R} = \frac{\sum_{i=1}^m R_i}{m}$	

TABLE 4.2
Constants for Constructing Control Charts

Sample Size, <i>n</i> ^a	<i>A</i> ₂	<i>A</i> ₃	<i>B</i> ₃	<i>B</i> ₅	<i>B</i> ₆	<i>d</i> ₂	<i>D</i> ₂	<i>D</i> ₃	<i>D</i> ₄
2	1.88	2.659	0	3.267	2.606	1.128	3.686	0	3.267
3	1.023	1.954	0	2.568	2.276	1.693	4.358	0	2.575
4	0.729	1.628	0	2.266	2.088	2.059	4.698	0	2.282
5	0.577	1.427	0	2.089	1.964	2.326	4.918	0	2.115
6	0.483	1.287	0.030	1.97	1.874	2.534	5.078	0	2.004
7	0.419	1.182	0.118	1.882	1.806	2.704	5.204	0.076	1.924
8	0.373	1.182	0.185	1.815	1.751	2.847	5.306	0.136	1.864
9	0.337	1.099	0.239	1.761	1.707	2.970	5.393	0.184	1.816
10	0.308	1.032	0.284	1.716	1.669	3.078	5.469	0.223	1.777

^aFor sample sizes greater than 10, see Montgomery (2001) or the *NIST/SEMATECH Handbook* at <http://www.itl.nist.gov/div898/handbook>.

and lower control limit (LCL) for the \bar{x} control chart are

$$\begin{aligned} \text{UCL} &= \bar{x} + A_2 \bar{R} \\ \text{Centerline} &= \bar{x} \\ \text{LCL} &= \bar{x} - A_2 \bar{R} \end{aligned} \tag{1}$$

where A_2 is a constant that depends on the sample size, n . Values of A_2 are provided in Table 4.2 for various sample sizes. The UCL, CL, and LCL for the range are

$$\begin{aligned} \text{UCL} &= D_4 \bar{R} \\ \text{Centerline} &= \bar{R} \\ \text{LCL} &= D_3 \bar{R} \end{aligned} \tag{2}$$

where D_3 and D_4 are constants that depend on the sample size, n . Values of D_3 and D_4 are provided in Table 4.2 for various sample sizes.

The parameter values are very straightforward to compute by hand. The value d_2 is often referred to as the mean of the measure “relative range,” defined as R/σ , and is dependent on the sample size, n . The formula for A_2 is

$$A_2 = \frac{3}{d_2 \sqrt{n}}$$

which is dependent only on the sample size n and can be calculated for various values of n as given in Table 4.2. The parameter values, D_3 and D_4 , are defined as

$$D_3 = 1 - 3 \frac{d_3}{d_2} \quad \text{and} \quad D_4 = 1 + 3 \frac{d_3}{d_2}$$

where d_3 is a function of the standard deviation R . As with A_2 , D_3 and D_4 are also constants dependent on the sample size n . Formulas for each of the parameters displayed in Table 4.2

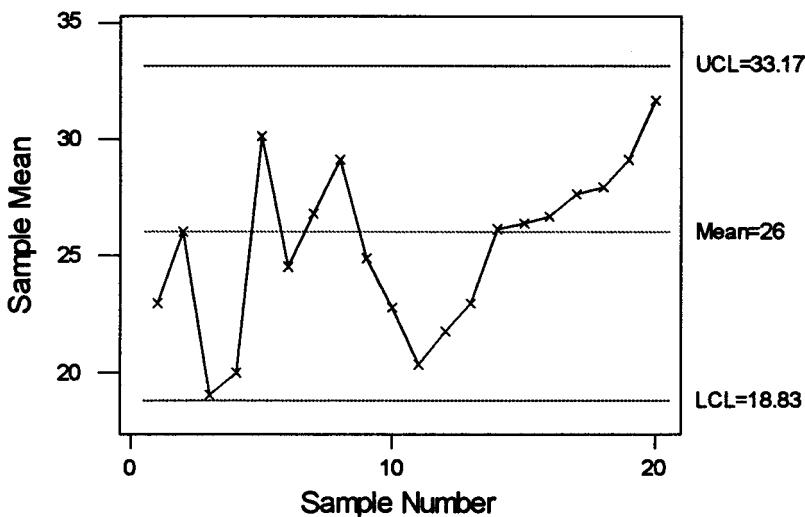


FIG. 4.1. An illustration of a standard control chart with an upward trend.

can be found in Montgomery (2001). The reader should also note the use of 3 (as in 3-sigma) in the formulas above and in many of the control limits to be discussed in this section. The use of three-sigma limits is a direct result of the underlying assumption of normality. It can be shown that if a dataset follows a normal distribution, then approximately 99.7% of all the data will lie within three standard deviations of the mean. Therefore, if an observation falls beyond three standard deviations from the mean, that data point is unusual (because the chance of this occurring is 0.3%) and may be a signal of an out-of-control situation in control charting. Similarly, it could be a signal of an intrusion in anomaly detection. Because the Shewhart control charts are based on the normality assumption, it is common to use three standard deviations in the construction of the control limits for these charts.

The control chart itself is created by plotting the subgroup averages (or ranges), LCL, centerline, and UCL against subgroup, m . If one or more sample average(s) (or ranges) plot beyond the control limits, then the process is considered out of control. Patterns or trends that are obvious over the subgroups are also an indication of an out-of-control process. To illustrate, consider the control chart displayed in Fig. 4.1. There is an obvious upward trend in the data starting with Observation 11 and continuing until the final observation (20). This trend indicates that the process is no longer producing random observations about some in-control mean, and the process should be investigated for an assignable cause.

Example 1. Consider the semiconductor wafer discussion given previously. Four wafers are randomly selected every 2 hours, and a particular thickness is measured. Twenty subgroups have been selected, and the results are given in Table 4.3. The engineer would like to monitor the mean wafer thickness using this data.

Solution. For this problem, $m = 20$ and $n = 4$. To construct the control charts, the sample mean and sample range for each subgroup are needed. The sample mean for each subgroup is simply the average of each subgroup. To illustrate, the sample mean for

TABLE 4.3
Wafer Thicknesses with the Sample Mean and Sample Range
for Each Subgroup

Subgroup <i>i</i>	Thickness				\bar{x}_i	R_i
1	15.4	14.1	12.7	13.8	14.000	2.7
2	15.3	14.2	14.1	14.8	14.600	1.2
3	15.5	15.0	17.0	14.5	15.500	2.5
4	15.7	15.1	15.8	15.1	15.425	0.7
5	14.7	16.8	14.1	15.0	15.150	2.7
6	14.3	15.8	13.2	14.5	14.450	2.6
7	14.3	15.8	14.2	13.9	14.550	1.9
8	14.4	15.3	15.7	13.7	14.775	2.0
9	14.2	16.0	14.4	15.1	14.925	1.8
10	13.2	14.9	16.2	14.8	14.775	3.0
11	15.4	15.6	15.0	15.1	15.275	0.6
12	13.9	14.0	15.3	15.1	14.575	1.4
13	15.2	13.4	15.1	14.2	14.475	1.8
14	17.7	15.6	15.7	16.2	16.300	2.1
15	13.5	14.8	15.5	16.1	14.975	2.6
16	14.3	14.0	15.9	14.3	14.625	1.9
17	14.3	16.2	15.8	15.1	15.350	1.9
18	14.6	15.9	13.2	15.0	14.675	2.7
19	15.7	15.9	16.5	16.4	16.125	0.8
20	15.9	14.9	13.8	14.1	14.675	2.1

Subgroup 1 is

$$\begin{aligned}
 \bar{x}_1 &= \frac{x_1 + x_2 + x_3 + x_4}{4} \\
 &= \frac{15.4 + 14.1 + 12.7 + 13.8}{4} \\
 &= \frac{56}{4} \\
 &= 14
 \end{aligned}$$

The sample range for Subgroup 1 is calculated as

$$\begin{aligned}
 R_1 &= \max - \min \\
 &= 15.4 - 12.7 \\
 &= 2.7
 \end{aligned}$$

The sample mean and sample range for all subgroups are displayed in Table 4.3. To find the grand average, calculate

$$\begin{aligned}
 \bar{\bar{x}} &= \frac{\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_{20}}{20} \\
 &= \frac{14.000 + 14.600 + \cdots + 14.675}{20} \\
 &= \frac{299.20}{20} \\
 &= 14.96
 \end{aligned}$$

The average range is found to be

$$\begin{aligned}\bar{R} &= \frac{R_1 + R_2 + \cdots + R_{20}}{20} \\ &= \frac{2.7 + 1.2 + \cdots + 2.1}{20} \\ &= \frac{39}{20} \\ &= 1.95\end{aligned}$$

The LCL, centerline, and UCL of the \bar{x} control chart can be found using the formulas provided in Equation 1. First, the constant A_2 must be found. From Table 4.2 with $n = 4$, we find $A_2 = 0.729$. The control limits and centerline for the \bar{x} control chart are

$$\text{UCL} = \bar{x} + A_2 \bar{R} = 14.96 + 0.729(1.95) = 16.38$$

$$\text{Centerline} = 14.96$$

$$\text{LCL} = \bar{x} - A_2 \bar{R} = 14.96 - 0.729(1.95) = 13.54$$

The control limits and centerline for the R chart can be calculated using the formulas in Equation 2. The constants D_3 and D_4 for sample size $n = 4$ are found in Table 4.2. These values are $D_3 = 0$ and $D_4 = 2.282$. The control limits and centerline are

$$\text{UCL} = D_4 \bar{R} = 2.282(1.95) = 4.449$$

$$\text{Centerline} = \bar{R} = 1.95$$

$$\text{LCL} = D_3 \bar{R} = 0(1.95) = 0$$

The resulting control charts for Example 1 are displayed in Fig. 4.2. The process appears to be in control, because there are no obvious patterns or trends and no points plot beyond the control limits. The control charts presented in Fig. 4.2 were constructed using Minitab software.

Finally, if a point or points had fallen beyond the control limits in either the \bar{x} chart or R chart, then the process should have been investigated to determine whether an assignable cause could be found. If the point or points is/are determined to be unusual, then the point(s) should be removed, the control limits recalculated, and revised control charts constructed. For more details about these control charts and their interpretation, see Montgomery (2001).

\bar{x} and S Control Charts

There are situations in which the sample standard deviation, S , would be a more preferable estimate of the process variability than the sample range. In general, it has been shown that the sample standard deviation is a better estimate of the true process standard deviation when the sample size n is large, say $n > 10$. As n gets larger, the range, R , is less precise and does not have the same statistical efficiency for large sample sizes. The sample standard deviation has also been shown to be a better estimator for the process standard deviation when the sample sizes are not constant from one subgroup to another.

The \bar{x} and S control charts are constructed similarly to the \bar{x} and R control charts, except the standard deviation is calculated instead of the range for each subgroup. Let x_1, x_2, \dots, x_n

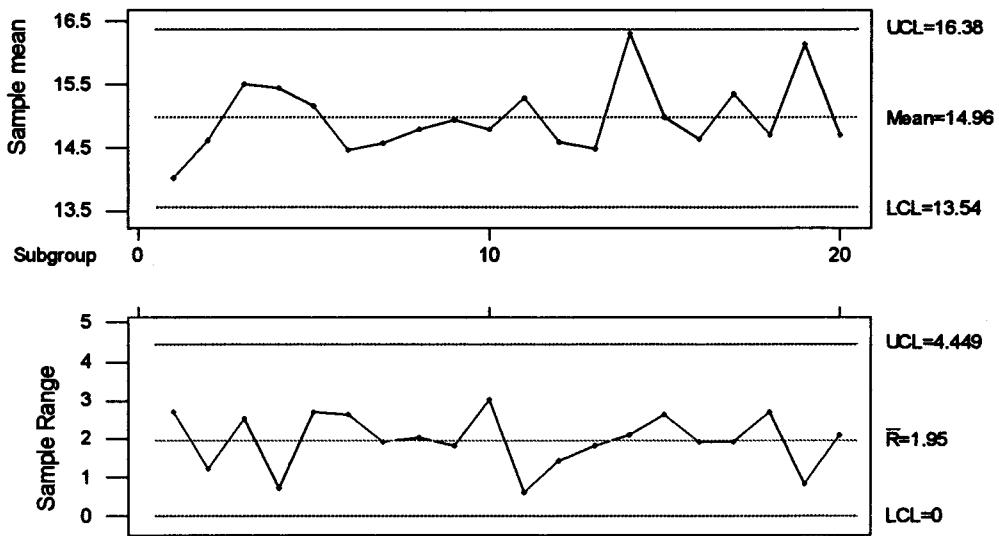


FIG. 4.2. The \bar{x} and R control charts for Example 1.

represent a sample of size n . The formula for the sample standard deviation of subgroup i is

$$S_i = \sqrt{\frac{\sum_{j=1}^n (x_j - \bar{x})^2}{n-1}}$$

The average standard deviation over all m subgroups is

$$\bar{S} = \frac{\sum_{i=1}^m S_i}{m}$$

The control limits and centerline for \bar{x} are then

$$\begin{aligned} \text{UCL} &= \bar{x} + A_3 \bar{S} \\ \text{Centerline} &= \bar{x} \\ \text{LCL} &= \bar{x} - A_3 \bar{S} \end{aligned} \tag{3}$$

where A_3 is a constant that depends on the sample size, n , and can be found in Table 4.2 for various sample sizes. The control limits and centerline for the S chart are

$$\begin{aligned} \text{UCL} &= B_4 \bar{S} \\ \text{Centerline} &= \bar{S} \\ \text{LCL} &= B_3 \bar{S} \end{aligned} \tag{4}$$

where B_3 and B_4 are constants that depend on the sample size, n , and can be found in Table 4.2 for various sample sizes.

TABLE 4.4
Wafer Thickness Data with Sample Means and Sample Standard Deviations for Example 2

Subgroup i	Thickness				\bar{x}_i	S_i
1	15.4	14.1	12.7	13.8	14.000	1.11
2	15.3	14.2	14.1	14.8	14.600	0.56
3	15.5	15.0	17.0	14.5	15.500	1.08
4	15.7	15.1	15.8	15.1	15.425	0.38
5	14.7	16.8	14.1	15.0	15.150	1.16
6	14.3	15.8	13.2	14.5	14.450	1.07
7	14.3	15.8	14.2	13.9	14.550	0.85
8	14.4	15.3	15.7	13.7	14.775	0.90
9	14.2	16.0	14.4	15.1	14.925	0.81
10	13.2	14.9	16.2	14.8	14.775	1.23
11	15.4	15.6	15.0	15.1	15.275	0.28
12	13.9	14.0	15.3	15.1	14.575	0.73
13	15.2	13.4	15.1	14.2	14.475	0.85
14	17.7	15.6	15.7	16.2	16.300	0.97
15	13.5	14.8	15.5	16.1	14.975	1.12
16	14.3	14.0	15.9	14.3	14.625	0.86
17	14.3	16.2	15.8	15.1	15.350	0.83
18	14.6	15.9	13.2	15.0	14.675	1.12
19	15.7	15.9	16.5	16.4	16.125	0.39
20	15.9	14.9	13.8	14.1	14.675	0.94

Example 2. To illustrate the construction of the \bar{x} and S charts, consider the wafer thickness data presented in Example 1. Instead of calculating the range for each subgroup, we now calculate the standard deviation for each subgroup. For example, consider the first subgroup. The sample standard deviation for this subgroup, S_1 , is

$$\begin{aligned}
 S_1 &= \sqrt{\frac{\sum_{j=1}^n (x_j - \bar{x})^2}{n-1}} \\
 &= \sqrt{\frac{(15.4 - 14)^2 + (14.1 - 14)^2 + (12.7 - 14)^2 + (13.8 - 14)^2}{4-1}} \\
 &= 1.11
 \end{aligned}$$

The standard deviations for each subgroup are displayed in Table 4.4 along with the original data and sample means. The average standard deviation is found to be

$$\begin{aligned}
 \bar{S} &= \frac{\sum_{i=1}^m S_i}{m} \\
 &= \frac{1.11 + 0.59 + \dots + 0.94}{20} \\
 &= \frac{17.24}{20} \\
 &= 0.862
 \end{aligned}$$

The control limits and centerline for the \bar{x} control chart are found using Equation 3 with $A_3 = 1.628$ (from Table 4.2 with $n = 4$)

$$\text{UCL} = \bar{x} + A_3 \bar{S} = 14.96 + 1.628(0.862) = 16.36$$

$$\text{Centerline} = \bar{x} = 14.96$$

$$\text{LCL} = \bar{x} - A_3 \bar{S} = 14.96 - 1.628(0.862) = 13.56$$

The control limits and centerline for the S chart are found using Equation 4 with $B_3 = 0$ and $B_4 = 2.266$ (again, from Table 4.2 with $n = 4$)

$$\text{UCL} = B_4 \bar{S} = 2.266(0.862) = 1.95$$

$$\text{Centerline} = \bar{S} = 0.862$$

$$\text{LCL} = B_3 \bar{S} = 0(0.862) = 0$$

The control charts for the sample mean and the sample standard deviation are displayed in Fig. 4.3. The process appears to be in control, because there are no obvious trends or patterns, and points plot within the control limits on both charts. Again, if a point had plotted out of control, it should have been investigated. If a point is found to be unusual, it should be removed and revised control limits calculated. The control charts in Fig. 4.3 were constructed using Minitab software.

Individuals Control Charts

Some situations exist in which the sample consists of a single observation, that is, $n = 1$. A sample of size one can occur when production is very slow or costly, and it is impractical to allow the sample size to be greater than one. An individuals control chart for variable data is appropriate for this type of situation. Later in this chapter two control charting procedures are introduced that are significantly better at monitoring the process than the individuals control chart.

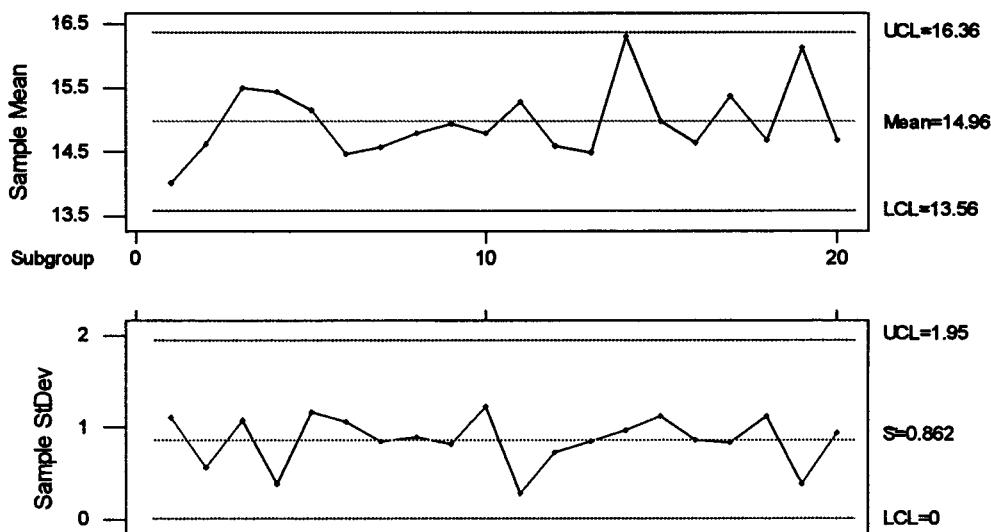


FIG. 4.3. The \bar{x} and S control charts for Example 2.

The individuals control chart uses the *moving range* of two successive samples for estimating the process variability (see Montgomery [2001] for a detailed discussion of moving range and individuals control charts in general). The moving range is given by

$$\text{MR}_i = |x_i - x_{i-1}|.$$

To illustrate, for m subgroups of size $n = 1$ each, $m - 1$ moving ranges are defined as $\text{MR}_2 = |x_2 - x_1|$, $\text{MR}_3 = |x_3 - x_2|, \dots, \text{MR}_m = |x_m - x_{m-1}|$. The average moving range is simply

$$\overline{\text{MR}} = \frac{\sum_{i=2}^m \text{MR}_i}{m - 1}$$

because only $m - 1$ moving range values are calculated (there is no moving range for Subgroup 1). Control charts are designed for the individual observations (x chart) and the moving range of the subgroups (MR chart). The MR chart is used to monitor the process variability.

The control limits and centerline of the x (or individuals) control chart are

$$\begin{aligned} \text{UCL} &= \bar{x} + 3 \frac{\overline{\text{MR}}}{d_2} \\ \text{Centerline} &= \bar{x} \\ \text{LCL} &= \bar{x} - 3 \frac{\overline{\text{MR}}}{d_2} \end{aligned} \tag{5}$$

where d_2 is a constant that depends on the number of observations used to calculate the moving range for each subgroup. Values for d_2 and various sample sizes are given in Table 4.2. Specifically, because $\text{MR}_i = |x_i - x_{i-1}|$, $n = 2$ for the individuals control charts. The control chart for individuals is constructed by plotting the actual observation, x_i , the control limits, and the centerline against the subgroup (or time) order.

The control limits and centerline for the moving range control chart are

$$\begin{aligned} \text{UCL} &= D_4 \overline{\text{MR}} \\ \text{Centerline} &= \overline{\text{MR}} \\ \text{LCL} &= D_3 \overline{\text{MR}} \end{aligned} \tag{6}$$

where D_3 and D_4 depend on the sample size ($n = 2$) and are provided in Table 4.2. The moving range control chart is constructed by plotting the $m - 1$ moving ranges, the control limits, and the centerline against the subgroup (or time) order.

Example 3. Twenty-ounce soft-drink bottles are filled by a machine and weighed. Weights for 25 successive bottles have been collected and are displayed in Table 4.5. The engineer wishes to determine if the filling process is indeed in control.

Solution. The moving ranges must be calculated using $\text{MR}_i = |x_i - x_{i-1}|$. To illustrate the calculation, consider the first moving range at Subgroup 2:

$$\text{MR}_2 = |x_2 - x_1| = |19.99 - 20.01| = 0.02$$

TABLE 4.5
Soft-Drink Bottle Data for Example 3

Bottle	Weight (x_i)	Moving Range
1	20.01	—
2	19.99	0.02
3	19.95	0.04
4	20.62	0.67
5	20.04	0.58
6	20.00	0.04
7	19.83	0.17
8	19.94	0.11
9	19.94	0.00
10	20.08	0.14
11	20.01	0.07
12	20.04	0.03
13	20.00	0.04
14	20.01	0.01
15	20.06	0.05
16	19.91	0.15
17	20.06	0.15
18	20.03	0.03
19	20.10	0.07
20	19.89	0.21
21	19.95	0.06
22	20.00	0.05
23	20.01	0.01
24	20.05	0.04
25	20.05	0.00

The remaining moving ranges are calculated accordingly and are provided in Table 4.5. To construct the control chart for individual observations (x control chart), the sample average weight for all 25 bottles is needed:

$$\begin{aligned}
 \bar{x} &= \frac{\sum_{i=1}^m x_i}{m} \\
 &= \frac{20.01 + 19.99 + \dots + 20.05}{25} \\
 &= \frac{500.575}{25} \\
 &= 20.023
 \end{aligned}$$

Next, the average moving range, \overline{MR} , can be calculated using the moving ranges for each of the 24 subgroups. The average moving range is

$$\begin{aligned}
 \overline{MR} &= \frac{\sum_{i=2}^m MR_i}{m - 1} \\
 &= \frac{0.02 + 0.04 + \dots + 0}{24} \\
 &= \frac{2.7408}{24} \\
 &= 0.1142
 \end{aligned}$$

The control limits and centerline for the individuals chart with moving ranges of size 2 using the limits in Equation 5 are

$$UCL = 20.023 + 3 \frac{0.1142}{1.128} = 20.33$$

$$\text{Centerline} = 20.023$$

$$LCL = 20.023 - 3 \frac{0.1142}{1.128} = 19.72$$

The control limits and centerline for the moving range chart are found using the limits given in Equation 6 with $D_3 = 0$ and $D_4 = 3.267$ from Table 4.2:

$$UCL = 3.267(0.1142) = 0.373$$

$$\text{Centerline} = 0.1142$$

$$LCL = 0(0.1142) = 0$$

The control charts for individual observations and for the moving range are provided in Fig. 4.4. Examining Fig. 4.4, it is obvious that some points are beyond the control limits on both the individuals chart and the moving range chart. Minitab identifies these points as Subgroup 4 on the individuals chart and Subgroups 4 and 5 on the moving range chart. Therefore, Subgroups 4 and 5 should be investigated to determine if they are truly unusual points and should be removed.

To illustrate how to revise control charts, assume that Subgroups 4 and 5 were found to be erroneous and should be removed. By removing these subgroups, the sample mean, average moving range, and control limits for both charts will change and must be recalculated. By removing Observation 4 ($x_4 = 20.62$) and Observation 5 ($x_5 = 20.04$), there are 23 subgroups. Let m^* denote the number of subgroups remaining with the removal of the unusual observations. Thus, $m^* = 23$ and the revised sample mean for the individual weights

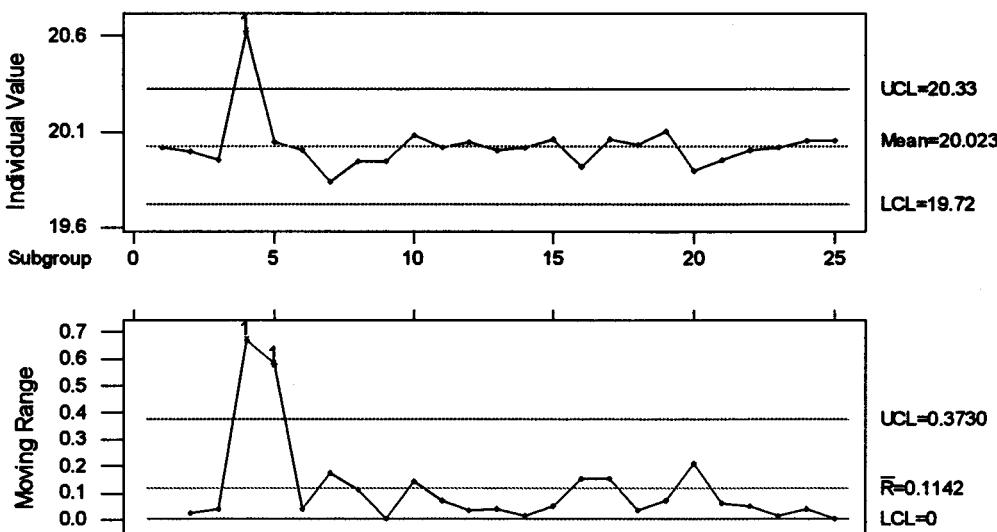


FIG. 4.4. \bar{x} and MR control charts for Example 3.

is approximately

$$\begin{aligned}\bar{x} &= \frac{\sum_{i=1}^{m^*} x_i}{m^*} \\ &= \frac{460}{23} \\ &= 20.00\end{aligned}$$

The moving ranges are also recalculated with the removal of Observations 4 and 5. The average moving range is approximately

$$\begin{aligned}\overline{MR} &= \frac{\sum_{i=2}^{m^*} MR_i}{m^* - 1} \\ &= \frac{1.50}{22} \\ &= 0.0682\end{aligned}$$

The revised control limits for the individuals chart, denoted UCL^* and LCL^* are

$$\begin{aligned}UCL^* &= 20.00 + 3 \frac{0.0682}{1.128} = 20.18 \\ \text{Centerline}^* &= 20.00 \\ LCL^* &= 20.00 - 3 \frac{0.0682}{1.128} = 19.82\end{aligned}$$

The revised control limits for the moving range chart are

$$\begin{aligned}UCL^* &= 3.267(0.0682) = 0.0.223 \\ \text{Centerline}^* &= 0.0682 \\ LCL^* &= 0(0.0682) = 0\end{aligned}$$

The revised control charts are displayed in Fig. 4.5. Notice that points 4 and 5, which were removed to revise the control limits, are represented on the chart. To preserve the time sequence or subgroup ordering, it is important to include the observations on the graph, but keep in mind that they are not used in the calculations. In other words, the removed observations are now simply “place holders” and do not affect the new control limits or interpretation of the chart. The revised control charts do not display any new observations as plotting beyond the control limits; therefore, the process is now considered in control.

It is important to note that the moving range control chart itself cannot be interpreted in the same way as the R chart with respect to patterns or trends. A pattern or trend on the moving range chart is not necessarily an indication that the process is out of control. The moving ranges are correlated (recall $MR_i = |x_i - x_{i-1}|$). That is, there may be a dependency between successive observations due to the subgrouping or time order. In this case, patterns or trends may occur naturally.

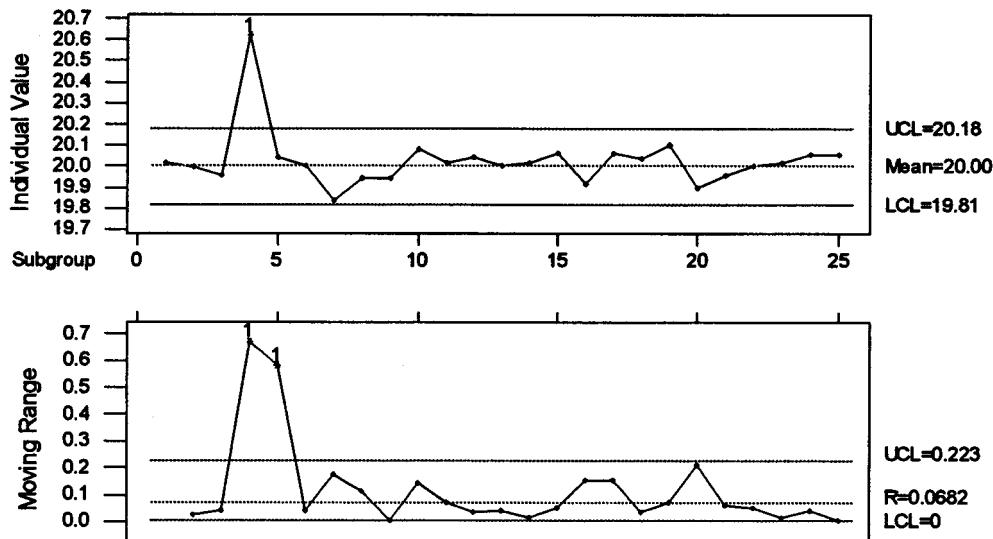


FIG. 4.5. Revised control charts for Example 3.

Attributes Control Charts

There are many processes or data collection activities that may not involve a true measurement (actual length, width, weight, etc.) on a particular part. There are many scenarios in which the quality characteristic of interest is simply to classify the measurement into a single category. For example, manufactured products may be measured but only classified as defective/nondefective, conforming/nonconforming, or pass/fail. Other situations may involve counting the number of nonconformities on a single product or item. For example, steel plates manufactured for use in pacemakers may be examined for defects such as scrapes, dents, and so forth. In these situations variables control charts are not necessary or appropriate. When the data can be classified, the appropriate control charts are called *attributes control charts*. The following subsection presents fraction nonconforming control charts (p and np control charts) and control charts for nonconformities (c and u control charts).

Fraction Nonconforming Control Charts

There are two Shewhart fraction nonconforming control charts, the p chart and the np chart. In practice these two charts give identical results (in-control or out-of-control) for a particular problem. The choice of chart to implement is arbitrary and often up to the practitioner.

Definitions and Descriptions. Before a formal presentation of the control charts, some notation and definitions are in order. For the fraction nonconforming control charts the quality characteristic of interest can be placed into one of exactly two categories. These categories may be pass/fail, defective/nondefective, yes/no, and so forth. For simplification we use the term “nonconforming” as a general reference regardless of the final categories to be used. The following notation is used in this subsection.

- n —the number of items examined, lot size, or sample size.
- m —the number of subgroups selected. (Each subgroup is of size n .)

- X —the number of nonconformities found in the sample of size n . (Obviously, $X \leq n$.)
- p —probability that any one item of interest will be nonconforming.
- \hat{p} —the *sample fraction nonconforming* found from our data. By definition, $\hat{p} = X/n$ and is calculated for each of the m subgroups.
- \bar{p} —the *average fraction nonconforming*. By definition, $\bar{p} = \frac{\sum_{i=1}^m \hat{p}_i}{m}$ and is an estimate of p defined above.

The goal of the fraction nonconforming control chart is to monitor the proportion of nonconforming items (fraction nonconforming) for a process of interest using the data collected over m samples each of size n .

The p Chart. The p chart is the charting technique used to monitor the proportion nonconforming directly. The control limits and centerline (when p is unknown) are

$$\text{UCL} = \bar{p} + 3\sqrt{\frac{\bar{p}(1 - \bar{p})}{n}}$$

$$\text{Centerline} = \bar{p}$$

$$\text{LCL} = \bar{p} - 3\sqrt{\frac{\bar{p}(1 - \bar{p})}{n}}$$

The control limits, centerline, and individual sample fraction nonconforming, \hat{p}_i , are plotted against the subgroup number, m . If one or more of the fraction nonconforming are outside the control limits, the process is considered out of control. Patterns or trends that are readily apparent would also be an indication of a possible out-of-control situation.

Example 4. A local bank collects data on the number of daily account activities that are recorded in error. There are records for 2,000 account activities per day, and data has been collected over a 30-day period. The number of account activities in error is recorded in Table 4.6. The bank would like to monitor the proportion of errors being committed by establishing control charts.

TABLE 4.6
Banking Accounts with Errors Over a 30-Day Period

Day	Accounts in Error	\hat{p}_i	Day	Accounts in Error	\hat{p}_i
1	81	0.0405	16	75	0.0375
2	77	0.0385	17	73	0.0365
3	75	0.0375	18	75	0.0375
4	67	0.0335	19	80	0.0400
5	73	0.0365	20	62	0.0310
6	98	0.0490	21	76	0.0380
7	72	0.0360	22	78	0.0390
8	75	0.0375	23	72	0.0360
9	84	0.0420	24	89	0.0445
10	69	0.0345	25	82	0.0410
11	79	0.0395	26	82	0.0410
12	83	0.0415	27	91	0.0455
13	95	0.0475	28	62	0.0310
14	79	0.0395	29	79	0.0395
15	78	0.0390	30	76	0.0380

Solution. First, the fraction nonconforming, \hat{p}_i , for each day must be computed. To illustrate the calculation, consider Day 1. The number of activities in error are $X = 81$, and the total number of activities investigated (the sample size) is $n = 2,000$ resulting in $\hat{p}_1 = 81/2,000 = 0.0405$. The remaining fraction nonconforming are found accordingly and provided in the third and sixth columns of Table 4.6. The average fraction nonconforming, \bar{p} , is found to be

$$\begin{aligned}\bar{p} &= \frac{\sum_{i=1}^m \hat{p}_i}{m} \\ &= \frac{\sum_{i=1}^{30} \hat{p}_i}{30} \\ &= \frac{0.0405 + 0.0385 + \dots + 0.0380}{30} \\ &= \frac{1.1685}{30} \\ &= 0.03895\end{aligned}$$

The control limits and centerline for the fraction nonconforming chart (p chart) are

$$UCL = \bar{p} + 3\sqrt{\frac{\bar{p}(1 - \bar{p})}{n}} = 0.03895 + 3\sqrt{\frac{0.03895(1 - 0.03895)}{2000}} = 0.05193$$

$$\text{Centerline} = 0.03895$$

$$LCL = \bar{p} - 3\sqrt{\frac{\bar{p}(1 - \bar{p})}{n}} = 0.03895 - 3\sqrt{\frac{0.03895(1 - 0.03895)}{2000}} = 0.02597$$

The resulting p chart is displayed in Fig. 4.6. There are no points beyond the control limits, and there are no obvious patterns or trends. Therefore, the process appears to be in statistical control.

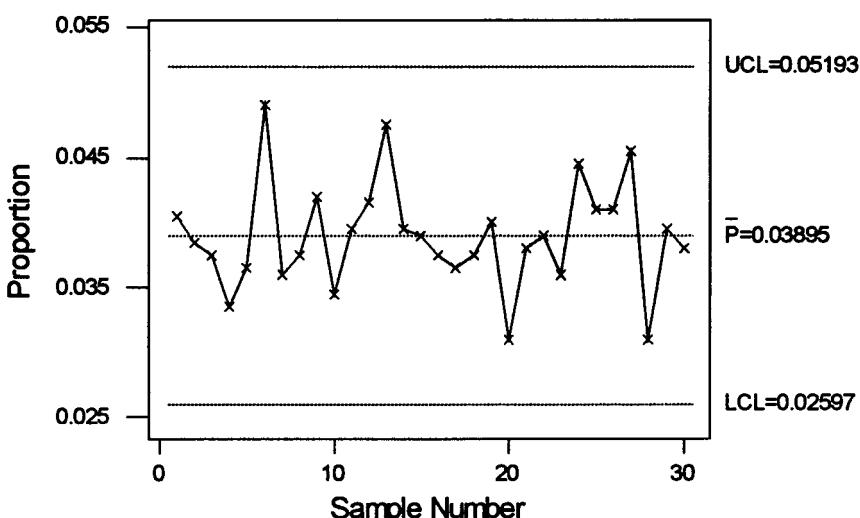


FIG. 4.6. Fraction nonconforming control chart for Example 4.

The np Control Chart. The np control chart is a variation of the p chart, except now the actual number of nonconforming items are plotted on the control chart. The control limits are based on the number of nonconforming units instead of the fraction nonconforming. The np chart and the p chart will give the same resulting information. That is, if the p chart indicates an out-of-control situation for a process, then the np chart for the same data will also signal out-of-control. One of the reasons the np chart may be used instead of the p chart is ease of interpretation by technicians and others collecting and reporting the data. Otherwise, there is no advantage or disadvantage to implementing one over the other.

The statistics that are necessary for this chart are simple. The average fraction nonconforming, \bar{p} , is the only value that must be calculated before constructing the control limits. The average fraction nonconforming can be found without having to calculate all the sample fraction nonconforming values, \hat{p}_i . For the np chart the average fraction nonconforming can be calculated using

$$\bar{p} = \frac{\sum_{i=1}^m X_i}{mn} \quad (7)$$

That is, sum up the total number of nonconforming items for all m subgroups and divide that value by the product mn . The control limits and centerline are calculated using

$$\text{UCL} = n\bar{p} + 3\sqrt{n\bar{p}(1 - \bar{p})}$$

$$\text{Centerline} = n\bar{p}$$

$$\text{LCL} = n\bar{p} - 3\sqrt{n\bar{p}(1 - \bar{p})}$$

The control limits, centerline, and the number of nonconforming items, X_i , are plotted against the subgroup. Interpretation of the chart is identical to that of the p chart.

Example 5. Consider the account activity data presented in Example 4. The data are displayed in Table 4.6. Instead of constructing the p chart for the fraction nonconforming, the np chart is to be constructed for the number of nonconforming items.

Solution. Recall that $n = 2,000$ accounts are examined over a 30-day ($m = 30$) period. The sample average nonconforming is needed, and from Example 4 it was found to be $\bar{p} = 0.03895$. But to illustrate Equation 7 we show that

$$\begin{aligned} \bar{p} &= \frac{\sum_{i=1}^m X_i}{mn} \\ &= \frac{81 + 77 + \dots + 76}{(30)(2000)} \\ &= \frac{2337}{(30)(2000)} \\ &= 0.03895 \end{aligned}$$

which is the same estimate found previously. The control limits and centerline for the

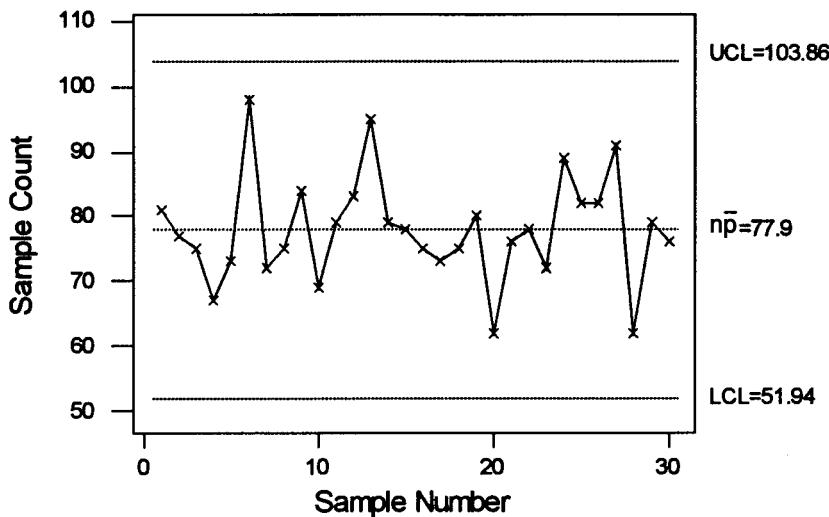


FIG. 4.7. The np control chart for account activity data.

np chart are

$$UCL = n\bar{p} + 3\sqrt{n\bar{p}(1 - \bar{p})} = 2000(0.03895) + 3\sqrt{2000(0.03895)(1 - 0.03895)} = 103.86$$

$$\text{Centerline} = n\bar{p} = 77.9$$

$$LCL = n\bar{p} - 3\sqrt{n\bar{p}(1 - \bar{p})} = 2000(0.03895) - 3\sqrt{2000(0.03895)(1 - 0.03895)} = 51.94$$

The control limits, centerline, and number of nonconforming items are plotted on the control chart displayed in Fig. 4.7. As with the corresponding p chart the process appears to be in statistical control.

Control Charts for Nonconformities

Control charts for nonconformities are similar to those discussed in the previous subsection for the number of nonconforming items. But the underlying statistical properties of the two are different and result in different control charting techniques. One way to think of the difference is that for the number of nonconforming items (discussed previously) we noted that the number of nonconforming items could not exceed the number of items in each sample investigated, that is, $X \leq n$. For monitoring nonconformities no such restriction exists. That is, the nonconformities are counted per unit. There could be an infinite number of nonconformities on the unit or units under investigation. For example, consider a process that manufactures glass soft-drink bottles. Ten bottles are selected every hour and examined for defects (scratches or chips). In one sample of 10 bottles, 17 defects were found. In this scenario there were 10 units investigated and 17 nonconformities (defects) discovered. This is an illustration of counting the nonconformities within a sample. It should be noted that there are some situations in which exactly one unit is selected for the sample and the number of nonconformities recorded. In this subsection the control charts for nonconformities that are presented include the c chart and the u chart.

The c Control Chart. The c chart is an appropriate control charting technique for nonconformities if the sample size, n , is constant from subgroup to subgroup. For the c chart

TABLE 4.7
Bottle Manufacturing Process Data for Example 6

Group	Number of Defects	Group	Number of Defects
1	20	13	16
2	14	14	14
3	17	15	17
4	18	16	13
5	14	17	13
6	23	18	21
7	28	19	25
8	11	20	14
9	22	21	18
10	19	22	16
11	19	23	13
12	21	24	26

we have:

- n —number of items inspected, sample size
- m —number of subgroups
- X —number of nonconformities per item inspected or per sample size
- \bar{c} —average number of nonconformities (over the entire data set)

$$\text{By definition, } \bar{c} = \frac{\sum_{i=1}^m X_i}{m}$$

The control limits and centerline for nonconformities when the sample size is constant are

$$\text{UCL} = \bar{c} + 3\sqrt{\bar{c}}$$

$$\text{Centerline} = \bar{c}$$

$$\text{LCL} = \bar{c} - 3\sqrt{\bar{c}}$$

Example 6. Consider a process that manufactures glass soft-drink bottles. Ten bottles are selected every hour and examined for defects (scratches or chips). Twenty-four such samples are collected with the number of nonconformities recorded for each. The results are displayed in Table 4.7. The process engineer wants to determine if the manufacturing process is in control.

Solution. A c chart is an appropriate monitoring technique for this problem because it is known that the sample size (10 bottles) is constant from subgroup to subgroup. The average number of nonconformities is

$$\begin{aligned}\bar{c} &= \frac{\sum_{i=1}^m X_i}{m} \\ &= \frac{20 + 14 + \dots + 26}{24} \\ &= \frac{432}{24} \\ &= 18\end{aligned}$$

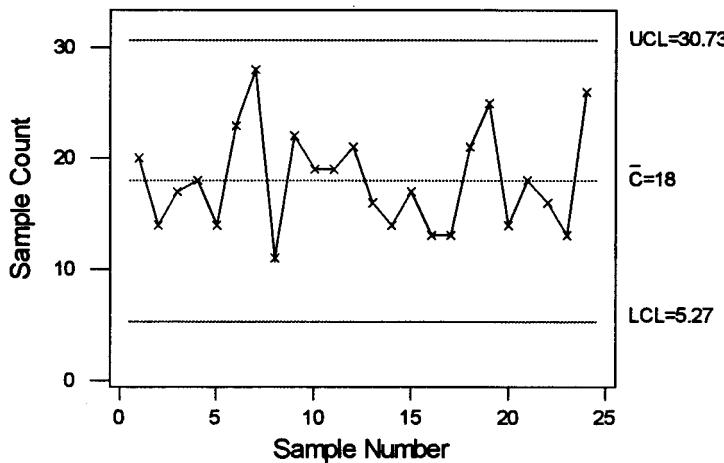


FIG. 4.8. The c chart for the bottle manufacturing process.

The control limits and centerline for the number of nonconformities are

$$UCL = \bar{c} + 3\sqrt{\bar{c}} = 18 + 3\sqrt{18} = 30.73$$

$$\text{Centerline} = \bar{c} = 18$$

$$LCL = \bar{c} - 3\sqrt{\bar{c}} = 18 - 3\sqrt{18} = 5.27$$

The control chart for the process is displayed in Fig. 4.8. The process does not exhibit any out-of-control situation. The process appears to be acceptable based on the data collected.

The u Control Chart. The u chart is an alternate control charting technique for nonconformities. The u chart monitors the average number of nonconformities (whereas the c chart monitors the number of nonconformities). As with the p and np charts, the resulting c and u charts for constant sample size provide identical results.

- n —number of items inspected, sample size
- m —number of subgroups
- X —number of nonconformities per item inspected or per sample size
- u —average number of nonconformities in a sample of n units. By definition, $u_i = X_i/n$. (Do not confuse u_i with \bar{c} stated previously. The value u_i is the average number of nonconformities per subgroup, \bar{c} , is the average number of nonconformities over all subgroups, m).
- \bar{u} —average number of nonconformities per unit. By definition, $\bar{u} = \frac{\sum_{i=1}^m u_i}{m}$

The control limits and centerline for average number of nonconformities are

$$UCL = \bar{u} + 3\sqrt{\frac{\bar{u}}{n}}$$

$$\text{Centerline} = \bar{u}$$

$$LCL = \bar{u} - 3\sqrt{\frac{\bar{u}}{n}}$$

The control limits, centerline, and u_i are plotted on the control chart against the subgroup.

TABLE 4.8
Bottle Manufacturing Process Data for Example 7

Group	Number of Defects	u_i	Group	Number of Defects	u_i
1	20	2.0	13	16	1.6
2	14	1.4	14	14	1.4
3	17	1.7	15	17	1.7
4	18	1.8	16	13	1.3
5	14	1.4	17	13	1.3
6	23	2.3	18	21	2.1
7	28	2.8	19	25	2.5
8	11	1.1	20	14	1.4
9	22	2.2	21	18	1.8
10	19	1.9	22	16	1.6
11	19	1.9	23	13	1.3
12	21	2.1	24	26	2.6

Example 7. Consider the soft-drink bottle manufacturing process presented in Example 6. The number of bottles examined, number of defects recorded, and the average number of nonconformities per bottle, u_i , are displayed in Table 4.8. To illustrate the calculation of u_i , consider the average number of nonconformities for Group 1:

$$u_i = \frac{X_1}{n} = \frac{20}{10} = 2$$

The average number of nonconformities per bottle over all subgroups is found to be

$$\begin{aligned}\bar{u} &= \frac{\sum_{i=1}^m u_i}{m} \\ &= \frac{2.0 + 1.4 + \dots + 2.6}{24} \\ &= \frac{43.2}{24} \\ &= 1.8\end{aligned}$$

The control limits and centerline are then

$$\text{UCL} = \bar{u} + 3\sqrt{\frac{\bar{u}}{n}} = 1.8 + 3\sqrt{\frac{1.8}{10}} = 3.073$$

$$\text{Centerline} = \bar{u} = 1.8$$

$$\text{LCL} = \bar{u} - 3\sqrt{\frac{\bar{u}}{n}} = 1.8 - 3\sqrt{\frac{1.8}{10}} = 0.527$$

The control limits, centerline, and u_i values are plotted against each subgroup. The resulting control chart is displayed in Fig. 4.9. Again, there is no indication of an out-of-control process.

Final Comments on Attributes Control Charts

The attributes control charts presented in this section are the basic and most commonly employed charts in statistical process control. Each of the four charting procedures presented here can be extended to situations in which the sample size, n , varies among the subgroups.

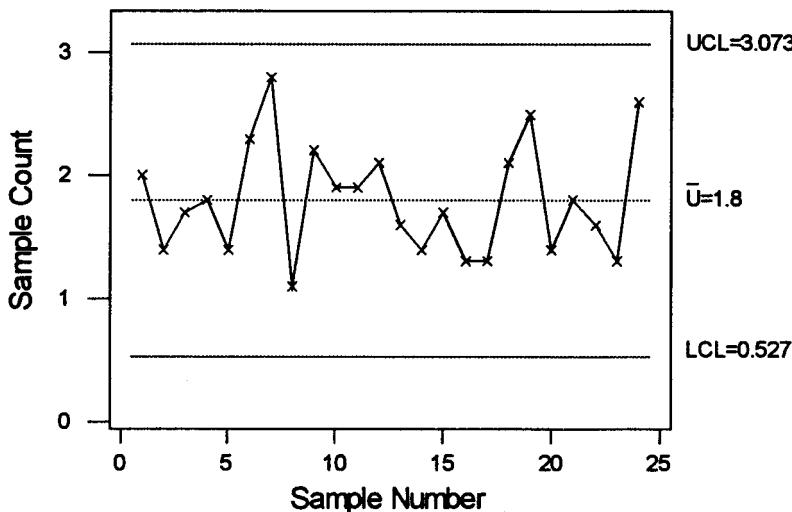


FIG. 4.9. The u chart for the bottle manufacturing process.

In this instance the practitioner can choose some standard techniques to deal with varying sample sizes. It is common to use varying control limits if the sample sizes are not constant. A second technique is to find the average sample size for a particular problem and apply the formulas provided in this section. The choice is up to the practitioner and will depend on the situation under investigation. The reader is encouraged to see Montgomery (2001) for details and examples of more applications of both variables and attributes control charts.

Cumulative Sum Control Charts

In the previous section Shewhart control charts were presented for monitoring various processes. As stated previously, these charts are most appropriate if the normality assumption is satisfied for the quality characteristic of interest. Unfortunately, for many problems the assumption of normality is not unknown or is not valid. Also, the Shewhart control charts have been shown to be poor indicators of small shifts in the process mean because they rely solely on the *current observation* (see Gan, 1991; Hawkins, 1981, 1993; Lucas, 1976; Montgomery, 2001; Woodall & Adams, 1993). One alternative to the use of Shewhart control charts is the *cumulative sum (CUSUM) control chart*.

The CUSUM control chart is more sensitive to small shifts in the process because it is based on not only the current observation, but also the most recent past observations.

The CUSUM control chart was first introduced by Page (1954) and includes information from successive sample observations. The control chart plots the cumulative sums of deviations of the observations from some target value. The CUSUM control chart plots the quantity

$$CS_i = \sum_{j=1}^n (\bar{x}_j - \mu_0)$$

against the subgroup i where

- CS_i is the cumulative sum up to and including the i th sample.
- \bar{x}_j is the mean of the j th sample.
- μ_0 is the target value for the process average.
- n is the sample size.

As long as the process remains “in control” at the target value μ_0 , then the cumulative sums, CS_i , will be approximately zero. Otherwise, if the process shifts away from the target mean, then CS_i become increasingly large in absolute value. That is, if the shift away from the target is in the upward direction, the CS_i will become a large positive value. If the shift away from the target is in the downward direction, the CS_i will become a large but negative value. Because the CUSUM uses information from a sequence of observations, it can detect small shifts in the process more quickly than a standard Shewhart chart. The CUSUM control charts can be implemented for both subgroup data and for individuals data. The two-sided CUSUM control chart for individuals will be presented next. One-sided CUSUM charts can be constructed if the practitioner is interested in a particular direction of shift—either downward or upward, but not both. See the references mentioned previously for more examples and discussion of the one-sided CUSUM charts.

The *tabular* form of the CUSUM involves two statistics, CS_i^+ and CS_i^- (see Lucas, 1976; and Montgomery, 2001). CS_i^+ is the accumulation of deviations above the target mean and is referred to as the one-sided upper cusum. CS_i^- is the accumulation of deviations below the target mean and is referred to as the one-sided lower CUSUM. CS_i^+ and CS_i^- are calculated using

$$\begin{aligned} CS_i^+ &= \max [0, x_i - (\mu_0 + K) + CS_{i-1}^+] \\ CS_i^- &= \max [0, (\mu_0 - K) - x_i + CS_{i-1}^-] \end{aligned}$$

with initial values of $CS_0^+ = CS_0^- = 0$, and where x_i is the i th observation.

The constant, K , is called the *reference value* where K is often calculated as

$$K = \frac{|\mu_1 - \mu_0|}{2}$$

where μ_0 is the target mean, and μ_1 is the out-of-control mean that we are interested in detecting (if the process average has shifted to this new mean, we want to be able to detect this shift fairly quickly).

If the out-of-control mean is unknown, there are methods for determining the value K . In this situation we can let $K = k\sigma$ where σ is the process standard deviation and k is some constant chosen so that a particular shift is detected. For example, say a shift from target of 1 standard deviation is important to detect (i.e., detect whether the target has shifted to $\mu_0 + 1\sigma$ or $\mu_0 - 1\sigma$, then $k = 1$, and $K = 1\sigma$). If the process standard deviation is not known, it must be estimated from the data provided. The two-sided CUSUM control chart plots the values of CS_i^+ and CS_i^- for each sample. If either statistic plots beyond a stated *decision interval* H , the process is considered out of control. (Important: when comparing the CS quantities to the decision interval, the respective sign is placed on any nonzero value. For example, because CS_i^- is a measure of a downward shift, once we calculate this value, the negative sign is placed on the value for comparing to the decision interval. This will be illustrated in more detail in the example.) The choice of H is not arbitrary but should be chosen after careful consideration. Often, $H = 5\sigma$ is an acceptable decision interval as this value, in general, results in a low false alarm rate for the process under study. The reader is referred to Hawkins (1993) and Woodall and Adams (1993) for more discussion on the design of CUSUM control charts.

Example 8. Thirty measurements have been taken on a critical dimension of a part in a manufacturing process. From past experience with the process the engineer reports that the

TABLE 4.9
Machining Data for Example 8

<i>Observation</i>	x_i	<i>Measurement</i>	
		CS_i^+	CS_i^-
1	156.96	0	1.44
2	164.23	2.63	0
3	163.44	4.47	0
4	162.13	5	0
5	161.41	4.81	0
6	161.77	4.98	0
7	153.96	0	4.44
8	169.94	8.34	0
9	157.99	4.73	0.41
10	161.89	5.02	0
11	162.24	5.66	0
12	167.51	11.57	0
13	159.83	9.80	0
14	157.91	6.11	0.49
15	160.71	5.22	0
16	166.23	9.85	0
17	160.11	8.36	0
18	164.02	10.78	0
19	159.95	9.13	0
20	165.87	13.40	0
21	154.26	6.06	4.14
22	159.05	3.51	3.49
23	162.56	4.47	0
24	165.97	8.84	0
25	161.46	8.70	0
26	160.32	7.42	0
27	157.13	2.95	1.27
28	159.23	0.58	0.44
29	160.40	0	0
30	157.84	0	0.56

standard deviation for the dimension of this part is $\sigma = 3.2$. The 30 measurements are displayed in Table 4.9. The process target is $\mu_0 = 160$. If the process mean shifts from this target by half of the process standard deviation, then the product eventually will be unacceptable. The engineer would like to design a two-sided CUSUM control chart and determine if the process is indeed in control at the target $\mu_0 = 160$.

Solution. To begin, determine some of the necessary quantities to set up the control chart. From the information given, the following is known:

- $\mu_0 = 160$
- $\sigma = 3.2$
- $K = 0.5\sigma = 0.5(3.2) = 1.6$; because a half of the process standard deviation, σ , is of interest in detecting, then $k = \frac{1}{2}$ or 0.5, and $K = k\sigma$.
- $H = 5\sigma = 5(3.2) = 16$

With the above information, the cumulative sums CS_i^+ and CS_i^- can be calculated for each

observation

$$\begin{aligned} CS_i^+ &= \max [0, x_i - (\mu_0 + K) + CS_{i-1}^+] \\ &= \max [0, x_i - (160 + 1.6) + CS_{i-1}^+] \\ CS_i^- &= \max [0, (\mu_0 - K) - x_i + CS_{i-1}^-] \\ &= \max [0, (160 - 1.6) - x_i + CS_{i-1}^-] \end{aligned}$$

In essence, if the process target has shifted, the cumulative sums will become nonzero. After calculating the cumulative sums, they are compared with the decision interval $(-H, H) = (-16, 16)$. If any one cumulative sum goes beyond the decision interval, the process is considered to be out of control.

To illustrate these calculations, consider the first observation, $x_1 = 156.96$. Initially, $CS_0^+ = 0$ and $CS_0^- = 0$, and as previously shown, $K = 1.6$. The first set of calculations is

$$\begin{aligned} CS_1^+ &= \max [0, x_1 - (\mu_0 + K) + CS_0^+] \\ &= \max [0, 156.96 - (160 + 1.6) + 0] \\ &= \max [0, -4.64] \\ &= 0 \\ CS_1^- &= \max [0, (\mu_0 - K) - x_1 + CS_0^-] \\ &= \max [0, (160 - 1.6) - 156.96 + 0] \\ &= \max [0, 1.44] \\ &= 1.44 \end{aligned}$$

(Because $CS_1^- = 1.44$, we would compare -1.44 with the decision interval.) Notice that both cumulative sums are within the decision interval $(-H, H)$ or $(-16, 16)$, so the process has not signaled out of control. The second set of calculations (for $x_2 = 164.23$) is then

$$\begin{aligned} CS_2^+ &= \max [0, x_2 - (\mu_0 + K) + CS_1^+] \\ &= \max [0, 164.23 - (160 + 1.6) + 0] \\ &= \max [0, 2.63] \\ &= 2.63 \\ CS_2^- &= \max [0, (\mu_0 - K) - x_2 + CS_1^-] \\ &= \max [0, (160 - 1.6) - 164.23 + 1.44] \\ &= \max [0, -4.39] \\ &= 0 \end{aligned}$$

Again, both values lie within the decision interval $(-16, 16)$; therefore, the process is still in statistical control at this point. The remaining cumulative sums are provided in Table 4.9. Examining the last two columns in Table 4.9, it is obvious that none of the cumulative sums will lie outside the decision interval $(-16, 16)$ and that the process is in control. Sometimes it is more convenient to see a graphical representation of the data. A CUSUM control chart consists of the decision values, the cumulative sums (with the corresponding signs), and a centerline at 0 and is presented in Fig. 4.10.

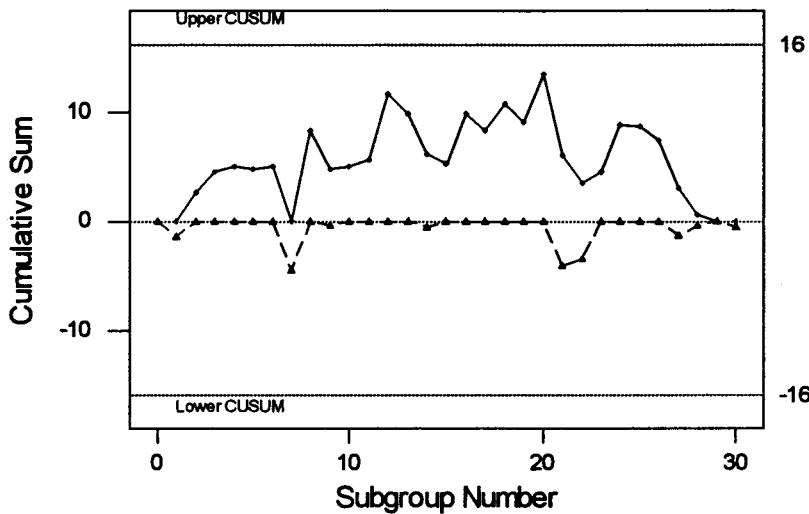


FIG. 4.10. CUSUM control chart for Example 8.

Exponentially Weighted Moving Average Control Charts

The exponentially weighted moving average (EWMA) control chart is another alternative to the Shewhart control charts when small shifts in the process mean are important. The EWMA control chart was first introduced by Roberts (1959). The EWMA statistic is defined by

$$z_i = \lambda x_i + (1 - \lambda)z_{i-1} \quad (8)$$

where λ is a weight, $0 < \lambda \leq 1$; x_i is the present observation; and z_{i-1} is the previous EWMA statistic in which the initial value for the procedure is $z_0 = \mu_0$, the process target mean. If the process target mean is not known, then \bar{x} is used as the initial value.

The definition given in Equation 8 includes information from past observation as well as the most current observation, x_i . Control limits can be placed on the values of z_i . If one or more z_i values fall beyond the control limits, then the process is considered to be out of control. The control limits for the EWMA for large values of i , are

$$\text{UCL} = \mu_0 + L\sigma\sqrt{\left(\frac{\lambda}{2-\lambda}\right)}$$

$$\text{LCL} = \mu_0 - L\sigma\sqrt{\left(\frac{\lambda}{2-\lambda}\right)}$$

where L is the width of the control limits. The values of L and λ can have considerable impact on the performance of the chart. Often, small values of λ work well in practice ($0.05 \leq \lambda \leq 0.25$). Values of L that also tend to work well are $2.6 \leq L \leq 3$. For more details on the EWMA control chart see Crowder (1989), Lucas and Saccucci (1990), and Montgomery (2001).

Example 9. Consider the dimension data presented in Example 8 for the CUSUM control chart. The engineer would like to construct an EWMA control chart for this data using $\lambda = 0.10$ and $L = 2.7$ (again, these values are not arbitrary; for more discussion on choosing these values, see the references mentioned previously).

Solution. Recall from the previous example that the target process mean is $\mu_0 = 160$ with $\sigma = 3.2$. With $L = 2.7$ and $\lambda = 0.10$, the EWMA statistic from Equation 8 is

$$\begin{aligned} z_i &= \lambda x_i + (1 - \lambda)z_{i-1} \\ &= 0.10x_i + (1 - 0.10)z_{i-1} \\ &= 0.10x_i + 0.90z_{i-1} \end{aligned}$$

To illustrate the EWMA statistic for each observation, consider the first observation, $x_1 = 156.96$. If we initialize the process using $z_0 = \mu_0 = 160$, z_1 is found to be

$$\begin{aligned} z_1 &= 0.10(156.96) + 0.90z_0 \\ &= 0.10(156.96) + 0.90(160) \\ &= 159.70 \end{aligned}$$

The remaining EWMA statistics are calculated accordingly and displayed in Table 4.10 along with the original data from Example 8.

TABLE 4.10
Machining Data for Example 9

<i>Observation</i>	<i>Measurement</i>	
	<i>x_i</i>	<i>z_i</i>
1	156.96	159.70
2	164.23	160.15
3	163.44	160.48
4	162.13	160.64
5	161.41	160.72
6	161.77	160.83
7	153.96	160.14
8	169.94	161.12
9	157.99	160.81
10	161.89	160.91
11	162.24	161.05
12	167.51	161.69
13	159.83	161.51
14	157.91	161.15
15	160.71	161.10
16	166.23	161.62
17	160.11	161.47
18	164.02	161.72
19	159.95	161.54
20	165.87	161.98
21	154.26	161.20
22	159.05	160.99
23	162.56	161.15
24	165.97	161.63
25	161.46	161.61
26	160.32	161.48
27	157.13	161.05
28	159.23	160.87
29	160.40	160.82
30	157.84	160.52

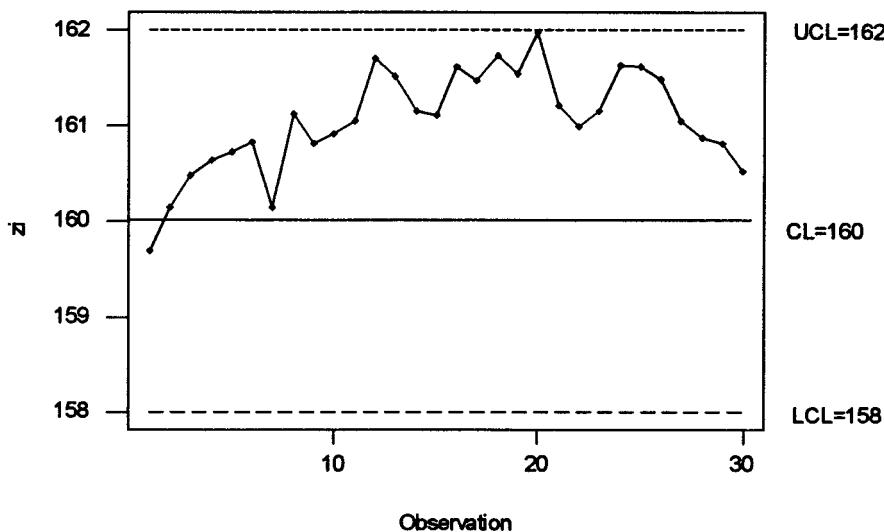


FIG. 4.11. EWMA control chart for Example 9.

The control limits are

$$\text{UCL} = \mu_0 + L\sigma \sqrt{\left(\frac{\lambda}{2 - \lambda}\right)} = 160 + 2.7(3.2)\sqrt{\left(\frac{0.10}{2 - 0.10}\right)} = 162$$

$$\text{Centerline} = 160$$

$$\text{LCL} = \mu_0 - L\sigma \sqrt{\left(\frac{\lambda}{2 - \lambda}\right)} = 160 - 2.7(3.2)\sqrt{\left(\frac{0.10}{2 - 0.10}\right)} = 158$$

The control limits and the EWMA statistics, z_i , are plotted on the EWMA control chart displayed in Fig. 4.11. The process appears to be in control because all EWMA statistics fall within the control limits.

Choice of Control Charting Techniques

In this chapter, Shewhart control charts, CUSUM control charts, and EWMA control charts are presented with definitions and examples. The decision as to which control-charting method should be used for a particular situation depends on the process under investigation. Shewhart control charts are easy to construct and implement, but they are less effective at detecting shifts in the process that may be quite small. It also has been shown that Shewhart control charts are very sensitive to the assumption of normality (see Borror, Montgomery, & Runger, 1999). That is, if the underlying distribution of the process is not normally distributed, then Shewhart control charts often signal out-of-control, when in fact the process is in control—this is referred to as a false alarm.

CUSUM and EWMA control charts, on the other hand, are quite robust to departures of the data from normality and tend to be better at detecting small shifts in the process mean. To demonstrate the difference between Shewhart control chart performance and the CUSUM and EWMA performances, consider the following example.

Example 10. In a particular manufacturing process a shift of 0.5σ in the target mean must be detected quickly and investigated. Two hundred random samples of the product have been selected and control charts applied to the process. To illustrate the superiority of the CUSUM and EWMA monitoring techniques over the Shewhart control charts, it is known that a shift in the process mean has occurred at the 100th observation. A good monitoring scheme will signal out of control soon after the 100th observation has occurred. In this example all three techniques have been applied to process. The resulting control charts are given in Fig. 4.12. In comparing the three charting techniques, it is obvious that the CUSUM and EWMA signaled within two observations after the 0.5σ shift occurred in the process target. The Shewhart control chart in Fig. 4.12a did not signal at all after the shift occurred. In this situation the EWMA and CUSUM are preferable to the Shewhart chart for detecting a very small shift in the process target.

Average Run Length

Several control-charting techniques are presented in this chapter. In this subsection the concept of the average run length (ARL) of a control chart is presented.

The ARL can be used to evaluate the performance of the control chart or to determine the appropriate values of various parameters for the control charts presented in this chapter. For example, to determine the most appropriate values of λ and L for the EWMA chart, we can state the acceptable value of the ARL and determine λ and L for a particular situation.

By definition, the ARL is the expected number of samples taken before an out-of-control signal appears on a control chart. In general, the average run length is

$$ARL_0 = \frac{1}{p}$$

where p is the probability of any one point exceeding the control limits. If the process is in control but an out-of-control signal is given, then a false alarm has occurred. For example, consider the \bar{x} control chart with the standard 3σ limits (presented earlier). It can be shown that for this Shewhart control chart, $p = 0.0027$ is the probability that a single point plotted on the \bar{x} control chart lies beyond the control limits when the process is in control. The in-control average run length for the \bar{x} control chart is

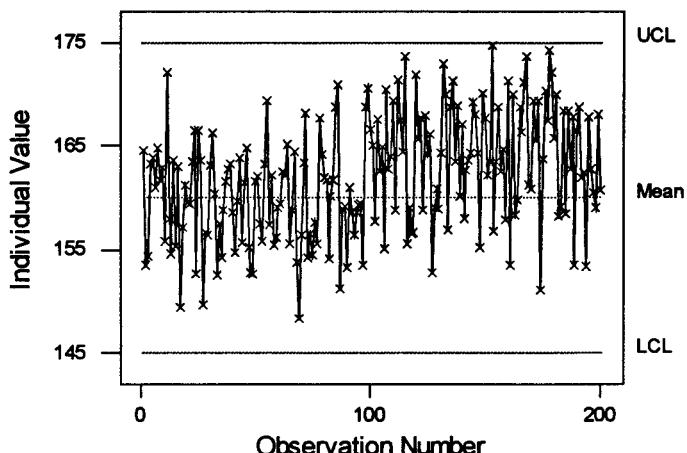
$$ARL_0 = \frac{1}{p} = \frac{1}{0.0027} = 370$$

That is, an out-of-control signal (or false alarm) is expected to occur every 370 samples, even if the process is actually in control. This would indicate that a false alarm would occur only every 370 samples or observations taken. Generally, a large value of the ARL is desired if the process is in control.

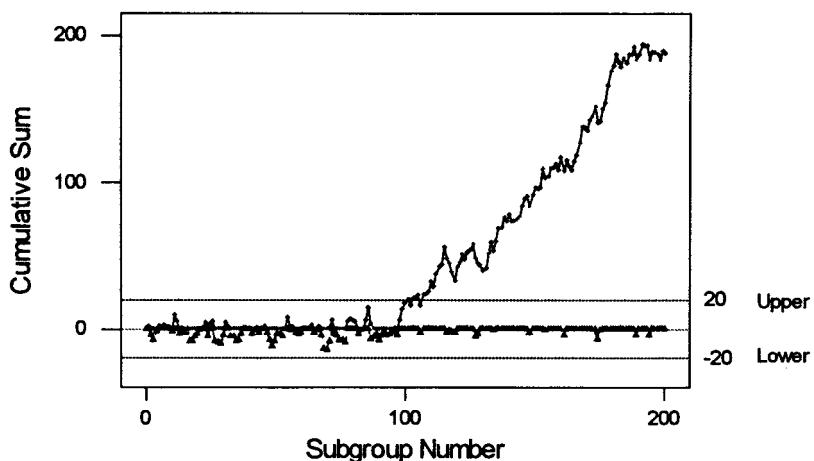
Also, if the process is actually out of control, then a small ARL value is desirable. A small ARL shows that the control chart will signal out-of-control soon after the process has shifted. The out-of-control ARL is

$$ARL_1 = \frac{1}{1 - \beta}$$

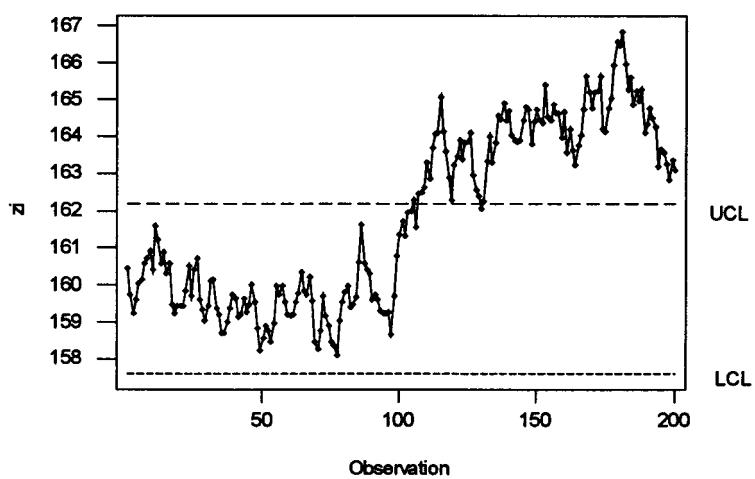
where β is the probability of not detecting a shift on the first sample after a shift has occurred. In other words, ARL_1 is the expected number of samples observed before a shift is detected.



(a)



(b)



(c)

FIG. 4.12. (a) Shewhart control chart. (b) CUSUM control chart. (c) EWMA control chart.

In general, the in-control ARL corresponds to the concept of a false alarm, whereas the out-of-control ARL can be related to the concept of a delay in the detection of an anomaly.

Average run lengths have been calculated for each of the control charting techniques discussed in this section. The calculations are carried out using knowledge of the underlying distribution of the data and the shift sizes of interest to detect. For more discussion on the calculation and use of ARLs for control charts see Hawkins (1993), Lucas and Saccucci (1990), and Montgomery (2001).

MULTIVARIATE CONTROL CHARTS

In the previous section, univariate control charting techniques were discussed. There are many situations in which several related quality characteristics are to be monitored simultaneously. In this section, two common multivariate techniques are discussed. The interested reader should see Jackson (1985); Crosier (1988); Hawkins (1993); Lowry, Woodall, Champ, & Rigidon (1992); Lowry & Montgomery (1995); Pignatiello & Runger (1990); Mason, Tracy, & Young (1995); and Montgomery (2001) for more details on the methods to be presented.

Data Description

When two or more quality characteristics are to be investigated, it is quite possible that there is some relationship (or correlation) between two or more of the variables. For example, the inside diameter and outside diameter of copper tubing may be two quality characteristics that must be monitored in the manufacturing process. Univariate control charts could be constructed for each quality characteristic (variables) individually, but the univariate chart may not capture the relationship between the variables being measured. In fact, if there are relationships between the variables, then univariate charts applied to the variables may give the wrong information, that is, the process may actually be out-of-control, but the univariate charts do not signal out-of-control. This type of phenomenon can occur when there is a correlation among the quality characteristics of interest, but they are investigated and monitored individually. For more information on multivariate data see Montgomery (2001).

Hotelling T^2 Control Chart

The first multivariate control chart to be presented is the Hotelling T^2 control chart. First presented by Hotelling (1947), multivariate quality control has gained a great deal of attention and a number of applications. The underlying distribution of the data that can be effectively monitored by multivariate control charts is the multivariate normal distribution. The multivariate normal distribution is simply the multiple variable counterpart to the normal distribution for a single variable.

The Hotelling T^2 control chart itself can be thought of as the multivariate counterpart to the Shewhart \bar{x} control chart. This control chart can be adapted to subgroup data or individual data.

Suppose that x_1, x_2, \dots, x_p are p quality characteristics that are to be monitored simultaneously. The statistic that is plotted and monitored is

$$T^2 = n(\bar{\mathbf{x}} - \bar{\mathbf{x}})' \mathbf{S}^{-1} (\bar{\mathbf{x}} - \bar{\mathbf{x}}) \quad (9)$$

where $\bar{\mathbf{x}}$ is the vector of sample means; $\bar{\mathbf{x}}$ is the vector of in-control means (estimated from the

process data); and \mathbf{S} is the averaged covariance matrix for the quality characteristics when the process is in control. In the case of individual observations ($n = 1$), the statistic to be plotted is

$$T^2 = (\mathbf{x} - \bar{\mathbf{x}})' \mathbf{S}^{-1} (\mathbf{x} - \bar{\mathbf{x}})$$

where $\bar{\mathbf{x}}$ is the vector of sample means, and \mathbf{S} is the averaged covariance matrix for the quality characteristics when the process is in control.

The construction of the control limits and statistics to be plotted is detailed and beyond the scope of this chapter. Details and implementation of the chart can be found in any of the references provided earlier in this section. An illustration of the multivariate control chart and its superiority over multiple univariate control charts will demonstrate the usefulness of these monitoring techniques in practice.

Example 11. Measurements on the inside and outside diameters of a particular copper tubing have been collected and are presented in Table 4.11 in coded form. The engineer wants to determine if the tube manufacturing process is in control.

Solution. Univariate control charts could easily be constructed for each quality characteristic individually. Specifically, the Shewhart control chart for individuals discussed

TABLE 4.11
Diameter Data for Example 11

Tube No.	Inside Diameter, x_1	Outside Diameter, x_2
1	27.2863	34.3854
2	22.2851	43.9282
3	30.8319	30.0322
4	29.0466	44.6336
5	25.6187	42.2535
6	29.6206	45.3383
7	27.3497	44.3806
8	23.6734	39.4534
9	29.3292	45.6672
10	28.6392	30.2690
11	25.3458	27.9130
12	26.1724	38.8171
13	25.6517	44.4404
14	23.6246	44.3296
15	26.9687	41.9942
16	28.1287	43.4413
17	24.5695	44.0917
18	27.2852	38.6781
19	23.5924	43.7138
20	29.1046	35.2320
21	30.0604	40.1509
22	25.1255	32.8442
23	18.0000	32.1295
24	26.5506	31.9827
25	25.9848	43.3962
26	23.4975	40.2245
27	23.7466	49.7419
28	28.0258	37.4064
29	24.3527	44.0640
30	22.5746	34.1724

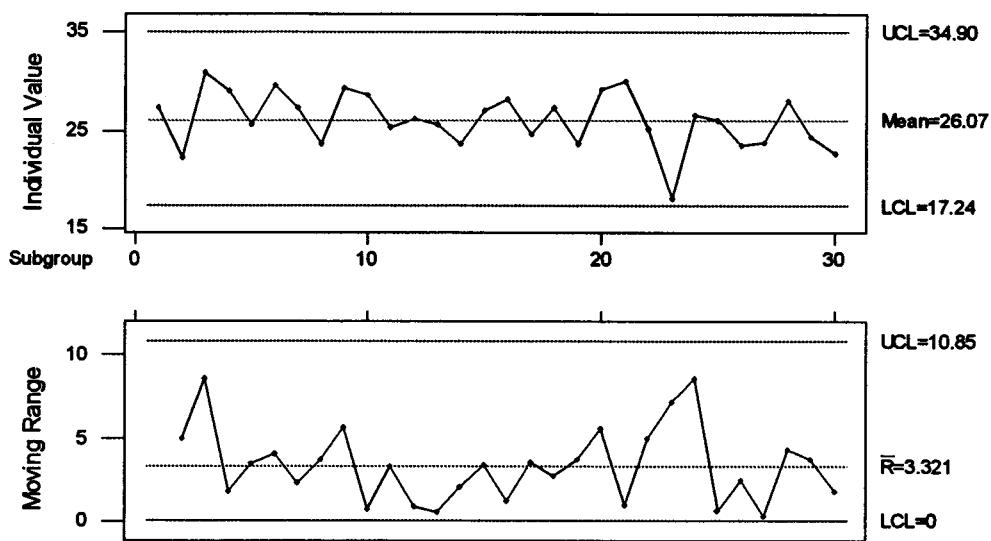


FIG. 4.13. Shewhart individuals and MR control charts for inside diameter.

previously can be constructed for both quality characteristics. The resulting control charts are displayed in Figs. 4.13 and 4.14. Based on the individuals control charts, the process appears to be in statistical control with regard to both inside diameter and outside diameter.

If only the univariate control charts were constructed, the process would be considered in control. But examine the Hotelling T^2 control chart created in Statgraphics (2000) and displayed in Fig. 4.15. It is obvious that the process is out of control because an observation has plotted beyond the UCL. This is because Hotelling's T^2 technique capitalizes on the relationship between the two variables, if there is one.

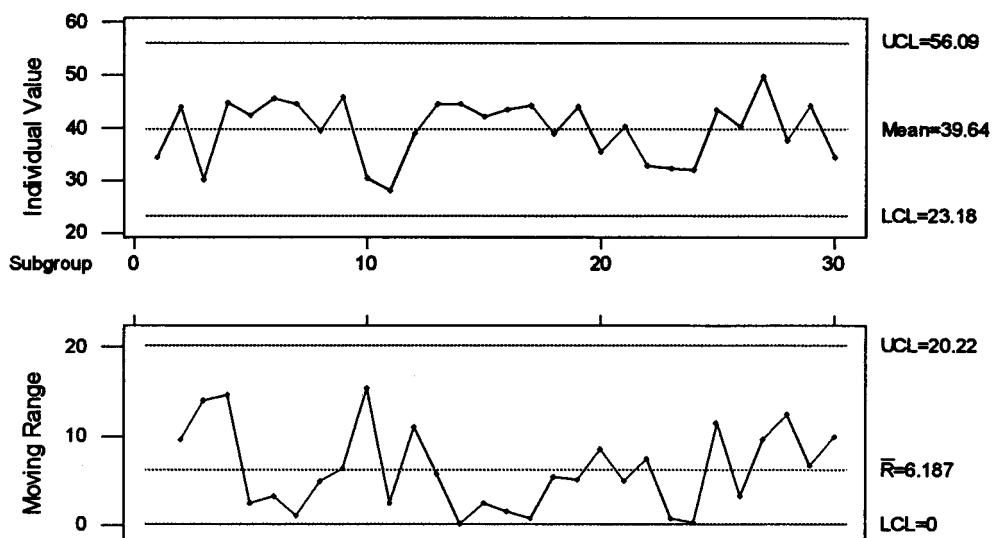


FIG. 4.14. Shewhart individuals and MR control charts for outside diameter.

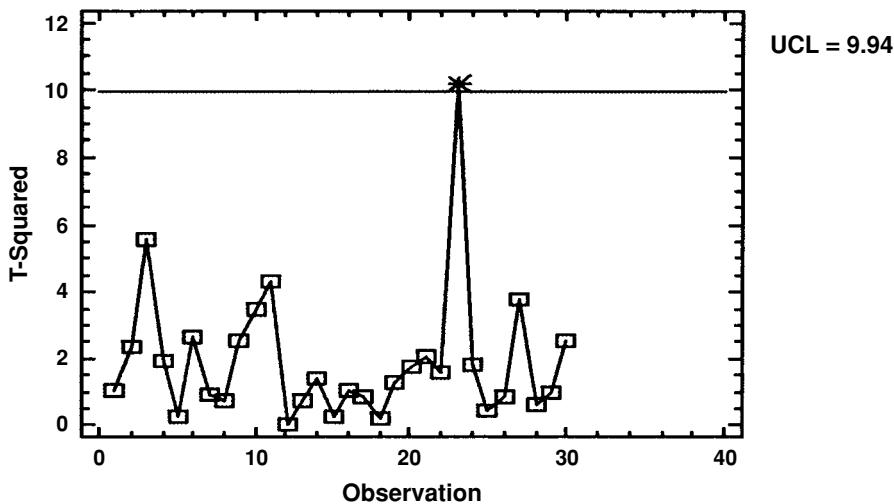


FIG. 4.15. Hotelling's T^2 control chart for inside and outside diameters.

In this case the multivariate control chart was much more informative than the univariate control charts used simultaneously. When monitoring multiple quality characteristics, multivariate control chart techniques are the most appropriate methods for identifying out-of-control processes.

Multivariate EWMA Control Charts

One of the drawbacks to Hotelling's T^2 control chart is also a drawback identified for the Shewhart univariate control charts previously—that only the current observation or subgroup information is used. Again, these charts are often insensitive to small shifts in the target process mean. An alternative to Hotelling's T^2 method is the multivariate analog to the univariate EWMA control chart.

The EWMA control chart for a single quality characteristic can be extended to multivariate situations. Lowry, Woodall, Champ, and Rigdon (1992) developed the multivariate EWMA (MEWMA) control chart, and several other authors have discussed the ARL properties of these charts. The MEWMA statistic is given by

$$\mathbf{Z}_i = \lambda \mathbf{x}_i + (1 - \lambda) \mathbf{Z}_{i-1}$$

where \mathbf{Z} and \mathbf{x} are in bold to represent vectors. That is, \mathbf{x}_i represents the i th observation in the dataset. For example, consider the diameter data given in Table 4.11. For this data $\mathbf{x}_1 = (27.2863, 34.3854)$ representing the first copper tube.

For the MEWMA the quantity plotted on the control chart is given by

$$\mathbf{T}_i^2 = \mathbf{Z}'_i \boldsymbol{\Sigma}_{z_i}^{-1} \mathbf{Z}_i$$

where

$$\boldsymbol{\Sigma}_{z_i} = \frac{\lambda}{2 - \lambda} [1 - (1 - \lambda)^{2i}] \boldsymbol{\Sigma}$$

is the covariance matrix (the multivariate analog of the variance of the univariate EWMA).

Prabhu and Runger (1997) provide tables and charts for the control limits and various values of the weighting factor λ . The reader is encouraged to see Lowry, Woodall, Champ, & Rigdon (1992), Prabhu and Runger (1997), or Montgomery (2001) for more information on the multivariate EWMA control chart.

SUMMARY

The importance of monitoring and controlling a process cannot be overstated. In this chapter several control-charting techniques have been presented for use in process control and monitoring. The techniques can easily be adapted for use in anomaly detection.

The standard Shewhart control charts are useful for situations in which small shifts in the process average (say less than 1.5σ) are not as important to detect as quickly as large shifts. For small shifts in the process mean the CUSUM or EWMA control charts have better performance properties in terms of the in-control and out-of-control average run lengths.

For multivariate processes Hotelling's T^2 control chart and MEWMA control chart are presented. These multivariate-monitoring techniques are superior to simultaneous implementation of univariate control charts for processes involving two or more process variables.

REFERENCES

- Borror, C., Montgomery, D., & Runger, G. (1999). Robustness of the EWMA control chart to nonnormality. *Journal of Quality Technology*, 31, 309–316.
- Crosier, R. (1988). Multivariate generalizations of cumulative sum quality control schemes. *Technometrics*, 30, 291–303.
- Crowder, S. (1989). Design of exponentially weighted moving average schemes. *Journal of Quality Technology*, 21, 155–162.
- Devore, J. (2000). *Probability and statistics for engineering and the sciences* (5th ed.). Pacific Grove, CA: Duxbury.
- Gan, F. F. (1991). An optimal design of CUSUM quality control charts. *Journal of Quality Technology*, 23, 279–286.
- Hawkins, D. (1981). A CUSUM for a Scale Parameter. *Journal of Quality Technology*, 13, 228–231.
- Hawkins, D. (1993). Cumulative sum control charting: An underutilized SPC tool. *Quality Engineering*, 5, 463–477.
- Hotelling, H. (1947). Multivariate quality control. In C. Eisenhart, M. W. Hastay, & W. A. Wallis, (Eds.), *Techniques of statistical analysis*. New York: McGraw-Hill.
- Jackson, J. (1985). Multivariate quality control. *Communications in Statistics—Theory and Methods*, 34, 2657–2688.
- Lowry, C., & Montgomery, D. (1995). A review of multivariate control charts. *IIE Transactions*, 26, 800–810.
- Lowry, C., Woodall, W., Champ, C., & Rigdon, S. (1992). A multivariate exponentially weighted moving average control chart. *Technometrics*, 34, 46–53.
- Lucas, J. (1976). The design and use of cumulative sum quality control schemes. *Journal of Quality Technology*, 31, 1–12.
- Lucas, J., & Saccucci, M. (1990). Exponentially weighted moving average control schemes: properties and enhancements. *Technometrics*, 32, 1–29.
- Mason, R., Tracy, N., & Young, J. (1995). Decomposition of T^2 for multivariate control chart interpretation. *Journal of Quality Technology*, 27, 99–108.
- Montgomery, D. (2001). *Introduction to statistical quality control* (4th ed.). New York: Wiley.
- Montgomery, D., & Runger, G. (2002). *Applied statistics and probability for engineers* (3rd ed.). New York: Wiley.
- Pignatiello, J., & Runger, G. C. (1990). Comparison of multivariate CUSUM charts. *Journal of Quality Technology*, 22, 173–186.
- Prabhu, S., & Runger, G. (1997). Designing a multivariate EWMA control chart. *Journal of Quality Technology*, 29, 8–15.
- Roberts, S. (1959). Control chart tests based on geometric moving averages. *Technometrics*, 1, 239–250.
- Woodall, W. H., & Adams, B. M. (1993). The statistical design of CUSUM charts. *Quality Engineering*, 5, 559–570.

5

Bayesian Data Analysis

David Madigan
Rutgers University

Greg Ridgeway
The RAND Corporation

Introduction	104
Fundamentals of Bayesian Inference	104
A Simple Example	104
A More Complicated Example	106
Hierarchical Models and Exchangeability	109
Prior Distributions in Practice	111
Bayesian Model Selection and Model Averaging	113
Model Selection	113
Model Averaging	114
Model Assessment	114
Bayesian Computation	115
Importance Sampling	115
Markov Chain Monte Carlo (MCMC)	116
An Example	117
Application to Massive Data	118
Importance Sampling for Analysis of Massive Data Sets	118
Variational Methods	120
Bayesian Modeling	121
BUGS and Models of Realistic Complexity via MCMC	121
Bayesian Predictive Modeling	125
Bayesian Descriptive Modeling	127
Available Software	128
Discussion and Future Directions	128
Summary	128

Acknowledgments	129
References	129

INTRODUCTION

The Bayesian approach to data analysis computes conditional probability distributions of quantities of interest (such as future observables) given the observed data. Bayesian analyses usually begin with a *full probability model*—a joint probability distribution for all the observable and unobservable quantities under study—and then use Bayes’ theorem (Bayes, 1763) to compute the requisite conditional probability distributions (called *posterior distributions*). The theorem itself is innocuous enough. In its simplest form, if Q denotes a quantity of interest and D denotes data, the theorem states:

$$p(Q | D) = p(D | Q) \times p(Q)/p(D).$$

This theorem prescribes the basis for statistical learning in the probabilistic framework. With $p(Q)$ regarded as a probabilistic statement of prior knowledge about Q before obtaining the data D , $p(Q | D)$ becomes a revised probabilistic statement of our knowledge about Q in the light of the data (Bernardo & Smith, 1994, p. 2). The marginal likelihood of the data, $p(D)$, serves as normalizing constant.

The computations required by Bayes’ theorem can be demanding, especially with large data sets. In fact, widespread application of Bayesian data analysis methods has occurred only in the last decade or so, having had to wait for computing power as well as breakthroughs in simulation technology. Barriers still exist for truly large-scale applications.

The primary advantages of the Bayesian approach are its conceptual simplicity and the commonsense interpretation of Bayesian outputs. The ability to incorporate prior knowledge can also be a boon. Many data mining applications provide copious data, but for models with thousands if not millions of dimensions or parameters, a limited amount of prior knowledge, often in the form of prior exchangeability information, can sharpen inferences considerably. Perhaps more commonly, though, the available data simply swamps whatever prior knowledge is available, and the precise specification of the prior becomes irrelevant.

The second section of this chapter uses a series of examples to introduce the basic elements of the Bayesian approach. Following sections describe strategies for Bayesian computation and model building. The remaining sections discuss some specific applications and describe currently available software.

FUNDAMENTALS OF BAYESIAN INFERENCE

A Simple Example

Consider estimating the “sex ratio,” that is, the proportion of births that are female, in a specific population of human births. Demographers have studied sex ratios for centuries, and variations across subpopulations continue to attract research attention. In 1781 the illustrious French scientist Pierre Simon Laplace presented a Bayesian analysis of the sex ratio (Stigler, 1990). He used data concerning 493,472 Parisian births between 1745 and 1770. Let $n = 493,472$ and y_1, \dots, y_n denote the sex associated with each of the births, $y_i = 1$ if the i th birth is

female $y_i = 0$ is the i th birth is male. For the Parisian data $\sum y_i = 241,945$, and we denote this by y . Denote by θ the probability that a given birth is female. Assuming that the births represent independent and identically distributed Bernoulli trials, we can compute the posterior distribution as:

$$p(\theta | y_1, \dots, y_n) \propto \theta^y (1 - \theta)^{n-y} p(\theta), \quad \theta \in [0, 1].$$

Ultimately we will want to use this posterior distribution to compute probabilities such $\Pr(\theta \geq \frac{1}{2})$ or $\Pr(0.49 \leq \theta \leq 0.50)$. However, to do this we will need to compute the normalizing constant on the right-hand side, that is,

$$\int \theta^y (1 - \theta)^{n-y} p(\theta) d\theta,$$

and we are getting a first glimpse as to why computation looms large in Bayesian analysis. Because the integral here is one-dimensional, it yields a straightforward numerical approximation irrespective of the precise choice of $p(\theta)$ (see the section headed Bayesian Computation). Laplace circumvented the integration by using a uniform prior distribution for θ (i.e., $p(\theta) = 1$, $\theta \in [0, 1]$) leading to:

$$p(\theta | y_1, \dots, y_n) \propto \theta^{241945} (1 - \theta)^{251527}, \quad \theta \in [0, 1].$$

This is the density of the so-called beta distribution and, in this case, the requisite normalizing constant is available in closed form:

$$p(\theta | y_1, \dots, y_n) = \frac{\Gamma(493474)}{\Gamma(241946)\Gamma(251528)} \theta^{241945} (1 - \theta)^{251527}, \quad \theta \in [0, 1].$$

The posterior mean and standard deviation are 0.490291 and 0.000712, respectively. Could the true probability be as big as a half? That was the scientific question that Laplace addressed, and the posterior density above yields:

$$\Pr\left(\theta \geq \frac{1}{2}\right) \approx 1.521 \times 10^{-42}.$$

Laplace concluded that it is “morally certain” that the true probability is indeed less than a half. Note that this analysis, unlike a classical p value, provides a direct probabilistic answer to the primary question of interest.

What about prediction? Denote by y_{n+1} the sex associated with the next birth. We compute the predictive density as follows:

$$p(y_{n+1} | y_1, \dots, y_n) = \int p(y_{n+1} | \theta) p(\theta | y_1, \dots, y_n) d\theta.$$

Note that we have used here the fact the θ renders y_{n+1} independent of y_1, \dots, y_n . Again, because we are dealing with a one-dimensional parameter, this integral is manageable. With Laplace’s particular choice of prior the predictive distribution takes an especially simple form:

$$\Pr(y_{n+1} = 1 | y_1, \dots, y_n) = \frac{y + 1}{n + 2} = 0.490291.$$

Laplace’s choice of the uniform prior led to closed-form expressions for the posterior and the predictive distributions. He could more generally have chosen any beta distribution as the

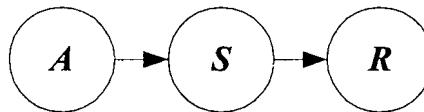


FIG. 5.1. Down syndrome: an acyclic directed probabilistic graphical model.

prior distribution for θ and still have closed-form outputs. The general form of the beta density for a θ is:

$$p(\theta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}, \quad \theta \in [0, 1].$$

We refer to this as a beta(α, β) distribution where α and β are *hyperparameters*. This leads to a beta($\alpha + y, \beta + n - y$) posterior distribution. For Bernoulli/binomial data the beta distribution is said to be the *conjugate prior distribution*. Conjugate priors stay in the same distributional family as they go from prior to posterior. Convenient conjugate priors are available for many standard distributions and, as long as they are flexible enough to accurately represent whatever prior knowledge is available, it often makes sense to use a conjugate prior. The BUGS manual (Spiegelhalter, Thomas, & Best, 1999) provides a list of distributions and their associated conjugate priors. Gelman, Carlin, Stern, and Rubin (1995) also analyze the example just described and provide a comprehensive introduction to Bayesian methods in general.

A More Complicated Example

The use of graphs to represent statistical models has a rich history dating back at least to the 1920s. Recently, probabilistic graphical models have emerged as an important class of models and have impacted fields such as data mining, causal analysis, and statistical learning. A probabilistic graphical model is a multivariate probabilistic model that uses a graph to represent a set of conditional independences. The vertices of the graph represent the random variables of the model, and the edges encode the conditional independences. In general, each missing edge corresponds to a conditional independence. Graphs with different types of edges—directed, undirected, or both—lead to different classes of probabilistic models. In what follows we consider only acyclic directed models, also known as *Bayesian networks*¹ (see, e.g., Pearl, 1988).

Spiegelhalter and Lauritzen (1990) presented a Bayesian analysis of acyclic directed probabilistic graphical models, and this topic continues to attract research attention. Here we sketch the basic framework with a stylized version of a real epidemiological application.

In Norway the Medical Birth Registry (MBR) gathers data nationwide on congenital malformations such as Down syndrome. The primary purpose of the MBR is to track prevalences over time and identify abnormal trends. The data, however, are subject to a variety of errors, and epidemiologists have built statistical models to make inference about true prevalences. For Down syndrome, one such model includes three dichotomous random variables: the reported Down syndrome status, R , the true Down syndrome status, S , and the maternal age, A , where age is dichotomized at 40, say.

Figure 5.1 displays a possibly reasonable model for these variables. This acyclic directed graph represents the assumption that the reported status and the maternal age are conditionally independent given the true status. The joint distribution of the three variables factors

¹This is somewhat of a misnomer because there is nothing Bayesian per se about Bayesian networks.

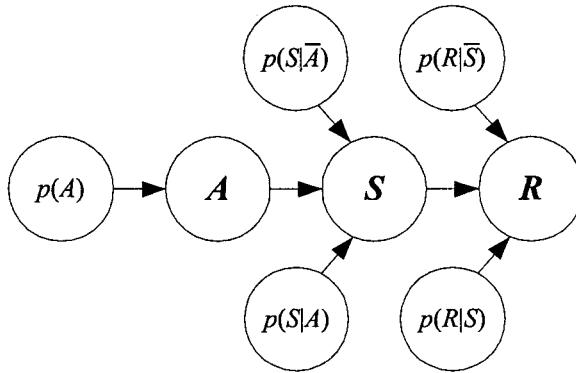


FIG. 5.2. Down syndrome: an acyclic directed Bayesian probabilistic graphical model.

accordingly:

$$\Pr(A, S, R) = \Pr(A)\Pr(S | A)\Pr(R | S). \quad (1)$$

This factorization features a term for every vertex, the term being the conditional density of the vertex given its parents. In general, this factorization implies that vertices (more correctly, the random variables corresponding to vertices) are conditionally independent of their nondescendants given their parents (Lauritzen, Dawid, Larsen, & Leimer, 1990).

The specification of the joint distribution of A , S , and R , in Equation 1 requires five parameters:

$$\Pr(R | S), \Pr(R | \bar{S}), \Pr(S | A), \Pr(S | \bar{A}) \quad \text{and} \quad \Pr(A) \quad (2)$$

where \bar{S} , for example, denotes the absence of Down syndrome. Once these probabilities are specified, the calculation of specific conditional probabilities such as $\Pr(R | A)$ can proceed via a series of local calculations without storing the full joint distribution (Dawid, 1992).

To facilitate Bayesian learning for the five parameters, Spiegelhalter and Lauritzen (1990) and Cooper and Herkovits (1992) make two key assumptions that greatly simplify subsequent analysis.

First, they assume that the parameters are independent a priori. Figure 5.2 embodies this assumption. For instance, $\Pr(S | A)$ in Fig. 5.2 has no parents. Therefore, it is marginally independent of, for instance, $\Pr(A)$, because this is not a descendant of $\Pr(S | A)$.

Second, they assume that each of the probabilities has a beta distribution (or Dirichlet distribution for categorical variables with more than two levels).

Calculation of the posterior distributions is straightforward. Suppose we have the following prior distributions for three of the parameters $\Pr(A) \sim \text{beta}(1,1)$, $\Pr(S | A) \sim \text{beta}(5,1)$, and $\Pr(R | \bar{S}) \sim \text{beta}(1,9)$, and we observe a single case, $d = (A, \bar{S}, R)$, that is, maternal age over 40 and incorrectly recorded by the MBR as Down syndrome. Then it is easy to show that $\Pr(d) = \frac{1}{1+1} \times \frac{1}{5+1} \times \frac{1}{1+9}$. Conditional on d , the posterior distributions of these parameters become: $\Pr(A) \sim \text{beta}(2,1)$, $\Pr(S | A) \sim \text{beta}(5,2)$, and $\Pr(R | \bar{S}) \sim \text{beta}(2,9)$. The posterior distributions of the remaining two parameters, $\Pr(S | \bar{A})$ and $\Pr(R | S)$, are unchanged. In this manner we can sequentially calculate the likelihood for a collection of cases, D , conditional on the model of Fig. 5.2. Heckerman, Geiger, and Chickering (1994) provide a closed-form expression for this likelihood.

TABLE 5.1
Down Syndrome Data for 1985–1988

<i>Doubly Sampled Data</i>							
	<i>A</i> ₁	<i>A</i> ₂	<i>A</i> ₃	<i>A</i> ₄	<i>A</i> ₅	<i>A</i> ₆	<i>Total</i>
<i>R</i> ₁ , <i>R</i> ₂	1	2	0	3	2	0	8
<i>R</i> ₁ , <i>R</i> ₂	5	2	3	0	2	1	13
<i>R</i> ₁ , <i>R</i> ₂	1	4	2	1	1	0	9
<i>R</i> ₁ , <i>R</i> ₂	7478	10247	7058	2532	504	28	27847
Total	7485	10255	7063	2536	509	29	27877

<i>Singly Sampled Data</i>							
	<i>A</i> ₁	<i>A</i> ₂	<i>A</i> ₃	<i>A</i> ₄	<i>A</i> ₅	<i>A</i> ₆	<i>Total</i>
<i>R</i> ₁	32	55	58	62	23	3	233
<i>R</i> ₁	48957	70371	49115	16834	3348	165	188790
Total	48989	70426	49173	16896	3371	168	189023

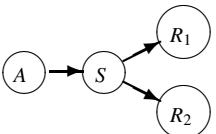
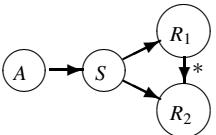
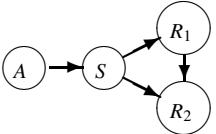
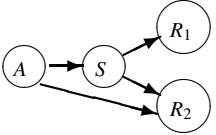
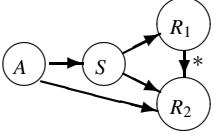
Note: R_1 represents case ascertainment through the national MBR registry and R_2 through the regional MIA registry. A represents the maternal age in six categories: ≤ 24 , 25–29, 30–34, 35–39, 40–44, and ≥ 45 .

We now discuss the actual Norwegian Down syndrome example in more detail. Because of growing concerns about incomplete ascertainment, an additional notification system titled “Melding om Fosterindiserte Aborter og Medfødte Misdannelser” (MIA) was introduced in 1985 in the county of Hordaland covering about 15% of all births in Norway. The MIA registration is based on prenatal diagnostics and pediatric follow-up including results from cytogenetic tests. Although it was expected that the MIA registration would be more accurate than the MBR, the MIA registration is subject to error. Table 5.1 presents data concerning Down syndrome collected between 1985 and 1988 (Lie, Heuch, & Irgens, 1991, 1994). The variables A and S continue to represent maternal age and true Down syndrome status, respectively. R_1 represents case ascertainment through the national MBR registry and R_2 through the regional MIA registry.

Denoting by Q the prevalence of Down syndrome, and by Y the observed data, the goal is to compute $\text{pr}(Q | Y)$. York, Madigan, Heuch, and Lie (1995) presented a Bayesian probabilistic graphical model analysis that addressed three substantive issues. First, there does exist prior knowledge about Down syndrome rates, and the analysis sought to incorporate this. Second, different models exist for these data that provide a reasonable fit to the data, but lead to Down syndrome prevalence estimates that are quite different. Third, the data are partially missing because R_2 is present for only a subset of the examples. Table 5.2 shows the results of analyses using various Bayesian probabilistic graphical models.

Each model used an informative prior distribution, based on historical data and expert knowledge for the various probabilities. Specifically, the parameters for the prior on $\text{pr}(S)$, the prevalence of Down syndrome, were based on raw data from the MBR registry for 1979–1984 without any adjustment for misclassification. During that period there were 0.97 observed cases per 1,000 births; we chose the $\text{beta}(0.0097, 9.9903)$ prior to have that rate as its expected value and so that our prior knowledge is equivalent to having observed 10 hypothetical births. The priors for the probability of a case being identified by the registries reflect our belief that the registries are more likely to find cases of the syndrome than not. For the national MBR registry we placed $\text{beta}(4,2)$ priors on $\text{pr}(R_1 | S)$ and $\text{pr}(R_2 | S)$. For the six parameters, $\text{pr}(S | A_i)$, we chose prior variances so that $\text{Var}(\text{pr}(S))$ is the same for all models, and prior expectations given by historical data for 1979–1984, which are 0.59, 0.59, 0.97, 3.04, 6.88, and 18.50 cases per

TABLE 5.2
Features of the Posterior Distribution for Down Syndrome Prevalence

Model	Posterior Probability	$10^3 \times \Pr(S)$		
		Mode	Mean	Std Dev
	0.282	1.81	1.92	0.292
	0.385	1.49	1.51	0.129
	0.269	1.60	1.70	0.252
	0.030	1.71	1.78	0.226
	0.016	1.50	1.52	0.129

Note: Prevalence is given as the rate per thousand. “Posterior Probability” is a normalized marginal likelihood. Only models with posterior probability larger than 0.01 are listed.

1,000 for age groups A_1, \dots, A_6 , as presented in Lie et al. (1991). York et al. (1995) describe the remaining prior distributions.

The analysis assumed that there were no false positives, which is reasonable in this context. Models with an asterisk on the R_1, R_2 link impose a special kind of dependence for which it is assumed that the MIA registry, R_2 , will find all cases missed by the national registry, R_1 . York et al. (1995) used a Markov chain Monte Carlo procedure to deal with the missing data (see the section headed Bayesian Computation).

Each model has an associated model score (“Posterior Probability” in Table 5.2) that reflects a tradeoff between how well the model fits the data and the complexity of the model. It is clear that different plausible models can lead to quite different inferences. This issue is discussed in some detail in the following section.

Hierarchical Models and Exchangeability

For a data miner the notion of “exchangeability” and the consequent models represent a powerful and underutilized tool. Here we introduce the basic idea by example.

TABLE 5.3
Hospital Mortality Data

	Hospital											
	A	B	C	D	E	F	G	H	I	J	K	L
No. of Ops (n)	27	148	119	810	211	196	148	215	207	97	256	360
No. of Deaths (x)	0	18	8	46	5	13	9	31	14	8	29	24

Table 5.3 presents data concerning mortality rates in $k = 12$ hospitals performing cardiac surgery in babies. The analytic goal here is to rank order the hospitals according to their true mortality rates. Hospital A has the lowest observed mortality rate (0), but has performed the fewest surgeries (27). Should the analysis rank Hospital A as number one?

Denote by r_i the true mortality rate for Hospital i . A Bayesian analysis must specify prior distributions for each of the r_i 's. In the absence of any further information about the hospitals, these prior distributions must treat the r_i 's symmetrically. In particular, in this situation the Bayesian analysis would specify identical marginal prior distributions for each of the rates:

$$p(r_i) \equiv p(r_j) \forall i, j$$

as well as identical marginal joint prior distributions for all pairs of rates:

$$p(r_i, r_{i'}) \equiv p(r_j, r_{j'}) \forall i, i', j, j',$$

all triples, and so forth. Probabilistically we can represent this symmetry through *exchangeability*: the parameters r_1, \dots, r_k are exchangeable if their joint distribution $p(r_1, \dots, r_k)$ is invariant to permutations of the indices $(1, \dots, k)$.

A remarkable result due to deFinetti (1931) says that in the limit as $k \rightarrow \infty$ any exchangeable sequence r_1, \dots, r_k can be written as:

$$p(r_1, \dots, r_k) = \int \left[\prod_{i=1}^k p(r_i | \phi) \right] p(\phi) d\phi$$

for some $p(\phi)$. So, exchangeability suggests that we assume that the parameters r_1, \dots, r_k comprise independent and identically distributed draws from some “mother distribution,” denoted here by $p(\phi)$. In the cardiac surgery example we could proceed as follows. Assume that within hospital i the number of deaths x_i has a binomial distribution with parameters n_i (this is known) and r_i (this is unknown). Next we assume that the r_i 's are independent and identically distributed where $\log(\frac{r_i}{1-r_i})$ is normally distributed with mean μ and precision τ . Finally, we put fairly flat prior distributions on μ and τ . The Bayesian literature refers to such models *hierarchical models*. Figure 5.3 shows a probabilistic graphical model representation.

Figure 5.3 uses some established graphical conventions. The box around n_i indicates that this is a known quantity. The box around n_i, r_i , and x_i is called a *plate*, indexed by $i = 1, \dots, k$ and is a shorthand for repeating each n, r , and x, k times. Every arrow entering the plate would also be repeated—for instance, there is an arrow from a to each of r_1, \dots, r_k .

A Markov chain Monte Carlo algorithm (see the section headed Bayesian Computation) computes posterior distributions for r_i 's. Table 5.4 shows the results.

The effect of the exchangeability assumption is to shrink the observed rates toward the overall average mortality rate (here 7.3%). For any given hospital the amount of shrinkage depends on the number of surgeries performed. For example, Hospital D performed 810 surgeries. The mean of the posterior distribution for Hospital D's rate (5.76%) is close to the raw rate (5.68%).

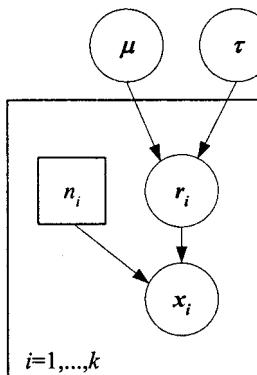


FIG. 5.3. Bayesian models from exchangeability.

Hospital A, on the other hand, performed only 27 surgeries. The posterior mean for Hospital A is 3.76% in contrast with raw rate of 0%. Indeed, ranking the hospitals according to their posterior means leads to the conclusion that Hospital E is the best hospital despite the fact that no deaths took place at Hospital A!

The phenomenon we are observing here is a sort of “borrowing strength.” The hospitals with a paucity of data borrow inferential strength from the hospitals with an abundance of data. The notion of exchangeability generalizes to partial exchangeability where observations form exchangeable subgroups (see, for example, Consonni & Veronese, 1995). Draper, Hodges, Mallows, & Pregibon (1993) provide an interesting discussion of the central role of exchangeability in data analysis (equally, in data mining).

Prior Distributions in Practice

We have already seen two examples involving the use of informative prior distributions. In practice, and especially in the context of large data sets, Bayesian analyses often instead use prior distributions that reflect some degree of prior indifference or ignorance. The basic requirement for such a prior distribution is that it be flat in the region of the parameter space favored by the likelihood. Certainly it is always advisable to make sure that the conclusions are robust to the particular choice of prior. Carlin and Louis (2000, section 6.1) provide a detailed discussion of different robustness analyses.

In an attempt to represent prior ignorance, some Bayesian analyses utilize so-called *improper priors*. These are priors that do not integrate to one but are uniform over, for instance, the real numbers. Hence, these improper priors are not probability distributions at all. In fact, improper priors can actually lead to “improper posteriors,” that is, posterior densities that do not integrate to one, an undesirable state of affairs. We recommend that data miners stick to proper prior distributions.

Concerning informative prior distributions, Chaloner (1996) surveys methodology for eliciting priors from human experts. Spiegelhalter, Freedman, and Parmar (1994) provide an exquisite example of pragmatic prior construction in the context of high-stakes decision making. Madigan, Gavrin, and Raftery (1995) describe an example where informative priors improved predictive performance.

The *empirical Bayes* approach learns the prior distribution from the data. Carlin and Louis (2000) provide a thorough introduction. DuMouchel (1999) and DuMouchel and Pregibon (2001) describe applications of empirical Bayes to frequent item set identification, both with methodology suitable for massive data sets.

TABLE 5.4
Hospital Mortality Rate Results (rates $\times 100$)

	<i>Hospital</i>											
	<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	<i>I</i>	<i>J</i>	<i>K</i>	<i>L</i>
No. of Ops (<i>n</i>)	27	148	119	810	211	196	148	215	207	97	256	360
Raw Rate (α/n)	0.00	12.16	6.72	5.68	2.37	6.63	6.08	14.42	6.76	8.25	11.33	6.67
Post. Mean	5.77	10.50	7.01	5.88	4.15	6.86	6.58	12.58	6.94	7.85	10.34	6.81
Post. SD	2.3	2.3	1.8	0.8	1.3	1.5	1.6	2.2	1.5	2.1	1.8	1.2
Raw Rank	1	11	7	3	2	5	4	12	8	9	10	6
Post. Rank	2	11	8	3	1	6	4	12	7	9	10	5

BAYESIAN MODEL SELECTION AND MODEL AVERAGING

The preceding example raised a challenging and ubiquitous issue: often there are many competing “full probability models” that may all seem reasonable in the light of the prior knowledge and of the data, but lead to different predictions.² The Bayesian literature provides two general strategies to deal with this issue. *Model selection* chooses the single “best” model and bases subsequent predictions on that model. *Model averaging* combines the models and computes predictions as weighted averages. Both approaches have strengths and weaknesses. Model selection is computationally more straightforward and provides a single model to examine and critique. However, predictions that condition on a single model ignore model uncertainty and can be badly calibrated. Model averaging can be computationally demanding and inscrutable but usually provide better predictions.

Model Selection

Suppose a particular analysis leads to set of candidate models $\{M_i, i \in I\}$ and denote by $D = \{x_1, \dots, x_n\}$ the observed data. The *posterior model probabilities* $\{p(M_i | D), i \in I\}$ play a central role in Bayesian model selection. If we believe that one of the models $\{M_i, i \in I\}$ actually generated the data, then, if our goal is to choose the true model, the optimal decision is to choose the model with the highest posterior probability (Bernardo & Smith, 1994, p. 389).

In practice, rather than trying to choose the “true” model, our goal is usually to make the best predictions possible. Furthermore, we usually do not believe that any of the models under consideration actually generated the data. Nonetheless, the posterior model probability is sometimes a reasonable choice as a model selection criterion (although not usually predictively optimal—see Barbieri & Berger, 2002). Note that:

$$p(M_i | D) \propto p(D | M_i) \times p(M_i)$$

so computing a posterior model probability involves two terms, the marginal likelihood and the prior model probability, as well as a normalization over the competing models. There exists a substantial literature on marginal likelihood computation—we refer the interested reader to section 6.3 of Carlin and Louis (2000) and to Han and Carlin (2001). Suffice it to say that the computation can sometimes prove challenging.

Bernardo and Smith (1984, section 6.1.6) argued for selecting the model that maximizes a cross-validated predictive score. For the case of the logarithmic score, this selects the model that maximizes:

$$\frac{1}{k} \sum_{j=1}^k \log p(x_j | M_i, \mathbf{x}_{n-1}(j))$$

where $\mathbf{x}_{n-1}(j)$ represents the data with observation x_j removed, $\{x_1, \dots, x_k\}$ represents a random sample from D , and the maximization is over $i \in I$.

For some applications computational complexity may rule out cross-validatory predictive model selection procedures. Carlin and Louis (2000, section 6.5.1) discussed alternatives. Spiegelhalter, Best, Carlin, and van der Linde (2002) proposed a model selection criterion

²Here we use the term “prediction” to mean inference for any unknown in the model, be it a future observable or not.

especially well suited to Bayesian hierarchical models and presented a decision-theoretic justification. See also Geisser (1993) and Shibata (1983).

Model Averaging

If we believe that one of the models $\{M_i, i \in I\}$ actually generated the data, then, if our goal is to minimize the squared loss for future predictions, the optimal strategy is to average over all models (Bernardo & Smith, 1994, p. 398). Specifically, if Q is the quantity of interest we wish to compute, Bayesian model averaging computes:

$$p(Q) = \sum_{i \in I} p(Q | D, M_i) p(M_i | D).$$

Bayesian model averaging is also optimal with regard to a logarithmic predictive score (Madigan & Raftery, 1994). As with model selection, we usually do not believe that one of the candidate models actually generated the data, but empirical evidence suggests that Bayesian model averaging usually provides better predictions than any single model (see, e.g., Hoeting, Madigan, Raftery, & Volinsky, 1999), sometimes substantially better. Predictive distributions from Bayesian model averaging usually have bigger variances, more faithfully reflecting the real predictive uncertainty. Draper (1995) provided examples in which failing to account for model uncertainty proved unfortunate.

Hoeting et al. (1999) discussed computing for Bayesian model averaging. Other references include Madigan and York (1995), Carlin and Chib (1995), and Carlin and Louis (2000, section 6.4).

Model Assessment

Both model selectors and model averagers need tools to assess model fit. Bayesian residuals provide a useful starting point:

$$r_i = y_i - E(y_i | D), \quad i = 1, \dots, m$$

where $\{y_1, \dots, y_m\}$ is a validation sample independent of the training data D , and E denotes expectation. Examination of these residuals can reveal failure in a distributional assumption or failure of some independence assumption, or whatever. Carlin and Louis (2000, section 2.4) discussed a cross-validatory approach, wherein the fitted value for y_i is computed conditional on all the data except y_i , namely, $\mathbf{y}_{(i)} \equiv (y_1, \dots, y_{i-1}, y_{i+1}, \dots, y_n)$, yielding:

$$r'_i = y_i - E(y_i | \mathbf{y}_{(i)}).$$

The *conditional predictive ordinate* (CPO), $p(y_i | \mathbf{y}_{(i)})$, is also quite useful. Individual data values having low a CPO are poorly fit by the model. Gelfand and Dey (1984) point out the interesting fact that

$$\frac{1}{p(y_i | \mathbf{y}_{(i)})} = \int \frac{1}{p(y_i | \mathbf{y}_{(i)}, \theta)} p(\theta | y) d\theta.$$

Hence, a Monte Carlo estimate of the harmonic mean of $p(y_i | \mathbf{y}_{(i)}, \theta)$ estimates the CPO. This is usually straightforward to compute using, for example, BUGS (see the later section on BUGS in this chapter). Harmonic means, however, can be numerically unstable.

Gelman et al. (1995, chap. 6) favored a technique that involves drawing simulated values from the posterior predictive distribution of replicated data and comparing these samples to the observed data using some discrepancy measure. Different choices of discrepancy measure can shed light on different aspects of model fit. We refer the reader to Gelman et al. for further details.

BAYESIAN COMPUTATION

Outputs of Bayesian data analyses often come in the form of expectations such as the marginal means, variances, and covariances of the quantity of interest. We compute the expected value of the quantity of interest, $h(\theta)$, using

$$E(h(\theta) | x_1, \dots, x_N) = \int h(\theta) f(\theta | x_1, \dots, x_N) d\theta \quad (3)$$

where $f(\theta | \mathbf{x})$ is the posterior distribution of the parameters given the observed data. Computation of these expectations requires calculating integrals that, for all but the simplest examples, are difficult to compute in closed form. Monte Carlo integration methods sample from the posterior, $f(\theta | \mathbf{x})$, and appeal to the law of large numbers to estimate the integrals,

$$\lim_{M \rightarrow \infty} \frac{1}{M} \sum_{i=1}^M h(\theta_i) = \int h(\theta) f(\theta | x_1, \dots, x_N) d\theta \quad (4)$$

where the θ_i compose a sample from $f(\theta | \mathbf{x})$.

The ability to compute these expectations efficiently is equivalent to being able to sample efficiently from $f(\theta | \mathbf{x})$. Sampling schemes are often difficult enough without the burden of large data sets. This burden usually causes each iteration of the Monte Carlo sampler to be slower. When the number of iterations already needs to be large, efficient procedures within each iteration are essential to timely delivery of results. So-called variational approximations can be useful for massive data sets beyond the reach of Monte Carlo methods, and we discuss those briefly at the end of this section.

Importance Sampling

Importance sampling is a general Monte Carlo method for computing integrals. As previously mentioned, Monte Carlo methods approximate integrals of the form in Equation 3. The approximation in Equation 4 depends on the ability to sample from $f(\theta | \mathbf{x})$. When a sampling mechanism is not readily available for the “target distribution,” $f(\theta | \mathbf{x})$, but one is available for another “sampling distribution,” $g(\theta)$, we can use importance sampling. Note that for Equation 3 we can write

$$\int h(\theta) f(\theta | x_1, \dots, x_N) d\theta = \int h(\theta) \frac{f(\theta | \mathbf{x})}{g(\theta)} g(\theta) d\theta \quad (5)$$

$$= \lim_{M \rightarrow \infty} \sum_{i=1}^M w_i h(\theta_i) \quad (6)$$

where θ_i is a draw from $g(\theta)$, and $w_i = f(\theta_i | \mathbf{x})/g(\theta_i)$. Note that the expected value of w_i under $g(\theta)$ is 1. Therefore, if we can compute the importance sampling weights, w_i , only up

to a constant of proportionality, we can normalize the weights to compute the integral.

$$\int h(\theta) f(\theta | x_1, \dots, x_N) d\theta = \lim_{M \rightarrow \infty} \frac{\sum_{i=1}^M w_i h(\theta_i)}{\sum_{i=1}^M w_i} \quad (7)$$

Naturally, for the sampling distribution to be useful, drawing from $g(\theta)$ must be easy. We also want our sampling distribution to be such that the limit converges quickly to the value of the integral. If the tails of $g(\theta)$ decay faster than $f(\theta | \mathbf{x})$, the weights will be numerically unstable. If the tails of $g(\theta)$ decay much more slowly than $f(\theta | \mathbf{x})$, we will frequently sample from regions where the weight will be close to zero, wasting computation time.

In Ridgeway and Madigan (2002) we show that when we set the sampling density to be $f(\theta | x_1, \dots, x_n)$, where $n \ll N$ so that we condition on a manageable subset of the entire data set, the importance weights for each sampled θ_i require only one sequential scan of the remaining observations. Before we begin that discussion, the next section introduces the most popular computational method for Bayesian analysis of complex models.

Markov Chain Monte Carlo (MCMC)

Importance sampling is a useful tool, but for complex models, crafting a reasonable sampling distribution can be extremely difficult. Gilks, Richardson, and Spiegelhalter (1996) provided a detailed introduction to MCMC along with a variety of interesting examples and applications.

As with importance sampling, the goal is to generate a set of draws from the posterior distribution $f(\theta | \mathbf{x})$. Rather than create independent draws and reweight, MCMC methods build a Markov chain, a sample of *dependent* draws, $\theta_1, \dots, \theta_M$, that have stationary distribution $f(\theta | \mathbf{x})$. It turns out that it is often easy to create such a Markov chain with a few basic strategies. However, there is still a bit of art involved in creating an efficient chain and assessing the chain's convergence.

Figure 5.4 shows the Metropolis-Hastings algorithm (Hastings, 1970), a very general MCMC algorithm. Assume that we have a single draw θ_1 from $f(\theta | \mathbf{x})$ and a proposal density for a new draw, $q(\theta | \theta_1)$. If we follow Step 2 of the MCMC algorithm, then the density of θ_2

1. Initialize the parameter θ_1

2. For i in $2, \dots, M$ do

Step (a) and/or (b) requires a scan of the dataset

(a) Draw a proposal θ' from $q(\theta | \theta_{i-1})$,

(b) Compute the acceptance probability

$$\alpha(\theta', \theta_{i-1}) = \min \left(1, \frac{f(\theta' | \mathbf{x}) q(\theta_{i-1} | \theta')}{f(\theta_{i-1} | \mathbf{x}) q(\theta' | \theta_{i-1})} \right) \quad (8)$$

(c) With probability $\alpha(\theta', \theta_{i-1})$ set $\theta_i = \theta'$.

Otherwise set $\theta_i = \theta_{i-1}$

FIG. 5.4. The Metropolis-Hastings algorithm.

will also be $f(\theta | \mathbf{x})$. This is one of the key properties of the algorithm. Iterating this algorithm, we obtain a sequence $\theta_1, \dots, \theta_M$ that has $f(\theta | \mathbf{x})$ as its stationary distribution.

MCMC methods have two main advantages that make them useful for Bayesian analysis. First, we can choose q 's from which it is easy to simulate. Special choices for q , which can depend on the data, simplify the algorithm. If q is symmetric—for example, a Gaussian centered on θ_{i-1} —then the entire proposal distributions cancel out in Equation 8. If we choose a q that proposes values that are very close to θ_{i-1} , then it will almost always accept the proposal, but the chain will move very slowly and take a long time to converge to the stationary distribution. If q proposes new draws that are far from θ_{i-1} and outside the region with most of the posterior mass, the proposals will almost always be rejected, and again the chain will converge slowly. With a little tuning the proposal distribution can usually be adjusted so that proposals are not rejected or accepted too frequently. Essentially, the only constraint on the choice of q is that it results in an irreducible, aperiodic and not transient chain. The second advantage is that there is no need to compute the normalization constant of $f(\theta | \mathbf{x})$, because it cancels out in Equation 8.

The Gibbs sampler (Geman & Geman, 1984) is a special case of the Metropolis-Hastings algorithm and is especially popular. If θ is a multidimensional parameter, the Gibbs sampler sequentially updates each of the components of θ from the full conditional distribution of that component given fixed values of all the other components and the data. For many models used in common practice, even the ones that yield a complex posterior distribution, sampling from the posterior's full conditionals is often a relatively simple task. Conveniently, with Gibbs sampling, the acceptance probability (see Equation 8) always equals one.

An Example

Consider again the hospital mortality example in the section headed Hierarchical Models and Exchangeability. Recall that the unknowns of are the hospital mortality rates, r_i , $i = 1, \dots, 12$. For simplicity here we assume that the r_i 's come from a beta distribution with parameters a and b , and that the analyst has endowed a and b with some prior distribution, say $p(a, b)$. Gibbs sampling here requires that we sample from the conditional distribution of each parameter, given all the other parameters and given the observed data. The trick here is to first write down the joint density of all the random quantities in the model:

$$p(r_1, \dots, r_{12}, x_1, \dots, x_{12}, a, b) = \prod_{i=1}^{12} \{p(x_i | r_i, n_i)p(r_i | a, b)\} p(a, b).$$

Because:

$$p(r_i | r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_{12}, x_1, \dots, x_{12}, a, b) \propto p(r_1, \dots, r_{12}, x_1, \dots, x_{12}, a, b),$$

we have that:

$$p(r_i | r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_{12}, x_1, \dots, x_{12}, a, b) \propto p(x_i | r_i, n_i)p(r_i | a, b)$$

(that is, just pick off the terms in the joint density that involve r_i). Because x_i is a binomial random variable and r_i has a beta prior distribution, we have:

$$p(r_i | r_1, \dots, r_{i-1}, r_{i+1}, \dots, r_{12}, x_1, \dots, x_{12}) \sim \text{Beta}(a + x_i, b + n_i - x_i).$$

Thus, the Gibbs sampler here simply cycles through the 12 rates in any order, sampling each time from a beta distribution. The Gibbs sampler will also sample from a and b , the specific distributions depending on the choice of $p(a, b)$.

Missing data fits into this scheme in a straightforward fashion. Suppose x_5 is missing. Then the Gibbs sampler expands to include a draw from the conditional distribution of x_5 given everything else. The veracity of this approach does depend on somewhat subtle assumptions about the missing data mechanism. See Gelman et al. (1995, chap. 17) for a detailed discussion.

Application to Massive Data

MCMC as specified, however, is computationally infeasible for massive data sets. Except for the most trivial examples, computing the acceptance probability (Equation 8) requires a complete scan of the data set. Although the Gibbs sampler avoids the acceptance probability calculation, precalculations for simulating from the full conditionals of $f(\theta | \mathbf{x})$ require a full scan of the data set, sometimes a full scan for each component! Because MCMC algorithms produce dependent draws from the posterior, M usually has to be large to reduce the amount of Monte Carlo variation in the posterior estimates. Although MCMC makes fully Bayesian analysis practical it is often impractical for massive data set applications.

Although this section has not given great detail about MCMC methods, the important ideas for the purpose of this chapter are that (a) MCMC methods make Bayesian analysis practical, (b) MCMC often requires an enormous number of laps through the dataset, and (c) given a θ drawn from $f(\theta | \mathbf{x})$, we can use MCMC to draw another value, θ' , from the same distribution.

Importance Sampling for Analysis of Massive Data Sets

So far we have two tools, importance sampling and MCMC, to draw samples from an arbitrary posterior distribution. Ridgeway and Madigan (2002) presented a particular form of importance sampling that helps perform Bayesian analysis for massive data sets.

Ideally, we would like to sample efficiently and take advantage of all the information available in the data set. A factorization of the integrand shows that this is possible when the observations, x_i , are exchangeable. Let D_1 and D_2 be a partition of the data set so that every observation is in either D_1 or D_2 .

As noted for Equation 3, we would like to sample from the posterior conditioned on all of the data, $f(\theta | D_1, D_2)$. Because sampling from $f(\theta | D_1, D_2)$ is difficult due to the size of the data set, we consider setting $g(\theta) = f(\theta | D_1)$ for use as our sampling distribution and using importance sampling to adjust the draws. If θ_i , $i = 1, \dots, M$, are draws from $f(\theta | D_1)$, then we can estimate the posterior expectation (Equation 3) as

$$\hat{E}(h(\theta) | D_1, D_2) = \frac{\sum_{i=1}^M w_i h(\theta_i)}{\sum_{i=1}^M w_i} \quad (9)$$

where the w_i 's are the importance sampling weights

$$w_i = \frac{f(\theta_i | D_1, D_2)}{f(\theta_i | D_1)}. \quad (10)$$

Although these weights still involve $f(\theta_i | D_1, D_2)$, they greatly simplify.

$$w_i = \frac{f(D_1, D_2 | \theta_i) f(\theta_i)}{f(D_1, D_2)} \frac{f(D_1)}{f(D_1 | \theta_i) f(\theta_i)} \quad (11)$$

$$= \frac{f(D_1 | \theta_i) f(D_2 | \theta_i) f(D_1)}{f(D_1 | \theta_i) f(D_1, D_2)} \quad (12)$$

$$= \frac{f(D_2 | \theta_i)}{f(D_2 | D_1)} \quad (13)$$

$$\propto f(D_2 | \theta_i) = \prod_{x_j \in D_2} f(x_j | \theta_i)$$

Line 10 follows from applying Bayes' theorem to the numerator and denominator. Equation 12 follows from 11; because the observations are assumed to be exchangeable, the observations in the data set partition D_1 are independent from those in D_2 given θ . Conveniently, Equation 13 is just the likelihood of the observations in D_2 evaluated at the sampled value of θ . Figure 5.5 summarizes this result as an algorithm. The algorithm maintains the weights on the log scale for numerical stability.

1. Load as much data into memory as possible to form D_1 , taking into account space requirements for the Monte Carlo algorithm
2. Draw M times from $f(\theta | D_1)$ via Monte Carlo or Markov chain Monte Carlo
3. Purge the memory of D_1
4. Create a vector of length M to store the logarithm of the weights and initialize them to 0
5. Iterate through the remaining observations. For each observation, x_j , update the log-weights on all of the draws from $f(\theta | D_1)$
 - for x_j in the partition D_2 do
 - {
 - for i in $1, \dots, M$ do
 - {
 - $\log w_i \leftarrow \log w_i + \log f(x_j | \theta_i)$
 - }
 - }
6. Rescale to compute the weights

$$w_i \leftarrow \exp(\log w_i - \max(\log w_i)) \quad (14)$$

FIG. 5.5. Importance sampling for massive data sets.

So rather than sample from the posterior conditioned on all of the data, D_1 and D_2 , which slows the sampling procedure, we need only sample from the posterior conditioned on D_1 . The remaining data, D_2 , simply adjusts the sampled parameter values by reweighting. The “for” loops in Step 5 of Fig. 5.5 are interchangeable. The trick here is to have the inner loop scan through the draws so that the outer loop only needs to scan D_2 once to update the weights. Although the same computations take place, in practice physically scanning a massive data set is far more expensive than scanning a parameter list. However, massive models as well as massive data sets exist, so that in these cases scanning the data set may be cheaper than scanning the sampled parameter vectors. Here we assume that scanning the data set is the main impediment to the data analysis.

We certainly can sample from $f(\theta | D_1)$ more efficiently than from $f(\theta | D_1, D_2)$, because simulating from $f(\theta | D_1)$ requires a scan of a much smaller portion of the data set. For reasons discussed in Ridgeway & Madigan (2002) the algorithm works best when D_1 is as large as possible. We also assume that, for a given value of θ , the likelihood is readily computable up to a constant, which is almost always the case. When some data are missing, the processing of an observation in D_2 requires integrating out the missing information. Because the algorithm handles each observation case by case, computing the observed likelihood as an importance weight is much more efficient than if it were embedded and repeatedly computed in a Metropolis-Hastings rejection probability computation. Placing observations with missing values in D_2 greatly reduces the number of times this integration step needs to occur, easing likelihood computations.

Ridgeway and Madigan (2002) described some of the limitations of the algorithm shown in Fig. 5.5 and described a more elaborate algorithm based on recent work on sequential Monte Carlo algorithms. We refer the interested reader to Ridgeway and Madigan (2002) and to Chopin for details.

Variational Methods

Variational approximations for Bayesian calculations have emerged in recent years as a viable alternative to MCMC. The variational approach scales well and can tackle problems beyond the reach of MCMC (see, e.g., Blei, Jordan, & Ng, in press). Here, following Jaakola and Jordan (2000), we sketch the basic idea in the specific context of logistic regression. Recall that a logistic regression takes the form:

$$p(Y = 1 | \mathbf{X}, \theta) = g(\theta^T \mathbf{X})$$

where $g(\theta^T X) = (1 + e^{-x})^{-1}$. Given training data indexed by $i = 1, \dots, n$, the posterior distribution of θ is:

$$p(\theta | \{Y_i, \mathbf{X}_i\}_{i=1}^n) \propto p(\theta) \prod_{i=1}^n (Y_i \times g(\theta^T \mathbf{X}_i) + (1 - Y_i) \times (1 - g(\theta^T \mathbf{X}_i)))$$

where $p(\theta)$ is some prior distribution for θ .

The first step is to symmetrize the logistic function:

$$\log g(x) = -\log(1 + e^{-x}) = \frac{x}{2} - \log(e^{x/2} + e^{-x/2})$$

and note that $f(x) = -\log(e^{x/2} + e^{-x/2})$ is a convex function in the variable x^2 . Thus, it is

possible to bound f below by a function that is linear in x^2 . Specifically:

$$f(x) \geq f(\xi) + \frac{\partial f(\xi)}{\partial (\xi^2)}(x^2 - \xi^2) \quad (15)$$

$$= -\xi/2 + \log g(\xi) + \frac{1}{4\xi} \tanh(\xi/2)(x^2 - \xi^2). \quad (16)$$

This bound is exact when $\xi^2 = x^2$. The essential idea of the variational approximation is used to this lower bound in place of $\log g()$ in the calculation of the posterior distribution of θ . Jaakola and Jordan (2000) used an EM algorithm to find the value for ξ that makes the bound as tight as possible across all the training data. Because the bound is quadratic in x , choosing, for instance, a Gaussian prior for θ will yield a closed form expression for the approximate posterior distribution. In a sense, the variational approach here replaces an integration problem with a simpler optimization problem.

BAYESIAN MODELING

We begin with a discussion of BUGS, a singular general purpose software tool for Bayesian data analysis. Then we briefly discuss specific Bayesian models and provide pointers to the literature.

BUGS and Models of Realistic Complexity via MCMC

BUGS is a useful tool for Bayesian data mining. The U.K. Medical Research Council at Cambridge developed BUGS over the last decade. The program is available free-of-charge from <http://www.mrc-bsu.cam.ac.uk/bugs/>. There are versions for Unix, DOS, and Windows (WinBUGS). The BUGS manual (Spiegelhalter et al., 1999, p. 5) describes BUGS as follows.

BUGS is a computer program that carries out Bayesian inference on statistical problems using Gibbs sampling.

BUGS assumes a Bayesian or full probability model, in which all quantities are treated as random variables. The model consists of a defined joint distribution over all unobserved (parameters and missing data) and observed quantities (the data); we then need to condition on the data in order to obtain a posterior distribution over the parameters and unobserved data. Marginalising over this posterior distribution in order to obtain inferences on the main quantities of interest is carried out using a Monte Carlo approach to numerical integration (Gibbs sampling).

There is a small set of BUGS commands to control a session in which a (possibly very complex) statistical model expressed using the BUGS language is analysed. A compiler processes the model and available data into an internal data structure suitable for efficient computation, and a sampler operates on this structure to generate appropriate values of the unknown quantities.

BUGS is intended for complex models in which there may be many unknown quantities but for which substantial conditional independence assumptions are appropriate. Particular structures include generalised linear models with hierarchical or crossed random effects, latent variable or frailty models, measurement errors in responses and covariates, informative censoring, constrained estimation, and missing data.

```

model;

const

N = 12,                      # number of hospitals
a = 2, b = 24;                # hyperparameters

var

r[N],                         # number of deaths
n[N],                         # total number of operations
p[N];                          # true probability of death

data r,n in 'surgical.dat';

{

for( i in 1 : N ) {
    r[i] ~ dbin(p[i],n[i])
    p[i] ~ dbeta(a,b)
}
}

```

FIG. 5.6. BUGS code for the cardiac surgery example.

For example, Fig. 5.6 shows the BUGS code for the cardiac surgery example used in an earlier section. Alternatively and equivalently, WinBUGS can represent the model graphically. Figure 5.7 shows a screendump. Here we selected the node $r[i]$, and the two lines at the top of the figure show the node's specification. We see that it is a *stochastic node* (represented by an oval) with a binomial distribution conditional on its parents in the graph, which are the binomial parameters $p[i]$ and $n[i]$. The two other node types are *constant nodes* (represented by rectangles) such as a , b , and $n[i]$; or *logical nodes* (represented by ovals with dashed incoming edges—see Fig. 5.7), which are functions of other nodes.

The primary BUGS output comprises summary statistics and density estimates for all the unknowns. BUGS can also provide the MCMC iterates for input to MCMC diagnostic programs such as CODA (CODA is distributed with BUGS). Figure 5.8 shows the density estimates for the first four hospitals in the cardiac surgery example.

The combination of the probabilistic graphical model framework, the Bayesian approach, and MCMC means that BUGS can build models of considerable complexity, when such complexity is warranted. To demonstrate this, we present here an application of moderate complexity that would challenge the capabilities of any commercial data analysis or data mining software. This example is from Draper and Madigan (1997).

In the United States the antiabortion campaign of the late 1980s and early 1990s generated much publicity and occasionally became violent. Sociologists, epidemiologists, and medical researchers have begun to study the effect that this campaign had on access to reproductive

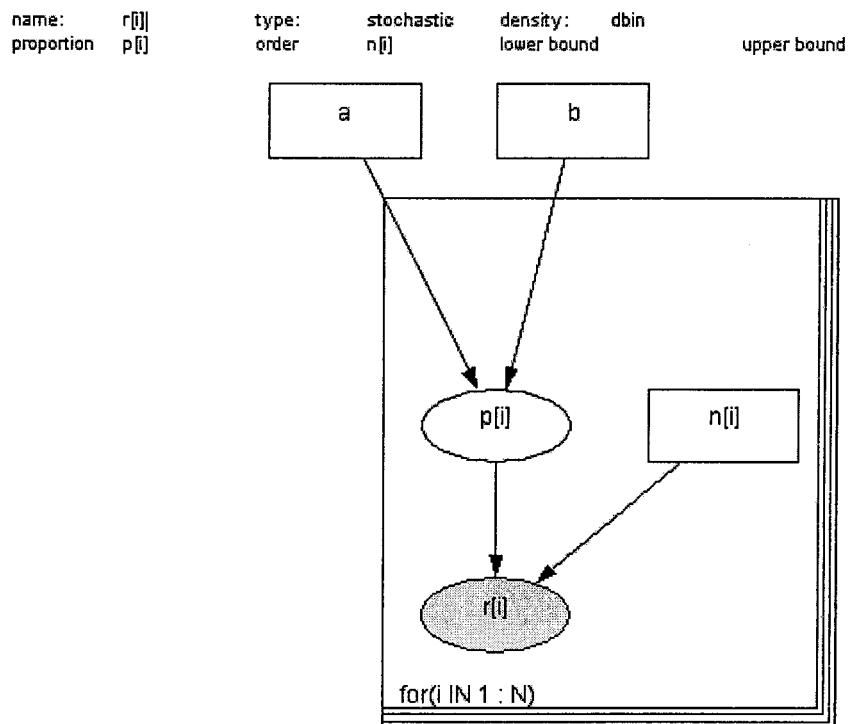


FIG. 5.7. BUGS graphical Markov model for the cardiac surgery model.

services and on pregnancy terminations. Interest mainly lies in modeling the incidence rates of pregnancy terminations and their changes over time at the county level, with particular attention focusing on possible changes in response to the antiabortion campaign. Available data include the observed number y_{ijk} of reported terminations in age group i , county j , and year k (across five age groups, 38 U.S. counties, and the years 1983–1994), together with the appropriate population denominators n_{ijk} and a selection of county-level predictor variables

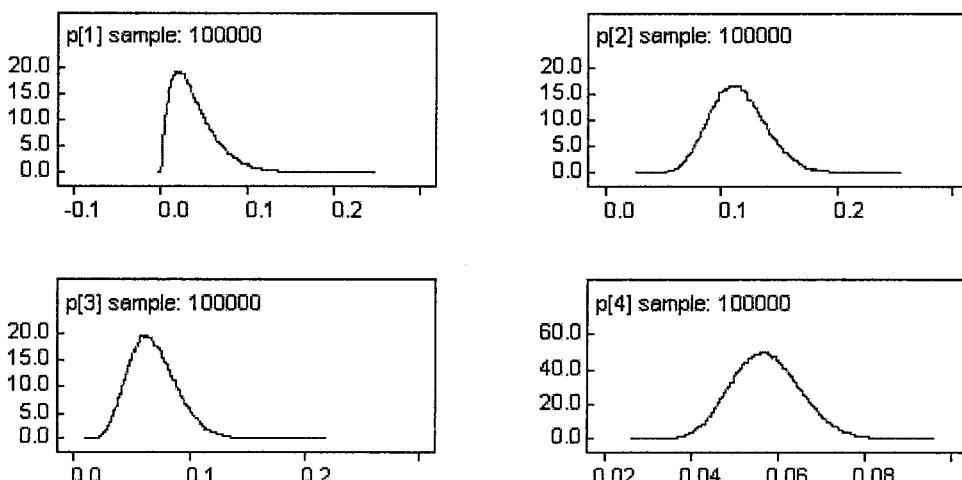


FIG. 5.8. BUGS output for Hospitals A–D in the cardiac surgery model.

x_j including, for instance, the estimated number of clinics providing reproductive services in 1983 and in 1993. A natural model for count data such as the y_{ijk} involves regarding the counts as realizations of Poisson random variables,

$$y_{ijk} \sim \text{Poisson}(\mu_{ijk}), \quad (17)$$

where the mean structure μ_{ijk} is related to age group, district, and year. Preliminary exploratory data analysis and examination of several simple models suggests the structure

$$\log \mu_{ijk} = \log n_{ijk} + \alpha_i^a + \alpha_j^c + (\beta_i^a + \beta_j^c)t_k + (\gamma_i^a + \gamma_j^c)z_k, \quad (18)$$

where z_k is an indicator variable for after 1990 versus before and, for example, α_i^a , β_i^a , and γ_i^a are the age effect on the intercept, the change per year, and the gradient after 1990. However, in light of the data, several deficiencies in this model present themselves and prompt the following elaborations.

- Overdispersion: The theoretical mean and variance of the Poisson distribution are equal, but with data such as these, sample variances often greatly exceed sample means. Thus, the model typically needs to be generalized to allow for this extra variability, which can be thought of as arising from the omission of additional unobserved predictor variables. This generalization may be accomplished by adding a Gaussian error term to Equation 18.
- Hierarchical modeling: The standard Bayesian modeling of the county-level coefficients α_j^c , β_j^c , and γ_j^c would involve specifying prior information—in the form of probability distributions—for each of them (i.e., 114 separate distributions). However, considering the counties as exchangeable pools the knowledge across them. This is accomplished mathematically by regarding, for example, the 38 α_j^c 's as having arisen from a single (e.g., Gaussian) distribution. That single distribution also may have a set of unknown hyperparameters that we could choose or estimate.
- Relationships among the coefficients: Prior experience suggests that it is unrealistic to consider the county-level coefficients α_j^c , β_j^c , and γ_j^c as independent, so the three distributions just described (one for each of α , β , and γ) need to be treated as correlated.
- Underreporting: Much is known about abortion underreporting rates, which is a form of missing data. It is necessary to incorporate this information into the model in a way that fully fleshes out the resulting uncertainty.

Figure 5.9 presents a summary of the resulting model. MCMC computation makes calculations for this Bayesian model straightforward. Moreover, complicated functions of the unknowns, such as the true relative rankings in abortion rates among the counties, are available trivially, together with measures of uncertainty (difficult to achieve correctly with non-Bayesian analyses) that fully reflect what is known and what is not. See Spiegelhalter (1998) for another example along these lines.

The text by Congdon (2001) discusses Bayesian data analysis with BUGS and WinBUGS and provides sample code. The BUGS documentation itself features two volumes of examples covering a broad range of applications.

Note that BUGS does not provide any facilities for learning model structure. That is, the user specifies the graphical Markov model, and then BUGS carries out the corresponding MCMC calculations. A BUGS module for learning model structure would be a useful addition.

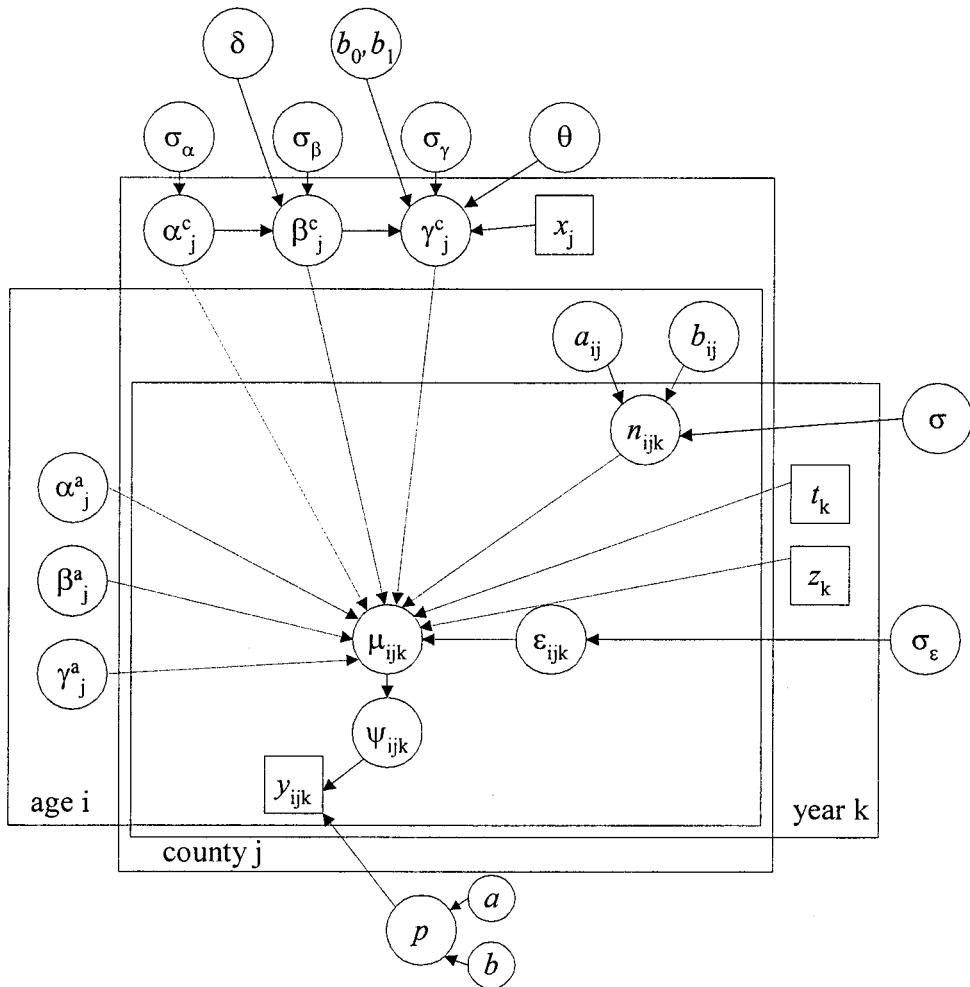


FIG. 5.9. Bayesian model for the reproductive services example.

Bayesian Predictive Modeling

Thus far, we have dealt mostly with full probability modeling. Many applications, however, single out one quantity, y , and characterize y as a function of another quantity or vector of quantities x . As such, these applications want the conditional distribution of y given x , parameterized as $p(y | \theta, x)$ say, under a model in which the n observations $(x, y)_i$ are exchangeable (Gelman et al., 1995, p. 233). The quantity y is called the *response* or *outcome variable*. The variables $x = \{x_1, \dots, x_k\}$ are the *explanatory variables* or *predictors*.

Gelman et al. (1995, p. 235) noted that a full probability model would also specify a distribution for x , say $p(x | \psi)$ for some parameter vector ψ , leading to the full probability model $p(x, y | \theta, \psi)$. However, if θ and ψ are independent in their prior distribution, that is, $p(\theta, \psi) = p(\theta)p(\psi)$, then the posterior distribution factors as:

$$p(\theta, \psi | x, y) = p(\psi | x)p(\theta | x, y)$$

and we can analyze the second factor by itself with no loss of information:

$$p(\theta | x, y) \propto p(\theta)p(y | x, \theta).$$

The practical advantage of this is that it is often much easier to devise a good conditional probability model for a single quantity than to come up with a full probability model for all quantities.

Many predictive applications use the *normal linear model* in which the distribution of y given x is a normal whose mean is a linear function of x :

$$E(y_i | \beta, x) = \beta_1 x_{i1} + \cdots + \beta_k x_{ik},$$

for $i = 1, \dots, n$. Gelman et al. (1995, chap. 8) discussed the Bayesian analysis of the normal linear model with a variety of variance structures. See also Raftery, Madigan, and Hoeting (1997). So-called “sparse priors” for predictive models can provide excellent predictive performance. See, for example, Tipping (2001).

Gelman et al. (1995) also discussed Bayesian analysis of hierarchical linear models and generalized linear models. Carlin and Louis (2000) included spatial and spatio-temporal models, nonlinear models, and longitudinal data models. West and Harrison (1997) is the classical reference on the Bayesian analysis of time series data. The companion text, Pole, West, and Harrison (1994), covers applications and software.

Survival analysis concerns the situation in which y is a lifetime and, invariably, is subject to various forms of censoring. Ibrahim, Chen, and Sinha (2001) provide a thorough introduction to survival analysis from the Bayesian perspective. See also Raftery, Madigan, and Volinsky (1996) and Volinsky, Madigan, Raftery, and Kronmal (1997).

Neal (1996) provides an excellent description of the Bayesian analysis of neural networks. See also the extensive material on David MacKay’s Web site: <http://www.inference.phy.cam.ac.uk/mackay/>.

Tree-based predictive models partition the predictor space into a set of rectangles and then fit a simple model (like a constant) in each one. Tree models are conceptually simple, handle both discrete and continuous responses and predictors, and elegantly deal with missing values. However, tree models suffer from an instability problem, namely, that a small change in the data can result in the tree-learning algorithm selecting a very different tree. Averaging procedures such as bagging or boosting effectively address the instability issue and can yield outstanding predictive accuracy. Hastie, Tibshirani, and Friedman (2001, section 9.2) provided an introduction and references. Breiman and Friedman (1984) provided a fascinating early account of scaling tree-learning algorithms to massive data. Chipman, George, and McCulloch (1998) and Denison, Malick, and Smith (1998) described Bayesian tree models and associated MCMC learning algorithms. Chipman, George, and McCulloch (2002) reported impressive predictive performance for their Bayesian tree algorithm. Bayesian model averaging may address the instability problem associated with tree models, but this has yet to be fully explored. Denison, Adams, Holmes, and Hand (in press) described a related class of Bayesian partition models. The “partitions” in their case refer to partitions of continuous predictors. The method seems to scale reasonably well and provided good predictive performance on a banking application.

Readers may be interested in the (non-Bayesian) Microsoft Research Win-Mine Toolkit. This software learns graphical models from data, using a decision tree to model the conditional distribution at each node. See <http://research.microsoft.com/~dmax/WinMine/Tooldoc.htm>.

We note that Bayesian predictive modelers can short-circuit some of the computational complexity of the fully Bayesian approach by making so-called Bayesian plug-in predictions.

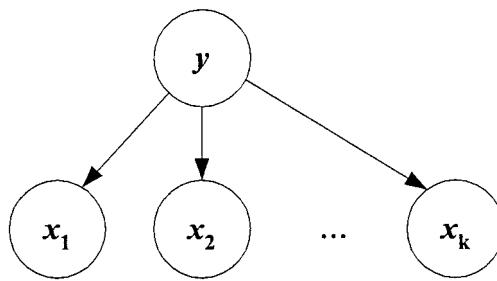


FIG. 5.10. The naive Bayes model.

The essential idea is to make predictions at a single parameter value, typically the posterior mode, instead of integrating over the posterior distribution. In practice, predictive accuracy is usually robust to this approximation and even outperforms the fully Bayesian approach on occasion.

Bayesian Descriptive Modeling

Earlier sections presented several examples of Bayesian probabilistic graphical models, and many commercial and noncommercial tools (such as BUGS) exist for learning such models from data. Kevin Murphy maintains a list at <http://www.cs.berkeley.edu/~murphyk/Bayes/bnsoft.html>. Of course, probabilistic graphical models can be used predictively simply by considering conditional distributions. Naive Bayes models represent an important subclass of graphical models that scale well and often yield surprisingly good predictive performance (see Hand & Yu, 2001; Lewis, 1998). The classical naive Bayes model (for example, Duda & Hart, 1973) imposes a conditional independence constraint, namely, that the predictor variables, say, x_1, \dots, x_k , are conditionally independent given the response variable y . Figure 5.10 shows a graphical Markov model representation of the naive Bayes model.

Outlier detection is an important data mining task, and many authors have presented Bayesian approaches. A key advantage of the Bayesian approach is that it directly computes the probability that an observation is an outlier. See Bayarri and Morales (2000), Hoeting, Raftery, and Madigan (1996), Justel and Pena (1999), and the references therein.

Cluster analysis has developed mainly as a set of ad hoc methods. More recently, many data analysts have found that basing cluster analysis on a full probability model yields significant advantages. In particular, the probabilistic approach enables clustering of nonstandard objects (such as World Wide Web page visits over time or gene expression data), can detect clusters within clusters or overlapping clusters, and can make formal inference about the number of clusters. Reversible jump MCMC (Green, 1995), an MCMC algorithm than incorporates jumps between spaces of varying dimension, provides a flexible framework for Bayesian analysis of model-based clustering. Richardson and Green (1997) is a key reference. See also Fraley and Raftery (1998) and the MCLUST software available from Raftery's website, <http://www.stat.washington.edu/raftery/Research/Mclust/mclust.html>, the AutoClass system (Cheeseman & Stutz, 1996), and the SNOB system (Wallace & Dowe, 2000). Cadez and Smyth (1999) and Cadez et al. (2001) present an EM algorithm for model-based clustering and describe several applications.

AVAILABLE SOFTWARE

An earlier section described the BUGS software. Carlin and Louis (2000, Appendix C) presented a comprehensive guide to Bayesian software. Berger (2000) included a list of Bayesian software web sites as well as an overview of Bayesian, philosophy and approaches, at <http://www.stat.duke.edu/~berger/papers/99-30.ps>.

DISCUSSION AND FUTURE DIRECTIONS

MCMC opened the floodgates to serious Bayesian data analyses. For example, the Sixth Workshop on Case Studies of Bayesian Statistics took place in 2001 and presented a broad array of real-world applications (see <http://lib.stat.cmu.edu/bayesworkshop/2001/Bayes01.html>).

Although data mining applications often feature massive data sets, predictive modelers often deal with a paucity of *labeled* data. Bayesian analyses, however, can readily absorb external information sources and usefully bootstrap the learning process. For example, much of the empirical work in text categorization research uses massive, publicly available collections of labeled documents. The Reuters Corpus of news articles, for instance, contains more than 800,000 labeled documents. Incorporation of prior information in such a context yields modest returns. Yet, real-world applications rarely feature large collections of labeled documents (Lewis, personal communication, June 1, 2002), and prior knowledge can play a critical role. We suspect the same is true of many real-world prediction problems.

Breiman (2001) argued that attempting to model mother nature's data generating mechanism is largely a futile exercise. He made a distinction between the "data modeling culture" (this includes the full probability modelers and the Bayesians) and the "algorithmic modeling culture." The algorithmic culture regards the data generating mechanism as complex and unknown, and focuses on algorithms, often very complicated algorithms, that yield good predictive performance. Breiman's distinction is perhaps not as sharp as he contended. For one thing, many of the algorithmic methods such as neural networks and support vector machines yield natural Bayesian analogues. See, for example, the *Gaussian Process* models of Williams (1998), or the *Bayes Point Machines* of Herbrich, Graepel, and Campbell (2001). In fact, Herbrich et al. present Bayes point machines as an algorithmic approximation to Bayesian inference for kernel-based predictive models.

Combining multiple complex hierarchical models such as that in Fig. 5.9 with model structures and priors tuned to optimize predictive performance or some other utility function combines aspects of the data modeling culture and the algorithmic culture. For *multilevel data* such combinations seem especially expedient. For example, in a medical context, we may have information about individual patients, information about the hospitals in which they stayed, and information about the geographic locales of the hospitals. Assuming exchangeability or partial exchangeability within each level can sharpen inference and provide better predictions. Between the levels the Bayesian literature usually features linear or mildly nonlinear parametric models. Yet, many of the popular algorithmic or kernel-based methods could play a role instead.

SUMMARY

The Bayesian approach to data analysis emphasizes full probability models and appropriate accounting for uncertainty. The approach can harness prior knowledge when it is available and provides outputs that are simple to interpret. The core related notions of "exchangeability" and

“hierarchical modeling” represent powerful concepts for data mining, especially for models with huge numbers of parameters.

Prior to the emergence of MCMC methods computational complexity limited Bayesian data analysis to small-scale problems. Now analysts conduct Bayesian data analyses of considerable complexity. Nonetheless, significant barriers remain for very large-scale applications, and further research is required. Variational approximations are especially promising.

Religious adherence to a Bayesian or non-Bayesian perspective, often a feature of past debates in the statistical community, has become an absurdity. Bayes is not a panacea and is certainly not always the optimal approach. In particular, for applications in which accounting for uncertainty is not so important, the fully Bayesian approach may be more trouble than it is worth. Nonetheless, it is a tool of considerable flexibility and elegance and offers many advantages for data miners.

ACKNOWLEDGMENTS

National Science Foundation grants supported Madigan’s research. We thank the editor and an anonymous reviewer for helpful comments. We are grateful to David D. Lewis for stimulating conversations.

REFERENCES

- Barbieri, M. M., & Berger, J. O. (2002). *Optimal predictive model selection*. Working Paper 02-02, Institute of Statistics and Decision Sciences, Duke University.
- Bayarri, M. J., & Morales, J. (2000). Bayesian measures of surprise for outlier detection. Retrieved [July 1, 2002], from <http://citeseer.nj.nec.com/bayarri00bayesian.html>.
- Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53, 370–418; 54, 296–325.
- Berger, J. (2000). Bayesian analysis: A look at today and thoughts of tomorrow. *Journal of the American Statistical Association*, 95, 1269–1276.
- Bernardo, J. M., & Smith, A. F. M. (1994). *Bayesian theory*. Chichester, England: John Wiley & Sons.
- Blei, D., Jordan, M., & Ng, A. (in press). Latent Dirichlet models for applications in information retrieval. In *Bayesian Statistics 7*. Oxford: Oxford University Press.
- Breiman, L. (2001). Statistical modeling: The two cultures (with discussion). *Statistical Science*, 16, 199–231.
- Breiman, L., & Friedman, J. (1984). Tools for large data set analysis. In E. J. Wegman and J. G. Smith (Eds.), pp. 191–197. *Statistical Signal Processing*, New York: M. Dekker.
- Cadez, I., & Smyth, P. (1999). Probabilistic clustering using hierarchical models. Technical Report UCI-ICS 99-16.
- Cadez, I. V., Smyth, P., Ip, E., & Mannila, H. (2001). Predictive profiles for transaction data using finite mixture models. Technical Report UCI-ICS 01-67. Irvine, CA: University of California, Department of Information & Computer Science.
- Carlin, B. P., & Chib, S. (1995). Bayesian model choice via Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society (Series B)*, 57, 473–484.
- Carlin, B. P., & Louis, T. A. (2000). *Bayes and empirical bayes methods for data analysis (second edition)*, New York: Chapman and Hall.
- Chaloner, K. (1996). The elicitation of prior distributions. In D. Berry & D. Stangl (Eds.), *Bayesian Biostatistics*, pp. 141–156. New York: Marcel Dekker.
- Cheeseman, P., & Stutz, J. (1996). Bayesian classification (AutoClass): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, & R. Uthurusamy (Eds.), *Advances in knowledge discovery and data mining*, pp. 000–000. Cambridge, MA: AAAI Press/MIT Press.
- Chipman, H. A., George, E. I., & McCulloch, R. E. (1998). Bayesian CART model search (with discussion). *Journal of the American Statistical Association*, 93, 935–960.
- Chipman, H. A., George, E. I., & McCulloch, R. E. (2002). Bayesian tree models. *Machine Learning*, 48, 299–320.
- Chopin, N. (). A sequential particle filter method for static models. *Biometrika*. vol. 89, 539–552.

- Congdon, P. (2001). *Bayesian Statistical Modelling*. London: Wiley.
- Consonni, G., & Veronese, P. (1995). A Bayesian method for combining results from several binomial experiments. *Journal of the American Statistical Association*, 90, 935–944.
- Cooper, G. F., & Herskovits, E. (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9, 309–347.
- Dawid, A. P. (1992). Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2, 25–36.
- de Finetti, B. (1931). Funzione caratteristica di un fenomeno aleatorio. Atti della R. Accademia Nazionale dei Lincei, Serie 6. Memorie, Classe di Scienze Fisiche, Mathematiche e Naturale, 4, 251–99.
- Denison, D., Mallick, B., & Smith, A. F. M. (1998). A Bayesian CART algorithm. *Biometrika*, 85, 363–377.
- Denison, D. G. T., Adams, N. M., Holmes, C. C., & Hand, D. J. (in press). Bayesian partition modelling. *Computational Statistics and Data Analysis*.
- Draper, D. (1995). Assessment and propagation of model uncertainty (with discussion). *Journal of the Royal Statistical Society (Series B)*, 57, 45–97.
- Draper, D., Hodges, J. S., Mallows, C., & Pregibon, D. (1993). Exchangeability and data analysis (with discussion). *Journal of the Royal Statistical Society (Series A)*, 156, 9–37.
- Draper, D., & Madigan, D. (1997). The scientific value of Bayesian statistical methods. *IEEE Intelligent Systems and their Applications*, 12, 18–21.
- Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.
- DuMouchel, W. (1999). Bayesian data mining in large frequency tables, with an application to the FDA Spontaneous Reporting System. *American Statistician*, 53, 177–202.
- DuMouchel, W., & Pregibon, D. (2001). Empirical Bayes screening for multi-item associations. *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovering and Data Mining* (pp. 67–76). New York: ACM Press.
- Fraley, C., & Raftery, A. E. (1998). How many clusters? Which clustering method?—answers via Model-Based Cluster Analysis. *Computer Journal*, 41, 578–588.
- Geisser, S. (1993). *Predictive inference*. London: Chapman and Hall.
- Gelfand, A. E., & Dey, D. K. (1994). Bayesian model choice: Asymptotics and exact calculations. *Journal of the Royal Statistical Society (Series B)*, 56, 501–514.
- Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. London: Chapman and Hall.
- Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741.
- Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov chain Monte Carlo in Practice*. London: Chapman and Hall.
- Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82, 711–732.
- Han, C., & Carlin B. P. (2001). MCMC methods for computing Bayes factors: A comparative review. *Journal of the American Statistical Association*, 96, 1122–1132.
- Hand, D. J., & Yu, K. (2001). Idiot's Bayes—not so stupid after all? *International Statistical Review*, 69, 385–398.
- Hastie, T., Tibshirani, T., & Friedman, J. (2001). *The elements of statistical learning: Data Mining, Inference & Prediction*. New York: Springer.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97–109.
- Heckerman, D., Geiger, D., & Chickering, D. M. (1994). Learning Bayesian networks: The combination of knowledge and statistical data. In R. Lopez de Mantaras & D. Poole (Eds.), *Uncertainty in Artificial Intelligence, Proceedings of the 10th Conference*, pp. 293–301. San Francisco: Morgan Kaufmann.
- Herbrich, R., Graepel, T., & Campbell, C. (2001). Bayes point machines. *Journal of Machine Learning Research*, 1, 245–279.
- Hoeting, J., Raftery, A. E., & Madigan, D. (1996). A method for simultaneous variable selection and outlier identification in linear regression. *Computational Statistics and Data Analysis*, 22, 251–270.
- Hoeting, J., Madigan, D., Raftery, A. E., & Volinsky, C. T. (1999). Bayesian model averaging—A tutorial (with discussion). *Statistical Science*, 14, 382–417.
- Ibrahim, J. G., Chen, M-H., & Sinha, D. (2001). *Bayesian survival analysis*. New York: Springer.
- Jaakkola, T., & Jordan, M. I. (2000). Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10, 25–37.
- Justel, A., & Pena, D. (1999). Heterogeneity and model uncertainty in Bayesian regression models. *Revista de la Real Academia de Ciencias*, 93, 357–366. <http://www.adi.uam.es/~ajustel/papers/jp3.pdf>.
- Lauritzen, S. L., Dawid, A. P., Larsen, B. N., & Leimer, H-G. (1990). Independence properties of directed markov fields. *Networks*, 20, 491–505.

- Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In ECML '98, 10th European Conference on Machine Learning, pp. 4–15.
- Lie, R. T., Heuch, I., & Irgens, L. M. (1991). A temporary increase of Down syndrome among births of young mothers in Norway: An effect of risk unrelated to maternal age? *Genetic Epidemiology*, 8, 217–230.
- Lie, R. T., Heuch, I., & Irgens, L. M. (1994). Estimation of the proportion of congenital malformations using double registration schemes. *Biometrics*, 50, 433–444.
- Madigan, D., & Raftery, A. E. (1994). Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89, 1535–1546.
- Madigan, D., & York, J. (1995). Bayesian graphical models for discrete data. *International Statistical Review*, 63, 215–232.
- Madigan, D., Gavrin, J., & Raftery, A. E. (1995). Eliciting prior information to enhance the predictive performance of Bayesian graphical models. *Communications in Statistics—Theory and Methods*, 24, 2271–2292.
- Neal, R. M. (1996). *Bayesian learning for neural networks*. New York: Springer-Verlag.
- Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible Inference*. Morgan Kaufman.
- Pole, A., West, M., & Harrison, J. (1994). *Applied Bayesian forecasting and time series analysis*. London: Chapman & Hall.
- Raftery, A. E., Madigan, D., & Volinsky, C. T. (1996). Accounting for model uncertainty in survival analysis improves predictive performance. In J. M. Bernardo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.). *Bayesian statistics V*, pp. 323–350. Oxford: Oxford University Press.
- Raftery, A. E., Madigan, D., & Hoeting, J. (1997). Bayesian model averaging for linear regression models. *Journal of the American Statistical Association*, 92, 179–191.
- Richardson, S., & Green, P. J. (1997). On Bayesian analysis of mixtures with an unknown number of components (with discussion). *Journal of the Royal Statistical Society (Series B)*, 59, 731–792.
- Ridgeway, G., & Madigan, D. (2002). Bayesian analysis of massive datasets via particle filters. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (pp. 5–13).
- Shibata, R. (1983). Asymptotic mean efficiency of a selection of regression variables. *Annals of the Institute of Mathematical Statistics*, 35, 415–426.
- Spiegelhalter, D. J. (1998). Bayesian graphical modeling: A case study in monitoring health outcomes. *Applied Statistics*, 47, 115–133.
- Spiegelhalter, D. J., & Lauritzen, S. L. (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20, 579–605.
- Spiegelhalter, D. J., Freedman, L. S., & Parmar, M. K. B. (1994). Bayesian approaches to randomized trials (with discussion). *Journal of the Royal Statistical Society (Series A)*, 157, 357–416.
- Spiegelhalter, D. J., Thomas, A., & Best, N. G. (1999). *WinBUGS version 1.2 user manual*. MRC Biostatistics Unit.
- Spiegelhalter, D. J., Best, N. G., Carlin, B. P., & van der Linde, A. (2002). Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society (Series B)*.
- Stigler, S. M. (1990). *The history of statistics: The measurement of uncertainty before 1900*. Cambridge, MA: Harvard University Press.
- Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1, 211–244.
- Volinsky, C. T., Madigan, D., Raftery, A. E., & Kronmal, R. A. (1997). Bayesian model averaging in proportional hazard models: Predicting strokes. *Applied Statistics*, 46, 433–448.
- Wallace, C. S., & Dowe, D. L. (2000). MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10, 73–83.
- West, M., & Harrison, J. (1997). *Bayesian Forecasting and Dynamic Models* (2nd ed.), New York: Springer.
- Williams, C. K. I. (1998). Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan (Ed.), *Learning and inference in graphical models*, (pp. 599–622). Norwell, MA: Kluwer Academic Press.
- York, J., Madigan, D., Heuch, I., & Lie, R. T. (1995). Estimating a proportion of birth defects by double sampling: A Bayesian approach incorporating covariates and model uncertainty. *Applied Statistics*, 44, 227–242.

6

Hidden Markov Processes and Sequential Pattern Mining

Steven L. Scott

*Marshall School of Business,
University of Southern California*

Introduction to Hidden Markov Models	134
Parameter Estimation in the Presence of Missing Data	136
The EM Algorithm	136
MCMC Data Augmentation	138
Missing Data Summary	140
Local Computation	140
The Likelihood Recursion	140
The Forward-Backward Recursions	141
The Viterbi Algorithm	142
Understanding the Recursions	143
A Numerical Example Illustrating the Recursions	143
Illustrative Examples and Applications	144
Fetal Lamb Movements	144
The Business Cycle	150
HMM Stationary and Predictive Distributions	153
Stationary Distribution of d_t	153
Predictive Distributions	154
Posterior Covariance of h	154
Available Software	154
Summary	154
References	155

INTRODUCTION TO HIDDEN MARKOV MODELS

A hidden Markov model (HMM) is a finite mixture model whose mixing distribution is a Markov chain. Suppose $\mathbf{d} = (d_1, \dots, d_n)$ represents a sequence of observed data, and that another unobserved (hidden) sequence $\mathbf{h} = (h_1, \dots, h_n)$ follows a Markov process with $h_t \in \mathcal{S} = \{0, \dots, S - 1\}$. One says that \mathbf{d} obeys a hidden Markov model if the full conditional distribution for d_t depends only on the current h_t and a parameter vector θ . Formally,

$$p(d_t | d_{-t}, \mathbf{h}, \theta) = p(d_t | h_t, \theta) \equiv P_{h_t}(d_t | \theta), \quad (1)$$

where d_{-t} is \mathbf{d} with d_t removed. Figure 6.1 illustrates the conditional independence structure describing Equation 1. This chapter assumes that the Markov process governing \mathbf{h} is stationary, so that $p(h_t | h_1, \dots, h_{t-1}) = q(h_{t-1}, h_t)$, although the theory can easily be modified to allow nonstationary transition probabilities (i.e., q may depend on t). In my notation, θ contains the Markov transition probabilities in addition to the parameters of the conditional distributions $P_s(d_t | \theta)$. Note that d_t can be any quantity to which a probability distribution can be assigned. Thus, d_t can be a univariate, multivariate, continuous, or discrete random variable, or possibly something more complicated such as a spatial stochastic process. The fact that d_t can be almost any data structure means that HMMs are a very general class of time series models. This generality is reflected in the variety of problems to which HMMs have been applied, including genetics (Churchill, 1989; Liu, Neuwald, & Lawrence, 1999, Chapter 24 in this volume), speech recognition (Juang & Rabiner, 1991), neurobiology (Fredkin & Rice, 1992), image analysis (Romberg, Choi, & Baraniuk, 2001), signal processing (Andrieu & Doucet, 1999), network intrusion detection (Scott, 1999, in press), econometrics (Albert & Chib, 1993a; Hamilton, 1989), atmospheric science (Zucchini & Guttorm, 1991), and many others. The purpose of this chapter is to familiarize readers with HMMs and with the methods used to estimate model parameters and the hidden Markov chain.

There are two views that one may take regarding the role of HMMs in data analysis. The first is that HMMs are a flexible class of time series models, but that \mathbf{h} is nothing more than a computational device used to represent dependence among observed data. The second attaches a physical meaning to \mathbf{h} . Although both views are valid, HMMs are much more compelling models when \mathbf{h} is a meaningful quantity, in which case estimating \mathbf{h} is often a primary inferential goal.

For a concrete example consider the record of fetal lamb movements over 240 consecutive 5-second intervals presented by Leroux and Puterman (1992), which is now ubiquitous in the HMM literature. From Fig. 6.2 one can see that the fetal lamb's activity level varies among either two or three states. Here d_t is the number of movements recorded in interval t . The state space of the hidden Markov chain describing the fetus' activity level is either $h_t \in \mathcal{S} = \{0 = \text{passive}, 1 = \text{active}\}$ or $\mathcal{S} = \{0 = \text{passive}, 1 = \text{somewhat active}, 2 = \text{very active}\}$. The conditional distributions are modeled as $P_s(d_t | \theta) = \text{Pois}(d_t | \lambda_s)$, where $\text{Pois}(\cdot | \lambda)$ is the

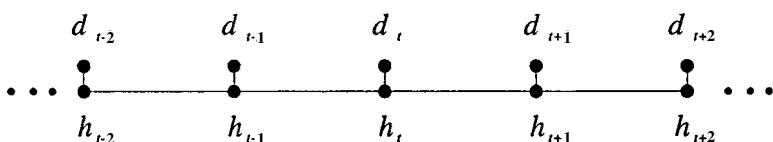


FIG. 6.1. Conditional independence graph for an HMM. The conditional distribution of the variable at a node, given all other variables, depends only on variables to which it is connected by an edge.

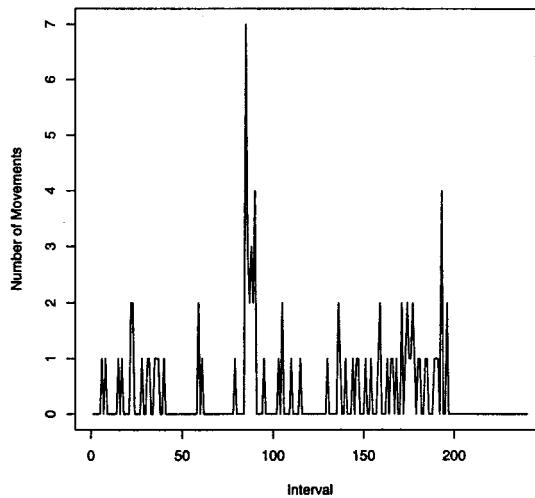


FIG. 6.2. Number of movements made by a lamb fetus during 240 consecutive 5-second intervals.

Poisson distribution with mean λ . The parameter θ contains the Poisson means $\lambda_0 < \dots < \lambda_{S-1}$ and the matrix of Markov transition probabilities $\mathbf{Q} = [q(r, s)]$.

The procedures used to estimate θ and \mathbf{h} are special cases of two more general statistical techniques: missing data methods and local computation. The following two sections explain missing data methods and local computation in general terms and in relation to HMMs. Missing data methods are appropriate because of the awkward HMM likelihood function, which occurs because \mathbf{h} is unobserved. If \mathbf{h}_1 has initial distribution π_0 , then the HMM likelihood is

$$p(\mathbf{d} | \theta) = \sum_{\mathbf{h} \in S^n} p(\mathbf{d}, \mathbf{h} | \theta) = \sum_{\mathbf{h} \in S^n} \pi_0(h_1) P_{h_1}(d_1 | \theta) \prod_{t=2}^n q(h_{t-1}, h_t) P_{h_t}(d_t | \theta). \quad (2)$$

The sum in Equation 2 is over S^n elements, so it quickly becomes impossible to compute directly, even for moderate n . This would appear to thwart straightforward attempts at estimating θ by maximum likelihood or Bayesian techniques. However, notice that if \mathbf{h} had been observed, then the complicated sum in Equation 2 would no longer be necessary, and the complete data likelihood function $p(\mathbf{d}, \mathbf{h} | \theta)$ could be evaluated easily. Missing data methods allow one to avoid calculating Equation 2 by repeatedly working with estimates of the simpler complete data likelihood.

Local computation attacks sums like Equation 2 by exploiting the conditional independence properties that produce the graphical structure in Fig. 6.1. Local computation methods calculate marginal distributions of subsets of \mathbf{d} or \mathbf{h} by recursively averaging over the marginal distribution of their neighbors in Fig. 6.1. The recursions allow marginal distributions to be computed with only $O(S^2 n)$ operations instead of the $O(S^n)$ suggested by directly evaluating Equation 2. The third section of this chapter explains three local computation methods for HMMs. The *likelihood recursion* computes Equation 2. The *forward-backward recursions* (Baum & Petrie, 1966; Baum, Petrie, Soules, & Weiss, 1970) compute the marginal distribution of each transition (h_{t-1}, h_t) given \mathbf{d} and θ . The *Viterbi algorithm* (Viterbi, 1967) constructs the most likely configuration of \mathbf{h} conditional on \mathbf{d} and θ . The forward-backward recursions and the Viterbi algorithm are the methods most often used to estimate \mathbf{h} .

The fourth section of the chapter illustrates the methods described earlier on the fetal lamb data from Fig. 6.2 and on a data set examining the presence or absence of recessions in the U.S. gross national product. The fifth section derives the stationary and predictive distributions of HMMs, which can be used as model diagnostics. The final sections list some of the few software packages used to implement HMMs and conclude with a summary.

Before moving on, it is worth mentioning the close relationship of HMMs to several other classes of models. HMMs are a special case of state space models (West & Harrison, 1997), although that term usually suggests \mathbf{h} follows a Gaussian process. The forward-backward recursions for HMMs correspond directly to the prediction and smoothing steps in the Kalman filter (Kalman, 1960) used for Gaussian state space models. HMMs are commonly used to model discrete valued time series. MacDonald and Zucchini (1997) compared HMMs to several other models used for that purpose. If one removes the time dimension from HMMs, so that h_t is an independent draw from a multinomial distribution, then an HMM becomes a finite mixture model (Titterington, Smith, & Makov, 1985). Finite mixture models are often used for classification and clustering problems. HMMs are very well suited for classification and clustering problems in which a subject's cluster membership changes over time. Finally HMMs are a special case of graphical models (Lauritzen, 1996, Chapter 5 in this volume), which are also called "Bayesian belief networks" and "probabilistic expert systems." The recursive methods employed by HMMs and state space models are special cases of a general local computation algorithm for graphical models.

PARAMETER ESTIMATION IN THE PRESENCE OF MISSING DATA

HMMs were an early and important application of statistical missing data methods, which are now a mature subfield of statistics. In addition to their use for HMMs, missing data methods have been applied to problems such as clustering using finite mixture models (Chapter 10 in this volume), factor analysis (Chapter 8 in this volume), fitting robust models based on the multivariate T distribution (Liu & Rubin, 1995), and fitting generalized linear models with random effects (Albert & Chib, 1993b). The current state of the art in missing data methods is reviewed by van Dyk and Meng (2001). This section reviews the two most commonly used missing data methods, the EM algorithm and Markov chain Monte Carlo (MCMC) data augmentation. These algorithms are discussed in general terms and in specific relation to HMMs. The notation consistently uses \mathbf{d} for observed data and \mathbf{h} for missing data, though in the general discussion the data need not come from an HMM.

Both the EM algorithm and MCMC data augmentation produce a sequence $\theta^{(1)}, \theta^{(2)}, \dots$ by using imputed values of missing quantities to produce the next $\theta^{(j+1)}$. In the EM algorithm the sequence of θ 's converges to a maximum likelihood estimate, whereas in MCMC data augmentation the sequence is a correlated Monte Carlo sample from the Bayesian posterior distribution $p(\theta | \mathbf{d})$.

The EM Algorithm

The EM Algorithm in the General Case

The EM algorithm (Dempster, Laird, & Rubin, 1977) is a technique for obtaining maximum likelihood estimates (or Bayesian maximum posterior estimates) of model parameters in the presence of missing data. The algorithm alternates between estimating the log-likelihood that

would have occurred if all data had been observed (the *complete data log likelihood*) and maximizing the expected complete data log likelihood. The EM algorithm begins with an initial value $\theta^{(1)}$. One iteration of EM is composed of two steps:

E(expectation)-step. Calculate $Q_j(\theta) = E_j[\log p(\mathbf{d}, \mathbf{h} | \theta)]$, where the expectation is taken over the distribution $p(\mathbf{h} | \mathbf{d}, \theta^{(j)})$. Note that $\theta^{(j)}$ is a numerical quantity, so the expectation replaces functions of \mathbf{h} with numerical quantities, *not* symbolic functions of θ .

M(aximization)-step. Assign $\theta^{(j+1)}$ the value of θ that maximizes $Q_j(\theta)$.

The Q_j function is a guess, based on the current value of $\theta^{(j)}$, of the log-likelihood you would write down if you could see all the data. Maximizing Q_j leads to a new parameter $\theta^{(j+1)}$, which produces a new guess for the complete data log-likelihood, and so forth until the algorithm converges.

The EM algorithm is a very stable method for maximizing likelihood functions because the sequence of $\theta^{(j)}$'s that it produces is nondecreasing, that is, $p(\mathbf{d} | \theta^{(j)}) \leq p(\mathbf{d} | \theta^{(j+1)})$. To understand why, observe that $\log p(\mathbf{d}, \mathbf{h} | \theta) = \log p(\mathbf{d} | \theta) + \log p(\mathbf{h} | \mathbf{d}, \theta)$. Solving for $\log p(\mathbf{d} | \theta)$ and taking expected values over $p(\mathbf{h} | \mathbf{d}, \theta^{(j)})$ yields

$$\log p(\mathbf{d} | \theta) = Q_j(\theta) - H_j(\theta), \quad (3)$$

where $H_j(\theta) = E_j[\log p(\mathbf{h} | \mathbf{d}, \theta)]$. One may use Equation 3 to decompose $\log p(\mathbf{d} | \theta^{(j+1)}) - \log p(\mathbf{d} | \theta^{(j)})$ into two pieces, both of which are nonnegative.

$$\log \frac{p(\mathbf{d} | \theta^{(j+1)})}{p(\mathbf{d} | \theta^{(j)})} = [Q_j(\theta^{(j+1)}) - Q_j(\theta^{(j)})] + [H_j(\theta^{(j)}) - H_j(\theta^{(j+1)})]. \quad (4)$$

It is obvious that $Q_j(\theta^{(j+1)}) \geq Q_j(\theta^{(j)})$ because $\theta^{(j+1)}$ is chosen to maximize Q_j . To see that $H_j(\theta^{(j)}) \geq H_j(\theta^{(j+1)})$ apply Jensen's inequality

$$\begin{aligned} H_j(\theta^{(j+1)}) - H_j(\theta^{(j)}) &= E_j \left[\log \frac{p(\mathbf{h} | \mathbf{d}, \theta^{(j+1)})}{p(\mathbf{h} | \mathbf{d}, \theta^{(j)})} \right] \\ &\leq \log E_j \left[\frac{p(\mathbf{h} | \mathbf{d}, \theta^{(j+1)})}{p(\mathbf{h} | \mathbf{d}, \theta^{(j)})} \right] \\ &= \log \sum_{\mathbf{h}} \frac{p(\mathbf{h} | \mathbf{d}, \theta^{(j+1)})}{p(\mathbf{h} | \mathbf{d}, \theta^{(j)})} p(\mathbf{h} | \mathbf{d}, \theta^{(j)}) \\ &= \log \sum_{\mathbf{h}} p(\mathbf{h} | \mathbf{d}, \theta^{(j+1)}) = \log 1 = 0. \end{aligned} \quad (5)$$

Equation 5 implies that any move away from $\theta^{(j)}$ reduces $H_j(\theta)$. Thus, any θ for which $Q_j(\theta) \geq Q_j(\theta^{(j)})$ satisfies $p(\mathbf{d} | \theta) \geq p(\mathbf{d} | \theta^{(j)})$. This fact has been used to produce several generalizations of the EM algorithm (e.g., Liu & Rubin, 1994; Meng & Rubin, 1993). Note that the sum in Equation 5 can be replaced by an integral with respect to an arbitrary measure without modifying the proof, so the result is general. Also note that the EM algorithm can be modified to produce Bayesian maximum posterior estimates under a prior distribution $p(\theta)$ by replacing $Q_j(\theta)$ with $Q_j(\theta) + \log p(\theta)$ in the M-step.

The EM Algorithm for HMMs

When applied to HMMs, the EM algorithm is sometimes called the Baum-Welch algorithm, as Baum et al. (1970) applied the algorithm to HMMs before Dempster et al. (1977) exposed its full generality. However, any disagreements about who “invented” the algorithm were put to rest by Lauritzen (1981) who notes its use by Thiele (1880) for a state space model almost 100 years earlier.

Applying the EM algorithm to HMMs merely involves writing the complete data log-likelihood in a form conducive to taking expected values.

$$\begin{aligned} \log p(\mathbf{d}, \mathbf{h} | \theta) &= \log \pi_0(h_1) + \log P_{h_1}(d_1) + \sum_{t=2}^n \log q(h_{t-1}, h_t) + \log P_{h_t}(d_t | \theta) \\ &= \sum_{s=0}^{S-1} I(h_t = s)[\log \pi_0(s) + \log P_s(d_1 | \theta)] \\ &\quad + \sum_{t=2}^n \sum_{r=0}^{S-1} \sum_{s=0}^{S-1} I(h_{t-1} = r) I(h_t = s)[\log q(r, s) + \log P_s(d_t | \theta)]. \end{aligned} \quad (6)$$

To compute $Q_j(\theta)$ one must calculate $p'_{trs} = p(h_{t-1} = r, h_t = s | \mathbf{d}, \theta^{(j)})$, which requires a forward-backward recursion developed later in the chapter. Once the p'_{trs} are available, Q_j becomes

$$\begin{aligned} Q_j(\theta) &= \sum_{s=0}^{S-1} \pi'_1(s | \theta^{(j)}) [\log \pi_0(s) + \log P_s(d_1 | \theta)] \\ &\quad + \sum_{t=2}^n \sum_{r=0}^{S-1} \sum_{s=0}^{S-1} p'_{trs} [\log q(r, s) + \log P_s(d_t | \theta)], \end{aligned} \quad (7)$$

where $\pi'_1(r | \theta^{(j)}) = \sum_s p'_{2rs}$. A later section presents examples of Q_j for specific choices of $P_s(d_t | \theta)$.

MCMC Data Augmentation

One drawback of the EM algorithm is that it provides no standard errors for the point estimates it produces. MCMC data augmentation is an alternative to EM that bases inference for θ on a sequence of random draws $(\theta^{(1)}, \mathbf{h}^{(1)}), (\theta^{(2)}, \mathbf{h}^{(2)}), \dots$ from a Markov chain, the stationary distribution of which is the Bayesian posterior distribution $p(\theta, \mathbf{h} | \mathbf{d})$. The random draws may be used to compute posterior expectations of functions of θ and \mathbf{h} by simply calculating means, medians, histograms, and so forth from the Monte Carlo sample. In particular, marginal distributions for quantities of interest can be estimated by forming histograms of the corresponding quantities in the Monte Carlo sample, ignoring the quantities to be marginalized out. These marginal distributions communicate the uncertainty associated with any estimated θ . Moreover, the marginal distributions estimated from the Monte Carlo sample are based on the full posterior distribution $p(\theta | \mathbf{d})$, so they are superior to classical “standard errors” that only consider the local behavior of $p(\theta | \mathbf{d})$ or $p(\mathbf{d} | \theta)$ at the mode.

The set of authors who have used MCMC data augmentation for HMMs is already very large and is growing. An incomplete list includes Albert and Chib (1993a), Chib (1996), Scott (1999, 2002b), Robert and Titterington (1998), Robert, Celeux, and Diebolt (1993), and Robert, Rydén, and Titterington (1999).

MCMC Data Augmentation: The General Case

MCMC data augmentation involves two steps:

Data augmentation: Draw $\mathbf{h}^{(j)}$ from $p(\mathbf{h} | \mathbf{d}, \theta^{(j)})$.

Posterior sampling: Draw $\theta^{(j+1)}$ from $p(\theta | \mathbf{d}, \mathbf{h}^{(j)})$.

Clearly the sequence $(\theta^{(j)}, \mathbf{h}^{(j)})$ is a Markov chain because the algorithm for sampling $(\theta^{(j+1)}, \mathbf{h}^{(j+1)})$ depends only on $(\theta^{(j)}, \mathbf{h}^{(j)})$. It should also be obvious that if $(\theta^{(j)}, \mathbf{h}^{(j)})$ is a random draw from $p(\theta, \mathbf{h} | \mathbf{d})$ then $(\theta^{(j+1)}, \mathbf{h}^{(j+1)})$ is a random draw with the same distribution, so $p(\theta, \mathbf{h} | \mathbf{d})$ is the stationary distribution of the chain.

If the data augmentation and posterior sampling steps can both be done directly then the algorithm described above is the data augmentation algorithm of Tanner and Wong (1987). If either step is difficult to do directly, it may be replaced by a sequence of Gibbs or Metropolis-Hastings steps that preserve $p(\theta, \mathbf{h} | \mathbf{d})$ as the target distribution. For example, if $p(\theta | \mathbf{d}, \mathbf{h})$ is difficult to sample, one may split θ into $(\theta_1, \dots, \theta_K)$ and draw each θ_k from $p(\theta_k | \mathbf{d}, \mathbf{h}, \theta_{-k})$. Drawing each θ_k from its full conditional distribution is known as the Gibbs sampler (Geman & Geman, 1984). If one or more θ_k is difficult to sample then one may draw θ_k^* from a proposal density $f(\theta_k^* | \theta_k^{(j)})$ and determine $\theta_k^{(j+1)}$ based on the Metropolis-Hastings probability (Hastings, 1970; Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953)

$$\alpha = \min \left\{ \frac{p(\theta_k^* | \theta_{-k}, \mathbf{h}^{(j)}, \mathbf{d}) / f(\theta_k^* | \theta_k^{(j)})}{p(\theta_k^{(j)} | \theta_{-k}, \mathbf{h}^{(j)}, \mathbf{d}) / f(\theta_k^{(j)} | \theta_k^*)}, 1 \right\}. \quad (8)$$

Assign $\theta_k^{(j+1)} = \theta_k^*$ with probability α , and $\theta_k^{(j+1)} = \theta_k^{(j)}$ with probability $1 - \alpha$. The proposal distribution $f(\theta_k^* | \theta_k^{(j)})$ may depend on the current $\theta_k^{(j)}$, but it does not have to. In particular, if f is the full conditional distribution of θ_k then both numerator and denominator in Equation 8 equal 1, so $\alpha \equiv 1$. Thus, the Gibbs sampler is a special case of the Metropolis-Hastings algorithm in which all proposals are accepted.

The Gibbs sampler and Metropolis-Hastings algorithms have been reviewed in numerous places. For further details see Chapter 5 in this volume, Besag, Green, Higdon, and Mengersen (1995), Casella & George (1992), Chib & Greenberg (1995), Gelfand & Smith (1990), Gilks, Richardson, and Spiegelhalter (1996), and Liu (2001).

MCMC Data Augmentation for HMMs

There is little one can say about the posterior sampling step without knowing the conditional distributions $P_s(d_t | \theta)$. However, these distributions often are chosen to make the posterior sampling step easy once \mathbf{h} has been drawn.

There are two methods for performing the data augmentation step for HMMs. The first uses the stochastic forward-backward recursions discussed in a later section to draw $\mathbf{h}^{(j)}$ directly from $p(\mathbf{h} | \mathbf{d}, \theta^{(j)})$. The second iteratively draws each h_t from

$$p(h_t | h_{-t}, d_t, \theta) \propto q(h_{t-1}, h_t)q(h_t, h_{t+1})P_{h_t}(d_t | \theta). \quad (9)$$

The random draw described in Equation 9 conditions on the most recently drawn h_{t-1} , h_{t+1} , and θ . Call Equation 9 “Gibbs data augmentation.” Scott (2002a) compares the Gibbs and forward-backward data augmentation methods. Gibbs data augmentation is an $O(Sn)$ algorithm, whereas forward-backward is $O(S^2n)$, so Gibbs augmentation requires less computer

time than forward-backward to complete one draw of \mathbf{h} . However, the Gibbs draw of \mathbf{h} is “less random” than the forward-backward draw, so the overall MCMC algorithm can mix more rapidly using the forward-backward recursions, especially when there is substantial dependence in \mathbf{h} .

Missing Data Summary

Bayesian MCMC methods have two advantages over the EM algorithm. First, the draw of θ in the posterior sampling step conditions on the value of \mathbf{h} drawn in the data augmentation step. With \mathbf{h} observed, there is no integral to do as there is in the E-step of EM. Second, Bayesian MCMC methods allow one to work directly with the exact Bayesian posterior distribution of quantities of interest. Thus, one need not worry about details such as whether the asymptotic distribution of the maximum likelihood estimator is normal, or if so, whether n is large enough so that normality has been achieved. The only approximation in MCMC is Monte Carlo error, which may be made irrelevant by a long enough simulation run. The disadvantage of MCMC is that it can be slow if long simulation runs are used, so that EM may be preferred if only a point estimate of θ is needed.

LOCAL COMPUTATION

This section describes local computation methods for computing probabilities relevant to HMMs. Local computation is required to implement the EM algorithm discussed earlier. Local computation is not required for MCMC data augmentation, but it can be used to make data augmentation more efficient (Scott, 2002a). Perhaps more important, local computation can be used to estimate \mathbf{h} once an estimate of θ has been obtained. This section describes three local computation algorithms. The likelihood recursion is a technique for computing $p(\mathbf{d} | \theta)$. The forward-backward recursions compute the posterior distribution of the t 'th transition in \mathbf{h} , which is required by the EM algorithm in Equation 7. The Viterbi algorithm can be used to obtain the most likely configuration of \mathbf{h} given \mathbf{d} and θ , which can be different from the most likely \mathbf{h} determined from the sequence of marginal distributions produced by the forward-backward recursions.

The Likelihood Recursion

The likelihood recursion is a method for evaluating $p(\mathbf{d} | \theta)$ in $O(S^2n)$ calculations. For an arbitrary vector $\mathbf{x} = (x_1, \dots, x_n)$, define $x_j^k = (x_j, \dots, x_k)$. Define $\mathcal{L}_t(s) = p(d_1^t, h_t = s | \theta)$ to be the joint likelihood contribution of all data up to time t and the event $\{h_t = s\}$. Clearly $\mathcal{L}_t(s) = p(d_1^t | \theta)\pi_t(s | \theta)$, where $\pi_t(s | \theta) = p(h_t = s | d_1^t, \theta)$. The likelihood contribution from d_1^t is $p(d_1^t | \theta) = \sum_s \mathcal{L}_t(s)$. The likelihood recursion is a recipe for computing \mathcal{L}_t from \mathcal{L}_{t-1} .

$$\begin{aligned}\mathcal{L}_t(s) &= \sum_{r=0}^{S-1} p(d_t, h_t = s, d_1^{t-1}, h_{t-1} = r | \theta) \\ &= P_s(d_t | \theta) \sum_{r=0}^{S-1} q(r, s) \mathcal{L}_{t-1}(r).\end{aligned}\tag{10}$$

For each s , Equation 10 requires a sum over S elements, which means moving from \mathcal{L}_{t-1} to \mathcal{L}_t is an $O(S^2)$ operation, and thus computing $p(\mathbf{d} | \theta)$ is $O(S^2n)$. Equation 10 is the simplest

expression of the likelihood recursion, but it is susceptible to underflow. Most statistical applications avoid computer underflow by computing log-likelihood $\ell_t = \log p(d_1^t | \theta)$ instead of likelihood. To develop a more stable version of the likelihood recursion let $a_{trs} = \log \{P_s(d_t | \theta)q(r, s)\pi_{t-1}(r | \theta)\}$, and let $m_t = \max_{r,s} a_{trs}$. Then it is easy to show that

$$\ell_t = \ell_{t-1} + m_t + \log \left\{ \sum_{r=0}^{S-1} \sum_{s=0}^{S-1} \exp(a_{trs} - m_t) \right\}. \quad (11)$$

Equation 11 has two useful properties. First, each $P_s(d_t | \theta)$ need only be evaluated on the log scale, so that a single outlier is unlikely to cause underflow. Second, the largest element in the sum is always 1, so the log that must be taken is computationally stable. The following section on forward-backward recursions shows that $\pi_t(s | \theta) \propto \sum_r \exp(a_{trs})$, which sets up the next step in the recursion.

Derivatives of log-likelihood may be computed by recursively applying the derivative operator to Equation 11. After cancelling m_t and taking derivatives one arrives at

$$\frac{\partial \ell_t}{\partial \theta} = \frac{\partial}{\partial \theta} \ell_{t-1} + \sum_r \sum_s p_{trs} a'_{trs} \quad (12)$$

where $p_{trs} = \exp(a_{trs}) / \sum_r \sum_s \exp(a_{trs})$, and $a'_{trs} = \partial a_{trs} / \partial \theta$. Evaluating a'_{trs} requires a second recursion to compute $\partial \log \pi_{t-1}(r | \theta) / \partial \theta$. By writing $\pi_t(s | \theta) = \sum_r \exp(a_{trs}) / \sum_r \sum_s \exp(a_{trs})$ one may show

$$\frac{\partial}{\partial \theta} \log \pi_t(s | \theta) = \sum_r p_{tr|s} a'_{trs} - \sum_r \sum_s p_{trs} a'_{trs}, \quad (13)$$

where $p_{tr|s} = \exp(a_{trs}) / \sum_r \exp(a_{trs})$. Equation 13 is a recursion because a'_{trs} involves the expression $\partial \log \pi_{t-1}(r) / \partial \theta$. Recursions similar to Equations 12 and 13 may be used to compute second and further derivatives. Derivatives of log-likelihood can be used to fit HMMs using maximum likelihood estimation without resorting to the EM algorithm. Some authors (e.g., MacDonald and Zucchini, 1997) advocate this approach because the EM algorithm is known to converge only linearly near the maximum. However, it seems wise to begin the maximization with at least a few steps of EM to take advantage of the EM algorithm's stability. Newton-Raphson or conjugate gradient methods can be used to polish maximum likelihood estimates once in the neighborhood of the maximum.

The Forward-Backward Recursions

The forward-backward recursions compute the marginal distribution of each transition $p'_{trs} = p(h_{t-1} = r, h_t = s | \mathbf{d}, \theta)$, which is required by the EM algorithm in Equation 7. The forward recursion computes $p_{trs} = p(h_{t-1} = r, h_t = s | d_1^t, \theta)$, the distribution of the t 'th transition given observed data up to time t . The backward recursion updates these probabilities so that they condition on all observed data. The forward recursion is

$$\begin{aligned} p_{trs} &\propto p(h_{t-1} = r, h_t = s, d_t | d_1^{t-1}, \theta) \\ &= \pi_{t-1}(r | \theta)q(r, s)P_s(d_t | \theta) \\ &= \exp(a_{trs}) \end{aligned} \quad (14)$$

with proportionality reconciled by $\sum_r \sum_s p_{trs} = 1$. One may compute $\pi_t(s | \theta)$ by marginalizing over p_{trs} . Equations 11 and 14 imply that the log-likelihood can be stably computed by simply accumulating the log of the normalizing constant for Equation 14.

The backward recursion replaces $\mathbf{P}_t = (p_{trs})$ with $\mathbf{P}'_t = (p'_{trs})$ once the forward recursion has completed. Obviously $\mathbf{P}_n = \mathbf{P}'_n$, and one computes \mathbf{P}'_t from \mathbf{P}_t and \mathbf{P}'_{t+1} using

$$\begin{aligned} p'_{trs} &= p(h_{t-1} = r | h_t = s, d_1^n, \theta) p(h_t = s | d_1^n, \theta) \\ &= p(h_{t-1} = r | h_t = s, d_1^t, \theta) \pi'_t(s | \theta) \\ &= p_{trs} \frac{\pi'_t(s | \theta)}{\pi_t(s | \theta)}. \end{aligned} \quad (15)$$

The marginal distribution $\pi'_t(s | \theta) = p(h_t = s | \mathbf{d}, \theta)$ is computed as $\pi'_{t-1}(r | \theta) = \sum_s p'_{trs}$.

There is a stochastic version of the forward-backward recursion that one may use to simulate \mathbf{h} from $p(\mathbf{h} | \mathbf{d}, \theta)$. To implement it, draw (h_{n-1}, h_n) from \mathbf{P}_n , then draw h_t from the distribution proportional to column h_{t+1} of either \mathbf{P}_{t+1} or \mathbf{P}'_{t+1} . The stochastic backward recursion may be understood by factoring

$$p(\mathbf{h} | d_1^n, \theta) = p(h_n | \mathbf{d}, \theta) \prod_{t=1}^{n-1} p(h_{n-t} | h_{n-t+1}^n, \mathbf{d}, \theta). \quad (16)$$

Each term in Equation 16 is proportional to $p(h_{t-1} = r | h_t = s, \mathbf{d}, \theta) \propto p'_{trs} \propto p_{trs}$. Equation 15 implies that column s of \mathbf{P}_t and column s of \mathbf{P}'_t differ only by a factor depending on s , which is why either \mathbf{P}_t or \mathbf{P}'_t can be used in the simulation. This is a consequence of the fact that h_t is independent of d_{t+1}^n once h_{t+1} has been drawn.

The Viterbi Algorithm

The Viterbi algorithm (Viterbi, 1967) is a method of producing $\hat{\mathbf{h}}$, the value of \mathbf{h} that maximizes $p(\mathbf{h} | \mathbf{d}, \theta)$. Thus, $\hat{\mathbf{h}}$ is sometimes called the maximum a posteriori (MAP) estimate of \mathbf{h} . Estimating \mathbf{h} by MAP ensures that $\hat{\mathbf{h}}$ captures dependence that might be present in $p(\mathbf{h} | \mathbf{d}, \theta)$ but absent in marginal distributions. One may view the Viterbi algorithm as a generalization of the forward-backward recursions in which one “marginalizes” using maximizations instead of averages (Cowell et al., 1999). The following implementation of the Viterbi algorithm follows Fredkin and Rice (1992).

Define $L_1(s) = \pi_0(s) P_s(d_1 | \theta)$ and $L_t(h_t) = \max_{h_1, \dots, h_{t-1}} p(h_1^t, d_1^t | \theta)$. In words, $L_t(s)$ is the largest contribution to the complete data likelihood that can be obtained if $h_t = s$. The quantities $L_t(\cdot)$ are computed recursively through

$$L_t(s) = \max_r [L_{t-1}(r) q(r, s)] P_s(d_t | \theta). \quad (17)$$

To find $\hat{\mathbf{h}}$, run Equation 17 for $t = 1, \dots, n$, storing each $L_t(\cdot)$ as the algorithm progresses. Choose h_t to maximize $L_n(s)$, and for $t = n-1, \dots, 1$ choose

$$\hat{h}_t = \arg \max_{r \in S} L_t(r) q(r, \hat{h}_{t+1}) \quad (18)$$

to maximize the complete data likelihood conditional on \hat{h}_{t+1} .

Understanding the Recursions

The notation used in the first two parts of this section is somewhat nonstandard. It was chosen to emphasize the connection between the local computation methods for HMMs and a general local computation algorithm for graphical models (Cowell, Dawid, Lauritzen, & Spiegelhalter, 1999; Lauritzen, 1996; Lauritzen & Spiegelhalter, 1988). The general local computation algorithm operates by calculating marginal distributions of *cliques* on a graph. A clique is a set of nodes that are all neighbors. In Figure 6.1 the cliques are $A_t = (h_{t-1}, h_t)$ and $B_t = (h_t, d_t)$. Once the marginal distribution of a clique is available, the marginal distribution of any variable in the clique may be calculated using low dimensional sums or integrals. The local computation algorithm arranges the cliques of the conditional independence graph into a tree called a *junction tree*. The junction tree for an HMM is a chain $A_{t-1} \rightarrow B_{t-1} \rightarrow A_t, \dots$. The matrix \mathbf{P}_t in the forward-backward recursions represents the marginal distribution of clique A_t . The marginal distribution of a clique in the tree, conditional on all observed information about its ancestors in the tree, may be computed by recursively averaging the conditional distribution of the clique given its parents in the tree over the marginal distributions of its parents. The process reverses once the marginal distributions for the terminal leaves of the tree (a single leaf, B_n , for HMMs) have been computed. The general backward recursion then updates the marginal distribution of the parent cliques in the tree given the fully informed marginal distribution of its children.

A Numerical Example Illustrating the Recursions

For concreteness, suppose you are estimating a two-state Markov-Poisson HMM of the type described in the first section of this chapter. Imagine that you have computed $\pi_{t-1} = (.3, .7)$, the next data value is $d_t = 3$, and the current parameter values are $\lambda_0 = 1.7$, $\lambda_1 = 3.2$, and

$$\mathbf{Q} = \begin{pmatrix} .8 & .2 \\ .1 & .9 \end{pmatrix}.$$

The next step in the forward recursion is

$$\mathbf{P}_t \propto \begin{pmatrix} (.3)(.8)1.7^3 \exp(-1.7) & (.3)(.2)3.2^3 \exp(-3.2) \\ (.7)(.1)1.7^3 \exp(-1.7) & (.7)(.9)3.2^3 \exp(-3.2) \end{pmatrix}.$$

A factor of $3!$ has been eliminated from the Poisson likelihood because it is common to all elements of \mathbf{P}_t , and so it may be ignored due to proportionality. After normalization \mathbf{P}_t becomes

$$\mathbf{P}_t = \begin{pmatrix} 0.180 & 0.067 \\ 0.052 & 0.701 \end{pmatrix}$$

which says $p(h_{t-1} = 1, h_t = 1 | d_t^i) = .701$, whereas $p(h_{t-1} = 0, h_t = 1 | d_t^i) = .067$. Notice that observing d_t provides information about h_{t-1} . For example, $p(h_{t-1} = 0 | d_t^i) = 0.247$, which is less than $\pi_{t-1}(0) = 0.30$, the same probability calculated before observing d_t . To set up the next step of the recursion, compute $\pi_t(0) = 0.180 + 0.052 = 0.232$ and $\pi_t(1) = 0.701 + 0.067 = 0.768$.

For the backward recursion, suppose you compute $\pi'_t = (.15, .85)$. Then

$$\begin{aligned}\mathbf{P}'_t &= \begin{pmatrix} 0.180(.15)/(.232) & 0.067(.85)/(.768) \\ 0.052(.15)/(.232) & 0.701(.85)/(.768) \end{pmatrix} \\ &= \begin{pmatrix} 0.116 & 0.074 \\ 0.034 & 0.776 \end{pmatrix}.\end{aligned}$$

Computing $\pi'_{t-1}(0) = 0.116 + 0.074 = 0.19$ and $\pi'_{t-1}(1) = 0.034 + 0.776 = 0.81$ sets up the recursion for \mathbf{P}'_{t-1} .

For the Viterbi algorithm suppose you have computed $L_{t-1}(0) = .001$ and $L_{t-1}(1) = .003$. The next step is to compute the matrix

$$\begin{pmatrix} (.001)(.8)1.7^3 \exp(-1.7) & (.001)(.2)3.2^3 \exp(-3.2) \\ (.003)(.1)1.7^3 \exp(-1.7) & (.003)(.9)3.2^3 \exp(-3.2) \end{pmatrix} = \begin{pmatrix} 0.000718 & 0.000267 \\ 0.000269 & 0.003606 \end{pmatrix},$$

which yields $L_t(0) = \max(.000718, .000269) = .000718$ and $L_t(1) = \max(.000267, .003606) = .003606$. Should you later determine that $\hat{h}_{t+1} = 1$, then $\hat{h}_t = 1$ because $.000718(0.1) < .003606(0.9)$, where $0.1 = q(0, 1)$ and $0.9 = q(1, 1)$. Should the values of $L_t(s)$ threaten to underflow, they may either be renormalized or one may work with the log of Equation 17 instead.

ILLUSTRATIVE EXAMPLES AND APPLICATIONS

This section uses two examples to illustrate the machinery developed in the previous two sections. The first part of this section analyzes the fetal lamb data using a Poisson HMM; the second fits a Gaussian HMM to a data set describing quarterly growth in the U.S. gross national product.

Fetal Lamb Movements

For the moment assume the data in Fig. 6.2 were generated by a two-state Poisson HMM $P_s(d_t | \theta) \propto \exp(-\lambda_s) \lambda_s^{d_t}$, $s \in \{0, 1\}$. To apply the EM algorithm, one merely substitutes the specific form of P_s into Equation 7. The Q function for the Poisson HMM is

$$Q_j(\theta) = \sum_r \sum_s \hat{n}_{rs} \log q(r, s) + \sum_s (\hat{d}_{s+} \log \lambda_s - \hat{n}_s \lambda_s). \quad (19)$$

The expected complete data sufficient statistics in Equation 19 are the expected transition count $\hat{n}_{rs} = \sum_t p'_{trs}$, the expected number of observations in each state $\hat{n}_s = \sum_t \pi'_t(s | \theta^{(j)})$, and the expected total count for each state $\hat{d}_{s+} = \sum_t d_t \pi'_t(s | \theta^{(j)})$. The M-step of EM estimates $\hat{q}(r, s) = \hat{n}_{rs}/n$ and $\hat{\lambda}_s = \hat{d}_{s+}/\hat{n}_s$.

Prior Distributions and Label Switching

Bayesian MCMC methods require one to specify a prior distribution $p(\theta)$. In practice, conjugate prior distributions are almost always used when they are available. A likelihood and prior distribution are *conjugate* if the posterior distribution that they induce is of the

same family as the prior. The conjugate prior for the rate parameter in the Poisson likelihood is gamma $\Gamma(\lambda | a, b) \propto \lambda^{a-1} \exp(-b\lambda)$. The conjugate prior for the probability vector \mathbf{q} in the multinomial likelihood is Dirichlet $\mathcal{D}(\mathbf{q} | \mathbf{v}) \propto \prod_s q_s^{v_s-1}$. Among other benefits of using conjugate prior distributions is that their parameters may be viewed as “prior observations.” For example, one may view $a - 1$ and b in the gamma distribution as a prior sum and prior sample size. Similarly, $v_s - 1$ in the Dirichlet distribution may be viewed as a prior count of multinomial level s . A standard practice is to choose a prior distribution corresponding to a small number of prior observations. See Gelman, Carlin, Stern, and Rubin (1995) for a complete discussion about the role of conjugate priors in Bayesian inference.

Some care is required when specifying prior distributions for HMM parameters because the HMM likelihood is invariant to permutations of the state labels. For example, under the obvious conjugate prior for θ each $\lambda_s \sim \Gamma(\lambda_s | a_s, b_s)$ and each $\mathbf{q}_r = (q(r, s)) \sim \mathcal{D}(\mathbf{q}_r | \mathbf{v}_r)$, independent of all other λ 's and \mathbf{q} 's. Then the complete data posterior distribution of θ is

$$p(\theta | \mathbf{d}, \mathbf{h}) = \prod_s \Gamma(\lambda_s | a_s + d_{s+}, b_s + n_s) \prod_r \mathcal{D}(\mathbf{q}_r | \mathbf{v}_r + \mathbf{n}_r). \quad (20)$$

The complete data sufficient statistics in Equation 20 are $d_{s+} = \sum_t I(h_t = s)d_t$, $n_s = \sum_t I(h_t = s)$, and $\mathbf{n}_r = (n_{rs})$, where $n_{rs} = \sum_t I(h_{t-1} = r, h_t = s)$. Equation 20 says that the posterior sampling step can be accomplished by drawing a collection of independent gamma and Dirichlet deviates, which is easy to do. However, Fig. 6.3(a) shows the effect of averaging Equation 20 over $p(\mathbf{h} | \mathbf{d})$ to obtain $p(\theta | \mathbf{d})$. Every point in the sample space of \mathbf{h} has a corresponding point \mathbf{h}' where $h'_t = 1 - h_t$ for all t . Consequently $p(\theta | \mathbf{d}, \mathbf{h}) = p(\theta' | \mathbf{d}, \mathbf{h}')$, where θ' is θ with the “0” and “1” labels switched. With no information in the prior to distinguish

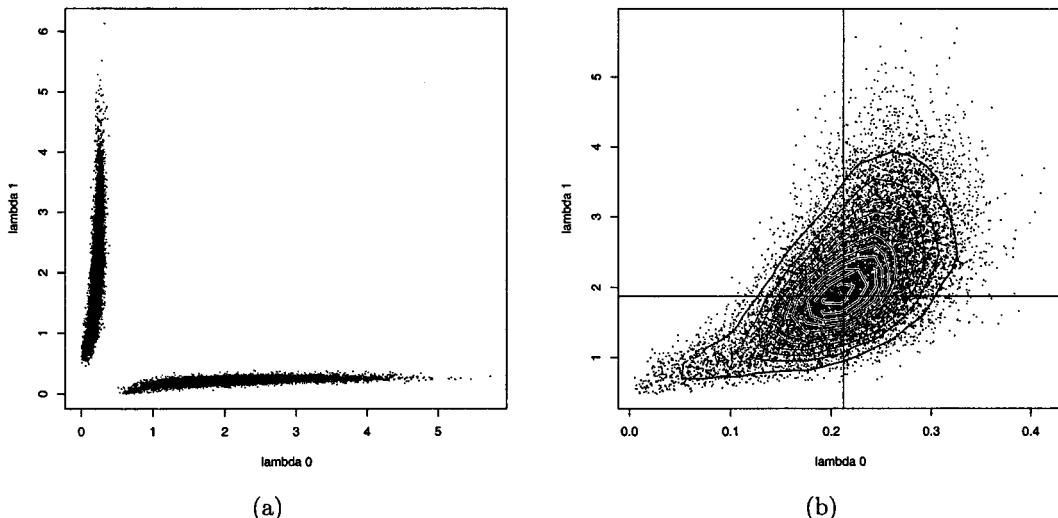


FIG. 6.3. Posterior samples of λ_0 and λ_1 for the two-state Poisson HMM applied to the fetal lamb data. (a) Exchangeable prior on λ_s , $q(r, s)$ with $a_s = b_s = v_{rs} = 1$ for all (r, s) . (b) Prior forcing $\lambda_1 > \lambda_0$. The prior on μ_0, μ_1 is independent gamma with $a_s = b_s = 1$ with $s = 0, 1$. The horizontal and vertical lines in panel (b) intersect at the posterior mode located using an EM algorithm. The contour lines reflect levels of the posterior density calculated using a kernel density estimator.

between the two modes, marginal posterior distributions for λ_0 and λ_1 are identical. For a general state space of size S , label switching produces $S!$ symmetric modes.

In practice, one typically defeats label switching by introducing a constraint that ties θ to a specific ordering of the state labels. Such a constraint may be considered part of the prior distribution. Scott (2002a) introduced a method for maintaining the natural ordering $\lambda_0 < \dots < \lambda_{S-1}$ in the Poisson model. Let $\mu_0 = \lambda_0$, and let $\mu_s = \lambda_s - \lambda_{s-1}$ represent the increment in λ_s . Let $d_t = D_{t0} + \dots + D_{tS-1}$ where

$$p(D_{ts} | h_t, \theta) = \begin{cases} \text{Pois}(D_{ts} | \mu_s) & \text{if } s \leq h_t \\ I(D_{ts} = 0) & \text{otherwise.} \end{cases} \quad (21)$$

One can understand Equation 21 by imagining that d_t is the sum of contributions from S independent regimes, which are sometimes inactive. If regime s is active then its contribution is a Poisson random variable with mean μ_s . The inactive regimes at time t are $h_{t+1}, \dots, S-1$. Regimes $0, \dots, h_t$ actively contribute to d_t . Clearly, Equation 21 implies that $P_s(d_t | \theta) = \text{Pois}(\mu_0 + \dots + \mu_s) = \text{Pois}(\lambda_s)$. However, now $\lambda_s > \lambda_{s-1}$ is enforced because μ_s must be positive to be a legal Poisson parameter. The natural prior for the ordered model assumes independent increments $\mu_s \sim \Gamma(\mu_s | a_s, b_s)$, where $a_s = 1$ and b_s may be thought of as a prior sum and sample size from regime s .

Equation 21 adds a second component to the MCMC data augmentation step. The first component samples \mathbf{h} from $p(\mathbf{h} | \mathbf{d}, \theta)$ using the forward-backward recursions. The second samples $D_t = (D_{t0}, \dots, D_{tS-1})$ from a multinomial distribution with total d_t and probability vector proportional to $(\mu_0, \dots, \mu_{h_t}, 0, \dots, 0)$. Note that drawing D_t at each step does not adversely affect the mixing properties of the MCMC algorithm because the draw of \mathbf{h} is independent of $\mathbf{D} = (D_1, \dots, D_n)$. The posterior sampling step draws from

$$p(\theta | \mathbf{d}, \mathbf{h}, \mathbf{D}) = \prod_r \mathcal{D}(\mathbf{q}_r | \mathbf{v}_r + \mathbf{n}_r) \prod_s \Gamma(\mu_s | a_s + D_{s+}, b_s + n_{\geq s}). \quad (22)$$

The complete data sufficient statistics in Equation 22 are the sum from regime s : $D_{s+} = \sum_t D_{st}$, and the number of times regime s was active: $n_{\geq s} = \sum_t I(h_t \geq s)$.

Figure 6.3(b) shows the effect of the ordering constraint. Under the exchangeable prior on (λ_0, λ_1) in Fig. 6.3(a) the sampler converges into different regions depending on its starting values. In Fig. 6.3(b) reversing the order of the starting values has no effect on the behavior of the sampler because it is sampling from an identified model. There is only one mode in the posterior distribution of the ordered model.

Note that label switching is a property of the HMM likelihood. It is not a property of a particular MCMC algorithm used to explore the posterior distribution. However, label switching is less of a problem for point estimation algorithms such as EM, which typically climb into one of the $S!$ symmetric modes and remain ignorant of the rest.

Parameter Estimation: MCMC Versus EM

We are now in a position to see the strengths and weaknesses of MCMC data augmentation methods relative to point estimation methods such as EM. The primary advantage of MCMC is that it provides a representation of the entire posterior distribution $p(\theta | \mathbf{d})$. Put another way, MCMC gives you all of Fig. 6.3(b), whereas EM gives only the posterior mode. An important consequence of having the entire posterior distribution of model parameters at your disposal is that inferences for model parameters are freed from asymptotics. For example, in Fig. 6.3(b)

a bivariate normal distribution centered on the mode fails to adequately describe the tails of the distribution. This particular approximation could be improved by examining the parameters on the log scale, but in general multivariate normality becomes less reliable as S grows.

The disadvantage of MCMC (or any other method of calculating the entire posterior distribution) is that it is more time-consuming than point estimation. The 10,000 posterior samples plotted in Fig. 6.3(b) were obtained in slightly under 18 seconds. Only .07 seconds were needed to discover the mode of the posterior distribution using the EM algorithm. These times are based on a forward-backward algorithm implemented in C on an 850 MHz Pentium III computer.

Predicting \mathbf{h}

MCMC can improve estimates of \mathbf{h} as well as estimates of θ . If MCMC is not an option, then \mathbf{h} must be estimated using a two-stage procedure. The first stage obtains $\hat{\theta}$, a point estimate of θ . The second stage computes a summary describing $p(\mathbf{h} | \mathbf{d}, \hat{\theta})$. The summary is typically either the MAP estimate $\hat{\mathbf{h}}$ computed using the Viterbi algorithm, or the sequence of marginal distributions $\pi_t'(h_t | \hat{\theta})$ from the forward-backward recursions. Many applications consider the overall configuration of \mathbf{h} to be much more interesting than the status of an individual h_t . Examples include speech recognition, where h_t is a phoneme in the word \mathbf{h} , and genetics, where h_t is a DNA base pair in the gene \mathbf{h} . In such cases the Viterbi (MAP) estimate is often preferred because it captures dependence in $p(\mathbf{h} | \mathbf{d}, \hat{\theta})$ which can be overlooked by marginal distributions. The marginal distributions produced by the forward-backward algorithm may be preferred if attention centers on one particular h_t . For example, in tracking problems the most recent h_t is of much greater interest than previous h 's. A significant advantage of $\pi_t(h_t | \hat{\theta})$ is that it communicates the uncertainty about each hidden state. Thus, practitioners may wish to employ the forward-backward recursions to calculate uncertainty estimates associated with Viterbi point estimates.

Figure 6.4 illustrates how HMMs allow neighboring states to communicate information about \mathbf{h} . Figure 6.4(a) compares the output from the forward-backward recursions and the Viterbi algorithm for the two-state model applied to the fetal lamb data. The marginal probability of fetal activity is slightly less than 0.5 for intervals 170 and 171. However, nearby intervals strongly indicate fetal activity, so $\hat{h}_t = 1$ for $t = 170, 171$. Figure 6.4(b) presents the marginal probabilities of fetal activity from a two-state Poisson finite mixture model. The finite mixture model assumes consecutive observations are independent, so it disregards information about fetal activity from neighboring intervals when calculating the distribution of h_t .

Many statisticians would characterize the methods described above as *empirical Bayes* procedures because they fix $\theta = \hat{\theta}$ in their posterior probability calculations. By doing so, empirical Bayes procedures ignore the fact that $\hat{\theta}$ is an uncertain quantity. Fully Bayes estimates differ from empirical Bayes by averaging over the posterior distribution of model parameters. In other words, fully Bayes estimates of \mathbf{h} come from $p(\mathbf{h} | \mathbf{d}) = \int p(\mathbf{h} | \mathbf{d}, \theta)p(\theta | \mathbf{d}) d\theta$. The difference between Bayes and empirical Bayes estimation may be understood by considering the familiar simple linear regression model $y \sim \mathcal{N}(\beta_0 + \beta_1 x, \sigma^2)$. A rough estimate of the prediction interval for y at any given x is given by $\hat{y} \pm 2s$. The interval is obtained from $p(y | \mathbf{x}, \beta = \hat{\beta})$, where β_0, β_1 , and s^2 are least squares estimates of β and σ^2 . However, students are taught to prefer

$$\hat{y} \pm 2s \sqrt{1 + \frac{1}{n} + \frac{(x - \bar{x})^2}{\sum_i (x_i - \bar{x})^2}}$$

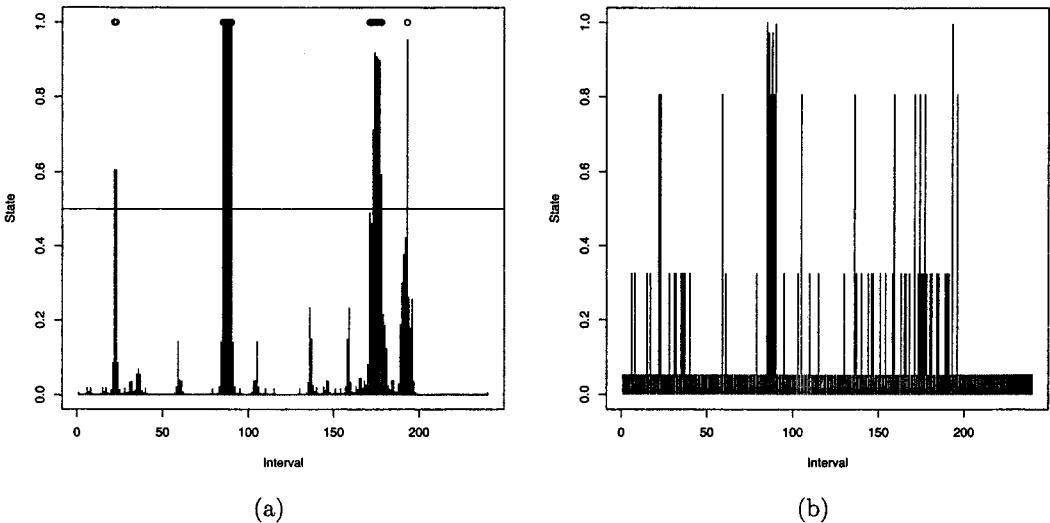


FIG. 6.4. (a) Forward-backward and Viterbi output for the two-state model applied to the fetal lamb data. The MAP estimate $\hat{\mathbf{h}}$ is zero everywhere except at points indicated by open circles. Vertical lines are marginal probabilities of fetal activity computed by the forward-backward algorithm. (b) Probability of fetal activity based on a two-state finite mixture model.

because it captures the uncertainty about the estimated model parameters. Incorporating model parameter uncertainty in predictions remains important once one moves beyond Gaussian linear models. However, model parameter uncertainty is often omitted from predictions because including it can be computationally challenging.

MCMC provides a simple method of including model parameter uncertainty in the marginal probabilities of hidden states. Let $\pi_t(s) = p(h_t = s | \mathbf{d})$. One may calculate $\pi_t(s)$ either by $\pi_t(s) \approx 1/m \sum_{j=1}^m \pi_t(h_t^{(j)} | \theta^{(j)})$ or $\pi_t(s) = 1/m \sum_{j=1}^m I(h_t^{(j)} = s)$. The first method is slightly less variable than the second. Figure 6.5 compares Bayes and empirical Bayes marginal probabilities of fetal activity. Bayes probabilities tend to be closer to $1/2$ than empirical Bayes probabilities, which correctly reflects the greater uncertainty one faces when model parameters are not known to equal exact values.

A further advantage of MCMC is that it allows one to compute the posterior distribution of any arbitrary function of \mathbf{h} . For example, suppose you are interested in the distribution of time of the k 'th transition from 0 to 1. MCMC allows you to simply record this time for each imputed $\mathbf{h}^{(j)}$ from the MCMC sequence and make a histogram of the results. It is generally not possible to construct forward-backward recursions for computing empirical Bayes distributions for arbitrary functions of \mathbf{h} .

Selecting S

Thus far, all computations have assumed $S = 2$. Choosing S is a model selection problem which can draw on the many statistical tools that have been developed for model selection. From the Bayesian perspective, S is an unknown quantity whose value is to be inferred from its posterior distribution given \mathbf{d} . An asymptotic approximation to $\log(p(S | \mathbf{d}))$ was given by Schwarz (1978). The Schwarz criterion is $C(S) = \ell(\hat{\theta}) - k_S \log(n)/2$, where $\ell(\theta)$ is the

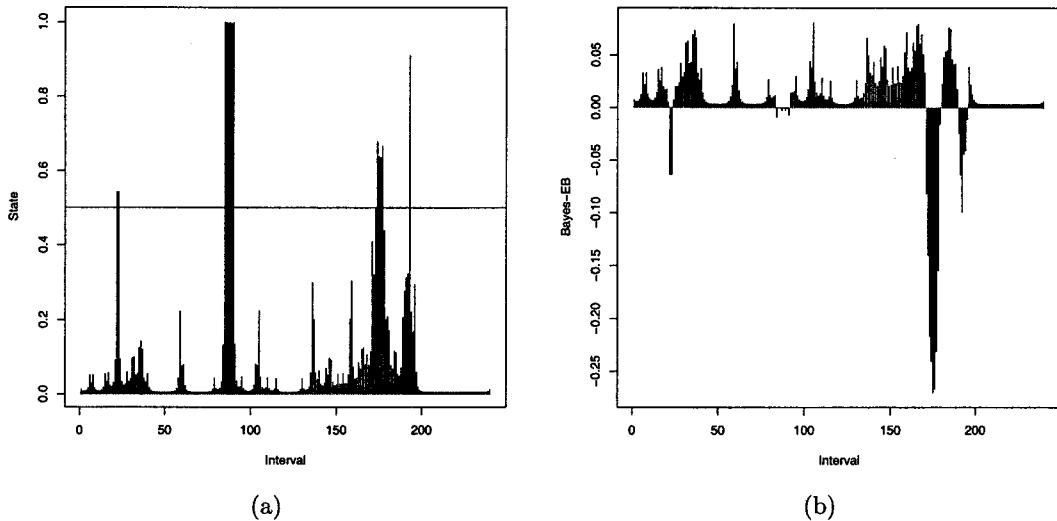


FIG. 6.5. Bayes versus empirical Bayes probabilities of fetal activity. (a) Bayes probabilities. (b) Difference between Bayes and empirical Bayes probabilities.

maximized log likelihood and k_S is the number of free parameters in a model with state space size S . The quantity $-2C(S)$ is called the Bayesian information criterion (BIC), which is a frequently used tool in model selection problems.

As with all asymptotic arguments, the correctness of BIC depends on having a sufficiently large sample size, which one cannot check unless exact methods are available. Scott (2002a) proposed the following method for computing $p(S | \mathbf{d})$ using MCMC. Let $\Theta = (\theta_1, \dots, \theta_{S_{\max}})$, where θ_S is the parameter of an HMM with state space size S . Assume a prior distribution in which $p(\Theta) = \prod_S p(\theta_S)$. Then the posterior distribution of S is

$$\begin{aligned} p(S | d_1^n) &= \int p(S | d_1^n, \Theta) p(\Theta | d_1^n) d\Theta \\ &\approx 1/m \sum_{j=1}^m p(S | d_1^n, \Theta^{(j)}), \end{aligned} \tag{23}$$

where $p(S | d_1^n, \Theta^{(j)}) \propto p(d_1^n | \theta_S^{(j)}, S)p(S)$, and $\Theta^{(j)} = (\theta_S^{(j)})$ is the j th draw of each θ_S from S_{\max} independent MCMC algorithms. The likelihood recursion is invoked to calculate the likelihood associated with each drawn θ_S .

Table 6.1 shows the calculations required to compute BIC, as well as the posterior probabilities of S computed using MCMC. The Schwarz approximation suggests that the two state model is $\exp[C(2) - C(3)] \approx 40$ times more likely than its closest competitor, the three state model. The MCMC calculation indicates that the three state model is clearly superior. The discrepancy occurs because the sample size of 240 is not large enough to ensure asymptotic normality. Figure 6.6 presents estimated marginal posterior distributions for the Markov transitions probabilities $q(r, s)$ under the two- and three-state models applied to the fetal lamb data. Several of these probabilities have nonnormal distributions, which violates one of BIC's central assumptions.

TABLE 6.1
MCMC versus BIC for Selecting S

S	Maximized Log-Posterior	k_S	$C(S)$	BIC	$p(S \mid \mathbf{d})$
1	-174.3	1	-177.0	354.0	0.000
2	-150.7	4	-161.6	323.2	0.051
3	-140.7	9	-165.3	330.6	0.650
4	-139.2	16	-183.1	366.2	0.229
5	-139.5	25	-208.0	416.0	0.066
6	-139.8	36	-238.4	476.8	0.004

Note: The posterior probabilities $p(S \mid \mathbf{d})$ were computed using independent MCMC runs of 20,000 iterations each.

The Business Cycle

This section presents a slightly simplified version of an HMM used by Hamilton (1989, 1990) to describe fluctuations in U.S. gross national product (GNP). Figure 6.7 shows quarterly differences in log GNP from 1947 to 1982, representing the economic growth rate during each quarter. Assume that during interval t the U.S. economy is either in a state of recession ($h_t = 0$) or expansion ($h_t = 1$), and that the economic growth rate during period t is $d_t \sim \mathcal{N}(\mu_{h_t}, \sigma^2)$. The Q function for the EM algorithm is

$$Q_j(\theta) = \sum_r \sum_s \hat{n}_{rs} \log q(r, s) + -\frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \left(d_+^2 - 2 \sum_s \hat{d}_{s+} \mu_s + \sum_s \hat{n}_s \mu_s^2 \right). \quad (24)$$

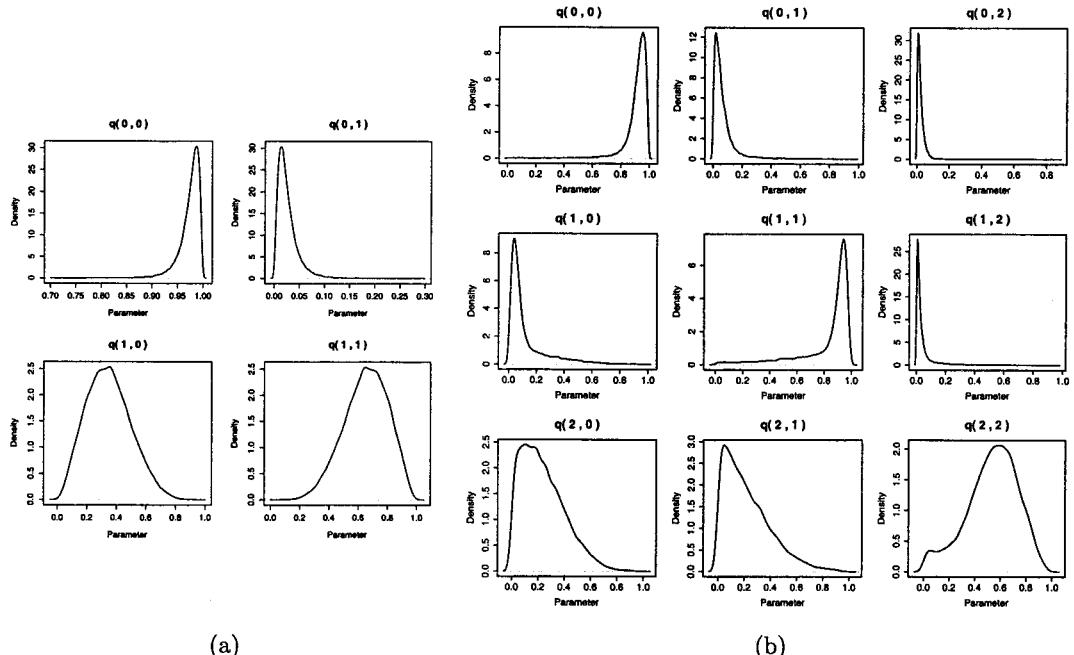


FIG. 6.6. Estimated posterior distributions of Markov transition probabilities for the (a) two- and (b) three-state Poisson model applied to the fetal lamb data. Several of these parameters have nonnormal posterior distributions.

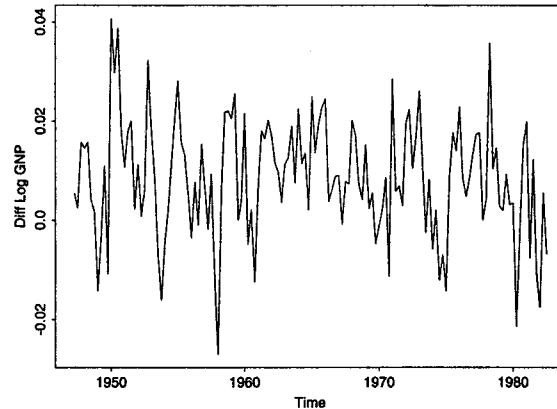


FIG. 6.7. Quarterly differences in U.S. log GNP. Source: Bureau of Economic Analysis (www.bea.doc.gov).

The expected complete data sufficient statistics in Equation 24 are as in Equation 19 plus the sum of squares $d_+^2 = \sum_t d_t^2$. The M-step of EM estimates $\hat{\mu}_s = \hat{d}_{s+}/\hat{n}_s$, $\hat{q}(r, s) = \hat{n}_{rs}/n$, and $\hat{\sigma}^2 = (d_+^2 - \sum_s \hat{n}_s \hat{\mu}_s^2)/n$.

One could allow σ^2 to depend on h_t as well, but doing so requires an informative prior on σ_s^2 to prevent a well-known singularity in the normal likelihood. The singularity occurs when state s is assigned only one point, say d_1 , and one estimates $\hat{\mu}_s = d_1$. Then the sum of squares in likelihood component s is exactly zero, which allows the likelihood to approach infinity as $\sigma_s^2 \rightarrow 0$.

The conjugate prior for this model assumes independent Dirichlet priors on the rows of the Markov transition probability matrix, and $p(\mu_0, \mu_1, \sigma^2) = \Gamma(1/\sigma^2 | DF/2, SS/2) \prod_s \mathcal{N}(\mu_s | M_s, \tau^2)$. Here DF is the prior degrees of freedom, and SS is the prior sum of squares. The prior expectation of $1/\sigma^2$ is DF/SS . The prior is further truncated to force μ_0 and μ_1 to both lie within the range of the data. Occasionally, the MCMC sampling algorithm will assign no observations to one of the states. When a state collapses in this way, its mean is drawn from the prior distribution. Truncating the support of the prior limits the time one must wait for a parameter from a collapsed state to return to a region of high posterior probability.

Figure 6.8(a) plots samples from the posterior distribution of (μ_0, μ_1) without any identifiability constraints. Label switching is clearly a more serious problem here than earlier (under Fetal Lamb Movements) because the two modes in the posterior distribution are poorly separated. Figure 6.8(a) is a single MCMC run of 20,000 iterations. Contrast this with Fig. 6.3(a), which combines two different MCMC runs with different starting values. The modes in Fig. 6.3(a) are sufficiently far apart that if only one MCMC run were done for the fetal lamb data then no label switching would occur. Evidently, there are times when label switching can be safely ignored and times when it cannot. Given that one cannot know in advance whether label switching will be a serious problem, the safest approach is to identify the model using a parameter ordering that reflects the intended meaning of the hidden states.

The natural ordering in this problem constrains $\mu_1 > \mu_0$ to comply with the notion that the economy grows faster during periods of expansion than during periods of contraction. The ordering may be enforced by truncating the support of the prior distributions so that $p(\mu_0, \mu_1) \propto \mathcal{N}(\mu_0 | M_0, \tau^2) \mathcal{N}(\mu_1 | M_1, \tau^2) I(\mu_0 < \mu_1)$. Under this prior the complete data

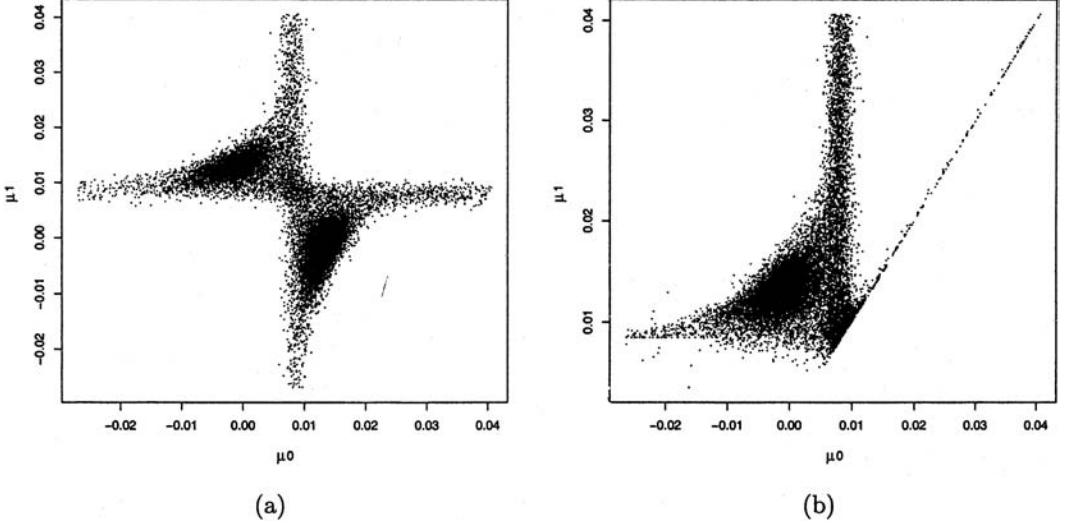


FIG. 6.8. Samples from the posterior distribution of μ_0 and μ_1 for the U.S. GNP data (a) ignoring label switching (b) constraining $\mu_1 > \mu_0$. Both panels have prior distributions truncated beyond range of data.

posterior distribution is

$$\begin{aligned}
 p(\theta | \mathbf{d}, \mathbf{h}) \propto & \prod_r \mathcal{D}(\mathbf{q}_r | \mathbf{n}_r + \mathbf{v}_r) \\
 & \times \Gamma\left(\frac{1}{\sigma^2} \left| \frac{DF + n}{2}, \frac{SS + \sum_t (d_t - \mu_{h_t})^2}{2}\right.\right) \\
 & \times \prod_s \mathcal{N}\left(\mu_s \left| \frac{d_{s+}/\sigma^2 + M_s/\tau^2}{n_s/\sigma^2 + 1/\tau^2}, \frac{1}{n_s/\sigma^2 + 1/\tau^2}\right.\right) I(\mu_0 < \mu_1) I(\mu_s \in R(\mathbf{d})),
 \end{aligned} \tag{25}$$

where $R(\mathbf{d})$ is the interval $[\min_t \{d_t\}, \max_t \{d_t\}]$. One method of sampling from Equation 25 is to independently sample the rows of \mathbf{Q} from a Dirichlet distribution. Then sample $1/\sigma^2$ from a gamma distribution. Finally, propose μ_s^* from a normal distribution with the given mean and variance. If μ_s^* conforms to the truncation restrictions, then set $\mu_s^{(j+1)} = \mu_s^*$; otherwise, retain $\mu_s^{(j+1)} = \mu_s^{(j)}$. Figure 6.8(b) plots the posterior distributions of (μ_0, μ_1) after imposing the constraint. Note that there is still a second mode in the distribution, but it is not caused by label switching. The second mode occurs because there is some support in the data for a single state model. Assuming that the “true model” has either one or two states, MCMC can be used to compute $p(S = 1 | \mathbf{d}, S_{\max} = 2) = 0.115$, and $p(S = 2 | \mathbf{d}, S_{\max} = 2) = 0.885$. The two-state model is more likely, but there is some support for $S = 1$.

Figure 6.9 compares Bayes probabilities of recession computed using the two state Gaussian HMM with ordered means to the corresponding probabilities computed using a finite mixture model. As discussed in an earlier section, the HMM’s ability to allow neighboring observations to communicate information about \mathbf{h} allows it to make a better judgment about the state of the economy at time t . The finite mixture model assumes the h_t ’s are independently generated, which forces it to rely only on the current observation when calculating $p(h_t = 0)$.

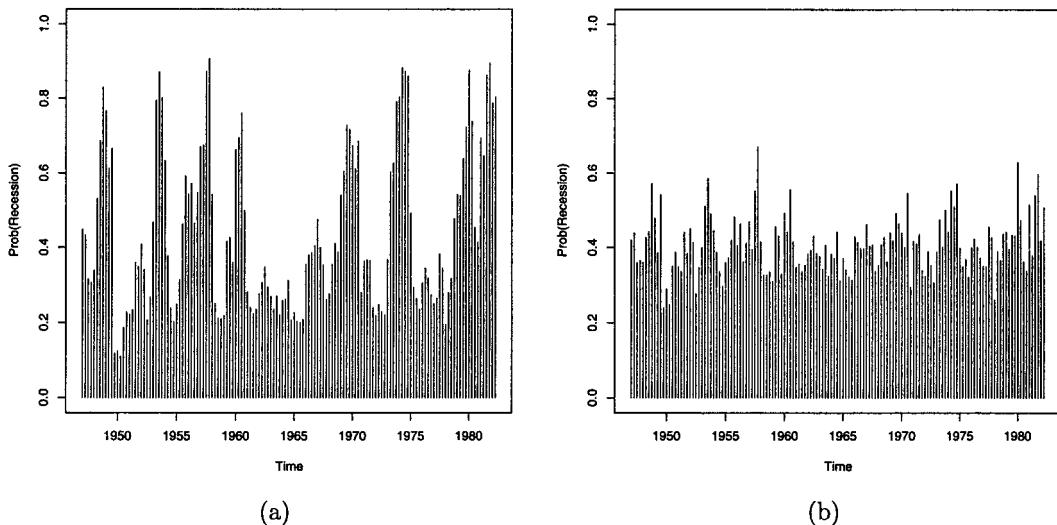


FIG. 6.9. Marginal Bayes probabilities of recession computed using (a) the two-state Gaussian HMM, and (b) the two-state Gaussian finite mixture model. Probabilities are based on MCMC runs of 20,000 iterations constraining $\mu_1 > \mu_0$.

HMM STATIONARY AND PREDICTIVE DISTRIBUTIONS

Deriving the stationary and predictive distributions for d_t , and their corresponding moments, is simplified by the following notation. Let the column vector $\boldsymbol{\pi}_0$ be the stationary distribution of the matrix of Markov transition probabilities \mathbf{Q} , so that $\boldsymbol{\pi}_0^\top \mathbf{Q} = \boldsymbol{\pi}_0^\top$. Let $\boldsymbol{\pi}_t = (\pi_t'(s | \theta))$ be the posterior distribution of h_t given d_1^t , written as a column vector. Let $\boldsymbol{\pi}_{\tau|t}^\top = \boldsymbol{\pi}_t^\top \mathbf{Q}^\tau$ be the predictive distribution of $h_{t+\tau}$ given d_1^t . Let μ_s and Σ_s be the conditional mean and variance of d_t given $h_t = s$. Let $\boldsymbol{\mu}$ be the matrix whose r th row is μ_r . The following calculations follow sections of MacDonald and Zucchini (1997), who used the fitted stationary distribution and HMM autocorrelation function as model diagnostics.

Stationary Distribution of d_t

The stationary distribution of d_t is simply $p(d_t | \theta) = \sum_s \pi_0(s) P_s(d_t | \theta)$. This distribution conditions on θ but not \mathbf{d} or \mathbf{h} . Its moments may be derived using the laws of iterated expectation $E(d_t) = E(E(d_t | h_t))$, $Var(d_t) = E(Var(d_t | h_t)) + Var(E(d_t | h_t))$, and the corresponding rule for covariances.

$$\begin{aligned} E(d_t | \theta) &= \boldsymbol{\pi}_0^\top \boldsymbol{\mu} \\ Var(d_t | \theta) &= \sum_s \boldsymbol{\pi}_0(s) \sum_s + \boldsymbol{\mu}^\top [\text{diag}(\boldsymbol{\pi}_0) - \boldsymbol{\pi}_0 \boldsymbol{\pi}_0^\top] \boldsymbol{\mu} \\ Cov(d_t, d_{t+\tau} | \theta) &= \boldsymbol{\mu}^\top [\text{diag}(\boldsymbol{\pi}_0) \mathbf{Q}^\tau - \boldsymbol{\pi}_0 \boldsymbol{\pi}_0^\top] \boldsymbol{\mu}. \end{aligned}$$

Predictive Distributions

The predictive distribution is $p(d_{t+\tau} | d_1^t, \theta) = \sum_s \pi_{\tau|t}(s) P_s(d_t | \theta)$. Because $\pi_{\tau|t} \rightarrow \pi_0$ as $\tau \rightarrow \infty$ the predictive distribution converges to the stationary distribution as $\tau \rightarrow \infty$. Notice that $\pi_{\tau|t}$ depends on all d_1^t , which implies that the relationship among successive data points in a hidden Markov model is not Markov. The moments of the predictive distribution are

$$E(d_{t+\tau} | d_1^t, \theta) = \boldsymbol{\pi}_{\tau|t}^\top \boldsymbol{\mu}$$

$$\text{Var}(d_{t+\tau} | d_1^t, \theta) = \sum_s \pi_{\tau|t}(s) \sum_s + \boldsymbol{\mu}^\top [\text{diag}(\boldsymbol{\pi}_{\tau|t}) - \boldsymbol{\pi}_{\tau|t} \boldsymbol{\pi}_{\tau|t}^\top] \boldsymbol{\mu}.$$

Posterior Covariance of \mathbf{h}

Let H_t be a column S-vector with 1 in position h_t and zeros elsewhere. Let $\mathbf{P}_t^* = (p_{trs}^*)$ where $p_{trs}^* = p'_{trs} / \sum_s p'_{trs} = p(h_t = s | h_{t-1} = r, \mathbf{d}, \theta)$. The covariance between H_t and $H_{t+\tau}$ given d_1^n , with $t < t + \tau < n$ may be written

$$\text{Cov}(H_t, H_{t+\tau} | \mathbf{d}, \theta) = \mathbf{P}_t^* \prod_{m=1}^{\tau-1} \mathbf{P}_{t+m}^* - \boldsymbol{\pi}_t \boldsymbol{\pi}_{t+\tau}^\top. \quad (26)$$

The product in Equation 26 is a matrix whose (r, s) element is $p(h_t = r, h_{t+\tau} = s | \mathbf{d}, \theta)$. The fact that this probability can be written as a product shows that \mathbf{h} remains a (nonstationary) Markov chain, even after conditioning on \mathbf{d} . One reason why an expression such as Equation 26 is valuable is that one occasionally wishes to avoid running the backward recursion all the way to the initial data point every time a new observation occurs. Instead, one could simply run the recursion until Equation 26 is sufficiently close to zero, beyond which point the updating will have little impact on earlier h_t 's.

AVAILABLE SOFTWARE

Most implementations of the likelihood, forward-backward, and Viterbi recursions are written by the end user in a high-level language like C or FORTRAN. MCMC inference can be carried out using BUGS (Spiegelhalter, Thomas, Best, & Gilks, 1996). Figures were prepared using R (Ihaka & Gentleman, 1996).

SUMMARY

This chapter provides a tutorial on HMMs and some of the techniques used to implement them. HMMs are useful both as flexible models for arbitrary time series data and as classification or clustering procedures when a subject's cluster membership varies over time. HMMs allow neighboring observations to communicate information about the hidden Markov chain. By pooling information from neighboring observations, HMMs can produce a more coherent reconstruction of the hidden states than is possible using finite mixture models or other clustering procedures that assume consecutive observations are independent.

The two missing data methods commonly used to estimate HMM parameters are MCMC data augmentation and the EM algorithm. MCMC is an ideal tool for the exploratory stage of a data mining problem. MCMC does not scale well to massive data sets (though chap. 5

of this volume suggests a possible improvement), but it can be used very effectively on moderately sized preliminary studies when the goal is to determine S , infer characteristics of θ , or determine the extent to which asymptotic arguments can be trusted in an implementation of a model on a massive data set. MCMC allows the user to easily compute full Bayes estimates of arbitrary functions of \mathbf{h} . It also frees inferences for θ and S from asymptotic assumptions. The EM algorithm is well suited to the screening stage of an analysis, when results are needed quickly, human intervention is difficult, and one would prefer to get “90% of the answer in 10% of the time.” Local computation methods for HMMs, the Viterbi and forward-backward algorithms, allow the models to screen very large data sets rapidly once a point estimate of θ has been obtained.

This chapter focusses on very simple applications of HMMs. More sophisticated applications exist. For example, Künsch, Geman, and Kehagias (1995) and Rydén and Titterington (1998) used hidden Markov random fields to model spatial data. Statistical inference for spatial lattice data is difficult because of a normalizing constant that cannot be directly calculated. Local computation techniques have not been as successful for spatial lattice data as they have for time series data. Because lattice data do not “separate” as easily as time series data, high dimensional distributions must still be computed. Romberg et al. (2001) combined wavelets with hidden Markov trees in an image analysis problem. Their hidden Markov process operates on a tree describing the need for a given level of wavelet detail. The tree structure means that local computation can be used effectively. A third variation occurs when HMMs are derived from other models, so that \mathbf{Q} is a parametric function of the parameters of the conditional distributions. One such example is Scott (1999), in which the Markov modulated Poisson process is expressed as an HMM to conduct a Bayesian inference for the process parameters.

Finally, the field of MCMC is expanding rapidly, and there are several MCMC algorithms applicable to HMMs that are not described here. Reversible jump MCMC (Green, 1995; Robert, Rydén, & Titterington, 2000) allows S to be one of the variables sampled by the MCMC algorithm. This removes the need to run several different MCMC samplers as done in the previous section on Fetal Lamb Movements. Stephens (2000a) provided another posterior sampling method, based on birth and death processes, which is competitive with reversible jump MCMC. Stephens (2000b) and Celeux, Hurn, and Robert (2000) offered alternative views of the label switching issue. These authors preferred to sample from posterior distributions without ordering constraints, which is difficult to do in cases such as Fig. 6.3(a), and obtain estimates of θ and \mathbf{h} using loss functions designed to handle multiple modes in the posterior distribution.

REFERENCES

- Albert, J. H., & Chib, S. (1993a). Bayes inference via Gibbs sampling of autoregressive time series subject to Markov mean and variance shifts. *Journal of Business and Economic Statistics*, 11, 1–15.
- Albert, J. H., & Chib, S. (1993b). Bayesian analysis of binary and polychotomous response data. *Journal of the American Statistical Association*, 88, 669–679.
- Andrieu, C., & Doucet A. (1999). Joint Bayesian model selection and estimation of noisy sinusoids via reversible jump MCMC. *IEEE Transactions on Signal Processing*, 47, 2667–2676.
- Baum, L. E., & Petrie, T. (1966). Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37, 1554–1563.
- Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41, 164–171.
- Besag, J., Green, P., Higdon, D., & Mengersen, K. (1995). Bayesian computation and stochastic systems (disc: p 41–66). *Statistical Science*, 10, 3–41.
- Casella, G., & George, E. I. (1992). Explaining the Gibbs sampler. *American Statistician*, 46, 167–174.

- Celeux, G., Hurn, M., & Robert, C. P. (2000). Computational and inferential difficulties with mixture prior distributions. *Journal of the American Statistical Association*, 95, 957–970.
- Chib, S. (1996). Calculating posterior distributions and modal estimates in Markov mixture models. *Journal of Econometrics*, 75, 79–97.
- Chib, S., & Greenberg, E. (1995). Understanding the Metropolis-Hastings algorithm. *American Statistician*, 49, 327–335.
- Churchill, G. A. (1989). Stochastic models for heterogeneous DNA sequences. *Bulletin of Mathematical Biology*, 51, 79–94.
- Cowell, R. G., Dawid, A. P., Lauritzen, S. L., & Spiegelhalter, D. J. (1999). *Probabilistic networks and expert systems*. New York: Springer.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (C/R: p 22–37). *Journal of the Royal Statistical Society, Series B, Methodological*, 39, 1–22.
- Fredkin, D. R., & Rice, J. A. (1992). Bayesian restoration of single-channel patch clamp recordings. *Biometrics*, 48, 427–448.
- Gelfand, A. E., & Smith, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85, 398–409.
- Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. London: Chapman & Hall.
- Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741.
- Gilks, W. R., Richardson, S., & Spiegelhalter, D. J. (1996). *Markov chain Monte Carlo in practice*. London: Chapman & Hall.
- Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika*, 82, 711–732.
- Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57, 357–384.
- Hamilton, J. D. (1990). Analysis of time series subject to changes in regime. *Journal of Econometrics*, 45, 39–70.
- Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57, 97–109.
- Ihaka, R., & Gentleman, R. (1996). R: A language for data analysis and graphics. *Journal of Computational and Graphical Statistics*, 5, 299–314.
- Juang, B. H., & Rabiner, L. R. (1991). Hidden Markov models for speech recognition. *Technometrics*, 33, 251–272.
- Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82, 35–45.
- Künsch, H., Geman, S., & Kehagias, A. (1995). Hidden Markov random fields. *Annals of Applied Probability*, 5, 577–602.
- Lauritzen, S. L. (1981). Time series analysis in 1880: A discussion of contributions made by T. N. Thiele. *International Statistical Review*, 49, 319–331.
- Lauritzen, S. L. (1996). *Graphical models*. Oxford: Oxford University Press.
- Lauritzen, S. L., & Spiegelhalter, D. J. (1988). Local computations with probabilities on graphical structures and their application to expert systems (c/r: p 194–224). *Journal of the Royal Statistical Society, Series B, Methodological*, 50, 157–194.
- Leroux, B. G., & Puterman, M. L. (1992). Maximum-penalized-likelihood estimation for independent and Markov-dependent mixture models. *Biometrics*, 48, 545–558.
- Liu, C., & Rubin, D. B. (1994). The ECME algorithm: A simple extension of EM and ECM with faster monotone convergence. *Biometrika*, 81, 633–648.
- Liu, C., & Rubin, D. B. (1995). ML estimation of the t distribution using EM and its extensions, ECM and ECME. *Statistica Sinica*, 5, 19–39.
- Liu, J. S. (2001). *Monte Carlo strategies in scientific computing*. New York: Springer.
- Liu, J. S., Neuwald, A. F., & Lawrence, C. E. (1999). Markovian structures in biological sequence alignments. *Journal of the American Statistical Association*, 94, 1–15.
- MacDonald, I. L., & Zucchini, W. (1997). *Hidden Markov and other models for discrete-valued time series*. London: Chapman & Hall.
- Meng, X.-L., & Rubin, D. B. (1993). Maximum likelihood estimation via the ECM algorithm: A general framework. *Biometrika*, 80, 267–278.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21, 1087–1092.
- Robert, C. P., Celeux, G., & Diebolt, J. (1993). Bayesian estimation of hidden Markov chains: A stochastic implementation. *Statistics & Probability Letters*, 16, 77–83.

- Robert, C. P., Rydén, T., & Titterington, D. M. (1999). Convergence controls for MCMC algorithms, with applications to hidden Markov chains. *Journal of Statistical Computation and Simulation*, 64, 327–355.
- Robert, C. P., Rydén, T., & Titterington, D. M. (2000). Bayesian inference in hidden Markov models through the reversible jump Markov chain Monte Carlo method. *Journal of the Royal Statistical Society, Series B, Methodological*, 62, 57–76.
- Robert, C. P., & Titterington, D. M. (1998). Reparameterization strategies for hidden Markov models and Bayesian approaches to maximum likelihood estimation. *Statistics and Computing*, 8, 145–158.
- Romberg, J. K., Choi, H., & Baraniuk, R. G. (2001). Bayesian tree-structured image modeling using wavelet-domain hidden Markov models. *IEEE Transactions on Image Processing*, 10, 1056–1068.
- Rydén, T., & Titterington, D. M. (1998). Computational Bayesian analysis of hidden Markov models. *Journal of Computational and Graphical Statistics*, 7, 194–211.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6, 461–464.
- Scott, S. L. (1999). Bayesian analysis of a two state Markov modulated Poisson process. *Journal of Computational and Graphical Statistics*, 8, 662–670.
- Scott, S. L. (2002a). Bayesian methods for hidden Markov models: Recursive computing in the 21st century. *Journal of the American Statistical Association*, 97, 337–351.
- Scott, S. L. (in press). A Bayesian paradigm for designing intrusion detection systems. *Computational Statistics and Data Analysis* special issue on network intrusion detection.
- Spiegelhalter, D. J., Thomas, A., Best, N. G., & Gilks, W. R. (1996). BUGS: Bayesian inference using Gibbs sampling, version 0.5, (version ii). <http://www.mrc-bsu.cam.ac.uk/bugs/>
- Stephens, M. (2000a). Bayesian analysis of mixtures with an unknown number of components—an alternative to reversible jump methods. *Annals of Statistics*, 28, 40–74.
- Stephens, M. (2000b). Dealing with label switching in mixture models. *Journal of the Royal Statistical Society, Series B, Methodological*, 62, 795–810.
- Tanner, M. A., & Wong, W. H. (1987). The calculation of posterior distributions by data augmentation (c/r: p 541–550). *Journal of the American Statistical Association*, 82, 528–540.
- Thiele, T. N. (1880). Om Anvendelse af mindste Kvadraters Methode i nogle Tilfælde, hvor en Komplikation af visse Slags uensartede tilfældige Fejlkilder giver Fejlene en ‘systematisk’ Karakter. *Det kongelige danske Videnskabernes Selskabs Skrifter, 5. Række, naturvidenskabelig og mathematisk Afdeling*, 12, 381–408.
- Titterington, D. M., Smith, A. F. M., & Makov, U. E. (1985). *Statistical analysis of finite mixture distributions*. New York: Wiley.
- van Dyk, D. A., & Meng, X.-L. (2001). The art of data augmentation (disc: p 51–111). *Journal of Computational and Graphical Statistics*, 10, 1–50.
- Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13, 260–269.
- West, M., & Harrison, J. (1997). *Bayesian forecasting and dynamic models*. New York: Springer.
- Zucchini, W., & Guttorm, P. (1991). A hidden Markov model for space-time precipitation. *Water Resources Research*, 27, 1917–1923.

7

Strategies and Methods for Prediction

Greg Ridgeway
The RAND Corporation

Introduction to the Prediction Problem	160
Guiding Examples	160
Prediction Model Components	161
Loss Functions—What We are Trying to Accomplish	162
Common Regression Loss Functions	162
Common Classification Loss Functions	163
Cox Loss Function for Survival Data	166
Linear Models	167
Linear Regression	168
Classification	169
Generalized Linear Model	172
Nonlinear Models	174
Nearest Neighbor and Kernel Methods	174
Tree Models	177
Smoothing, Basis Expansions, and Additive Models	179
Neural Networks	182
Support Vector Machines	183
Boosting	185
Availability of Software	188
Summary	189
References	190

INTRODUCTION TO THE PREDICTION PROBLEM

Many data mining problems depend on the construction of models, equations, or machines that can predict future outcomes. Although prediction is an important component of data mining, the abundance of methods such as linear models, neural networks, decision trees, and support vector machines can make a seemingly simple prediction problem rather confusing. Other chapters in this volume focus on particular methods. In this chapter we briefly assemble these ideas into a common framework within which we can begin to understand how all these methods relate to one another.

In many applications we are interested not only in having accurate predictions in the future but also in learning the relationship between the features of an observation and the outcomes. For example, we consider an example of predicting which students at age 12 are likely to drop out of high school before age 18. Certainly, we wish to have an accurate assessment of dropout risk, but we also wish to enumerate those factors that place certain students at greater risk. Understanding the mechanism relating the student features to the outcome helps formulate rules of thumb for identifying at-risk students and interventions targeting those students. Whether our goal is prediction alone or understanding the underlying mechanism or both, a well-structured and well-estimated prediction model is the first step in the process.

In this chapter we introduce the basics of prediction problems. We offer strategies on how to relate the choice of prediction method to applications and describe several of the frequently used prediction methods. This is far from a complete catalog of available tools and strategies but rather, like the rest of this handbook, it represents a starting place from which the reader could springboard into other, more technical developments of the methods. After reading this chapter the data miner will know the fundamental prediction concepts and be able to think critically and creatively about solving specific prediction problems. Throughout this chapter we use the following notation. Generically we have a data set containing N observations (y_i, \mathbf{x}_i) where $i = 1, \dots, N$. These observations come from some unknown process, generating mechanism, or probability distribution. The features, \mathbf{x}_i , often referred to as covariates, independent variables, or inputs, may be a vector containing a mix of continuous, ordinal, and nominal values. We wish to construct our prediction model, denoted as $f(\mathbf{x})$, so that it predicts the outcome, y_i , also called the response, the output, or the target, from the features.

Guiding Examples

This chapter focusses on three examples to help explain the concepts and techniques. The applications generally focus on public policy issues, but the same methods apply to problems spanning the scientific domains.

Regression: Cost of Stroke Rehabilitation

Regression problems (some engineering literature refers to these as estimation problems) involve the prediction of continuous outcomes. In 1997 the Balanced Budget Act mandated that the Centers for Medicare and Medicare Services (CMS) develop a prospective payment system for inpatient rehabilitation facilities. This system would determine how to allocate a \$4.3 billion budget to facilities that provide care for individuals covered under Medicare, the United States' federal healthcare system for the elderly and disabled. Part of the development involved building a cost model that predicts cost of rehabilitation from patient features. From CMS billing records we obtained each patient's age, reason for the stay (stroke, hip fracture, etc.),

and cost of care. From secondary sources we obtained functional ability scores that measured the patients' motor and cognitive abilities. The prediction problem involves developing a model so that for any future patient we can accurately predict cost of rehabilitation from the patient's features for accurate hospital reimbursement.

Classification: Detecting High School Dropouts

The National Education and Longitudinal Study of 1988 (NELS:88) is an extensive data source surveying the attitudes and behaviors of a nationally representative sample of American adolescents. NELS:88 first surveyed its respondents as eighth graders and has conducted three waves of follow-up measures in 1990, 1992, and 1994. Student, family, and school level measures are included in each wave of the survey. Offering multiple item indicators of student's goals, ability, past achievement, and involvement in school, NELS:88 also includes detailed data on parenting style, parent/child behavior and interactions, religion, race/ethnicity, and parents' occupation(s) and income, along with numerous other measures of family background. The strengths of this data set result from its large number of cases (over 15,000 students in this chapter's analyses), its comprehensiveness (measuring more than 6,000 variables), and its longitudinal design (allowing temporal as well as cross-sectional analyses). The examples utilize data from the first three survey waves, analyzing information from the baseline grade eight data to predict failure to complete high school.

Survival: Survival Time of PBC Patients

To show the breadth of loss functions available for consideration and the flexibility of prediction methods, we include a clinical trial example. In this example the problem is to estimate survival time of patients suffering from primary biliary cirrhosis of the liver (PBC). Although analyses using survival models predict time until death or time in remission for medical treatment studies, the models are applicable outside the domain of medicine. Applications also include time until failure of a machine or part, time until a customer churns by abandoning their current telecommunication provider for a competitor, and time between when a gun is purchased and when it is confiscated or recovered at a crime scene.

Prediction Model Components

Prediction methods may differ in three main ways: the *loss function* or performance measure that they seek to optimize, the *structural form* of the model, and the manner of obtaining model *parameter estimates* from training data. When considering a new or unfamiliar method, understanding these three basic components can go a long way toward realizing a method's limitations and advantages. Several methods may have the same structural form but differ on performance measures or scalability due to differences in how we estimate or learn the model from data. We see that three established classification methods, naive Bayes, logistic regression, and linear discriminant analysis, all have exactly the same structural form but differ on the loss function and parameter estimation method. In addition, the chapter on tree models in this handbook discusses models that all share the tree structure but may have different methods of forming splits and estimating the number of terminal nodes from the available data.

The next section discusses the most popular loss functions in use in statistics, machine learning, and data mining practice. Following that we give a concise catalog of some of the structural forms used in practice. Even after selecting a loss function and a structural form for

our predictor, the main problem facing data miners today is getting those models fit to massive data sets. We comment on the scalability issue as it arises, but it continues to be an active area of research and progress. Some accurate methods that were assumed to be not scalable to large data sets now have been tuned and optimized for practical use.

LOSS FUNCTIONS—WHAT WE ARE TRYING TO ACCOMPLISH

When developing a prediction model we usually have some performance measure that we want our model to optimize. The *loss function* is a function that takes as input a prediction model and produces a single number that indicates how well that prediction model performs. This section reviews some of the most commonly used loss functions. The notation $J(f)$ indicates the loss function J evaluated for prediction model f .

Common Regression Loss Functions

For regression problems the most widely used loss function is squared prediction error, which is the expected squared difference between the true value and the predicted value

$$J(f) = E_{y,x}(y - f(\mathbf{x}))^2 \quad (1)$$

where the $E_{y,x}$ represents the expectation operator that averages over all (y, \mathbf{x}) pairs drawn from some common distribution. By minimizing Equation 1 we assure ourselves that, on average, new predictions will not be too far from the true outcome. The properties of expectation indicate that the $f(\mathbf{x})$ that minimizes Equation 1 is $f(\mathbf{x}) = E(y | \mathbf{x})$, the average outcome at each value of \mathbf{x} . The probability distribution that generates the (y, \mathbf{x}) pairs is unknown, and so we cannot compute Equation 1 directly. Instead, we rely on a sample based estimate

$$\hat{J}(f) = \frac{1}{N} \sum_{i=1}^N (y_i - f(\mathbf{x}_i))^2. \quad (2)$$

Remember that it is almost *always* Equation 1 that we really want to minimize but resort to Equation 2 to guide us to a solution. There are many $f(\mathbf{x})$ that can make Equation 2 arbitrarily small but usually only one that minimizes Equation 1. The model fitting process will find an f that minimizes Equation 2 subject to some constraints on its structural form. An unbiased evaluation of the performance of a particular choice for f requires a separate test data set or some form of cross validation. The chapter on performance analysis and evaluation in this volume describes this process in more detail.

Although squared error loss is the dominant loss function in most applied regression work, decades of work on robustness have demonstrated that squared error is highly sensitive to outliers, unusually large outcomes potentially from data contamination and spurious measurements. Absolute prediction error

$$J(f) = E_{y,x} |y - f(\mathbf{x})| \quad (3)$$

has its minimum when $f(\mathbf{x}) = \text{median}(y | \mathbf{x})$, the median outcome at each value of \mathbf{x} . For data mining applications prone to contamination the absolute prediction error may be preferable.

Other regression loss functions use variations on the above theme. For example, Huber (1964) proposed a loss function that behaves like squared-error near 0 and like absolute error when $y - f(\mathbf{x})$ exceeds some cutoff, providing some protection against extreme y values. Support vector machine regression methods commonly use a loss function that is zero when $y - f(\mathbf{x})$ is less than some cutoff, and then behaves like absolute error for deviations greater than the cutoff. At this point, simply note that there is considerable flexibility in terms of specifying what it means to have a prediction be “close” to the true value and that the different choices result in different prediction models. The implications follow shortly in some examples.

Common Classification Loss Functions

Although regression problems focus on predicting continuous outcomes, classification methods attempt to label observations as one of k categories. The most common loss functions used in classification problems include misclassification rate, expected cost, and log-likelihood. Misclassification rates are generally the primary measure on which methods are compared. In fact, it is generally what problem solvers are aiming to minimize when considering a particular classification problem. In almost all problems, however, a false positive has a different cost than a false negative. High school dropouts are thought to cost 20 times more than graduates in terms of societal costs. The false negatives in this case are the more expensive mistake. Let c_0 be the cost of misclassifying a true 0 case and c_1 be the cost of misclassifying a 1 case. For a two class classification problem our classifier, $f(\mathbf{x})$, predicts values 0 or 1. Then the expected cost is

$$\begin{aligned} J(f) &= E_{\mathbf{x}} E_{y|\mathbf{x}} c_0 I(y = 0)f(\mathbf{x}) + c_1 I(y = 1)(1 - f(\mathbf{x})) \\ &= E_{\mathbf{x}} c_0(1 - P(y = 1 | \mathbf{x}))f(\mathbf{x}) + c_1 P(y = 1 | \mathbf{x})(1 - f(\mathbf{x})) \end{aligned} \quad (4)$$

where $I(\cdot)$ is the indicator function that is 1 if the expression is true and 0 otherwise. Minimizing the expression pointwise at each \mathbf{x} we see that ideally $f(\mathbf{x})$ should equal 0 whenever $c_0(1 - P(y = 1 | \mathbf{x})) > c_1P(y = 1 | \mathbf{x})$. Equivalently, the best classifier is

$$f(x) = \begin{cases} 0 & \text{if } P(y = 1 | \mathbf{x}) < c_0/(c_0 + c_1) \\ 1 & \text{otherwise} \end{cases} \quad (5)$$

We actually do not need an estimate of $P(y = 1 | \mathbf{x})$ to obtain a good decision rule. It is sufficient to have a method that determines on which side of $c_0/(c_0 + c_1)$ the probability would fall. In fact, some excellent classifiers produce poor estimates of $P(y = 1 | \mathbf{x})$. Note that if $c_0 = 1$ and $c_1 = 20$, as in the high school dropout problem, then any student with a dropout probability exceeding 0.047 needs special attention.

Although many classification methods advertise their ability to obtain low misclassification costs, many classification procedures minimize cost indirectly. Classification trees are among the few procedures that directly aim to minimize cost. Many other procedures aim for good estimates of $P(y = 1 | \mathbf{x})$, which Equation 5 as previously mentioned shows is sufficient but not necessary for developing a decision rule for any choice for c_0 and c_1 . In some settings a probabilistic prediction itself is necessary to have a complete risk assessment.

The likelihood principle, studied in detail in Berger and Wolpert (1984), implies that any inference about parameters of interest should depend on the data only through the likelihood function, the probability that the model would generate the observed data. So although Equation 4 is the loss function for minimizing misclassification cost, when seeking good

probability estimates for the two-class classification problem we should look to the *Bernoulli likelihood*,

$$L(p) = \prod_{i=1}^N p(\mathbf{x}_i)^{y_i} (1 - p(\mathbf{x}_i))^{1-y_i} \quad (6)$$

where $p(\mathbf{x}) = P(y = 1 | \mathbf{x})$ and is what we want to estimate and study. Although up to this point we have used $f(\mathbf{x})$ to denote the prediction model that we are trying to estimate, here we use $p(\mathbf{x})$ to remind ourselves that it is a probability and must be on the interval $[0, 1]$. Many statistical procedures are based on estimates of $p(\mathbf{x})$ that maximize the likelihood, intuitively the $p(\mathbf{x})$ that makes the observed data most likely. Before we discussed finding $f(\mathbf{x})$ to minimize a loss function, but here the goal is to find a $p(\mathbf{x})$ to maximize a likelihood. The log-likelihood is the more commonly used form of this loss function and, again, we are not simply interested in maximizing it for our finite sample but in expectation over a new observation drawn from the same distribution that generated our data set.

$$\begin{aligned} J(p) &= E_{y,\mathbf{x}} \log L(p) \\ &= E_{y,\mathbf{x}} [y \log p(\mathbf{x}) + (1 - y) \log(1 - p(\mathbf{x}))] \end{aligned} \quad (7)$$

We see in a later section that logistic regression procedures are based on *maximum likelihood* estimates of $p(\mathbf{x})$.

The Bernoulli log-likelihood naturally extends to multiclass classification via the *multinomial log-likelihood*,

$$E_{y,\mathbf{x}} L(p_1, p_2, \dots, p_K) = E_{y,\mathbf{x}} I(y_i = k) \log p_k(\mathbf{x}_i), \quad (8)$$

where $p_k(\mathbf{x}) = P(y = k | \mathbf{x})$ and the $p_k(\mathbf{x})$ sum to 1. With this loss function we will seek k functions, each of which estimates one of the class probabilities. Also, *ordinal regression* methods are available when the K classes are ordered as in preference rating scales.

Using the Bernoulli log-likelihood as a loss function focusses on obtaining good probability estimates, but it is unclear what “good” means in this context. Meteorologists especially have been studying accurate probability assessments, decomposing prediction accuracy into discrimination and calibration components. Discrimination, which generally gets the most attention, is the ability to separate the classes, whereas calibration is the ability to assign meaningful probabilities to events. When we are dealing with a binary outcome (y is either 0 or 1) the Brier score (Brier, 1950) shown in Equation 9 offers an interesting assessment of probabilistic assignments.

$$J(p) = E_{y,\mathbf{x}} (y - p(\mathbf{x}))^2 \quad (9)$$

The Brier score is small when our probability estimate is small and $y = 0$ and when our probability estimate is near 1 when in fact $y = 1$. Clearly, the Brier score is minimized when we have perfect forecasts. Yates (1982) discusses the *Murphy decomposition* of the empirical Brier score exposing different aspects of prediction quality.

$$\begin{aligned} \frac{1}{N} \sum_{i=1}^N (y_i - p(\mathbf{x}_i))^2 &= \bar{y}(1 - \bar{y}) - \frac{1}{N} \sum_{k=1}^K n_k (\bar{y}_k - \bar{y})^2 + \frac{1}{N} \sum_{k=1}^K n_k (p_k - \bar{y}_k)^2 \\ &= \text{uncontrollable variation} + \text{resolution} + \text{calibration} \end{aligned} \quad (10)$$

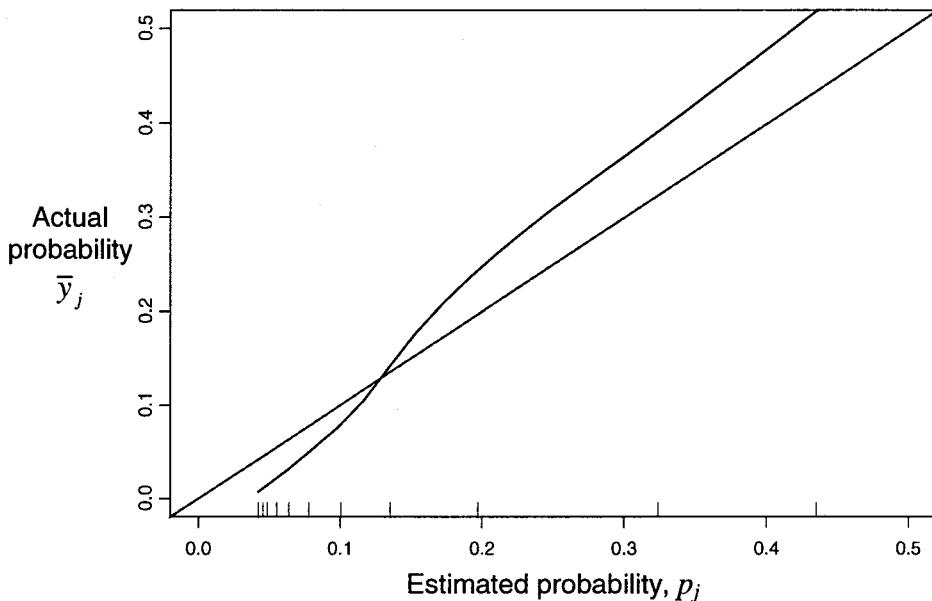


FIG. 7.1. Calibration plot for the high school dropout example.

The first term is the variance of the outcome. It is only small when the y_i 's are all 0 or all 1, something over which we have no control. The variance term also represents the best we can do. In the second term the n_k represents the number of observations that are given the probabilistic prediction p_k , and \bar{y}_k is the average of the outcomes given the prediction p_k . This *resolution* term is large (very negative) when we are able to discriminate the 0 outcomes from the 1s. In that situation the average outcome given prediction p_k is far from the baseline rate, near 0 or 1. The last term measures *calibration*, the ability to assign meaningful probabilities to the outcomes.

To understand calibration let us again turn to the high school dropout problem. Assume we have a probabilistic prediction model. A new set of students arrives on which we assess the dropout risk. If the model is well calibrated, then among the collection of students to which the model assigned a dropout probability of, for example, 0.3, 30% would actually dropout. Figure 7.1 shows a smoothed calibration plot for a boosted classification model (section 4.6) for the high school dropout example. The 45° line is the perfectly calibrated predictor. The tick marks along the bottom of the figure mark the deciles of the estimated probabilities, most of which are below 0.1. Note that for all the students with an estimated dropout probability around 0.3, 36% of them actually dropped out.

We learn from the Murphy decomposition that when measuring classification performance with the Brier score both discrimination and calibration are important. In practice, we will have to decide which attributes are most important for the problem at hand. Good classification accuracy alone at times may be insufficient.

Classification is conceptually quite simple. We want to put the right labels on observations. But that conceptual simplicity is not easily translated into a loss function without additional information on the goals we seek. Choosing to minimize cost or opting to go for the best probability estimates leads to different choices for loss functions and, therefore, prediction methods. One of the early strategic steps in a classification problem is defining the goals, and usually that translates into a natural choice for the loss function.

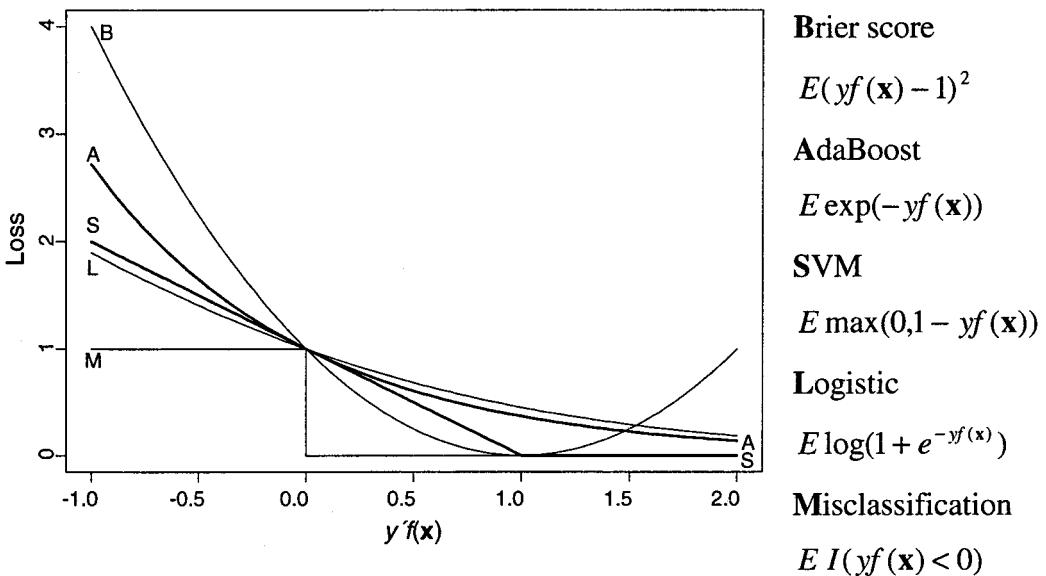


FIG. 7.2. Classification loss functions.

So far we have presented three candidates for the loss function, but others have been proposed. Figure 7.2 shows the similarities and differences among the various classification loss functions. First consider relabeling the two classes as -1 and 1 , $y' = \frac{1}{2}(y + 1)$, which simplifies the comparison. Therefore, when $y'f(\mathbf{x})$ is positive the observation is correctly classified. The curve labeled M in Fig. 7.2 reflects this. Note that all of the loss functions are bounds for the misclassification loss function. Also shown in bolder lines are the loss functions for support vector machines and AdaBoost (which are discussed later in this chapter). The different loss functions determine how much we penalize our predictor for certain mistakes. The Brier score strongly penalizes mistakes but is the only loss function that also penalizes overconfidence in predictions, indicated by the upswing in the Brier loss for $y'f(\mathbf{x}) > 1$.

Cox Loss Function for Survival Data

The last specific loss function discussed here is often used for survival data. We include it here partially for those interested in lifetime data but also to demonstrate creative developments of loss functions for specific problems. Survival analysis continues to be a thriving area in biostatistics, and this section focuses on a small part, the Cox model for *proportional hazards regression*. For each observation we observe the subject features, \mathbf{x}_i , the time the subject was last observed, t_i , and a 0/1 indicator of whether the subject failed at time t_i , δ_i . If we know the exact failure time for all observations, that is $\delta_i = 1$ for all i , then we can turn to some of the standard loss functions discussed earlier. When we have many observations that have not failed, such as customers who have not switched their long distance carrier yet, we would lose information if we did not include these observations in the estimation process. To make this more concrete, Customer A in South Dakota with no previously filed complaints has used long distance Company X for 100 days and remains a customer. The observation for this customer would be

$$(\mathbf{x} = \{\text{State} = \text{SD}, \text{Complaint} = 0\}, \delta = 0, t = 100).$$

Because Customer A remains a customer at Day 100, they have not yet “failed,” and so $\delta = 0$. Customer B, on the other hand, is from North Dakota, has filed three complaints, and on Day 65 of service switches long distance carriers. The observation record for this customer would be

$$(\mathbf{x} = \{\text{State} = \text{ND}, \text{Complaint} = 3\}, \delta = 1, t = 65).$$

So from a data set of observations $(\mathbf{x}_i, \delta_i, t_i)$, for i in $1, \dots, N$, we want to construct a model to predict actual failure time t_i from \mathbf{x}_i .

The proportional hazards model assumes that the *hazard function*, the instantaneous probability that a subject with feature \mathbf{x} fails in the next small interval of time given survival up to time t , is

$$h(t, \mathbf{x}) = \lambda(t) \exp(f(\mathbf{x})) \quad (11)$$

where $\lambda(t)$ is the baseline hazard. This model assumes that the relevance of a particular feature does not change over time. If we can estimate $f(\mathbf{x})$, then we can determine those indicators that accelerate the rate of failure and those observations that have an accelerated risk of failure. Given that an observation in the data set with N observations failed at time t' , the probability that it was observation i is

$$\frac{I(t_i \geq t') \lambda(t') \exp(f(\mathbf{x}_i))}{\sum_{j=1}^N I(t_j \geq t') \lambda(t') \exp(f(\mathbf{x}_j))}. \quad (12)$$

Conveniently, the baseline hazard, $\lambda(t)$, cancels in Equation 12. We can then write down the likelihood that the first observed failure would have failed at its failure time and the second observed failure would have failed at its failure time and so on.

$$\prod_{i=1}^N \left[\frac{\exp(f(\mathbf{x}_i))}{\sum_{j=1}^N I(t_j \geq t_i) \exp(f(\mathbf{x}_j))} \right]^{\delta_i} \quad (13)$$

If we can find an $f(\mathbf{x})$ that makes the *Cox partial likelihood* (Equation 13) large, this indicates that $f(\mathbf{x})$ can put the N observations approximately in order of when they will fail. Subsequently we can use this $f(\mathbf{x})$ to determine which subjects that have not failed yet are most at risk of failure in the near future. Even though we do not know the exact form of the baseline hazard $\lambda(t)$, we can still estimate $f(\mathbf{x})$ using only the order of failure times, rather than the actual failure times themselves. We can then estimate $\lambda(t)$ although $f(\mathbf{x})$ is sufficient for identifying observations that are prone to shorter times to failure.

Now that we have a handful of loss functions for measuring predictive performance, the next section begins a discussion of finding the ideal f to optimize our selected loss function.

LINEAR MODELS

In this section we begin looking at prediction models that have a linear structure. Although the structure at first may seem naive, the development is an important one. First, these methods have a long history and are still in widespread use. Second, although apparently simplistic these methods can perform particularly well when the data is sparse. And third, several of the most modern methods build on the basic linear structure and share some of the model building tools.

Linear Regression

The squared error loss function was introduced earlier. Now let us restrict the model to have the form

$$f(\mathbf{x}) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_d x_d = \boldsymbol{\beta}' \mathbf{x} \quad (14)$$

where $d + 1$ is the dimension of the feature vector \mathbf{x} , and the first element of \mathbf{x} is 1. Rather than having a completely unspecified $f(\mathbf{x})$ we now only have $d + 1$ model parameters to estimate. The coefficient, β_j , represents a difference between two subjects that have the same feature vector but differ by 1 unit on feature x_j . When fitting linear regression models with a categorical feature with k levels, we can create $k - 1$ 0/1 *dummy variables*. If variable x_1 has $k = 3$ levels, then we can fit the model

$$f(\mathbf{x}) = \beta_0 + \beta_{11} I(x_1 = 1) + \beta_{12} I(x_1 = 2) + \beta_2 x_2 + \cdots + \beta_d x_d. \quad (15)$$

Now β_{11} represents a difference between Category 1 subjects and Category 3 subjects.

To fit the model we simply select the vector of coefficients to minimize the empirical squared error loss

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \frac{1}{N} \sum_{i=1}^N (y_i - \boldsymbol{\beta}' \mathbf{x}_i)^2 = \arg \min_{\boldsymbol{\beta}} \frac{1}{N} (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})' (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \quad (16)$$

where \mathbf{X} is the $N \times d + 1$ feature matrix, and \mathbf{y} is a column vector with the N outcomes. The solution to Equation 16 is

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}' \mathbf{X})^{-1} \mathbf{X}' \mathbf{y} \quad (17)$$

solvable in a single pass through the data set. Figure 7.3(a) shows the result of a linear least squares fit to stroke data from 1996. The jagged curve in the plot is the average cost at each motor score. In this simple example we have enough data at most values of the motor score to get reasonable estimates. The line generally runs through the pointwise averages and shows

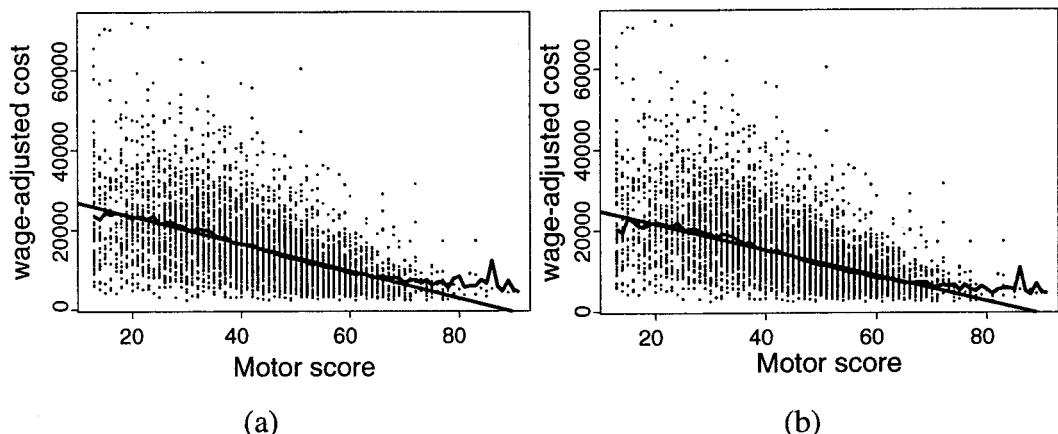


FIG. 7.3. Predicting cost of stroke rehabilitation from motor ability score. Least squares linear fit and the pointwise average fit (a) and the least absolute deviations linear fit and the pointwise median fit (b).

considerably less instability for the sparsely sampled stroke patients with high motor scores. The β that minimizes squared error loss is $(30251.0, -342.6)$ and the resulting average squared error is 5.88×10^7 .

Although this choice of β minimizes squared error for the 1996 data, we would hope that such a model holds up over time. If we use the model fit to 1996 data to predict costs for the 1997, 1998, and 1999 data, the average squared error for those years is 5.94×10^7 , 5.90×10^7 , and 6.19×10^7 . In general the error on the training data will be an overoptimistic estimate of the performance on future observations.

Medical costs are particularly prone to extreme observations. Modeling the median rather than the mean offers a more stable model. The minimizer of absolute error does not have a convenient closed form expression; however, moderately efficient algorithms do exist (Bloomfield & Steiger, 1983). Figure 7.3(b) displays the linear model that minimizes absolute prediction error along with pointwise median estimates that are well approximated by the line. The β that minimizes average absolute loss is $(28014.0, -320.2)$ producing a linear fit with a gentler slope than one obtained using the squared error loss.

Classification

In this section we focus on procedures for 0/1 classification problems but will make brief mention of the multiple class case at the end.

Linear Logistic Regression

Statistical methods for binary classification usually are designed to produce good estimates of $p(\mathbf{x}) = P(Y = 1 | \mathbf{x})$ using the Bernoulli likelihood shown in Equation 6. Linear logistic regression, one of the earliest techniques, assumes a particular parametric form for $p(\mathbf{x})$,

$$p(\mathbf{x}) = \frac{1}{1 + \exp(-f(\mathbf{x}))}, \quad \text{where } f(\mathbf{x}) = \beta' \mathbf{x}. \quad (18)$$

As in linear regression, the prediction depends on the feature vector only through a linear combination of the components of \mathbf{x} . Again, this greatly reduces the complexity of the model by reducing the problem to estimating only the $d + 1$ β 's. The logistic function transforms the linear combination from the whole real line to the $[0, 1]$ interval. Rewriting Equation 18, we can see that this model assumes that the log-odds are a linear function of \mathbf{x} .

$$\log \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} = \beta' \mathbf{x}. \quad (19)$$

Inserting Equation 18 into the Bernoulli likelihood from Equation 6 and taking the logarithm we obtain

$$L(\beta) = \sum_{i=1}^N y_i \beta' \mathbf{x}_i - \log(1 + \exp(\beta' \mathbf{x}_i)). \quad (20)$$

To fit the model by maximum likelihood we select the β that maximizes Equation 20. No closed form solution exists, but we can utilize a simple Newton-Raphson procedure to numerically maximize the likelihood. The first and second derivatives of Equation 20 are

$$\frac{\partial L(\beta)}{\partial \beta} = \sum_{i=1}^N \mathbf{x}_i \left(y_i - \frac{1}{1 + \exp(\beta' \mathbf{x}_i)} \right) = \mathbf{X}'(\mathbf{y} - \mathbf{p}) \quad (21)$$

$$\frac{\partial^2 L(\beta)}{\partial \beta^2} = - \sum_{i=1}^N \mathbf{x}_i \mathbf{x}'_i \frac{1}{1 + \exp(\beta' \mathbf{x}_i)} \left(1 - \frac{1}{1 + \exp(\beta' \mathbf{x}_i)} \right) = -\mathbf{X}' \mathbf{W} \mathbf{X} \quad (22)$$

where \mathbf{p} is the vector of predicted probabilities, and \mathbf{W} is a diagonal matrix with diagonal equal to $p(\mathbf{x}_i)(1 - p(\mathbf{x}_i))$. After selecting an initial starting value for β , the Newton-Raphson update is

$$\begin{aligned}\beta &\leftarrow \beta - (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'(\mathbf{y} - \mathbf{p}) \\ &= (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}(\mathbf{X}\beta - \mathbf{W}^{-1}(\mathbf{y} - \mathbf{p})) \\ &= (\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}'\mathbf{W}\mathbf{z}.\end{aligned}\quad (23)$$

Note the similarity between Equation 23 and the solution to the linear regression model in Equation 17. Rather than simply having the \mathbf{y} as in Equation 17, we have a *working response* \mathbf{z} and we have the weight matrix \mathbf{W} . The Newton update, therefore, is a weighted linear regression with observation i having weight $p_i(1 - p_i)$, where the features \mathbf{x}_i predict the working response z_i . This algorithm is known as *iteratively reweighted least squares* (IRLS). In practice, convergence usually occurs after three to four iterations of IRLS. We can also begin to think about improvements including nonlinear predictors in Equation 19 or using nonlinear regression inside the IRLS algorithm. We visit these issues shortly. Note also that the weight is largest when p_i is close to $1/2$, close to the equal cost decision boundary, an issue that will arise again in our discussion of boosting.

Figure 7.4 shows a linear logistic regression model fit to the high school dropout data set with two predictors. As expected, the model structure forces the contours to be parallel. From

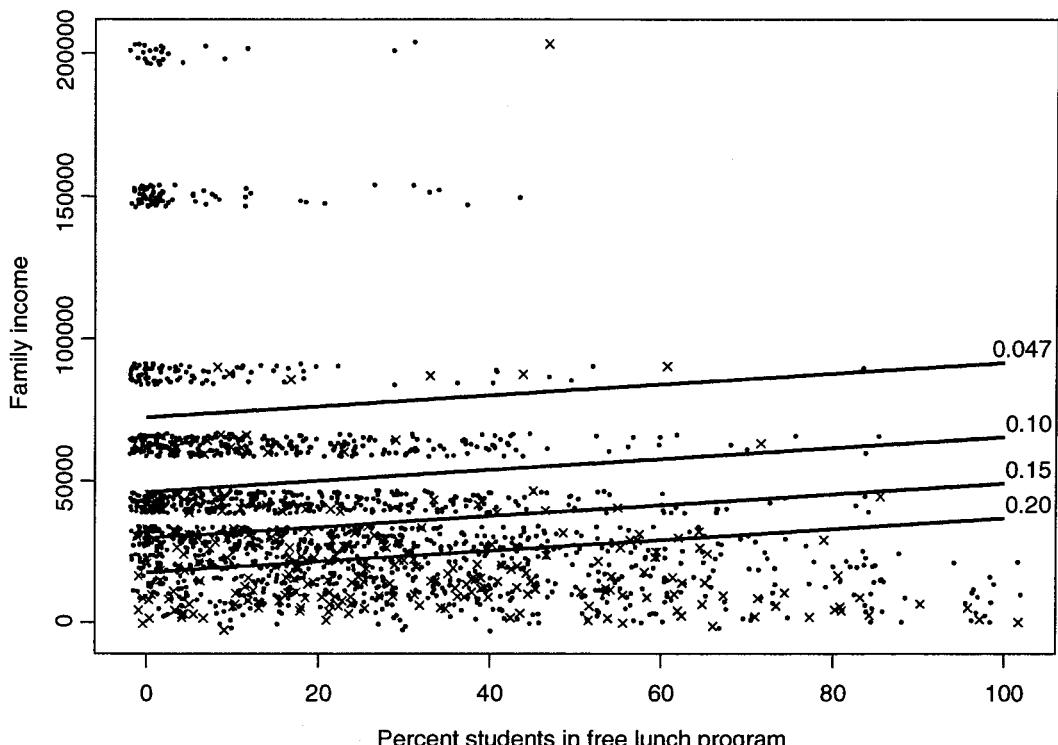


FIG. 7.4. Probability of dropout predicted from family income and percentage of students at school in a free lunch program. The four lines indicate contours of equal probability of dropping out. The x's mark the dropouts and the dots mark the graduates.

the predicted probabilities we can apply our decision rule and estimate the expected cost per student. As before, assuming that failing to detect a dropout is 20 times more costly than failing to identify a graduate, the decision boundary is $p = 0.047$, marked by the upper contour in Fig. 7.4. All students above that line are classified as graduates, and those below that line are classified as dropouts. Clearly this model puts the majority of cases as dropouts. Had we assumed everyone would graduate, our expected cost would be 3.3. Assuming everyone is a dropout would cost us 0.83. But using the information in the two predictors, our expected costs are reduced slightly to 0.81. Although this is a modest decrease, perhaps additional predictors may further reduce misclassification costs. On the other hand, perhaps the rigidity of the linear logistic model will prevent us from identifying the characteristics of the dropouts.

The Naive Bayes Classifier

The naive Bayes classifier (Hand & Yu, 2001) is, in spite of its name, a very powerful classifier. It is simple to program, fit to data, and is easy to interpret.

If Y is the class variable that we would like to predict using predictors \mathbf{x} , then the naive Bayes classifier has the form

$$P(Y = y | \mathbf{x}) \propto P(Y = y)P(\mathbf{x} | Y = y) \quad (24)$$

$$= P(Y = y) \prod_{j=1}^d P(x_j | Y = y) \quad (25)$$

Equation 24 is a direct application of Bayes theorem. To transition to Equation 25 requires a naive assumption, that given the true class the features are independent. For example, given that a particular child dropped out, knowing that he or she had poor grades gives no additional information about their socioeconomic status. This implies that dropout status is sufficient information to estimate any of the child's features. Chapter 5 in this volume diagrams the naive Bayes classifier as a graphical model. Refer to the figure there to see this graphically.

The naive Bayes assumption gives the naive Bayes classifier the same structural form as linear logistic regression described in the previous section. Letting $p(\mathbf{x}) = P(Y = 1 | \mathbf{x})$, we see that the naive Bayes classifier is additive on the log odds, or logit, scale.

$$\begin{aligned} \log \frac{p(\mathbf{x})}{1 - p(\mathbf{x})} &= \log \frac{P(Y = 1)}{P(Y = 0)} + \sum_{j=1}^d \log \frac{P(x_j | Y = 1)}{P(x_j | Y = 0)} \\ &= w_0 + w_1(x_1) + w_2(x_2) + \cdots + w_d(x_d) \end{aligned} \quad (26)$$

Note that the structural form is similar to the form used for linear logistic regression in Equation 19. If all of the x_j are categorical, then the functional forms are exactly the same. Standard practice has been to discretize the continuous x_j 's, creating histogram estimates of $P(x_j | Y)$. This creates an additive model where the w_j components are step functions rather than linear functions of x_j .

Logistic regression assumes that $P(Y = 1 | \mathbf{x}) = P(Y = 0 | \mathbf{x})\exp(\beta' \mathbf{x})$, where the $P(Y = 0 | \mathbf{x})$ can have an arbitrary form. In some respects naive Bayes has a more restrictive assumption, specifying a distributional form for both classes. The estimation procedure separately estimates the components for $y = 0$ and $y = 1$ before combining into a single classifier. This assumption has some advantages for large data set applications. The assumption allows us to estimate this model in a single pass through the data set, and missing x_j values can be ignored. In the next section we look at a third linear classifier with yet another set of associated assumptions.

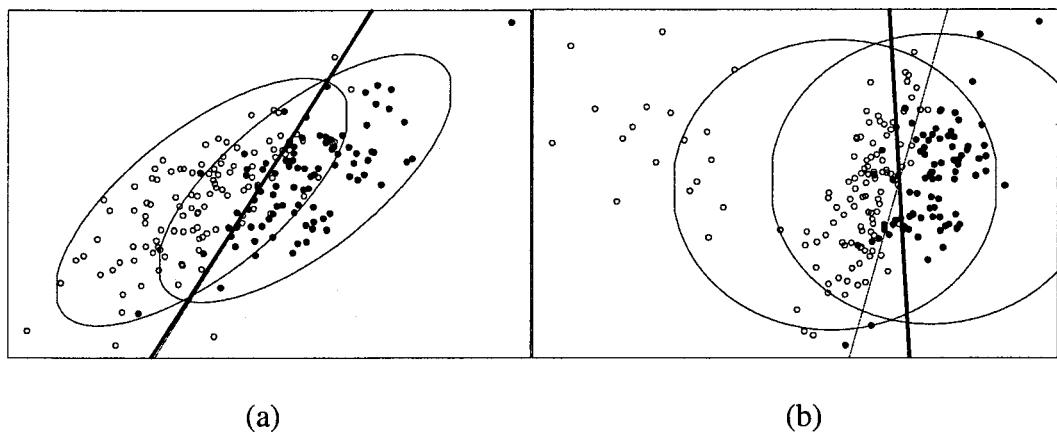


FIG. 7.5. LDA decision boundaries, where \mathbf{x} is multivariate normal with each class (a) and when one class is contaminated with outliers (b). The heavy line is the LDA boundary, and the lighter line is the logistic regression boundary.

Linear Discriminant Analysis

Fisher (1936) proposed linear discriminant analysis (LDA) for classification problems. As with the naive Bayes classifier, LDA uses Bayes theorem to reverse the conditional probability (Equation 24) and makes an assumption about the distribution of the features within each class. Rather than assume conditional independence as in Equation 25, LDA assumes that $P(\mathbf{x} | Y = y)$ is a multivariate normal density where all the classes share a common covariance matrix. With these two assumptions the log-odds again has a form that is linear in \mathbf{x} . Unlike logistic regression and naive Bayes, LDA is very sensitive to outliers in \mathbf{x} and in general performs quite poorly. Figure 7.5 shows an LDA fit to simulated data. When the features are truly multivariate normal as in Fig. 7.5(a), both LDA and logistic regression produce approximately the same decision boundary. When one class is contaminated with outliers the decision boundary can move substantially and perform worse than logistic regression Fig. 7.5(b).

In the last 70 years discriminant analysis, like other methods, has undergone considerable modernization. Although the simple form described here tends to perform poorly, LDA's descendants can perform quite well. They generally relax the normal distribution assumption and allow the classes to have separate distributions, each of which we can model with a more flexible density estimator. See Hastie, Tibshirani, and Buja (1994) and Ridgeway (2002) for examples of these extensions.

Generalized Linear Model

In this section we briefly mention the connection to a large class of regression models that one can understand as variations on the linear logistic regression development. An earlier section showed a particular transformation of $f(\mathbf{x})$ onto the probability scale using the logistic transform (Equation 18) and then used the Bernoulli likelihood to determine the best fitting linear model. Statisticians often prefer the logistic transform as the *link function*, the function relating the probability to $f(\mathbf{x})$, mostly because it allows interpretation of model coefficients as log odds-ratios. Economists, on the other hand, have tended to use *probit regression*, differing from logistic regression by its use of the inverse Gaussian cumulative distribution as the link

function, $p(\mathbf{x}) = \Phi^{-1}(f(\mathbf{x}))$. There is considerable flexibility in choosing the link function, although the logit and probit are by far the most common. Besides the choice of link function for logistic regression, we can vary the likelihood itself to capture an enormous class of useful prediction problems. The Bernoulli distribution is particular to 0/1 outcomes. The *Poisson distribution* is often used to model outcomes involving counts (items purchased, cigarettes smoked, etc.) and has the form

$$P(Y = y | \mathbf{x}) = \frac{\lambda(\mathbf{x})^y \exp(-\lambda(\mathbf{x}))}{y!} \quad (27)$$

where $\lambda(\mathbf{x})$ represents the expected count for an observation with features \mathbf{x} . Often an observation has a time measurement in addition, such as time as a customer or time since released from treatment. In such instances researchers commonly parameterize the Poisson model as

$$P(Y = y | \mathbf{x}, t) = \frac{(\lambda(\mathbf{x})t)^y \exp(-\lambda(\mathbf{x})t)}{y!} \quad (28)$$

so that $\lambda(\mathbf{x})$ represents a rate of occurrences. Using the log link function, $\log \lambda(\mathbf{x}) = \beta' \mathbf{x}$, we are assured that the rate will always be positive. For this reason Poisson regression often is referred to as log-linear modeling. If we have N independent observations (y_i, \mathbf{x}_i, t_i) , then we can write down the log-likelihood as

$$\log L(\beta) = \log \prod_{i=1}^N \frac{(\lambda(\mathbf{x}_i)t_i)^{y_i} \exp(-\lambda(\mathbf{x}_i)t_i)}{y_i!} \quad (29)$$

$$= \sum_{i=1}^N y_i \beta' \mathbf{x}_i - t_i \exp(\beta' \mathbf{x}_i) + y_i \log t_i - \log y_i! \quad (30)$$

Except for the last two terms that do not involve β , Equation 30 closely resembles Equation 20. Inside the sum in both cases we have the outcome, y , times the linear predictor minus a term that has the expected value of y as its derivative. Fitting the Poisson model also involves a few iterations of a simple IRLS algorithm.

The Bernoulli and Poisson prediction methods are special cases of the class of *generalized linear models* (GLM). After selecting the variables for the linear predictor, the distribution of the response, and a link function, the GLM framework packages together a likelihood based loss function and an IRLS algorithm for fitting the models. Even the linear least squares model under the earlier section on Linear Regression falls into this framework with a Gaussian distribution for the response and an identity link function. Simply replace the Poisson distribution in Equation 29 with the Gaussian distribution. Then setting the derivative of the log-likelihood equal to 0 and solving for β produces exactly the least squares solution we saw earlier.

Other useful GLMs include *multinomial logistic regression* for multiclass classification, *Gamma regression* for skewed outcome distributions (like cost), and *negative binomial regression* for count data with extra-Poisson variation. McCullagh and Nelder (1989) provided a complete guide to the basics of the GLM. Greene (1999) also discussed these methods with respect to econometrics. Although the development of the GLM in this section is brief, this overview should give the impression that one has considerable flexibility in determining how to model the outcome variable. For continuous, binary, and count outcomes the GLM framework is one good starting place. The linear part is easily replaceable with any other functional form, as we see in the next section.

NONLINEAR MODELS

In spite of their computational simplicity, stability, and interpretability, linear models have an obvious potential weakness. The actual process may not be linear, and such an assumption introduces uncorrectable bias into the predictions. When data is sparse or the dimension of x is large, linear models often capture much of the information in the data as shown in Fig. 7.6(a). There the linear model seems to capture much of the information. Detecting nonlinear features requires more data with a low signal-to-noise ratio. Data mining applications inherently involve large data sets, and so the general trend is almost always to use nonlinear methods, implying that most data miners feel that their data is closer to the situation in Fig. 7.6(b). Although the same mechanism generated both data sets, the increase in data in Fig. 7.6(b) makes the linear model less appealing. Problems with a large number of features require caution. Such situations will more closely resemble Fig. 7.6(a). Nonlinear models run a much greater risk of being overfit to the training data set, and the chapter on performance analysis and evaluation in this handbook requires careful attention (see chap. 17).

This section explores a few of the popular nonlinear prediction methods. These methods generalize the previous discussion by allowing $f(\mathbf{x})$ to take on a more flexible form.

Nearest Neighbor and Kernel Methods

The k nearest neighbor (knn) prediction model simply stores the entire data set. As the name implies, to predict for a new observation the predictor finds the k observations in the training data with feature vectors close to the one for which we wish to predict the outcome. The prediction depends on the loss function and in general is

$$f(\mathbf{x}) = \arg \min_{\theta} J_{N(k, \mathbf{x})}(\theta) \quad (31)$$

where θ is a constant and $J_{N(k, \mathbf{x})}$ represents the loss function computed for only the k closest observations in a neighborhood near \mathbf{x} . For example, the knn predictor for squared error loss is

$$f(\mathbf{x}) = \arg \min_{\theta} \frac{1}{k} \sum_{i \in N(k, \mathbf{x})} (y_i - \theta)^2 = \frac{1}{k} \sum_{i \in N(k, \mathbf{x})} y_i \quad (32)$$

the average of the outcomes for the k observations nearest to \mathbf{x} . The knn classifier works

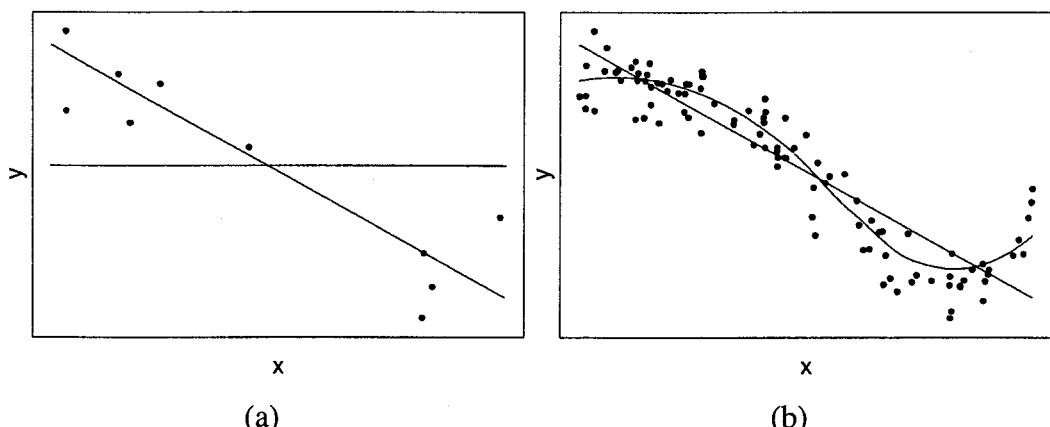


FIG. 7.6. Utility of linear and nonlinear models.

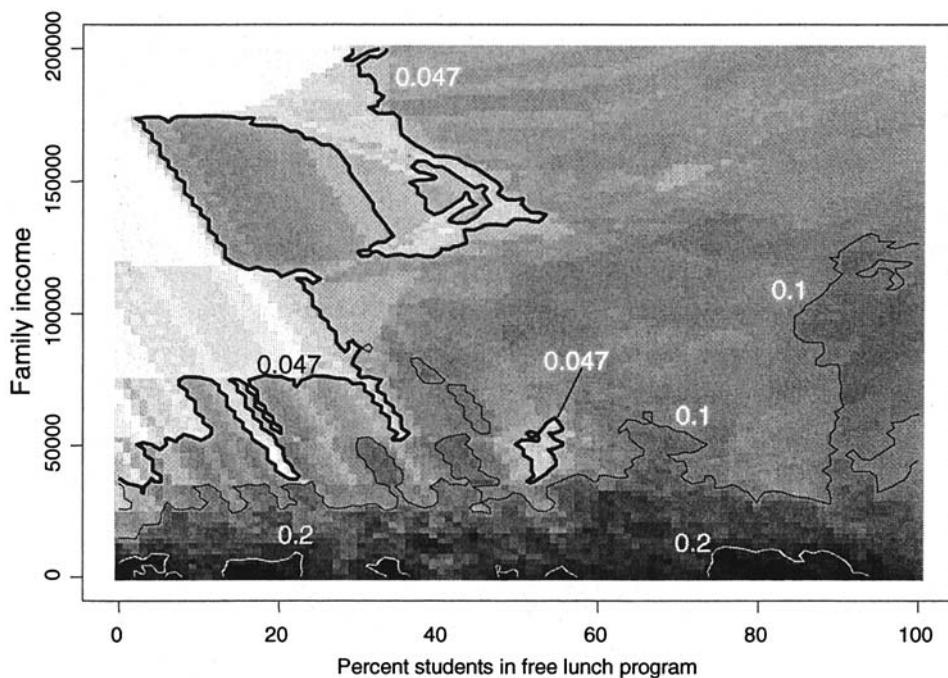


FIG. 7.7. The 100 nearest neighbor classifier for the high school dropout data. The darker regions of the figure represent greater dropout risk. The lighter regions of the figure indicate a near 0 dropout risk.

similarly. It collects the k nearest observations and predicts the class that minimizes cost, the most popular class in the case of equal misclassification costs. Although the method may seem naive, it is often competitive with other more sophisticated prediction methods. Figure 7.7 shows the knn classifier predicting high school dropout probability from family income and where $k = 100$. The features were rescaled for the distance calculations so that they both had unit variance. The heavy contour line marks the decision boundary between predicting a dropout and predicting a graduate. Only the students from the wealthier families in schools with few students on a free lunch program will not be classified as a dropout risk.

Where the linear model is rigid, the knn predictor is extremely flexible, as Fig. 7.7 clearly demonstrates. Compare Fig. 7.7 to Fig. 7.4. That flexibility can be abused by allowing k to be too small. Recalling the earlier discussion of common classification loss functions, knn tends to offer poor probability estimates but nevertheless tends to be quite good at minimizing misclassification cost. We can look at how different choices for k affect prospective predictive performance as shown in Fig. 7.8. Assuming that failure to identify a dropout is 20 times more costly than failure to identify a graduate, we can compute the average cost per student for our decision rule. Minimizing the expected cost heavily depends on careful selection of k . If we classify every student as a graduate, the expected cost is 3.3, about what we see with the 1-nearest neighbor classifier. Classifying all students as dropouts, the decision rule when k gets very large, produces an expected cost of 0.83 shown as the horizontal line in Fig. 7.8. The minimum expected cost, 0.75, occurs when $k = 90$. The 90 nearest neighbor classifier puts 83% of the students at a greater than 4.7% chance of dropout. The linear logistics regression model has an expected cost of 0.81 and classified 90% of the students as dropouts.

As N gets arbitrarily large and k grows at a certain rate (much slower than N) this predictor converges to the true optimal predictor, a property known as *Bayes risk consistency*. However,

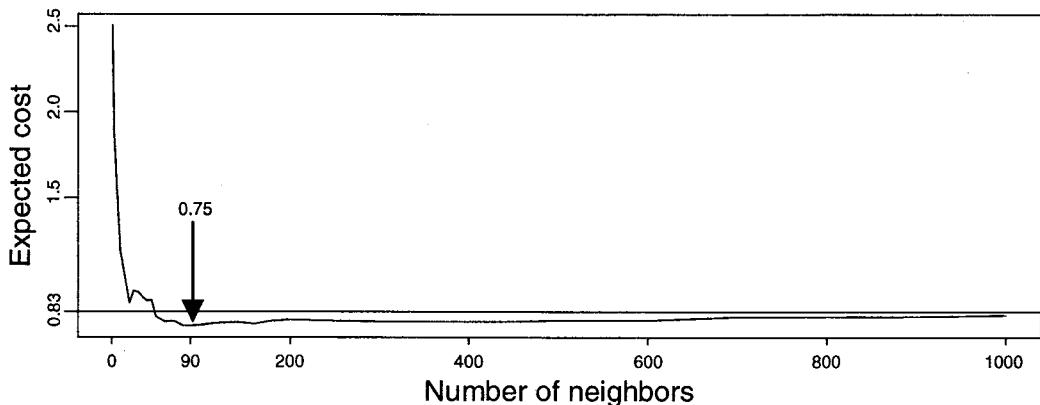


FIG. 7.8. Predictive performance for different values of k . Expected cost uses probability of dropout exceeding 4.7% as the decision boundary.

the performance of the predictor for data sets of practical size depends heavily on k , the metric used to determine which observations are close, and the dimension of the feature vector.

A natural generalization of the knn predictor $f(\mathbf{x})$ involves having every observation contribute its outcome to the prediction weighted by its distance to \mathbf{x} . Returning again to squared error

$$f(\mathbf{x}) = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N K(\mathbf{x}_i, \mathbf{x})(y_i - \theta)^2 = \sum_{i=1}^N w_i y_i \Bigg/ \sum_{i=1}^N w_i \quad (33)$$

a weighted average of the outcomes where $w_i = K(\mathbf{x}_i, \mathbf{x})$, a function that decreases as \mathbf{x}_i moves further from \mathbf{x} . The knn predictor is a special case of Equation 33, when $K(\mathbf{x}_i, \mathbf{x})$ takes value 0 or 1 depending on whether \mathbf{x}_i is among the k closest observations. First considering the case with a single continuous predictor, let $K(x_i, x)$ be the Gaussian density

$$K(x_i, x) = \exp\left(-\frac{1}{2}((x_i - x)/\sigma)^2\right) \quad (34)$$

with mean equal to x and standard deviation σ , known as the *bandwidth* for the kernel regression model. Figure 7.9 shows stroke rehabilitation cost models using two kernel regression estimates with different bandwidth settings and the linear model from Fig. 7.3. The cost axis is rescaled from Fig. 7.3 to reveal details of the model differences. As the bandwidth gets small the kernel regressor resembles the pointwise average estimate shown in Fig. 7.3 and exhibits a lot of variance in regions where there are fewer data points. The larger bandwidth is smoother and shows a lot of stability even in the extreme motor scores. Although all the methods align for the most common motor scores, the linear model reveals its *bias* in the extreme motor score values. In many prediction problems, data often show the presence of saturation effects (at some point additional improvements in motor ability do not decrease cost) and threshold effects (decreases in cost do not begin until motor exceeds some threshold). Note that if we observe only patients with motor score in the 30 to 60 range, the linear model works extremely well, and we would have little reason to consider other models. Other prediction methods can easily outperform linear models when a substantial portion of the data set lies to the right of the saturation point and to the left of the threshold point. Kernel regression methods generalize to multivariate feature vectors, and the reader is referred to Hastie, Tibshirani, and Friedman

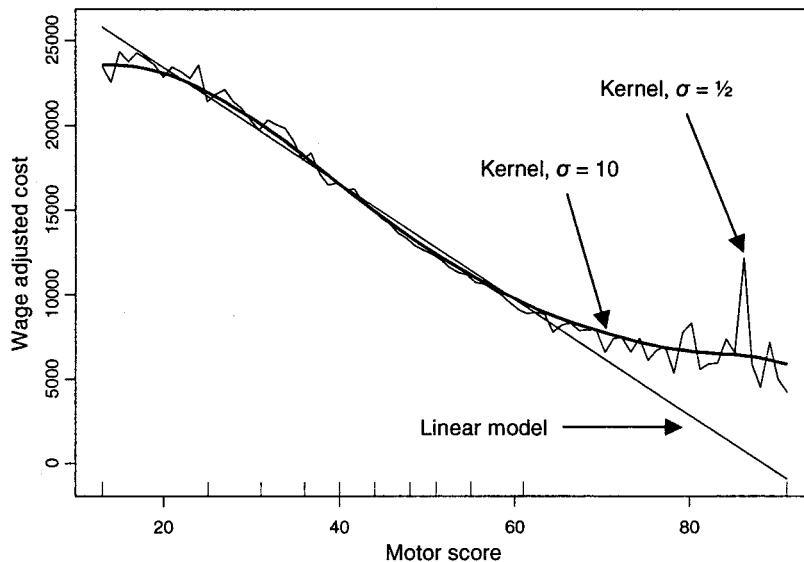


FIG. 7.9. A kernel regression prediction model.

(2001) for more details. Support vector machines, discussed in a later section, are in fact a particular kind of kernel regression method. The primary disadvantage of the knn predictor and naive implementations of kernel methods for data mining applications is the need to store the entire data set and the expense of searching for nearest neighbors or computing kernels to make a prediction for a single new observation. Some algorithms exist for trimming down the storage needs for these models. However, they still generally receive little attention as competitive data mining procedures in spite of their utility for nonlinear modeling.

For the data miner interested in prediction methods, nearest neighbor and kernel methods have instructional value. The heuristic is that when predicting an outcome at \mathbf{x} , we borrow information from those points in the data set that are close to \mathbf{x} . We trust that the underlying function is fairly smooth so that nearness in terms of \mathbf{x} implies similar outcomes. The linear models discussed earlier share this idea but assume the rigid functional form to interpolate between points. Remember that all prediction methods that minimize the same loss function differ only in how they interpolate between the points in the data set. In some fashion they combine the outcomes from points in a neighborhood near \mathbf{x} . How a method selects that neighborhood and how it combines the outcomes determine its scalability, its interpretability, and its predictive performance.

Tree Models

Chapter 1 in this handbook extensively discusses the use of tree structured prediction methods. Tree structured predictors usually assume that $f(\mathbf{x})$ is a piecewise constant function in which splits on the individual feature axes define the pieces. The terminal node of the tree defines the neighborhood, and the constant that minimizes the loss function within the terminal node becomes the node prediction.

The advantages of tree predictors for data mining is that fast algorithms exist to construct them from data, prediction for new observations is quick, the method handles all types of

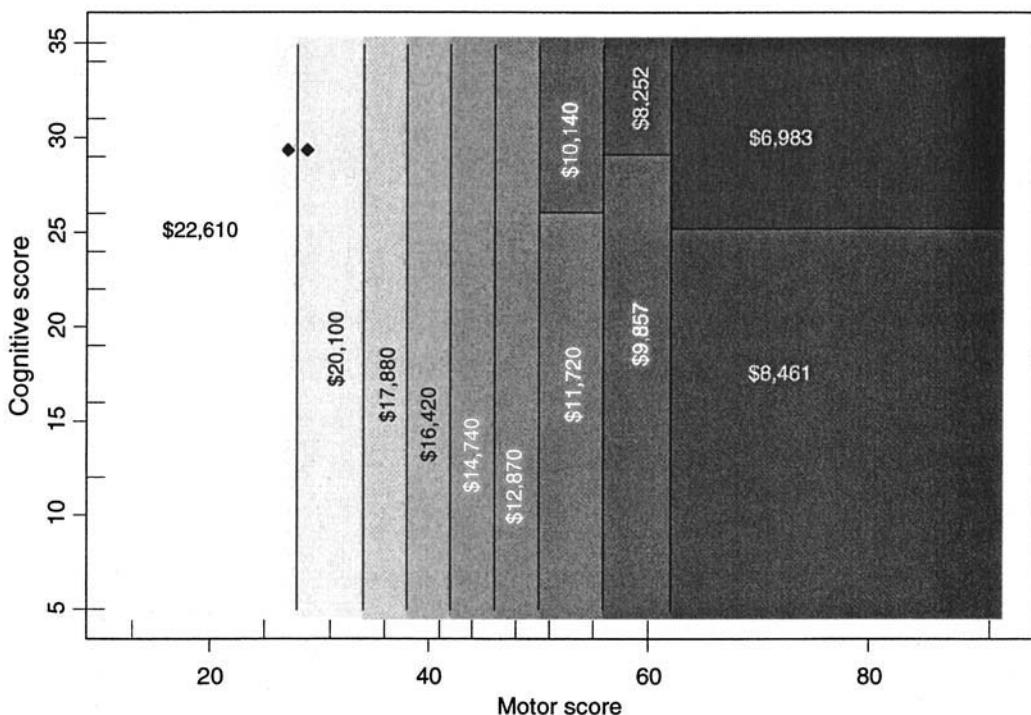


FIG. 7.10. CART for cost of stroke rehabilitation. The two diamonds refer to the hypothetical patients discussed in the text. The inward ticks on the axes mark the deciles of the distribution of the motor and cognitive scales.

input variables, and the model is storable in a compact form. This differs sharply from the nearest neighbor methods. However, as linear models can be criticized for their rigid functional form, the constant within-node prediction is rigid in a different way. Figure 7.10 shows how CART partitions stroke patients by motor and cognitive score into groups with relatively homogeneous cost. Note that in Fig. 7.10 the model predicts that the patient with a motor score of 27 and the patient with a motor score of 28 (marked with diamonds in the figure) have costs that differ by \$2,510, the difference in the predicted cost from each region. We really do not believe that cost varies so abruptly. Such biases around the partition edges can reduce prediction accuracy. Like the nearest neighbor predictor, tree structured procedures are generally Bayes risk consistent (Gordon & Olshen, 1984). That is, as the data set grows and the number of terminal nodes grows at a certain rate, the model converges to the minimizer of the population loss function. Although this is an interesting mathematical fact, it should be taken as a caution rather than a benefit. Trees can have a lot of variability in addition to edge biases.

Although trees often are appreciated for their apparent interpretability, one should use caution due to their instability. To demonstrate, we randomly split the high school dropout data into two parts and fit a CART classification tree to each. Figure 7.11 shows the resulting models. The two trees give very different perspectives concerning what is important in determining the risk of dropout. When several variables are correlated with each other and the outcome, the tree must select a single variable for the splitting variable, hiding the importance of other interesting inputs. Again, it is easy to be fooled by the tree's transparent functional form and overlook important variables not included in the tree.

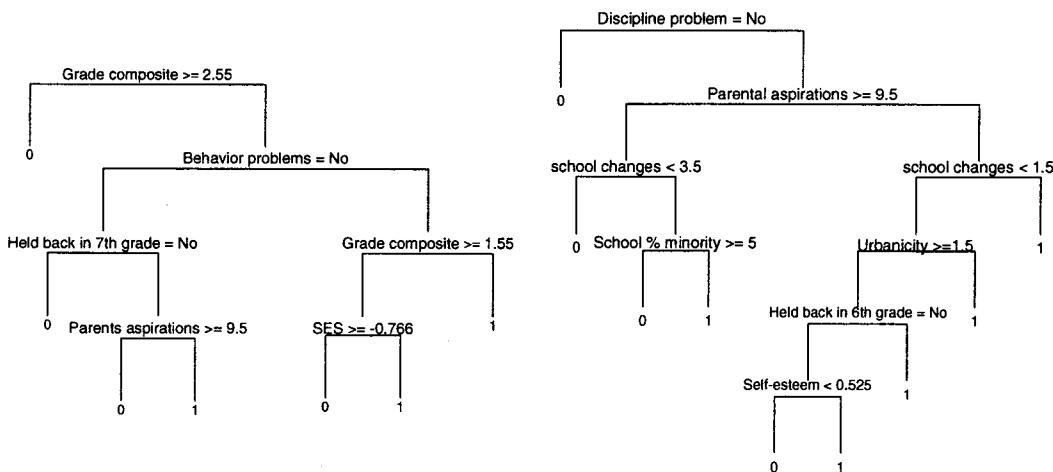


FIG. 7.11. CART fit to two random splits of the high school dropout data set.

Smoothing, Basis Expansions, and Additive Models

One disadvantage of tree models is that they produce discontinuous predictions, and the main disadvantage of the linear models is that they enforce strict smoothness everywhere. This section considers methods between these two extremes.

Splines utilize piecewise polynomials (usually linear or cubic) to model $f(x)$. Remember that trees use piecewise constants and that linear models use a single linear equation. Splines are, therefore, one further step in this progression. As with trees, spline methods need to select split points, known in spline terminology as *knots*. Between the knots we fit a polynomial of our choice to minimize our loss function. *Natural cubic splines* are among the most popular. Between knots these splines fit cubic polynomials but fit linear models in the range outside the first and last knots. In addition, they enforce continuity of $f(x)$, $f'(x)$, and $f''(x)$ at every knot. Figure 7.12 shows two examples of natural splines with differing numbers of knots. One can fit fairly flexible curves, including saturation and threshold effects previously mentioned, with only a handful of knots.

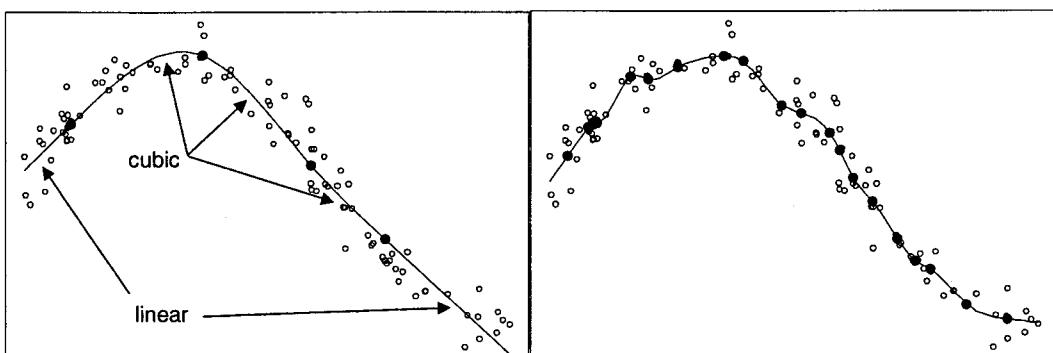


FIG. 7.12. Natural splines with 4 and 20 knots on simulated data. The knots, shown as solid points, are placed at equally spaced quantiles of the x variable.

The motivation for splines given earlier is that they seem a natural generalization of piecewise constant and linear models. Natural splines also arise in a curious way from a redefined loss function. Rather than simply minimize squared prediction error, consider minimizing it subject to a penalty on the magnitude of the function's second derivative.

$$\hat{J}(f) = \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i))^2 + \lambda \int_{-\infty}^{\infty} f''(x)^2 dx \quad (35)$$

If λ , known as the *smoothing parameter*, is 0, then the function can be arbitrarily jagged and interpolate the data. The best model when λ is large has $f''(x) = 0$ everywhere, the ordinary linear model. However, when λ takes on other, moderate values, the function that minimizes Equation 35 smoothly fits the data. It turns out that the minimizer of Equation 35 over a very rich class of functions is unique and is a *natural spline*! The minimizing natural spline has knots at every observed x_i , but the coefficients of each cubic are further restricted by an amount that depends on λ . We can use cross-validation to select λ .

A discussion on the details for fitting natural splines comes later, but as discussed so far, they can be computationally expensive for use with massive data sets. An easy adaptation is to reduce the number of knots as originally shown. Rather than placing knots at every data point, 5 to 20 knots are likely sufficient for many univariate applications. A potential downside of splines is that they are *not* invariant to transformations in the x variable. If instead of using x we use $\log x$ as the feature, we obtain a different natural spline for the same λ . Although tree models are invariant to transformations in x , for smooth models one needs to think more carefully about the scale used for x . A later section shows that boosted trees allow fairly smooth predictions but are also invariant to transformations of the features.

The natural spline discussion focussed on problems with a single feature. So far, for problems involving more than one feature we have considered only models that are linear in \mathbf{x} or are local averages in neighborhoods of \mathbf{x} . *Basis expansions* represent yet another broad class of models. Assume that the best predictor $f(\mathbf{x})$ is well approximated by a sum of *basis functions*.

$$f(\mathbf{x}) = \sum_{k=1}^K \beta_k g_k(\mathbf{x}) \quad (36)$$

The basis functions in the collection, $g_k(\mathbf{x})$, are often simple functions that can be combined to describe more complex functions.

Many of the models we have discussed fall into this framework. Note that if $g_k(\mathbf{x}) = x_{k-1}$, with $x_0 = 1$, then Equation 36 reduces to the linear model previously discussed. Trees also can be recast as a basis expansion where $g_k(\mathbf{x}) = I(\mathbf{x} \in N_k)$, where N_k is a defined by upper and lower limits on certain components of \mathbf{x} , such as the 12 partitions shown in Fig. 7.10. Tree learning algorithms vary on how they select the number of basis functions and the form of N_k . One can naturally imagine other choices for the basis functions.

A univariate cubic spline with K knots c_1, \dots, c_K is decomposable into $K+4$ basis functions

$$g_1(x) = 1 \quad g_2(x) = x \quad g_3(x) = x^2 \quad g_4(x) = x^3 \quad g_{k+4}(x) = (x - c_k)_+^3 \quad (37)$$

where $(x)_+ = \max(0, x)$. The cubic spline maintains a cubic fit before c_1 and after c_K where the natural spline uses a linear fit. To fit these models we can utilize all the methods developed for fitting linear models. Note that if we have $K = 2$ knots our predictor looks like

$$f(x) = \beta_1 + \beta_2 x + \beta_3 x^2 + \beta_4 x^3 + \beta_5 (x - c_1)_+^3 + \beta_6 (x - c_2)_+^3 \quad (38)$$

precisely a linear model with five features. With a little effort we can see that Equation 38 is a continuous piecewise cubic function with continuous first and second derivatives at the knots. Because Equation 38 is simply a linear model, we can use Equation 17 to fit the cubic spline.

Fitting cubic splines using the basis functions shown in Equation 37 requires $O(NK^2 + K^3)$ operations. Other sets of basis functions for cubic and natural splines are more efficient. In particular, *B-splines*, have computational properties that allow the models to be fit in $O(N \log N + K)$ operations.

General multivariate extensions to natural splines exist, known as *thin-plate splines*, but have not been developed to the computationally efficiency needs of data mining yet. *Additive models*, on the other hand, can take advantage of smoothing methods to create particularly powerful methods for exploratory analysis as well as prediction. Additive models use the basis functions $g_k(\mathbf{x}) = g_k(x_k)$, so that each basis function is a function of only one of the features. In this way we can generalize from the simple linear model while maintaining a fairly simple, stable model.

For the cost of rehabilitation data we can fit a regression model where the predictor has the form

$$f(\mathbf{x}) = \beta_0 + g_1(\text{motor}) + g_2(\text{cognitive}) \quad (39)$$

requiring that g_1 and g_2 are smooth functions of their arguments by modeling them as natural splines. Additive models are not Bayes risk consistent in most applications. That is, as the data set grows they cannot capture true interaction effects. However, they can often outperform other methods in practice, especially when the data is noisy. In addition, plotting the $g_k(x_k)$ can give much insight into the contribution of the individual features to the outcome. Figure 7.13

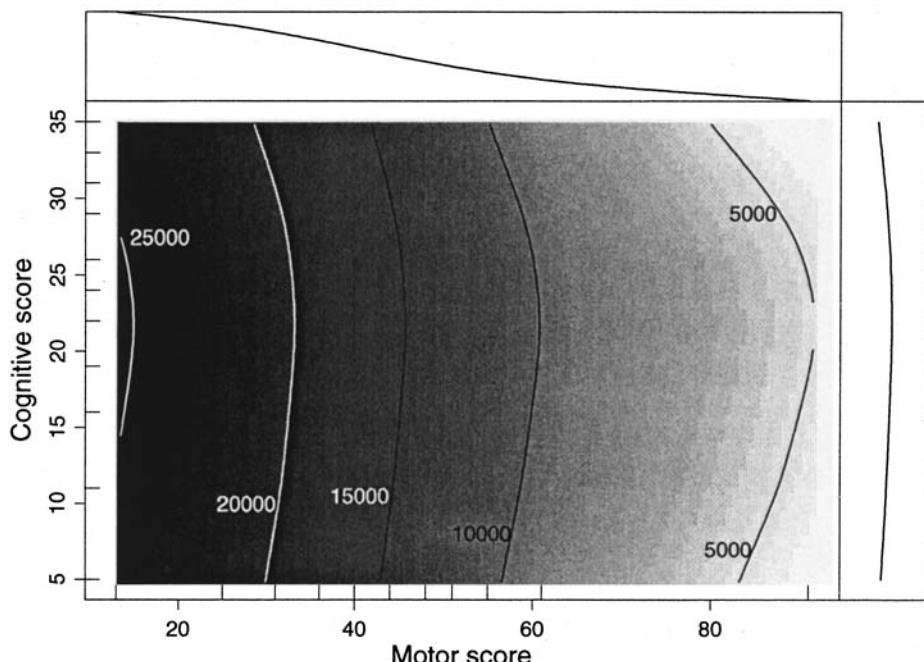


FIG. 7.13. The estimates of g_1 and g_2 for cost of rehabilitation. The curves plotted in the margin are g_1 (motor) and g_2 (cognitive).

shows the additive model fit to the cost of stroke rehabilitation data. Compare the contours with Fig. 7.10. The marginal plots in the figure show the estimates of g_1 and g_2 and are scaled equivalently. From the marginals alone we can see that the motor score is really driving the cost. The cognitive score has some information and seems to indicate that costs are slightly lower for patients at the extremes of the cognitive scale.

The *backfitting algorithm* is the common method of fitting additive models of this form. The constant β_0 is fixed at the mean of the y_i 's and the iterative algorithm begins by fitting the univariate natural spline to predict y from motor score. Then the algorithm builds a natural spline to fit the cognitive score to the residuals left over from the motor score component. The motor score component is refit, and the process iterates until convergence. This algorithm is fast as it generally involves only a few iterations, each of which uses an efficient B-spline implementation. Hastie and Tibshirani (1990) provided more details on additive models, including extensions to the GLMs discussed earlier. The interested reader might also look into multivariate adaptive regression splines (MARS) (Friedman, 1991). Just as CART implements a greedy search to construct a tree, MARS greedily constructs a model by introducing features into the prediction model (usually) through linear splines. It selects the variable and the spline transformation that offers the greatest decrease in prediction error through an exhaustive search similar to CART. The result is a modeling framework that allows some variables to enter as main effects (CART is only one big interaction term) and others to be interacted with one another. Many variations on this theme are possible.

Neural Networks

Although chapter 3 in this handbook discusses neural network models in much greater detail, the brief discussion here casts them into the framework developed in this chapter. Like all the methods discussed so far, there are variations in the functional form of the neural network predictors and the algorithms for fitting them. Generally, however, the single hidden layer neural network can be expressed in terms of a basis expansion

$$f(\mathbf{x}) = \sum_{k=1}^K \beta_k s(\alpha'_k \mathbf{x}) \quad (40)$$

where $s(z) = 1/(1 + e^{-z})$, the logistic transformation. The K basis functions are known as the hidden units, and the number of them is often fixed but may be selected by cross-validation.

For univariate classification problems neural networks model $P(Y = 1 | \mathbf{x})$ like the logistic regression model Equation 18. For M -class classification problems we create a multivariate outcome for each observation, \mathbf{y}_i , where the m^{th} value takes on the value 1 if the observation is in class m . Maintaining a common set of α_k 's as in Equation 40, we allow each dimension of \mathbf{y} to have a different β_{km} and the prediction as

$$f_m(\mathbf{x}) = \sum_{k=1}^K \beta_{km} s(\alpha'_k \mathbf{x}) \quad \text{and} \quad P(Y = m | \mathbf{x}) = \frac{\exp f_m(\mathbf{x})}{\sum_{m'=1}^M \exp f_{m'}(\mathbf{x})}. \quad (41)$$

See chapter 3 concerning methods for estimating the parameters of neural networks. Their inclusion in this section is to show that they share the same building blocks as all the other methods. Neural networks make specific choices for their basis functions but then can be used to minimize any loss function that we adopt for an application. The term *universal approximator* has been used for neural networks to mean that they can approximate any regression function. As the data set grows and K is allowed to grow, they are Bayes risk consistent (Barron, 1991).

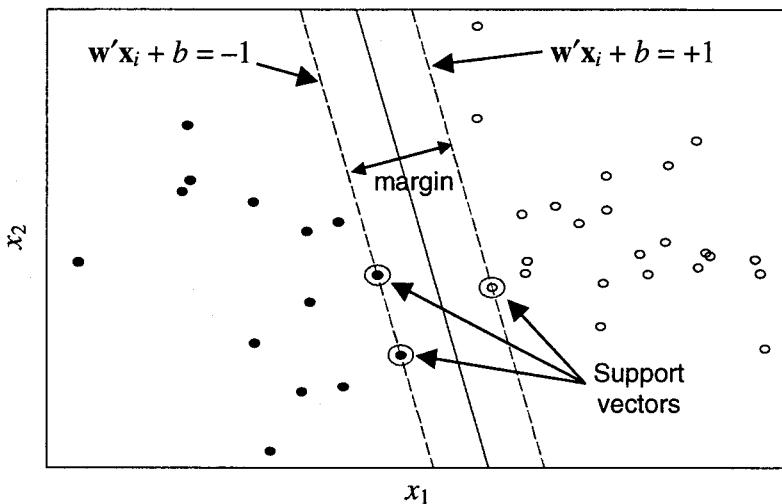


FIG. 7.14. Geometry of the linear SVM. x_1 and x_2 represent two arbitrary continuous features.

These are properties also shared by trees, nearest neighbors, and thin-plate splines. However, neural networks have the curious mathematical property that the rate at which the estimated predictor converges to the best predictor does not depend on the dimension number of features (Barron, 1993).

Support Vector Machines

Several of the previous classification examples focused on obtaining good estimates of $P(Y = 1 | \mathbf{x})$ and then perhaps applying Equation 5 to get a cost minimizing decision rule. The support vector machine (SVM) in its basic form aims directly for the decision boundary.

We begin with an example of a linearly separable data set shown in Fig. 7.14. There are two continuous features, and the class labels are $y = +1$ or $y = -1$, marked by clear and solid dots, respectively. Many lines can separate the two classes. SVMs select the line that maximizes the *margin*, the space between the decision boundary and the closest points from each of the classes. With separable classes we can find a separating line having the form $\mathbf{w}'\mathbf{x} + b = 0$ such that

$$\mathbf{w}'\mathbf{x}_i + b \geq +1 \quad \text{when } y_i = +1 \quad (42)$$

$$\mathbf{w}'\mathbf{x}_i + b \leq -1 \quad \text{when } y_i = -1. \quad (43)$$

We can rescale \mathbf{w} and b so that for some point(s) equality holds in Equations 42 and 43. A little geometry shows that the two planes defined by $\mathbf{w}'\mathbf{x}_i + b = 1$ and $\mathbf{w}'\mathbf{x}_i + b = -1$ are parallel, have no points between them, and are separated by a distance of d where

$$d^2 = 4 \sqrt{\sum_{j=1}^d w_j^2} = 4/\|\mathbf{w}\|^2. \quad (44)$$

Putting these steps together, fitting the SVM corresponds to maximizing the margin Equation 44 subject to the constraints of Equations 42 and 43 solvable as a quadratic optimization problem.

In the more practical case when the classes are not completely separable, SVMs still maximize the margin but allow for a limited number of observations to be on the wrong side of the decision boundary. We can introduce slack variables, $\xi_i \geq 0$, into the constraints Equations 42 and 43 as

$$\mathbf{w}'\mathbf{x}_i + b \geq 1 - \xi_i \quad \text{when } y_i = +1 \quad (45)$$

$$\mathbf{w}'\mathbf{x}_i + b \leq -1 + \xi_i \quad \text{when } y_i = -1. \quad (46)$$

$$\sum_{i=1}^N \xi_i \leq B \quad (47)$$

Again SVMs maximize the margin subject to Equations 45 and 46 but also penalizes the sum of the ξ_i Equation 47, restricting the budget we have for observations being too far from the decision boundary. The details of the fitting algorithm are fairly tedious, but two important ideas come in their development. To fill in the details not covered here see the excellent SVM tutorial in Burges (1998). The SVM optimization problem can be recast in its Wolfe dual form as

$$J(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i' \mathbf{x}_j \quad (48)$$

along with several constraints on the α_i 's relating to the observed data and the ξ_i . The solution for the prediction rule is

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}'\mathbf{x} + b) = \text{sign}\left(\sum_{i=1}^N \alpha_i y_i \mathbf{x}_i' \mathbf{x} + b\right). \quad (49)$$

Usually most of the α_i 's turn out to be 0. Those observations corresponding to a positive α_i are called the *support vectors*. Conveniently, the prediction rule depends only on the support vectors.

In Fig. 7.14 there were only three support vectors.

As with the linear models described earlier, SVMs are easily extended by expanding the set of features used in forming the decision boundary. We could assume that the decision boundary has a basis expansion form like Equation 36. An interesting feature of the SVM is that the features only enter the fitting stage Equation 48 and the prediction stage Equation 49 through the inner product of the features, $\mathbf{x}'\mathbf{x}$. Rather than having to specify the set of basis functions explicitly, only the inner product of the basis functions needs definition. The common replacements for $\mathbf{x}'\mathbf{x}$ are

$$\begin{aligned} \text{radial basis} \quad K(\mathbf{x}_i, \mathbf{x}) &= \exp(-\|\mathbf{x}_i - \mathbf{x}\|^2/c) \\ \text{polynomial} \quad K(\mathbf{x}_i, \mathbf{x}) &= (1 + \mathbf{x}'\mathbf{x})^c \\ \text{neural network} \quad K(\mathbf{x}_i, \mathbf{x}) &= \tanh(c_1 \mathbf{x}_i' \mathbf{x} + c_2). \end{aligned} \quad (50)$$

These choices for the kernel are flexible and quick to compute. Therefore, to use SVMs for prediction one needs to select a kernel, the kernel tuning parameters, and the roughness of the decision boundary determined by B from Equation 47.

To make the comparison of SVMs with other classifications, we can reformulate the optimization problem in Equations 45–47 as a loss function shown in Equation 51.

$$J(f) = \sum_{i=1}^N [1 - y_i f(\mathbf{x}_i)]_+ + \lambda \sum_{j=1}^d w_j^2. \quad (51)$$

Revisit Fig. 7.2 to compare this to the other classification loss functions. The first part of the SVM loss function encourages y_i and $f(\mathbf{x}_i)$ to agree on the sign. When they disagree, the penalty increases linearly in the magnitude of $|f(\mathbf{x}_i)|$. The second component penalizes the magnitude of the linear coefficients, equivalently controlling the margin. As with the smoothing splines (Equation 35), λ is a user specified smoothing parameter. SVMs, therefore, are characterized as using Equation 51 as the loss function, having structural form (Equation 49) that may be generalized with a kernel choice from Equation 50, and utilizing quadratic programming to fit the models to data.

Boosting

Although boosting has a technical definition in terms of computational learning theory, it has been more generally applied to methods that use a flexible gradient ascent approach to fit a prediction model. Schapire and Singer (1998) introduced the *Real AdaBoost* algorithm for classification problems, one of the most commonly used forms of boosting. Friedman, Hastie, and Tibshirani (2000) decomposed this algorithm into its basic components: the loss function, the functional form, and the estimation procedure.

First consider the classification problem for which y takes on values +1 or -1. The Real AdaBoost loss function is

$$J(f) = E_{y|\mathbf{x}} \exp(-yf(\mathbf{x})). \quad (52)$$

If y and $f(\mathbf{x})$ frequently agree on the sign, then Equation 52 is small. Furthermore, the AdaBoost loss function is an upper bound to the misclassification rate as shown in Fig. 7.2. So intuitively, if on average y and $f(\mathbf{x})$ frequently agree on the sign, then Equation 52 is small and the misclassification rate will be small.

Boosting's main innovation comes when we look at how it estimates $f(\mathbf{x})$. Assume that $f_0(\mathbf{x})$ is our current guess for the best predictor. To improve on our current guess, we can consider adding a new function, $g(\mathbf{x})$, to "fix" it. Of course, we want to choose the $g(\mathbf{x})$ that helps us to decrease the loss function (Equation 53) the most.

$$J(f) = E_{y|\mathbf{x}} \exp(-y(f_0(\mathbf{x}) + g(\mathbf{x}))). \quad (53)$$

Setting the derivative of Equation 53 with respect to $g(\mathbf{x})$ equal to zero gives us the "direction" in which $f_0(\mathbf{x})$ needs the most improvement,

$$g(\mathbf{x}) = \frac{1}{2} \log \frac{P_w(y = +1 | \mathbf{x})}{P_w(y = -1 | \mathbf{x})}, \quad (54)$$

where $P_w(y | \mathbf{x})$ is a weighted probability estimate with weight $\exp(-yf_0(\mathbf{x}))$. The brief derivation here deals with expectations, but this is easily translated into data applications.

Begin with a naïve guess for $f_0(\mathbf{x})$, perhaps a constant, c .

For t in $1, \dots, T$ do the following

1. Assign weights to the observations, $w_i = \exp(-yf_{t-1}(\mathbf{x}_i))$.
2. Obtain an estimate for $P_w(y = 1 | \mathbf{x})$ using any prediction model that can handle weights and puts out a probabilistic prediction. Classification trees are by far the most popular.
3. Form $g_t(\mathbf{x})$ as in Equation 54.
4. Update the current guess as $f_t(\mathbf{x}) \leftarrow f_{t-1}(\mathbf{x}) + g_t(\mathbf{x})$.

For a new observation \mathbf{x} the model predicts that $y = \text{sign}(f_T(\mathbf{x}))$.

It is helpful at this point to note some similarities with the linear logistic regression discussion earlier. Like AdaBoost, the IRLS algorithm for linear logistic regression repeatedly fits weighted regression models to the data set to obtain the best fit. In that situation the model is restricted to have a linear form, and so the updates get directly absorbed into the current guess for β . In addition, the IRLS weights were $p_i(1 - p_i)$, which are largest when $p_i = 1/2$, the point at which the equal cost decision rule is most in question. This behavior is replicated in the AdaBoost weights. Those weights are largest when y and $f_0(\mathbf{x})$ greatly disagree, perhaps the more difficult observations to classify.

Having discussed the loss function and estimation process, the last component is the predictor's form. This depends much on the base classifier selected for the $P_w(y = 1 | \mathbf{x})$ estimate. If we use the naive Bayes classifier, AdaBoost leaves it unchanged from the linear form (approximately so for the earlier discrete AdaBoost algorithm, Ridgeway, Madigan, Richardson, & O'Kane, 1998). If we use stumps, decision trees with only a single split, the final model falls into the class of additive models of the form in Equation 39. To see this, note that the model that AdaBoost produces has a basis expansion form

$$f_T(\mathbf{x}) = c + g_1(\mathbf{x}) + g_2(\mathbf{x}) + \cdots + g_{T-1}(\mathbf{x}) + g_T(\mathbf{x}). \quad (55)$$

In the additive model discussion the basis functions were decided on ahead of time then fit simultaneously to the data. Boosting selects the basis functions in Equation 55 greedily rather than simultaneously. If we use stumps, then each $g_t(\mathbf{x})$ is a function of one feature alone. We can then collect all those $g_t(\mathbf{x})$'s that split on variable j into a single component.

$$f_T(\mathbf{x}) = c + g_1^*(x_1) + g_2^*(x_2) + \cdots + g_d^*(x_d). \quad (56)$$

A two split decision tree for the base classifier results in an additive model with each term potentially being a function of two features. Therefore, the depth of the tree allows us to select the complexity of the feature interactions.

For data mining applications decision trees turn out to be an excellent choice for the base model. Trees seamlessly handle continuous, ordinal, nominal, and missing data. The other methods that this chapter discusses do not necessarily handle such a mix of data types easily. Trees are also invariant to one-to-one transformations of the features, using only the order information to determine the optimal partitions. For example, whether we use income or $\log(\text{income})$ as a feature, the tree will have the same form and produce the same predictions either way. The split points will be on the corresponding scale but all else will be the same. The performance of other methods can be particularly sensitive to feature transformations. This invariance in turn adds a bit of robustness to the boosted tree model.

Generalizations of the AdaBoost algorithm amount to selecting a different loss function, applying the same estimation procedure, and perhaps examining different base classifiers. Boosting can be generalized to all of the other loss functions discussed in the section on loss functions. Friedman (2001) gives a general framework for developing boosting algorithms, deriving specific algorithms for least squares, robust, and logistic regression.

To demonstrate the flexibility of boosting, consider the Cox loss function for survival data (see also Ridgeway, 1999). We apply the boosting ideas to maximize the log-partial likelihood, the logarithm of Equation 13.

$$J(f) = \sum_{i=1}^N \delta_i \left[f(\mathbf{x}_i) - \log \left(\sum_{j=1}^N I(t_j \geq t_i) \exp f(\mathbf{x}_j) \right) \right]. \quad (57)$$

As with AdaBoost we can find a function to add to $f(\mathbf{x}_i)$ to improve the model. The derivative of Equation 57 pointwise with respect to $f(\mathbf{x}_i)$ indicates the direction that adjusts $f(\mathbf{x}_i)$ to increase the likelihood. The derivative equals

$$z_i = \delta_i - \sum_{j=1}^N \delta_j \frac{I(t_i \geq t_j) \exp f(\mathbf{x}_i)}{\sum_{k=1}^N I(t_k \geq t_j) \exp f(\mathbf{x}_k)}. \quad (58)$$

This means that for some step size, ρ , if all the $f(\mathbf{x}_i)$ in Equation 57 were replaced with $f(\mathbf{x}_i) + \rho z_i$ then the likelihood would increase. In the AdaBoost algorithm ρ is exactly 1 but more generally we need to set it. Generally, z_i has information indicating how to adjust the current guess for $f(\mathbf{x})$. To pull in the features, we fit a regression tree, $g(\mathbf{x})$, predicting z_i from \mathbf{x}_i . If $g(\mathbf{x})$ can capture the gradient information from z_i a good update is $f(\mathbf{x}) \leftarrow f(\mathbf{x}) + \rho g(\mathbf{x})$, where ρ can be selected with a line search.

To examine the performance of boosting Cox's proportional hazards model, we turn to a clinical trial for testing the drug DPCA for the treatment of primary biliary cirrhosis of the liver (PBC). This data set has been the subject of several modern data analyses (Dickson, Grambsch, Fleming, Fisher, & Langworthy, 1989; Fleming & Harrington, 1991; Raftery, Madigan, & Volinsky, 1996). The data consist of 310 patients with complete observations on the predictors. Of these, 124 patients died during the study and the remaining 186 were censored observations. Of the eight features Raftery et al. (1996) considered, we selected the six continuous features for use in the model.

We tested this method by comparing the out-of-sample predictive performance of the linear Cox model to the boosted Cox model using regression stumps as $g_i(\mathbf{x})$, the base regressor. To judge the out-of-sample predictive performance of the two models, we trained each on half of the observations, reserving the rest for a test set. Figure 7.15 shows the value of the validation log-likelihood as the algorithm proceeds. To obtain a very smooth prediction surface, we shrunk the ρ by a factor of 1,000, making each adjustment very slight and requiring many more iterations. We see that after 40,000 iterations the boosted estimate has surpassed the

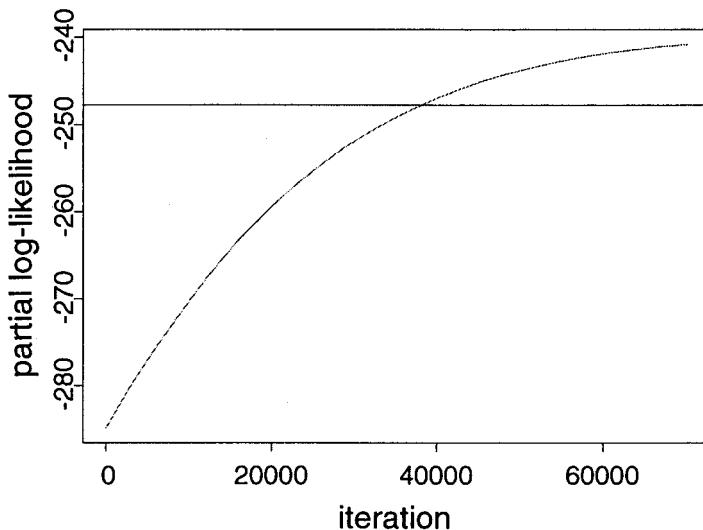


FIG. 7.15. Partial log-likelihood for PBC data. The upward trending curve indicates the boosted trees performance on test data. The horizontal line indicates the partial likelihood from the linear Cox model.

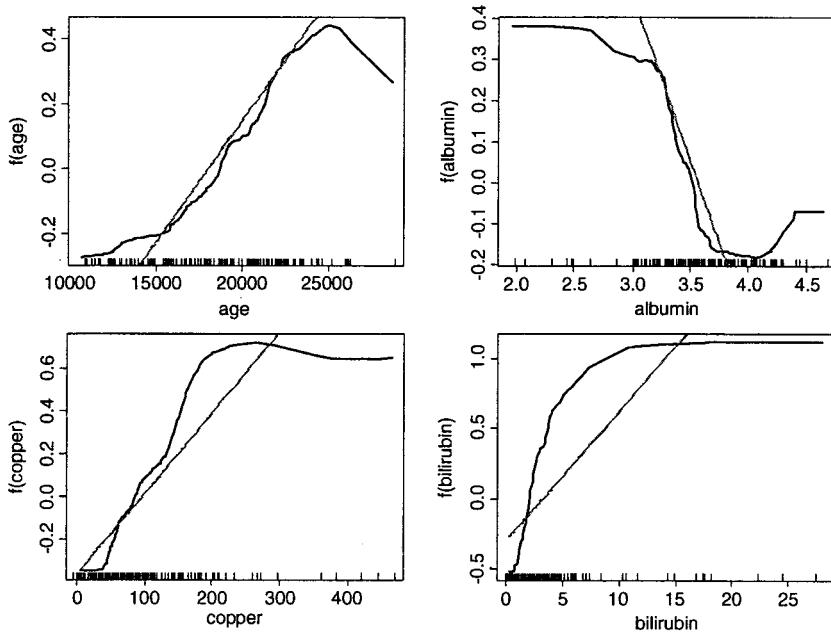


FIG. 7.16. Boosted estimates of the main effects of the PBC data. The curves are the boosted estimates of the functions, and the superimposed lines are the estimates from the linear model.

linear Cox model. Furthermore, for the next 30,000 iterations the boosted version continues to improve, although at a diminishing rate of return. Even though the model fitting used many iterations, it takes about a minute to complete the iterations on this small data set.

Because the base regressors were one-split decision trees, we can investigate the estimates of the main additive effects as in Equation 56. Figure 7.16 shows the boosted estimates of four of those effects along with those based on the linear model. Every variable shows evidence of either a threshold effect, a saturation effect, or both. In the region where most of the data points are concentrated, for most of the variables (except bilirubin) the underlying regressor is nearly linear. For this reason we would expect the linear model to perform reasonably well. However, for a patient with a more extreme value for any of the variables the boosted model is far superior. When effects depart substantially from linearity, as in the PBC data set, accurate survival prediction for all patients depends on a nonlinear estimation procedure such as boosting.

AVAILABILITY OF SOFTWARE

This section returns to the methods reviewed in this chapter indicating what software tools are available. Many of the available tools come in the form of add-ons to other packages, such as S-plus, Gauss, or Matlab. In general, a simple web search for the method will likely turn up everything from original source code, to Matlab scripts, to full stand-alone commercial versions. The R project (www.r-project.org) is a great place to start for statistical algorithms. It is a free, full-featured statistical package maintained by many in the statistical community and is often the place where the latest statistical algorithms first appear. Weka

(<http://www.cs.waikato.ac.nz/ml/weka/>) is another well integrated, ever improving, package that implements many algorithms popular in the machine learning community. There are stand-alone packages that implement individual algorithms, but they often lack convenient methods for managing the data sets.

Generalized linear models. All standard statistical packages, SAS, SPSS, Stata, S-plus, R, etc., include GLM procedures as well as the Cox model and other survival models.

Generalized additive models. Original source code is available from StatLib (www.lib.stat.cmu.edu/general). The standard S-plus (gam()) and R distributions (mgcv library) come with GAM. SAS as of version 8.2 has PROC GAM.

K nearest neighbors. Available in R (knn() in the class library) and Weka (weka.classifiers.IBk).

Tree structured models. Salford Systems (www.salford-systems.com) maintains and develops the official implementation of CART. Rpart, developed at the Mayo clinic, is part of the standard R distribution and is full-featured including surrogate splitting for missing values. S-plus (tree()) and SPSS (AnswerTree) include tree structured modeling but do not implement all the features of the CART algorithm. Also look for implementations of Quinlan's C4.5 and C5.0 on the web.

MARS. Salford Systems maintains and develops MARS. There are various other implementations, usually add-ons for other packages such as S-plus or Gauss.

Support vector machines. Many research packages are now available. SVM-Light is one of the most popular and is freely available at svmlight.joachims.org.

Neural networks. There are many variants on neural networks and at least as many programs out there. Matlab users may want to look at Netlab (www.ncrg.aston.ac.uk/netlab) to become more familiar with the technique. For a large catalog of free and commercial packages visit www.emsl.pnl.gov:2080/proj/neuron/neural.

Naive Bayes. Available in Weka. Also, Bayda exclusively implements naive Bayes (www.cs.helsinki.fi/research/cosco/Projects/NONE/SW).

Boosting. To accompany his paper, Gradient Boosting (Friedman, 2001), there is an R add-on for the LogitBoost algorithm, least squares, and robust regression methods. (www-stat.stanford.edu/~jhf). Boosting methods for AdaBoost and other loss functions including Poisson regression and the Cox model are available at www.i-pensieri.com/gregr as an S-plus or R add-on, additional functionality forthcoming. Weka can apply the AdaBoost algorithm to any of its standard classifiers.

SUMMARY

For prediction problems the process begins by choosing a loss function, then selecting the type of predictor, and lastly determining how best to fit the model to potentially massive data sets. Of course, this ordering is ideal, but in practice we may find that computational complexity or a policy constraint prevents us from modeling exactly how we had hoped. As was shown, algorithms designed to minimize particular loss functions can be slightly modified to minimize other loss functions that we might prefer. Indeed, SVMs have been used to maximize the logistic log-likelihood and knn has been used to minimize squared error loss. So although this chapter focussed only on particular combinations commonly used in practice, data analysts can and should think creatively to match loss, model structure, and fitting algorithm to the problem at hand. In public policy problems as well as business and scientific applications, interpretability

is frequently a constraint. At times the SVM or the boosted tree model might predict the outcome better than the more easily interpreted models. However, applications often require a model fit that can be translated into humanly understandable relationships between the features and the outcome. When a model like the linear model adequately approximates the optimal predictor, interpretation is not in doubt. One should use great caution when trying to interpret an apparently interpretable model when more complex models exist that substantially outperform it. Clearly, we should be careful when creating policies and actions based on interpreting models that do not fit the data well. See Breiman (2001) and the accompanying discussion for both sides of this debate. As noted, interpreting the apparently interpretable decision tree actually requires a fair bit of care. Research continues to develop in the area of converting the competitive “black box” models into interpretable patterns. Friedman (2001), for example, makes substantial gains in turning complex boosted regression models into understandable relationships between features and outcomes. Although interpretability potentially is an issue, scalability is the main problem facing data miners. The simplest models, such as the linear models, can be fit to data in a small number of passes through the data, sometimes only once. But the benefit of having large data sets is the ability to model complex, nonlinear, deeply interacted functions. As discussed in chapter 1, researchers have made substantial progress in creating scalable algorithms for decision trees. Research also has shown that subsampling within boosting iterations not only decreases the computational complexity but also incidentally improves predictive performance. Computational advances are making SVMs potentially applicable to large data mining problems. Watch for continued advances in each of these areas over the next few years.

REFERENCES

- Barron, A. R. (1991). Complexity regularization with application to neural networks. In G. Roussas (Ed.), *Nonparametric functional estimation and related topics* (pp. 561–576). Dordrecht, Netherlands: Kluwer Academic Publishers.
- Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoid function. *IEEE Transactions on Information Theory*, 39, 930–945.
- Berger, J., & Wolpert, R. (1984). *The likelihood principle*. Institute of Mathematical Statistics, Hayward, CA.
- Bloomfield, P., & Steiger, W. L. (1983). *Least absolute deviations: Theory, applications, and algorithms*. Boston: Birkhauser.
- Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78, 1–3.
- Breiman, L. (2001). Statistical modeling: The two cultures. *Statistical Science*, 16(3), 199–231.
- Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Knowledge Discovery and Data Mining*, 2, 121–167.
- Dickson, E. R., Grambsch, P. M., Fleming, T. R., Fisher, L. D., & Langworthy, A. (1989). Prognosis in primary biliary cirrhosis: Model for decision-making. *Hepatology*, 10, 1–7.
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, 179–188.
- Fleming, T. R., & Harrington, D. P. (1991). *Counting processes and survival analysis*. New York: Wiley.
- Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics*, 19, 1–82.
- Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29, 1189–1232.
- Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion). *Annals of Statistics*, 28, 337–374.
- Gordon, L., & Olshen, R. A. (1984). Almost surely consistent nonparametric regression from recursive partitioning schemes. *Journal of Multivariate Analysis*, 15, 147–163.
- Greene, W. H. (1999). *Econometric analysis* (4th ed.). Englewood Cliffs, NJ: Prentice Hall.
- Hand, D. J., & Yu, K. (2001). Idiot's Bayes—not so stupid after all? *International Statistical Review*, 69, 385–398.
- Hastie, T., & Tibshirani, R. (1990). *Generalized additive models*. London: Chapman & Hall.
- Hastie, T., Tibshirani, R., & Buja, A. (1994). Flexible discriminant analysis by optimal scoring. *Journal of the American Statistical Association*, 89, 1255–1270.

- Hastie, T., Tibshirani, R., & Friedman, J. (2001). *Elements of statistical learning: Data mining, prediction, and inference*. New York: Springer.
- Huber, P. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35, 73–101.
- McCullagh, P., & Nelder, J. (1989). *Generalized linear models*. London: Chapman & Hall.
- Raftery, A. E., Madigan, D., & Volinsky, C. T. (1996). Accounting for model uncertainty in survival analysis improves predictive performance. In J. M. Bernardo, J. O. Berger, A. P. Dawid, & A. F. M. Smith (Eds.), *Bayesian statistics 5* (pp. 323–349). Oxford, U.K.: Oxford University Press.
- Ridgeway, G. (1999). The state of boosting. *Computing Science and Statistics*, 31, 171–181.
- Ridgeway, G. (2002). Looking for lumps: Boosting and bagging for density estimation. *Computational Statistics and Data Analysis*, 38, 379–392.
- Ridgeway, G., Madigan, D., Richardson, T., & O’Kane, J. (1998). Interpretable boosted naive Bayes classification. In R. Agrawal, P. Stolorz, & G. Piatetsky-Shapiro (Eds.), *Proceedings of the fourth international conference on knowledge discovery and data mining* (pp. 101–104). New York: ACM Press.
- Schapire, R. E., & Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. In P. Bartlett and Y. Mansour (Eds.), *Proceedings of the 11th annual conference on computational learning theory* (pp. 80–91). New York: ACM Press.
- Yates, J. F. (1982). External correspondence: Decompositions of the mean probability score. *Organizational Behavior and Human Performance*, 30, 132–156.

8

Principal Components and Factor Analysis

Daniel W. Apley
Texas A&M University

Introduction	194
Examples of Variation Patterns in Correlated Multivariate Data	194
Overview of Methods for Identifying Variation Patterns	197
Representation and Illustration of Variation Patterns in Multivariate Data	197
Principal Components Analysis	198
Definition of Principal Components	199
Using Principal Components as Estimates of the Variation Patterns	199
Factor Rotation	202
Capabilities and Limitations of PCA	202
Methods for Factor Rotation	203
Blind Source Separation	205
The Classic Blind Source Separation Problem	205
Blind Separation Principles	206
Fourth-Order Blind Separation Methods	208
Additional Manufacturing Applications	211
Available Software	211
Summary	212
References	212

INTRODUCTION

When the different variables in a sample of multivariate data are correlated or interrelated in some manner, this indicates that one or more distinct variation patterns are present in the data. Generally speaking, each variation pattern may be viewed as the result of some underlying phenomenon or root cause that governs the behavior of the variables over the data sample.

The data mining objective considered in this chapter is to identify as precisely as possible the nature of each distinct variation pattern, based entirely on the data sample with no a priori knowledge of the patterns. Proper identification of the patterns can provide much insight into the interdependencies and correlations between the different variables and, ultimately, the phenomena that cause the patterns.

Examples of Variation Patterns in Correlated Multivariate Data

Although we illustrate the basic concepts and tools covered in this chapter with examples from manufacturing data analysis, the methods have much broader applicability than this. Indeed, many of the basic techniques such as principal components analysis (PCA) were first developed and applied in areas such as the social sciences and econometrics. Manufacturing provides a particularly fertile ground for analyzing variation patterns in multivariate data, however, due largely to the dramatic advances in in-process measurement and data collection technology that have occurred in recent years. It is not uncommon to have hundreds, or even thousands, of different process/product variables that are measured, often for 100% of the parts that are produced. One example is printed circuit board (PCB) assembly. In PCB assembly laser-optical measurement is commonly used to obtain detailed dimensional characteristics of the wet solder paste after it is deposited onto the board during the screen printing stage. After the electronic components are placed in position on the board and the solder is cured in the reflow oven, additional dimensional characteristics of each cured solder joint are obtained via X-ray laminography (Glass & Thomsen, 1993). In state-of-the-art PCB assembly operations the solder paste is measured in-process for 100% of the boards produced, and the cured solder joints are measured for nearly 100% of the boards.

Another example is automobile body assembly, in which laser-optical measurement stations are commonly built into an assembly line at various stages, typically immediately after major subassemblies are completed. In each measurement station well over 100 key autobody dimensional characteristics distributed over the subassembly may be measured. Moreover, 100% of the autoboies produced are measured. Figure 8.1 shows the measurement layout on the rear quarter panel subassembly of an autobody. The subassembly consists of a quarter panel joined to a D-pillar reinforcement. The y/z -plane dimensional coordinates of 5 separate features are measured on each subassembly, providing a total of 10 measured variables (denoted x_1 through x_{10} in Fig. 8.1). Here, the z -direction denotes the up/down direction, and the y -direction denotes the left/right direction. The x_1/x_2 and x_5/x_6 variables are the y/z -plane coordinates of two features located on the D-pillar. The other 6 variables are the coordinates of 3 features located on the quarter panel. The measurements are obtained after the subassembly is joined to the bodyside and represent the deviation of the coordinates from their nominal positions with respect to the position of the bodyside. Although there were more than 200 dimensional coordinates measured on the entire autobody, we illustrate the methods of this chapter with only the 10 quarter panel subassembly measurements.

Figure 8.2 shows scatter plots for selected pairs of variables from a sample of 200 measured autoboies. Many of the variables are quite strongly correlated, which indicates the presence

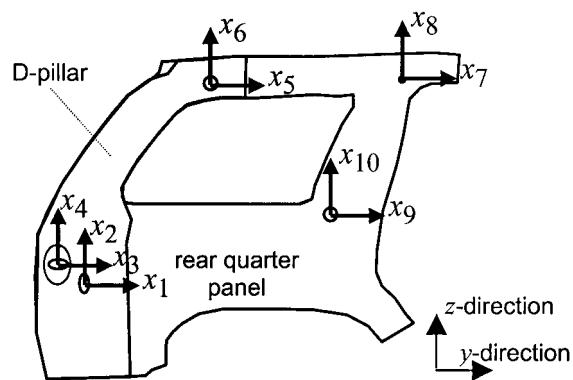


FIG. 8.1. The measurement layout on the quarter panel subassembly of an autobody.

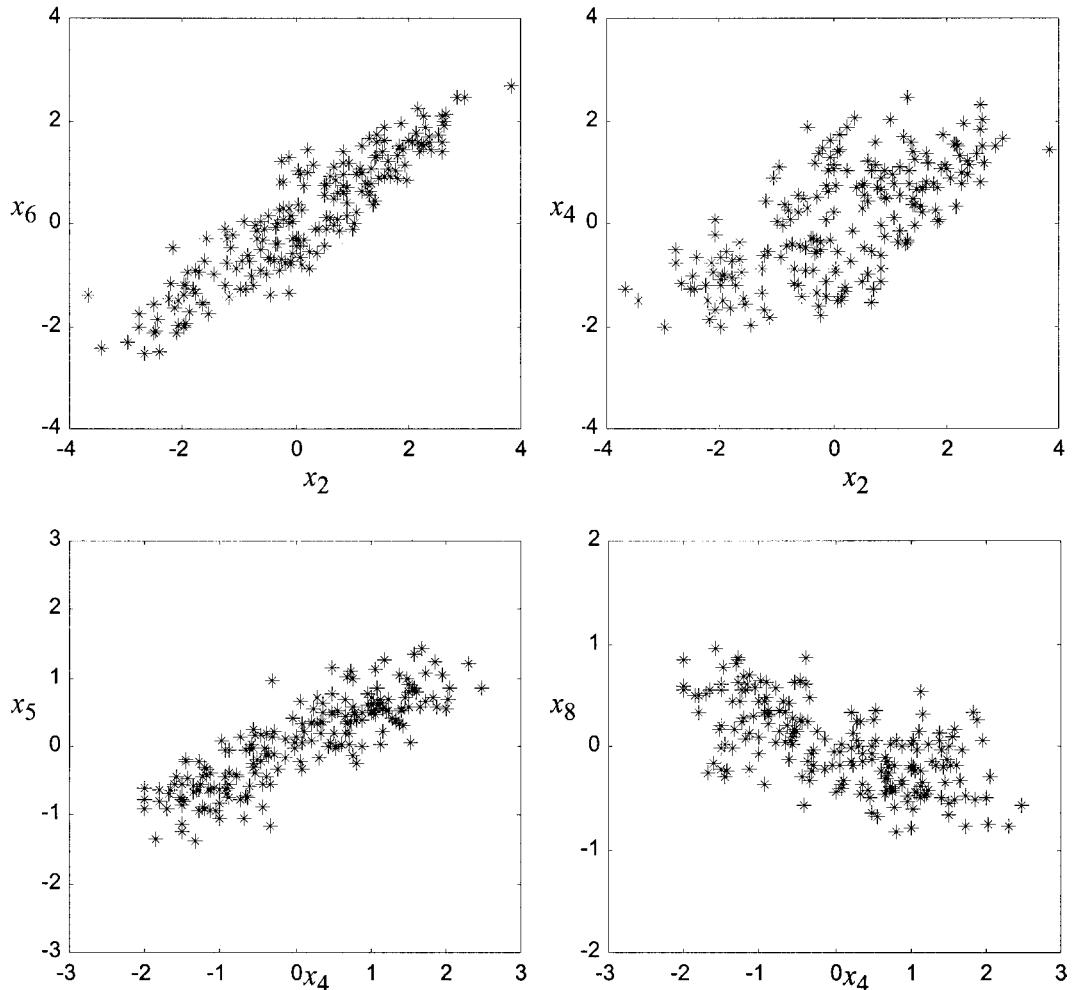


FIG. 8.2. Scatter plots of pairs of quarter panel measurements exhibiting variation patterns.

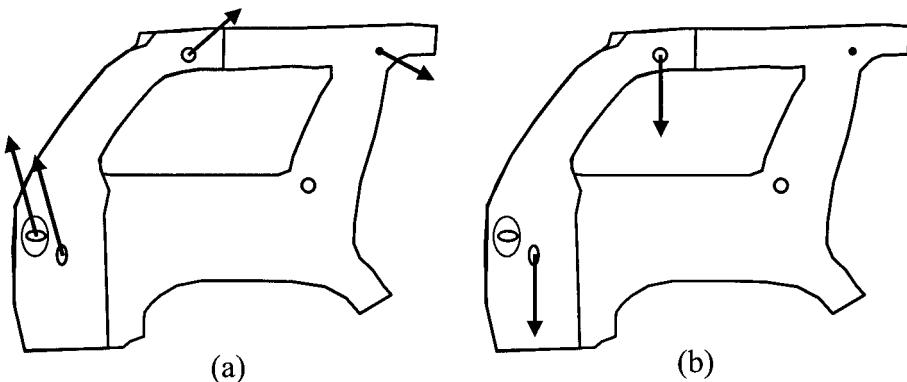


FIG. 8.3. Illustration of two linear spatial variation patterns in autobody assembly: (a) a rotation of the quarter panel subassembly and (b) a translation of the D-pillar in the z -direction.

of spatial variation patterns occurring across the different variables. To more precisely describe what is meant by a variation pattern in the context of this chapter, consider the situation depicted in Fig. 8.3. The arrows in Fig. 8.3(a) illustrate one distinct variation pattern and represent the interrelated manner in which the five measured features vary together (from quarter panel to quarter panel) due to the variation pattern. The lengths of the arrows are scaled to be proportional to the amount each feature varies. For example, suppose that on a particular quarter panel the variation pattern causes the x_1/x_2 feature to move up and to the left by one unit. Then, on the same quarter panel the variation pattern causes the x_3/x_4 feature to move up and to the left by approximately one unit, the x_5/x_6 feature to move up and to the right by approximately 0.7 units, the x_7/x_8 feature to move down and to the right by approximately 0.6 units, and the x_9/x_{10} feature is unaffected. Hence, the variation pattern depicted in Fig. 8.3(a) represents a rotation of the entire quarter panel subassembly about the x_9/x_{10} feature.

The root cause of this pattern was found to be a loose tooling element in the fixture that is used to locate the quarter panel subassembly when it is welded to the bodyside. The loose tooling element allowed the subassemblies to rotate by small amounts about the x_9/x_{10} feature (a locating hole) after being placed into the fixture. When a subassembly was subsequently clamped into place and welded to the bodyside, it retained the incorrect, rotated position. The subassemblies rotated clockwise on some autobodyes and counterclockwise on others. When the sample of 200 autobodyes was measured, the distinct “rotational” variation pattern depicted in Fig. 8.3(a) was present in the data. Note that although the arrows in Fig. 8.3(a) show a clockwise rotation, they are intended to represent a *variation* pattern (as opposed to a mean shift) in which the panels rotate in both the clockwise and counterclockwise directions. Hence, if the directions of all arrows in Fig. 8.3(a) were simultaneously reversed, their interpretation would be the same.

The arrows in Fig. 8.3(b), which illustrate a different variation pattern, can be interpreted in a manner similar to that of Fig. 8.3(a) described in the preceding paragraphs. Hence, the pattern depicted in Fig. 8.3(b) represents a z -direction translation of the D-pillar with respect to the quarter panel. The root cause of this variation pattern also was related to fixturing problems. A design flaw in the fixture used to locate the D-pillar when it was joined to the quarter panel allowed the D-pillar to translate freely by small amounts in the z -direction prior to being clamped into place and welded to the quarter panel. The D-pillars translated upward on some autobodyes and downward on others. The result of this was a second distinct “translational” variation pattern in the sample of measurement data.

Overview of Methods for Identifying Variation Patterns

A reasonably generic data mining objective is to identify the nature and explain the root causes of variation patterns in large multivariate data sets with correlation between different variables. From the scatter plots in Fig. 8.2, it is clear that variation patterns are present. The high positive correlation between x_2 and x_6 hints at the presence of the translational variation pattern and/or the rotational variation pattern that were described in the preceding paragraphs and illustrated in Fig. 8.3(a) and 3(b). The positive correlation between x_4 and x_5 , coupled with the negative correlation between x_4 and x_8 , also hints at the presence of the rotational pattern. If we had no a priori knowledge of the nature of these variation patterns, however, it would be extremely difficult to identify the precise nature of the patterns from simple pairwise scatter plots.

We may consider a more quantitative approach, in which the correlation matrix (consisting of the pairwise correlation coefficients) is analyzed. The exact nature of the variation patterns may not be obvious from the correlation matrix either, because the presence of a second variation pattern often tends to obscure the nature of the first pattern. For example, suppose that only the rotational pattern shown in Fig. 8.3(a) were present. In this case x_2 and x_4 would most likely have a very high correlation coefficient. Figure 8.2 shows that x_2 and x_4 have only moderate correlation, however. The reason is the second translational pattern affects x_2 , but not x_4 , and therefore tends to reduce their correlation due to the first pattern. The situation becomes even more complex when the data set is of higher dimension. If hundreds of variables were considered, instead of the 10 variables considered in the quarter panel example, realistically there would be far too many combinations of pairwise scatter plots or correlation coefficients to inspect.

Fortunately, there are a number of effective data mining techniques for identifying the nature of variation patterns in large multivariate data sets. The methods that will be covered in this chapter were, in fact, used to identify the variation patterns illustrated in Fig. 8.3. When these patterns were shown to process operators and engineers having some knowledge of the fixture layout and tooling, the root causes discussed above were almost immediately apparent. One should keep in mind that the purpose of identifying the nature of the variation patterns in the context of data mining is generally to gain insight into the phenomena causing the patterns.

The following section of this chapter discusses a model for representing the variation patterns. A discussion of PCA, which forms the basis for most of the methods for identifying variation patterns and is quite useful in its own respect, follows that. Two sections discuss factor rotation and blind separation methods, which are intended to improve on the results of PCA and provide more accurate identification of the variation patterns. The final two sections discuss additional examples from manufacturing, and provide a brief list of available software.

REPRESENTATION AND ILLUSTRATION OF VARIATION PATTERNS IN MULTIVARIATE DATA

To develop methods for identifying variation patterns, it is essential to have a model for representing the patterns. To this end, let $\mathbf{x} = [x_1, x_2, \dots, x_n]'$ be an $n \times 1$ random vector that represents a set of n measured variables. Let $\mathbf{x}_i, i = 1, 2, \dots, N$, be a sample of N observations of \mathbf{x} . In the quarter panel assembly example, \mathbf{x} would represent the vector of all 10 measured dimensional characteristics across a given quarter panel subassembly, and N would be the number of autobodies in the sample.

It is assumed that \mathbf{x} obeys the standard linear orthogonal factor analysis model (Johnson & Wichern, 1998)

$$\mathbf{x} = \mathbf{C}\mathbf{v} + \mathbf{w}, \quad (1)$$

where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_p]$ is an $n \times p$ constant matrix with linearly independent columns. The vector $\mathbf{v} = [v_1, v_2, \dots, v_p]'$ is a $p \times 1$ zero-mean random vector with independent components, each scaled (without loss of generality, because each column of \mathbf{C} may be rescaled accordingly) to have unit variance. The interpretation is that there are p independent *variation sources* $\{v_i: i = 1, 2, \dots, p\}$ that affect the measurement vector \mathbf{x} . The effect of the i th source is represented by the variation *pattern vector* \mathbf{c}_i . Because v_i is scaled to have unit variance, \mathbf{c}_i also indicates the magnitude or severity of the i th source. In factor analysis terminology, the v_i 's are called the common factors and the elements of \mathbf{C} are called the factor loadings. The vector $\mathbf{w} = [w_1, w_2, \dots, w_n]'$ is an $n \times 1$ zero-mean random vector that is independent of \mathbf{v} and that represents the aggregated effects of measurement noise and any inherent variation not attributed to the sources. The vectors \mathbf{v} and \mathbf{w} , and thus \mathbf{x} , are assumed to be zero-mean. If not, the mean of \mathbf{x} should first be subtracted from the data.

If in the autobody assembly example of the previous section we refer to the rotational variation pattern in Fig. 8.3(a) as the first pattern, v_1 is the unit variance random variable proportional to the angle of rotation of a quarter panel subassembly. The vector \mathbf{c}_1 would indicate the pattern is a rotation of the quarter panel about the x_9/x_{10} locating hole. For illustrative purposes suppose the numerical value for \mathbf{c}_1 is $\mathbf{c}_1 = [-0.30 \ 1.06 \ -0.30 \ 1.06 \ 0.56 \ 0.5000 \ 0.58 \ -0.30 \ 0.0]'$. The elements of \mathbf{c}_1 are plotted as arrows in Fig. 8.3(a) at the locations of the features to which they correspond. The scaling is for visual convenience, and the y - and z -direction coordinates (i.e., the left/right and up/down coordinates) of each feature have been combined into a single arrow. For example, consider the arrow plotted at the x_1/x_2 feature in Fig. 8.3(a). The y -direction component of this arrow is proportional to the first element of \mathbf{c}_1 (-0.30), and the z -direction component is proportional to the second element of \mathbf{c}_1 (1.06). Likewise, the y - and z -direction components of the arrow plotted at the x_3/x_4 feature are proportional to the third element of \mathbf{c}_1 (-0.30) and fourth element of \mathbf{c}_1 (1.06), respectively, and so on.

If we refer to the translational pattern shown in Fig. 8.3(b) as the second pattern, v_2 is the unit variance random variable proportional to the amount a D-pillar translates. Again for illustrative purposes suppose the numerical value for \mathbf{c}_2 is $\mathbf{c}_2 = [0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0]'$. The elements of \mathbf{c}_2 are plotted as arrows in Fig. 8.3(b) at the locations of the features to which they correspond, similar to what was described in the preceding paragraph for \mathbf{c}_1 .

In the remainder of this chapter, identifying the variation patterns refers to estimating \mathbf{C} . The autobody assembly example illustrates not only the applicability of the linear model in Equation 1, but also how the results of estimating \mathbf{C} could be used to gain insight into the root causes of the variation patterns. Suppose one of the methods to be described produced estimates of \mathbf{c}_1 and \mathbf{c}_2 that, when plotted on a figure of the autobody, appeared as in Fig. 8.3(a) and 3(b). \mathbf{c}_1 would most likely be interpreted as a rotation of the quarter panel subassembly about the x_9/x_{10} locating hole, and \mathbf{c}_2 as an up/down translation of the D-pillar. As mentioned above, when this was presented to process operators and engineers, the root causes of the variation patterns became evident.

PRINCIPAL COMPONENTS ANALYSIS

All of the methods discussed in this chapter utilize PCA, which involves analyzing the eigenvectors and eigenvalues of the covariance matrix of \mathbf{x} (Jackson, 1980) as the first step. I describe the methods in the context that the true covariance matrix of \mathbf{x} , defined as $\Sigma_x = E[\mathbf{x}\mathbf{x}']$ is known.

Here, $E[\cdot]$ denotes the expected value, and $'$ denotes the transpose of a matrix or vector. In the definition of Σ_x , I have left out the mean of \mathbf{x} , because it is assumed to be zero. In practice, one must work with an estimate of Σ_x obtained from the sample of data. To implement the methods discussed in this chapter, all distributional parameters would be replaced by their sample estimates. In particular, one would use the sample covariance matrix

$$\hat{\Sigma}_x = \frac{1}{N-1} \sum_{i=1}^N (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})'$$

where $\bar{\mathbf{x}}$ is the sample average of the N observations of \mathbf{x} .

Definition of Principal Components

A more detailed presentation of the following results can be found in Johnson & Wichern (1998) or Jackson (1980). Let $\{\mathbf{z}_i: i = 1, 2, \dots, n\}$ denote an orthonormal set of eigenvectors of Σ_x , and let $\{\lambda_i: i = 1, 2, \dots, n\}$ denote the corresponding eigenvalues, arranged in descending order. Furthermore, let $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n]$ be the $n \times n$ orthogonal matrix constructed from the eigenvectors and $\Lambda = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ be the $n \times n$ diagonal matrix constructed from the eigenvalues. Define the $n \times 1$ vector $\mathbf{u} = \mathbf{Z}'\mathbf{x}$. In other words, the i th element of \mathbf{u} is the linear combination $u_i = \mathbf{z}_i' \mathbf{x}$. Because $\mathbf{u} = \mathbf{Z}'\mathbf{x}$ and \mathbf{Z} is an orthogonal matrix, it follows that \mathbf{x} can be represented as

$$\mathbf{x} = \mathbf{Z}\mathbf{u} = \sum_{i=1}^n \mathbf{z}_i u_i. \quad (2)$$

This can be viewed simply as using the eigenvectors as a different (orthonormal) bases for representing \mathbf{x} . The component of \mathbf{x} along the direction of the eigenvector \mathbf{z}_i is just u_i , which is referred to as the i th principal component of \mathbf{x} . The significance of this PCA representation of \mathbf{x} stems from the basic results (Johnson & Wichern, 1998) that the principal components $\{u_i: i = 1, 2, \dots, n\}$ are all uncorrelated and that the variance of u_i is λ_i . Because we have arranged the eigenvalues in decreasing order, it follows that the component of \mathbf{x} with largest variance is the component u_1 along the \mathbf{z}_1 direction. The component of \mathbf{x} with next largest variance is the component u_2 along the \mathbf{z}_2 direction, and so on. In fact, one can show that the PCA representation is optimal in the sense that there is no unit-norm linear transformation of \mathbf{x} with larger variance than the first principal component $u_1 = \mathbf{z}_1' \mathbf{x}$. Moreover, of all unit-norm linear transformations of \mathbf{x} that are *uncorrelated* with u_1 , there is none that has larger variance than the second principal component $u_2 = \mathbf{z}_2' \mathbf{x}$. Continuing sequentially, of all unit-norm linear transformations of \mathbf{x} that are *uncorrelated* with $\{u_1, u_2, \dots, u_{i-1}\}$ there is none that has larger variance than the i th principal component $u_i = \mathbf{z}_i' \mathbf{x}$. Hence, the PCA representation of \mathbf{x} aggregates as much variance as possible into the fewest number of (uncorrelated) components.

Using Principal Components as Estimates of the Variation Patterns

There is a close relationship between the PCA representation in Equation 2 and the model in Equation 1 that we have used to represent variation patterns. The relationship is most apparent when there are only a few dominant eigenvalues. Suppose only the first (say) $p \ll n$ eigenvalues are large, and the remaining eigenvalues are all close to zero. Because the variance of u_i is λ_i ,

and λ_i is small for $i > p$, we can neglect the last $n-p$ terms in the summation in Equation 2 and approximate x as

$$\mathbf{x} = \sum_{i=1}^n z_i u_i \cong \sum_{i=1}^p z_i u_i = \sum_{i=1}^p (\lambda_i^{1/2} z_i) (\lambda_i^{-1/2} u_i). \quad (3)$$

Now suppose we neglect the noise term w , so that Equation 1 becomes $\mathbf{x} \cong \sum_{i=1}^p c_i v_i$. In this case, we see that Equation 3 is of identical structure to the model in Equation 1 with c_i corresponding to the scaled eigenvector $\lambda_i^{1/2} z_i$, and v_i corresponding to the scaled principal component $\lambda_i^{-1/2} u_i$. Moreover, $\{\lambda_i^{-1/2} u_i : i = 1, 2, \dots, p\}$ are all unit variance and uncorrelated, just as the variation sources $\{v_i : i = 1, 2, \dots, p\}$ are assumed to be.

Hence, PCA could be used to provide an estimate of C of the form

$$\hat{\mathbf{C}} = \mathbf{Z}_p \Lambda_p^{1/2}, \quad (4)$$

where $\mathbf{Z}_p = [z_1, z_2, \dots, z_p]$ and $\Lambda_p = \text{diag}\{\lambda_1, \lambda_2, \dots, \lambda_p\}$ are formed from only the p dominant eigenvalue/eigenvector pairs.

Example 1. To illustrate the use and interpretation of PCA, consider the sample of 200 autobody from which the scatter plots in Fig. 8.2 were constructed. The eigenvalues of $\hat{\Sigma}_x$ are plotted in descending order in Fig. 8.4. The first two eigenvalues were 4.57 and 1.15, and the remaining eigenvalues were all between 0.05 and 0.10. It was therefore concluded that there were two dominant eigenvalues and thus $p = 2$ variation sources. In this case

$$\Lambda_p = \begin{bmatrix} 4.57 & 0 \\ 0 & 1.15 \end{bmatrix}.$$

The average of the smallest eight eigenvalues was 0.086, which serves as an estimate of σ^2 (see the following section). The eigenvectors associated with the two dominant eigenvalues

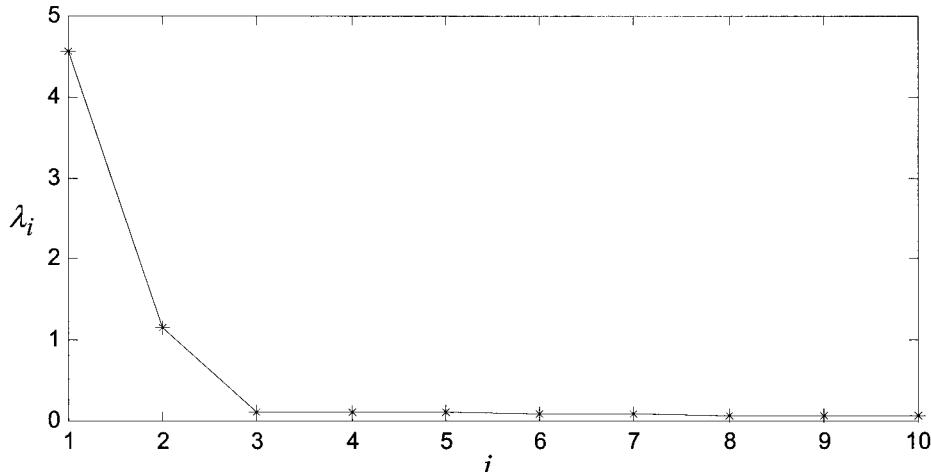


FIG. 8.4. Plot of the eigenvalues (in descending order) from PCA for the autobody assembly example.

were

$$\mathbf{Z}_p = \begin{bmatrix} -.10 & .18 \\ .68 & .27 \\ -.11 & .19 \\ .41 & -.57 \\ .21 & -.33 \\ .48 & .53 \\ .22 & -.35 \\ -.11 & .12 \\ -.01 & -.01 \\ .01 & .01 \end{bmatrix}$$

Using Equation 4, the estimate $\hat{\mathbf{C}} = \mathbf{Z}_p \Lambda_p^{1/2}$ can be calculated. Note that the estimates of \mathbf{c}_1 and \mathbf{c}_2 are nothing more than scaled (by the square root of the eigenvalues) versions of \mathbf{z}_1 and \mathbf{z}_2 . These PCA estimates of the variation pattern vectors \mathbf{c}_1 and \mathbf{c}_2 are shown in Table 8.1 and illustrated in Fig. 8.5(a) and 5(b), respectively. In Fig. 8.5, the estimates of \mathbf{c}_1 and \mathbf{c}_2 were plotted in the same manner that \mathbf{c}_1 and \mathbf{c}_2 were plotted in Fig. 8.3 (refer to the previous section for details). For example, the y - and z -direction components of the arrow plotted at the x_1/x_2 feature in Fig. 8.5(a) are (proportional to) the first two elements of the estimate of \mathbf{c}_1 (-0.22 and 1.46).

Although the estimate of \mathbf{c}_1 shown in Fig. 8.5(a) resembles a distorted version of a rotation, the estimate of \mathbf{c}_2 shown in Fig. 8.5(b) differs substantially from the translational pattern vector shown in Fig. 8.3(b). As such, the interpretation of the variation patterns would most likely have been misleading and not offered much insight into the fixture-related root causes. As will be shown in the following section, the reason is that when multiple variation sources are present (in this example there were two sources) the PCA estimate Equation 4 will in general differ from the true \mathbf{C} . This is somewhat obvious given that the eigenvectors are orthogonal. If the true \mathbf{C} is such that its columns are not orthogonal (as is the case in this example, by inspection of Fig. 8.3) the PCA estimate Equation 4 cannot possibly equal \mathbf{C} . The following two sections discuss methods that are intended to correct this shortcoming of PCA when multiple variation sources are present.

TABLE 8.1
Estimated Variation Pattern Vectors for the Quarter Panel Example Using Four Different Methods

	PCA		Varimax		Apley and Shi (2001)		Blind Separation	
	\mathbf{c}_1	\mathbf{c}_2	\mathbf{c}_1	\mathbf{c}_2	\mathbf{c}_1	\mathbf{c}_2	\mathbf{c}_1	\mathbf{c}_2
x_1	-0.22	0.19	-0.29	-0.04	-0.29	0.02	-0.29	0.03
x_2	1.46	0.28	0.74	1.29	1.02	1.06	1.04	1.04
x_3	-0.24	0.20	-0.31	-0.05	-0.31	0.02	-0.31	0.02
x_4	0.88	-0.61	1.04	0.27	1.06	0.02	1.05	0.00
x_5	0.45	-0.35	0.56	0.10	0.56	-0.02	0.55	-0.04
x_6	1.04	0.57	0.24	1.16	0.52	1.04	0.54	1.03
x_7	0.48	-0.38	0.60	0.12	0.60	-0.02	0.60	-0.03
x_8	-0.24	0.13	-0.25	-0.10	-0.26	-0.03	-0.26	-0.03
x_9	-0.01	-0.01	0.00	-0.01	-0.01	-0.01	-0.01	-0.01
x_{10}	0.02	0.01	0.00	0.02	0.01	0.02	0.01	0.02

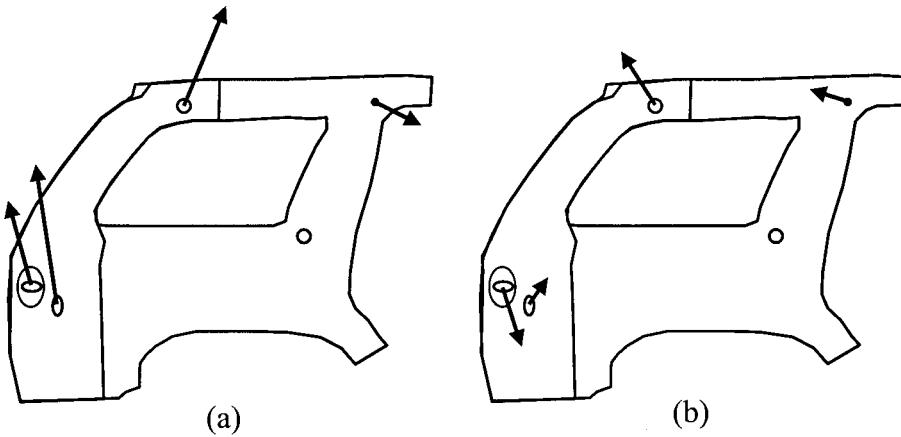


FIG. 8.5. Illustration of the scaled eigenvectors from PCA, as estimates of the first variation pattern (a) and second variation pattern (b).

PCA is still an important tool, however, because it forms the basis for the more advanced methods discussed in the subsequent sections. Moreover, when there is only a single variation source, the PCA estimate $\lambda_1^{1/2}z_1$ does coincide with the pattern vector c_1 . This also will be shown in the following section.

FACTOR ROTATION

As was demonstrated in Example 1 in the previous section, the PCA estimates of the variation patterns do not necessarily coincide with the true C . The factor rotation methods discussed in this section are techniques for rotating (forming orthogonal linear combinations of) the eigenvectors from PCA in order to obtain more “interpretable” estimates of the variation patterns.

Capabilities and Limitations of PCA

To better understand the limitations of PCA and the motivation for factor rotation, we must look at the covariance structure for x that Equation 1 implies. Because the covariance of x depends on the covariance of w (denoted Σ_w), this requires we assume a particular structure for Σ_w . In the factor analysis literature (Jackson, 1981; Johnson & Wichern, 1998) it is usually assumed that the covariance of w is $\Sigma_w = \text{diag}\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$, a diagonal matrix. Strictly speaking, the interpretations throughout the remainder of this chapter are valid only when either (a) the variances $\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$ of the noise variables are negligible relative to the variances due to the patterns (i.e., w is small relative to Cv), or (b) $\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$ are all equal, in which case we can write $\Sigma_w = \sigma^2\mathbf{I}$, a scalar multiple of the identity matrix. Negligible noise would be a reasonable assumption if we expected that almost all of the variability in x was due to the patterns. Although the noise is not negligible in most manufacturing applications, the assumption that $\Sigma_w = \sigma^2\mathbf{I}$ is often valid, in particular when $\{x_1, x_2, \dots, x_n\}$ are similar entities obtained via similar measurement principles.

If this is not the case and $\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$ are neither negligible nor identical, all is not lost, however. Providing that $\{\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2\}$ are known (up to a constant scale factor) or can be reliably estimated, we can work with a transformed version of the data $\Sigma_w^{-1/2}x = \Sigma_w^{-1/2}Cv + \Sigma_w^{-1/2}w$. In this case the transformed noise vector $\Sigma_w^{-1/2}w$ does have covariance

equal to a scalar multiple of the identity matrix. This approach, as well as strategies for estimating the noise variances, are discussed in more detail in Apley and Lee (2003). Unless otherwise noted, the remainder of this chapter assumes that either $\Sigma_w = \sigma^2\mathbf{I}$ or that the data has been transformed so that this is the case.

The reader is referred to Apley and Shi (2001) for details on the following derivations. From the model structure and assumptions, the covariance of \mathbf{x} is

$$\Sigma_x = E[(\mathbf{C}\mathbf{v} + \mathbf{w})(\mathbf{C}\mathbf{v} + \mathbf{w})'] = \mathbf{CC}' + \sigma^2\mathbf{I}. \quad (5)$$

It follows from Equation 5 that the eigenvalues are $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p > \sigma^2 = \lambda_{p+1} = \lambda_{p+2} = \dots = \lambda_n$. Thus, the number of sources p is given by the number of dominant eigenvalues, and σ^2 is equal to any of the smallest $n-p$ eigenvalues. When applying PCA to the sample covariance matrix, it may not be clear how many eigenvalues are dominant. Apley and Shi (2001) discussed various methods for estimating p . The average of the $n-p$ smallest eigenvalues can then be used as an estimate of σ^2 .

Σ_x can also be expressed in terms of its PCA decomposition:

$$\Sigma_x = \sum_{i=1}^n \lambda_i z_i z_i' = \sum_{i=1}^p (\lambda_i - \sigma^2) z_i z_i' + \sigma^2 \sum_{i=1}^n z_i z_i' = \mathbf{Z}_p [\Lambda_p - \sigma^2 \mathbf{I}] \mathbf{Z}_p' + \sigma^2 \mathbf{I} \quad (6)$$

For the covariance structures in Equations 5 and 6 to be consistent, \mathbf{C} must be of the form

$$\mathbf{C} = \mathbf{Z}_p [\Lambda_p - \sigma^2 \mathbf{I}]^{1/2} \mathbf{Q}, \quad (7)$$

for some $p \times p$ orthogonal matrix \mathbf{Q} . For the specific situation in which there is only a single variation source ($p = 1$), $\mathbf{C} (= \mathbf{c}_1)$ can be uniquely determined from $\mathbf{c}_1 = z_1 [\lambda_1 - \sigma^2]^{1/2}$, because $\mathbf{Q} = \pm 1$ is the only 1×1 orthogonal matrix. For the more general situation in which multiple sources are present, however, \mathbf{Q} cannot be determined from the covariance information used in PCA. The reason is that any orthogonal matrix that is assumed for \mathbf{Q} will result in a covariance structure in Equation 5 that is consistent with Equation 6. If one uses some incorrect $p \times p$ orthogonal matrix \mathbf{U} instead of \mathbf{Q} , the estimate

$$\hat{\mathbf{C}} = \mathbf{Z}_p [\Lambda_p - \sigma^2 \mathbf{I}]^{1/2} \mathbf{U} \quad (8)$$

will differ from the true \mathbf{C} , and the results may be noninterpretable. Note that the PCA estimate Equation 4 is exactly the estimate Equation 8 with $\mathbf{U} = \mathbf{I}$, aside from the inclusion of the $\sigma^2 \mathbf{I}$ term in Equation 8.

Although \mathbf{Q} cannot be uniquely determined from the covariance information in PCA, PCA does provide a great deal of information and is the first step for all of the methods discussed in this chapter. Specifically, PCA can be used to find p , \mathbf{Z}_p , Λ_p , and σ^2 , so that the remainder of the problem reduces to estimating the $p \times p$ orthogonal matrix \mathbf{Q} . If this can be accomplished, we can then obtain an estimate of \mathbf{C} from Equation 8 with \mathbf{U} equal to the estimate of \mathbf{Q} .

Methods for Factor Rotation

The objective of factor rotation is to find the orthogonal matrix \mathbf{U} that provides the clearest interpretability of the resulting estimate of \mathbf{C} in Equation 8. The term “rotation” stems from the fact that the postmultiplication of $\mathbf{Z}_p [\Lambda_p - \sigma^2 \mathbf{I}]^{1/2}$ by an orthogonal matrix in Equation 8 represents a rotation of its rows. Most factor rotation methods determine \mathbf{U} by making either

implicit or explicit assumptions regarding the structure of \mathbf{C} . The most popular factor rotation method is the *varimax* method, which is intended to produce an estimate of \mathbf{C} the elements of which are either large in magnitude or small in magnitude, with as few moderate sized elements as possible (Johnson & Wichern, 1998). Because for any valid estimate of \mathbf{C} the sum of the squares of the elements of any one of its rows must be a fixed quantity (equal to the variance of the corresponding element of \mathbf{x} , minus σ^2), the varimax method seeks an estimate of \mathbf{C} the structure of which is as close as possible to what we will refer to as the ideal varimax structure:

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_{1,1} & & \\ & \mathbf{c}_{2,2} & \\ & & \ddots \\ & & & \mathbf{c}_{p,p} \end{bmatrix} \quad (9)$$

where $\mathbf{c}_{i,i}$ is an $n_i \times 1$ vector with $\sum_{i=1}^p n_i = n$. This assumes an appropriate reordering of the elements of \mathbf{x} and the variation sources. Hence, the ideal varimax structure is that the p variation sources affect p disjoint subsets of the variables in \mathbf{x} . The first source affects only the first n_1 variables $\{x_1, x_2, \dots, x_{n_1}\}$, and its effects on these variables are given by $\mathbf{c}_{1,1}$. The second source affects only the n_2 variables $\{x_{n_1+1}, x_{n_1+2}, \dots, x_{n_1+n_2}\}$ with its effects given by $\mathbf{c}_{2,2}$, and so on.

The motivation for the varimax criterion (and a host of other similar criteria) is that the variation sources are easiest to interpret when each one is associated with its own distinct set of variables. This motivation is best justified in social science applications, where there may be no real underlying physics that dictate the structure of the true \mathbf{C} . In this case any reasonable interpretation of the variation patterns may suffice. In manufacturing applications, however, there will generally be some underlying process physics that dictate the structure of \mathbf{C} , as with the fixture-related variation shown in Fig. 8.3 for the autobody assembly example. If the structure of the true \mathbf{C} differs from the ideal varimax structure in Equation 9, the varimax estimate will most likely differ from \mathbf{C} and may not yield the correct interpretation of the patterns.

Given that there are many situations in which \mathbf{C} will not possess the ideal varimax structure, Apley and Shi (2001) developed a method in which \mathbf{C} is assumed to have only the ragged lower triangular structure

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_{1,1} & & & & \\ \mathbf{c}_{2,1} & \mathbf{c}_{2,2} & & & \\ \mathbf{c}_{3,1} & \mathbf{c}_{3,2} & \mathbf{c}_{3,3} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \mathbf{c}_{p,1} & \mathbf{c}_{p,2} & \mathbf{c}_{p,3} & \cdots & \mathbf{c}_{p,p} \end{bmatrix}, \quad (10)$$

where $\mathbf{c}_{i,j}$ is an $n_i \times 1$ vector with $\sum_{i=1}^p n_i = n$. Comparing Equations 9 and 10, it is clear that the ideal varimax structure is a rather restrictive special case of the structure assumed in Apley and Shi (2001). Hence, the method of Apley and Shi (2001) would be expected to produce a more accurate estimate of \mathbf{C} than the varimax method in many situations. Consider the autobody assembly example and the two variation patterns depicted in Fig. 8.3. Because the second pattern shown in Fig. 8.3(b) affects only the two variables x_2 and x_6 , if we reorder the elements of \mathbf{x} via $\mathbf{x} = [x_1, x_3, x_4, x_5, x_7, x_8, x_2, x_6]'$ it follows that \mathbf{C} possesses the structure in Equation 10. Because the first pattern shown in Fig. 8.3(a) also affects x_2 and x_6 , however, \mathbf{C} does not possess the ideal varimax structure in Equation 9.

Although due to space limitations I cannot describe the method of Apley and Shi (2001) in detail, it is fairly straightforward to implement. As discussed in a later section of this chapter,

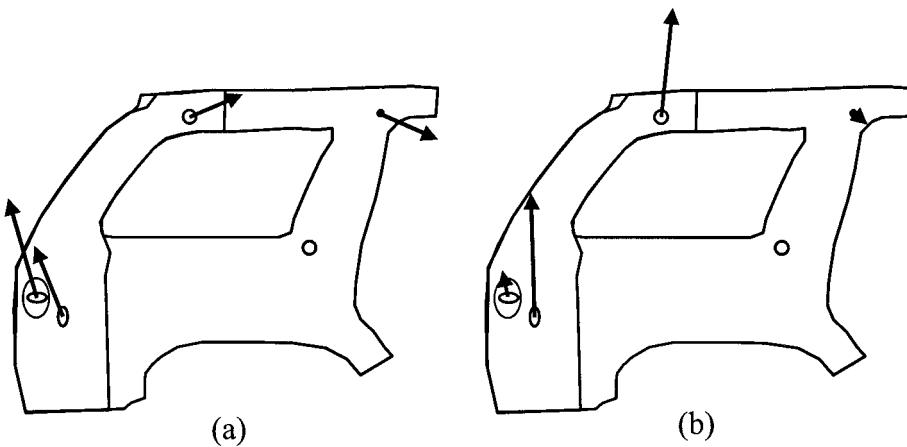


FIG. 8.6. Illustration of the varimax factor rotation estimates of the first variation pattern (a) and second variation pattern (b).

the varimax algorithm is available in many commercial statistical software packages. The estimates of c_1 and c_2 using the varimax and the Apley and Shi (2001) methods are also shown in Table 8.1. When illustrated on a figure of the quarter panel, the estimates from the method of Apley and Shi (2001) were indistinguishable from the pattern vectors shown in Fig. 8.3. The varimax estimates are illustrated in Fig. 8.6. Note that the varimax estimates are closer to the pattern vectors shown in Fig. 8.3 than are the PCA estimates. Note also that they are not as close as the estimates using the Apley and Shi (2001) method. This is reasonable since the \mathbf{C} matrix illustrated in Fig. 8.3 possesses the assumed structure of Apley and Shi (2001) but not the ideal varimax structure.

BLIND SOURCE SEPARATION

The Classic Blind Source Separation Problem

Blind source separation is a term used to describe a number of related signal processing problems in which there is an array of n spatially distributed sensors, each of which picks up signals from p distinct signal-emitting sources (Cardoso, 1998; Haykin, 2000). Applications include radar and sonar signal processing, biomedical (e.g., EEG, EKG, fetal heartbeat) and geophysical signal monitoring, wireless communications, and speaker localization. Although the typical signal processing application differs in many respects from the data mining applications considered in this chapter, the model that is utilized to represent the relationship between the source signals and the sensor signals is of identical structure to Equation 1. Specifically, the sensor signals $\{x_i: i = 1, 2, \dots, n\}$ are each assumed to be a linear combination of the source signals $\{v_j: j = 1, 2, \dots, p\}$ plus additive noise.

The blind source separation objective is typically to estimate (or separate) each of the p source signals using only a data sample of sensor signal observations, with no a priori knowledge of the source/sensor relationship given by \mathbf{C} . To accomplish this, it is first necessary to blindly estimate \mathbf{C} from the data sample, which gives rise to the term “blind source separation.” Once an estimate of \mathbf{C} is obtained, standard linear regression may be used to estimate the source signals.

Blind Separation Principles

Because the classic blind source separation model is identical to Equation 1, many of the blind separation methods apply directly to the data mining objective of identifying the variation pattern vectors in \mathbf{C} . Although there are many approaches to blind separation, this section discusses only a single class of methods, sometimes referred to as fourth-order methods. Even within this class, there are many variants. Rather than give a comprehensive survey of the different variants, I focus on a single method that illustrates the main principles and that is relatively straightforward to implement. More comprehensive discussions can be found in Apley and Lee (2003), Cardoso (1998), Hyvärinen (1999), and Hyvärinen and Oja (2000). These references also provide more detailed background on the blind separation problem in sensor array signal processing.

As in factor rotation, the first step in blind separation is to use PCA to find p , \mathbf{Z}_p , Λ_p , and σ^2 , and the remainder of the problem reduces to finding \mathbf{Q} . The estimate of \mathbf{C} is then given by Equation 8 with $\mathbf{U} = \mathbf{Q}$. As the name suggests, fourth-order methods utilize fourth-order statistics to uniquely estimate \mathbf{Q} under the specific assumptions that (a) no more than one of the p sources follows a Gaussian distribution, and (b) the noise does follow a Gaussian distribution (or, alternatively, the noise is negligible). Fourth-order methods can be derived as approximate maximum likelihood estimation (MLE) methods (Cardoso, 1998). In addition to the above assumptions, exact MLE methods typically assume that some additional characteristics of the source distributions are known (e.g., that the sources follow uniform distributions). In this sense the fourth-order methods involve a more relaxed set of assumptions and less a priori knowledge than exact MLE methods. There also exist computationally efficient algorithms for their implementation.

Rather than working directly with the measurements \mathbf{x} , blind separation methods usually work with a transformed version with spatially whitened (uncorrelated) components. The p -length whitened vector of measurements is defined as $\mathbf{y} = \mathbf{W}^{-1}\mathbf{x}$, where the $n \times p$ matrix \mathbf{W} is defined as $\mathbf{W} = \mathbf{Z}_p[\Lambda_p - \sigma^2\mathbf{I}]^{1/2}$. The $p \times n$ matrix $\mathbf{W}^{-1} = [\Lambda_p - \sigma^2\mathbf{I}]^{-1/2}\mathbf{Z}'_p$ is a left inverse of \mathbf{W} , because $\mathbf{W}^{-1}\mathbf{W} = \mathbf{I}$. Because $\mathbf{C} = \mathbf{Z}_p[\Lambda_p - \sigma^2\mathbf{I}]^{1/2}\mathbf{Q} = \mathbf{W}\mathbf{Q}$, it follows that

$$\mathbf{y} = \mathbf{W}^{-1}\mathbf{x} = \mathbf{W}^{-1}[\mathbf{C}\mathbf{v} + \mathbf{w}] = \mathbf{Q}\mathbf{v} + \mathbf{W}^{-1}\mathbf{w}. \quad (11)$$

It can be shown that \mathbf{y} has diagonal covariance matrix $\mathbf{I} + \sigma^2[\Lambda_p - \sigma^2\mathbf{I}]^{-1}$. Hence, \mathbf{y} is spatially white, and \mathbf{W}^{-1} is referred to as the whitening matrix.

To motivate the principles behind fourth-order methods, temporarily suppose the noise is negligible so that Equation 11 becomes $\mathbf{y} = \mathbf{Q}\mathbf{v}$. Let \mathbf{U} be an arbitrary $p \times p$ orthogonal matrix (a potential estimate of \mathbf{Q}) and consider the transformation

$$\mathbf{u} = \mathbf{U}'\mathbf{y} = \mathbf{U}'\mathbf{Q}\mathbf{v}. \quad (12)$$

The p -length vector \mathbf{u} has uncorrelated components for *any* orthogonal \mathbf{U} , because the product $\mathbf{U}'\mathbf{Q}$ represents an orthogonal transformation and the covariance matrix of \mathbf{v} is the identity matrix. The vector \mathbf{u} will not necessarily have *independent* elements, however, unless \mathbf{U} happens to coincide with \mathbf{Q} . In this case $\mathbf{U}'\mathbf{Q}$ is the identity matrix so that $\mathbf{u} = \mathbf{v}$ does have independent elements.

Example 2. To illustrate, consider a hypothetical example where there are $p = 2$ variation sources, and that v_1 and v_2 are both uniformly distributed on the interval $[-3^{1/2}, 3^{1/2}]$. Furthermore, suppose that

$$\mathbf{Q} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (13)$$

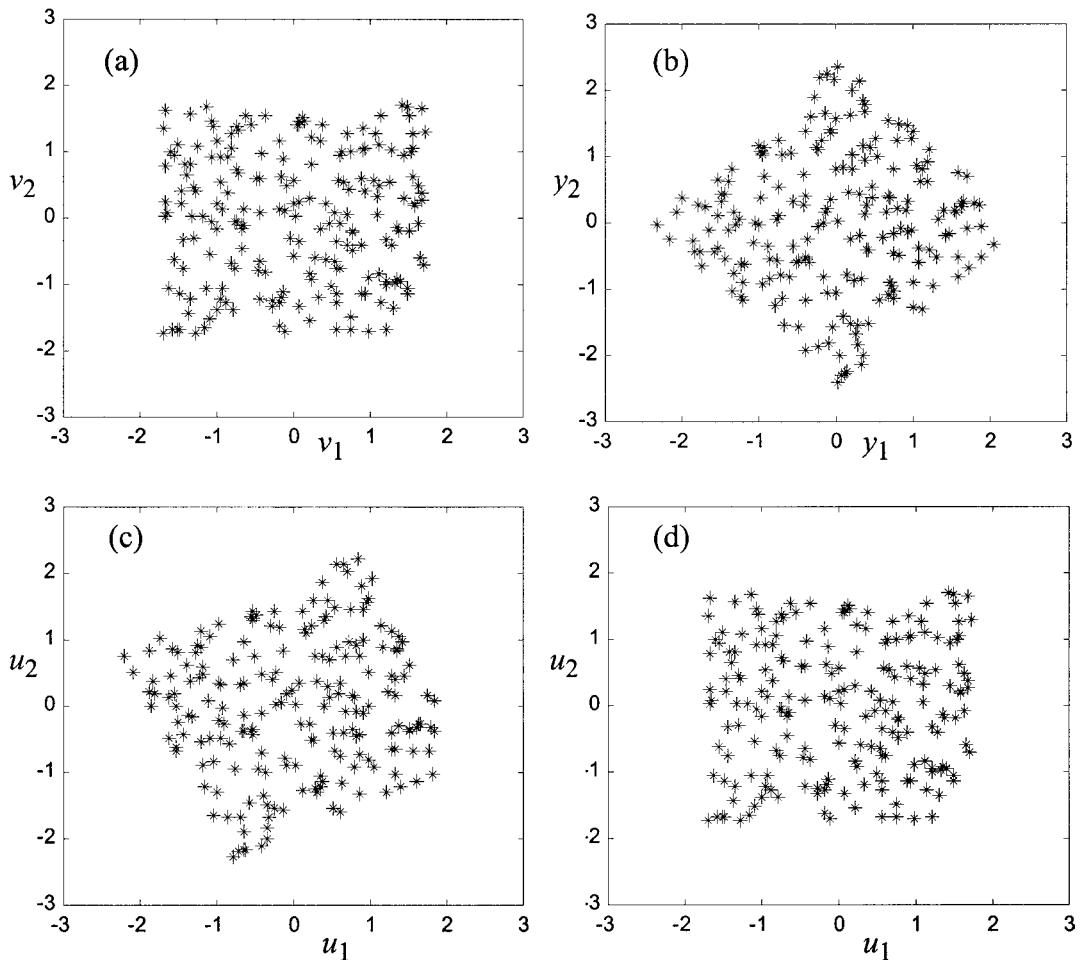


FIG. 8.7. Scatter plots of (a) the original sources v , (b) the whitened data y , (c) the incorrectly transformed data u with $\theta = 20$ degrees, and (d) the correctly transformed data $u (= v)$ with $\theta = 45$ degrees.

where $\theta = 45$ degrees. In other words, \mathbf{Q} represents a 45 degree (counter clockwise) rotation matrix. Fig. 8.7(a) and 7(b) show scatter plots of 200 typical realizations of the elements of v and the elements of $y = \mathbf{Q}v$, respectively. The elements of v are independent, whereas the elements of y are uncorrelated but clearly not independent. The vector y can be viewed as u with \mathbf{U} equal to the identity matrix. Fig. 8.7(c) and 7(d) show scatter plots of the elements of the transformed vector u for two additional \mathbf{U} matrices. In Fig. 8.7(c), \mathbf{U} was the rotation matrix of Equation 13 with $\theta = 20$ degrees, in which case the elements of u are still not independent. In Fig. 8.7(d), \mathbf{U} was the correct 45 degree rotation matrix \mathbf{Q} , in which case u equals v and its elements are now independent.

Example 2 illustrates the mechanisms behind fourth-order blind separation methods. The objective is to find the $p \times p$ orthogonal matrix \mathbf{U} such that the u vector obtained via Equation 12 has elements that are as independent as possible. This value of \mathbf{U} is then taken as the estimate of \mathbf{Q} . This bears a close relationship to PCA, in which the data is transformed to have uncorrelated, but not necessarily independent, components. Because of this, Comon (1994)

has referred to blind separation methods of this type as independent components analysis. This also illustrates why at most one source is allowed to be Gaussian, because uncorrelated Gaussian random variables are always independent as well.

Fourth-Order Blind Separation Methods

As a measure of independence of the elements of \mathbf{u} , fourth-order methods utilize the fourth-order cumulants of \mathbf{u} . For an arbitrary zero-mean random vector $\mathbf{u} = [u_1, u_2, \dots, u_p]'$, the fourth-order cumulant of its i th, j th, k th, and l th elements, $1 \leq i, j, k, l \leq p$, is defined as

$$C_{i,j,k,l}(\mathbf{u}) = E[u_i u_j u_k u_l] - E[u_i u_j]E[u_k u_l] - E[u_i u_k]E[u_j u_l] - E[u_i u_l]E[u_j u_k]. \quad (14)$$

Note that $C_{i,i,i,i}(\mathbf{u})$ is the kurtosis of u_i . Three important cumulant properties are (Rosenblatt, 1985; Stuart & Ord, 1987): (a) If \mathbf{u} is Gaussian, all of its fourth-order cumulants are zero; (b) if \mathbf{u} and \mathbf{z} are independent and of equal dimension, $C_{i,j,k,l}(\mathbf{u} + \mathbf{z}) = C_{i,j,k,l}(\mathbf{u}) + C_{i,j,k,l}(\mathbf{z})$; and (c) if the elements of \mathbf{u} are independent, all cross-cumulants of \mathbf{u} are zero. A cross-cumulant is defined as $C_{i,j,k,l}(\mathbf{u})$ with $i, j, k, l \neq i, i, i, i$.

Because \mathbf{w} is assumed Gaussian and independent of \mathbf{v} , properties (a) and (b) and Equation 11 imply that $C_{i,j,k,l}(\mathbf{U}'\mathbf{y}) = C_{i,j,k,l}(\mathbf{U}'\mathbf{Q}\mathbf{v})$, even when the noise is non-negligible. Thus, when \mathbf{U} is the desired orthogonal matrix \mathbf{Q} , $\mathbf{U}'\mathbf{Q}\mathbf{v} = \mathbf{v}$ has independent components, and all cross-cumulants of $\mathbf{U}'\mathbf{y}$ are zero by property (c). Hence, the objective of fourth-order methods is to find the orthogonal matrix \mathbf{U} that minimizes the cross-cumulants of $\mathbf{U}'\mathbf{y}$, and \mathbf{Q} is then taken to be the minimizer.

Comon (1994) suggested taking \mathbf{Q} to be the minimizer of the sum of the squares of the entire set of cross-cumulants of $\mathbf{U}'\mathbf{y}$. Cardoso and Souloumiac (1993) proposed taking \mathbf{Q} to be the minimizer of a similar criterion

$$\sum_{\substack{1 \leq i, j, k, l \leq p \\ l \neq k}} C_{i,j,k,l}^2(\mathbf{U}'\mathbf{y}), \quad (15)$$

which involves only a subset of the cross-cumulants. For both of these criteria, $\mathbf{U} = \mathbf{Q}$ is the unique minimizer if there is at most one Gaussian source (Cardoso & Souloumiac, 1993). The advantage of Equation 15, which is referred to as the joint approximate diagonalization of eigenmatrices (JADE) criterion, is that there exists a computationally efficient method for finding its minimizer. Cardoso and Souloumiac (1993) have shown that an equivalent expression for Equation 15 is

$$\sum_{\substack{1 \leq i, j, k, l \leq p \\ l \neq k}} C_{i,j,k,l}^2(\mathbf{U}'\mathbf{y}) = \sum_{\substack{1 \leq i, j, k, l \leq p \\ l \neq k}} [\mathbf{U}'\mathbf{M}(i, j)\mathbf{U}]_{k,l}^2 \quad (16)$$

where $[\cdot]_{k,l}$ denote the k th-row, l th-column element of a matrix, and each $p \times p$ cumulant matrix $\mathbf{M}(i, j)$ ($1 \leq i, j \leq p$) is defined such that $[\mathbf{M}(i, j)]_{k,l} = C_{i,j,k,l}(\mathbf{y})$.

From Equation 16, the JADE criterion is equivalent to finding the orthogonal matrix \mathbf{U} that minimizes the sum of the squares of the off-diagonal elements of the p^2 transformed cumulant matrices $\mathbf{U}'\mathbf{M}(i, j)\mathbf{U}$ ($1 \leq i, j \leq p$). Hence, the “joint diagonalization” term in the JADE acronym. The “approximate” term in the acronym comes from the fact that with sample data, no orthogonal transformation will result in all sample cross-cumulants exactly equal to zero. The sample cumulant matrices can only be approximately diagonalized in the sense that

Equation 16 is minimized. The sample cumulants are defined in the obvious way, where the expectations of the quantities in Equation 14 are replaced by their sample averages.

Fortunately, there is a computationally efficient numerical method for jointly approximately diagonalizing the set of cumulant matrices, which is based on the Jacobi technique and Givens rotations (Golub & Loan, 1989). Details of the algorithm can be found in Cardoso and Souloumiac (1993). Because of its excellent performance the JADE algorithm is often used as a benchmark for evaluating other algorithms (Reed & Yao, 1998; Wax & Sheinvald, 1997).

I emphasize that the fourth-order blind separation method requires that at most one source follows a Gaussian distribution. If this assumption fails, then \mathbf{Q} cannot be uniquely determined using fourth-order methods. In contrast, the factor rotation methods discussed in the previous section require that the true \mathbf{C} possesses a specific structure, such as the ideal varimax structure of Equation 9 or the structure of Equation 10 assumed in Apley and Shi (2001). Hence, blind separation methods tradeoff one set of assumptions (regarding the structure of \mathbf{C}) for a different set of assumptions (regarding the distribution of the sources). Which method would produce a more accurate estimate of \mathbf{C} depends on which set of assumptions are better satisfied. One advantage of the blind separation approach is that it is quite straightforward to verify whether the assumption that at most one source is Gaussian holds. This can be done using histograms of the estimated source signals, as illustrated in Example 3. As discussed in Apley and Lee (2003), it is much less straightforward, and sometimes impossible, to verify whether the true \mathbf{C} possesses the structures required by the factor rotation methods.

An additional advantage of the fourth-order method is that its assumptions can be relaxed somewhat if one uses a modified method in which the autocovariance of the data is considered in the blind separation algorithm. The details of this approach, which are omitted here for lack of space, can be found in Lee and Apley (2003). In their approach, more than one Gaussian source is allowed, as long as the autocorrelation functions for each pair of Gaussian sources differ for at least one time lag.

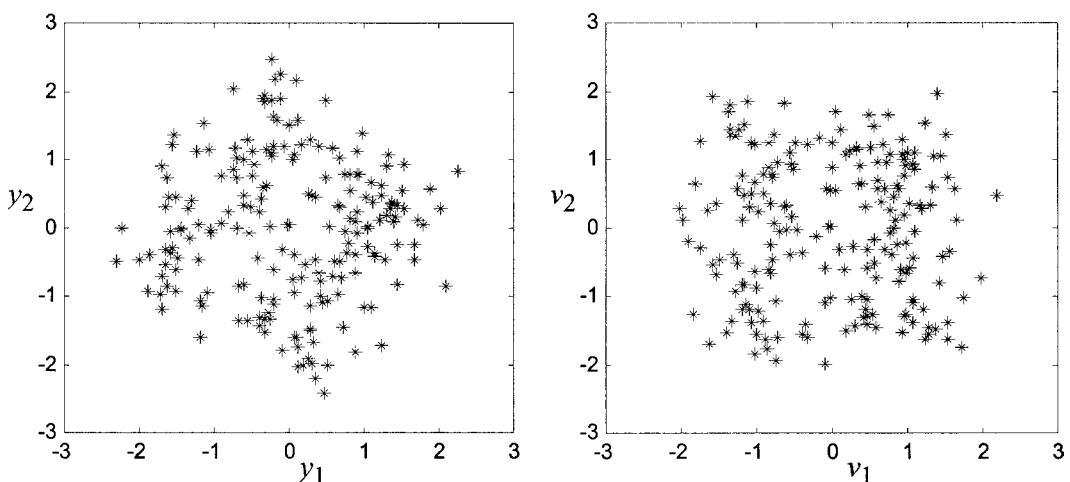


FIG. 8.8. Scatter plots of the whitened data \mathbf{y} and the estimated source signals \mathbf{v} using the blind separation method for the quarter panel example. The estimated \mathbf{Q} matrix was a rotation matrix with $\theta = -34$ degrees.

Example 3. To illustrate the application of the fourth-order method, consider a continuation of Example 1. Using the results from PCA that were discussed in a previous section, the whitened data vector \mathbf{y} for each autobody in the sample was calculated via Equation 11. A scatter plot of y_1 versus y_2 is shown in Fig. 8.8. The blind separation estimate \mathbf{U} of \mathbf{Q} was the rotation matrix in Equation 13 with $\theta = -34$ degrees, that is,

$$\mathbf{U} = \begin{bmatrix} \cos(34^\circ) & -\sin(34^\circ) \\ \sin(34^\circ) & \cos(34^\circ) \end{bmatrix} = \begin{bmatrix} 0.83 & 0.56 \\ -0.56 & 0.83 \end{bmatrix}. \quad (13)$$

Recall that this estimate of \mathbf{Q} is such that the estimated source signals $\mathbf{v} = \mathbf{U}'\mathbf{y}$ have components that are as independent as possible. Figure 8.8 also shows a scatter plot of the estimated v_1 versus v_2 . The scatter plots indicate that the estimated source signals are independent, whereas y_1 and y_2 are clearly not. Given the estimate of \mathbf{Q} , the estimates of \mathbf{c}_1 and \mathbf{c}_2 can then be obtained via Equation 8 using the values for Λ_p , \mathbf{Z}_p and σ^2 from the section headed Principal Components Analysis. Hence,

$$\begin{aligned} \hat{\mathbf{C}} &= \mathbf{Z}_p[\Lambda_p - \sigma^2 \mathbf{I}]^{1/2} \mathbf{U} \\ &= \begin{bmatrix} -.10 & .18 \\ .68 & .27 \\ -.11 & .19 \\ .41 & -.57 \\ .21 & -.33 \\ .48 & .53 \\ .22 & -.35 \\ -.11 & .12 \\ -.01 & -.01 \\ .01 & .01 \end{bmatrix} \left[\begin{bmatrix} 4.57 & 0 \\ 0 & 1.15 \end{bmatrix} - 0.086 \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix} \right]^{1/2} \begin{bmatrix} 0.83 & 0.56 \\ -0.56 & 0.83 \end{bmatrix} \\ &= \begin{bmatrix} -.10 & .18 \\ .68 & .27 \\ -.11 & .19 \\ .41 & -.57 \\ .21 & -.33 \\ .48 & .53 \\ .22 & -.35 \\ -.11 & .12 \\ -.01 & -.01 \\ .01 & .01 \end{bmatrix} \left[\begin{bmatrix} 1.75 & 1.19 \\ -0.57 & 0.85 \end{bmatrix} \right] = \begin{bmatrix} -.29 & .03 \\ 1.04 & 1.04 \\ -.31 & .02 \\ 1.05 & 0 \\ .55 & -.04 \\ .54 & 1.03 \\ .60 & -.03 \\ -.26 & -.03 \\ -.01 & -.01 \\ .01 & .02 \end{bmatrix}. \end{aligned}$$

For comparison with the other methods, these blind separation estimates of \mathbf{c}_1 and \mathbf{c}_2 also are shown in Table 8.1. When illustrated on a figure of the quarter panel, they were indistinguishable from the pattern vectors shown in Fig. 8.3. Hence, they would likely be interpreted as the rotational and translational variation patterns having as root causes the fixture-related problems discussed earlier.

Whether the assumption that at most one source is Gaussian is satisfied can be verified using histograms of the estimated source signals. From the histograms of v_1 and v_2 shown in Fig. 8.9, it appears that the assumption was satisfied, because v_1 appears to have a bimodal, non-Gaussian distribution.

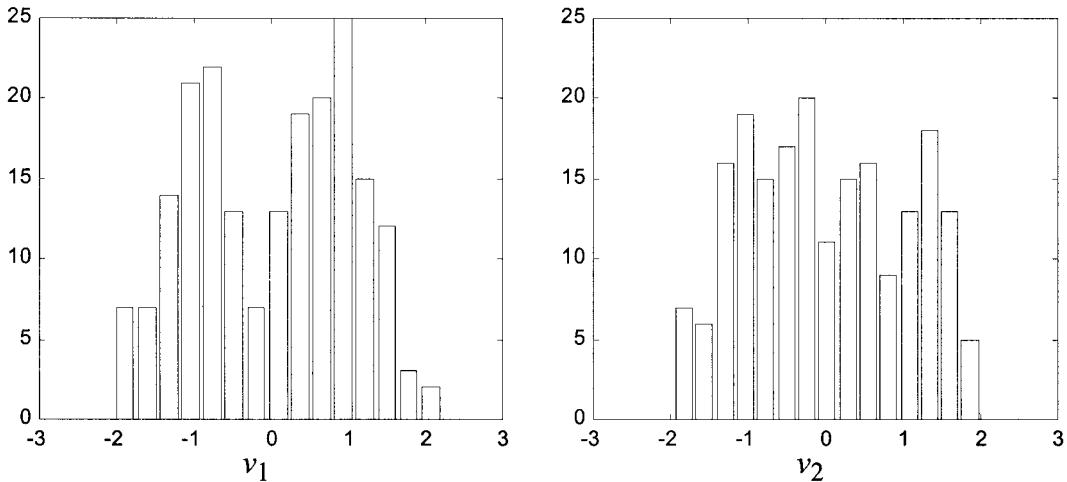


FIG. 8.9. Histograms of the estimated source signals using the blind separation method for the quarter panel example.

ADDITIONAL MANUFACTURING APPLICATIONS

The applicability of the methods described in this chapter depends predominantly on the applicability of the linear model of Equation 1 for representing the effects of variation sources. At first glance the linear structure of the model may seem somewhat restrictive, because in practice many manufacturing variation patterns would involve some degree of nonlinearity. The model is in fact a reasonably generic and widely applicable representation of the effects of variation sources, however, if one views it as the linearization of a more exact nonlinear relationship between the sources and the measurements. For example, the rotational pattern depicted in Fig. 8.1(a) is nonlinear, because the exact relationship between the elements of \mathbf{x} involves geometric functions such as sines and cosines. This relationship is closely approximated as linear, however, when small angles of rotation are involved. The interpretation of the model in Equation 1 as the linearization of a more exact nonlinear relationship is discussed in more detail in Apley and Shi (1998) and Apley and Shi (2001).

Given the interpretation of the model in Equation 1 as a linearized approximation, one can understand why the model and the factor analysis methods discussed in this chapter have found such widespread usage in a variety of areas. Recent applications in manufacturing include Barton and Gonzalez-Bareto (1996), who employ the linear model for monitoring variation in electronics assembly. Apley and Shi (2001) applied the methods to autobody assembly with much higher dimensional data sets, in which the measurements were distributed across the entire autobody. Apley and Lee (2003) have applied the methods in an automotive crankshaft manufacturing process, in which the measurement vector consisted of a large number of crankshaft dimensional characteristics.

AVAILABLE SOFTWARE

To implement PCA, one needs only to calculate the eigenvectors and eigenvalues of a sample covariance matrix. For this, a variety of statistical and linear algebra software packages can be used, such as Matlab, SPSS, or SAS. The varimax factor rotation algorithm is available

only in specialized statistical packages such as SAS, SPSS, and Minitab. The factor rotation method of Apley and Shi (2001) is not currently a part of any commercial software package. It is, however, relatively straightforward to implement using a high-level programming language such as Matlab. Although blind separation is nothing more than a form of factor rotation, the algorithms only recently were developed and are not included in commercial statistical software packages. The steps required to implement the blind separation algorithm are:

1. Estimate p , \mathbf{Z}_p , Λ_p , and σ^2 by conducting PCA on the sample covariance $\hat{\Sigma}_x$, as discussed in the sections covering principal components analysis and factor rotation.
2. Form the $p \times n$ whitening matrix $\mathbf{W}^{-1} = [\Lambda_p - \sigma^2 \mathbf{I}]^{-1/2} \mathbf{Z}_p$ and calculate the whitened data $\mathbf{y} = \mathbf{W}^{-1} \mathbf{x}$.
3. Calculate the set of fourth-order sample cumulants $C_{i,j,k,l}(\mathbf{y})$ via Equation 14 with the expectations replaced by their sample averages.
4. Calculate the set of $p \times p$ cumulant matrix $\mathbf{M}(i, j)$ ($1 \leq i, j \leq p$), defined such that $[\mathbf{M}(i, j)]_{k,l} = C_{i,j,k,l}(\mathbf{y})$.
5. Find the $p \times p$ orthogonal matrix \mathbf{U} that jointly approximately diagonalizes the set of cumulant matrixes (Matlab code for the joint approximate diagonalization is available on request from the author).
6. Take the estimate of \mathbf{C} to be $\hat{\mathbf{C}} = \mathbf{Z}_p [\Lambda_p - \sigma^2 \mathbf{I}]^{1/2} \mathbf{U}$.

SUMMARY

Although this chapter focusses on the statistical aspects of estimating variation patterns, visualization of the statistical results are no less important to understanding the nature of the patterns. In the quarter panel example used to illustrate the concepts throughout this chapter, visualization of the estimated patterns was accomplished with the aid of arrows plotted across a figure of the part. If the measurements were much more spatially dense and were obtained (for example) through optical scanning technology, then more advanced visualization methods such as surface rendering and/or software animation would be quite useful. When the data do not represent measured features distributed across some three-dimensional object such as an automobile, alternative visualization techniques must be employed.

All variation patterns were assumed to follow the linear model in Equation 1. As discussed, the model is reasonably generic if one views it as the linearization of a more exact nonlinear representation. If the nonlinearities are so pronounced that a linearization is inappropriate, however, alternative models and methods must be used. Hastie and Stuetzle (1989) developed a nonlinear counterpart to principal components, referred to as principal curves, which may be useful in these situations. Apley and Zhang (2002) combined elements of PCA with principal curve analysis to develop an approach to identifying nonlinear variation patterns with improved computational efficiency and accuracy.

REFERENCES

- Apley, D. W., & Lee, H. Y. (2003). Identifying spatial variation patterns in multivariate manufacturing processes: A blind separation approach. *Technometrics*, in press.
- Apley, D. W., & Shi, J. (2001). A factor-analysis method for diagnosing variability in multivariate manufacturing processes. *Technometrics*, 43, 84–95.

- Apley, D. W., & Shi, J. (1998). Diagnosis of multiple fixture faults in panel assembly. *ASME Journal of Manufacturing Science and Engineering*, 120, 793–801.
- Apley, D. W., & Zhang, F. (2002). Principal curve estimation via linear principal components analysis. Manuscript submitted for publication.
- Barton, R. R., & Gonzalez-Bareto, D. R. (1996). Process-oriented basis representations for multivariate process diagnostics. *Quality Engineering*, 9, 107–118.
- Cardoso, J. F. (1998). Blind signal separation: Statistical principles. *Proceedings of the IEEE*, 86, 2009–2025.
- Cardoso, J. F., & Souloumiac, A. (1993). Blind beamforming for non-Gaussian signals. *IEE Proceedings*, 140, 362–370.
- Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, 36, 287–314.
- Glass, S., & Thomsen, J. (1993). How SMT boards are assembled. *Printed Circuit Fabrication*, 16, 42–47.
- Golub, G. H., & Loan, C. F. V. (1989). *Matrix Computations*. Baltimore: Johns Hopkins University Press.
- Hastie, T., & Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84, 502–516.
- Haykin, S., (Ed.). (2000). *Unsupervised adaptive filtering*, Vol. 1. New York: Wiley.
- Hyvärinen, A. (1999). Survey on independent component analysis. *Neural Computing Surveys*, 2, 94–128.
- Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, 13, 411–430.
- Jackson, J. E. (1980). Principal components and factor analysis: Part I—principal components. *Journal of Quality Technology*, 12, 201–213.
- Jackson, J. E. (1981). Principal components and factor analysis: Part II—additional topics related to principal components. *Journal of Quality Technology*, 13, 46–58.
- Johnson, R. A., & Wichern, D. W. (1998). *Applied multivariate statistical analysis* (4th ed). Upper Saddle River, NJ: Prentice Hall.
- Lee, H. Y., & Apley, D. W. (2003). Diagnosing manufacturing variation using second-order and fourth-order statistics. *International Journal of Flexible Manufacturing Systems*, in press.
- Reed, C. W., & Yao, K. (1998). Performance of blind beamforming algorithms. In *Ninth IEEE Signal Processing Workshop* (pp 256–259). Los Alamitos, CA: IEEE Computer Society.
- Rosenblatt, M. (1985). *Stationary sequences and random fields*. Boston: Birkhauser.
- Stuart, A., & Ord, J. K. (1987). *Kendall's advanced theory of statistics*, Vol. 1, *Distribution* (5th ed.). New York: Wiley.
- Wax, M., & Sheinvald, J. (1997). A Least-squares approach to joint diagonalization. *IEEE Signal Processing Letters*, 4, 52–53.

9

Psychometric Methods of Latent Variable Modeling

Edward Ip

University of Southern California

Igor Cadez and Padhraic Smyth

University of California, Irvine

Introduction	216
Basic Latent Variable Models	217
The Basic Latent Class Model	217
The Basic Finite Mixture Model	221
The Basic Latent Trait Model	224
The Basic Factor Analytic Model	226
Common Structure	229
Extension for Data Mining	229
Extending the Basic Latent Class Model	229
Extending the Basic Mixture Model	232
Extending the Latent Trait Model	233
Extending the Factor Analytic Model	234
An Illustrative Example	236
Hierarchical Structure in Transaction Data	236
Individualized Mixture Models	237
Data Sets	238
Experimental Results	238
References and Tools	241
References	241
Tools	243
Summary	244
References	244

INTRODUCTION

This chapter describes data analysis methodologies using latent variables used in psychometrics and discusses strategies for adapting some of these methods for use in data mining. Psychometrics is the area of psychology that focuses on the measurement of human behavior. Historically, among the important psychometric tools that have been developed for understanding human behavior have been latent variable modeling, multidimensional scaling, and cluster analysis. Latent variable modeling in particular is commonly used in psychometrics and is especially well suited for analyzing and mining individual-level transaction data such as retail basket data and web browsing data.

We use transaction data as examples to illustrate the concepts and techniques of latent variable modeling. Massive transaction data sets collected from individuals (customers, Internet users, callers) are collected routinely and stored for use in a variety of applications, including telecommunications, retail commerce, and World Wide Web (WWW) site management. The mining and analysis of such data have attracted increasing interest in recent years. Data mining of transaction data can produce predictive models that have the capacity to anticipate how individuals will behave in specific settings, giving businesses the chance to improve services and cut costs. Examples include projecting the digital “trajectory” of how a user will navigate within a WWW portal, so that the interaction with the user can be more effective; profiling shoppers, so that product offers can be customized; and predicting callers’ behavior in phone calls, so that they can be efficiently routed to appropriate specialists.

Data mining methods, particularly model-based methods for individual-level data, have strong roots in a number of traditional disciplines such as statistics, econometrics, and psychometrics. We focus on a traditional psychometric area of study—latent variable models—and show how these models can be expanded for use in modern data mining. Understanding the underlying concepts can help data mining practitioners avoid spending valuable resources “reinventing the wheel.” Furthermore, understanding how these methods and tools evolved over time can help to broaden the scope of model-based data mining applications.

In the fields of psychology and the other social sciences, latent variable models grew out of the necessity to measure unobserved constructs such as human intelligence, political inclination, and specific abilities for performing certain tasks. Although these types of variables cannot be measured directly, there are often observable related variables, sometimes called manifest variables, that can be measured. There are various types of latent variable models. A useful taxonomy distinguishes four basic types of models, based on two modes of measurement (continuous and discrete) on two variables of interest (latent and manifest). Table 9.1 shows the cross-classification of the four types of models—latent class models, finite mixture models, latent trait models, and factor analysis models. Table 9.2 describes some examples of latent and manifest variables for each of the four categories.

TABLE 9.1
A Taxonomy of Latent Variable Models

	<i>Observed Variable</i>	
	<i>Discrete</i>	<i>Continuous</i>
Latent Variable	discrete	Latent class models
	continuous	Finite mixture models
		Latent trait models
		Factor analysis models

TABLE 9.2
Examples of Latent and Observed (Manifest) Variables

<i>Latent class</i>	
<i>Latent Variable (Discrete)</i>	<i>Observed Variable (Discrete)</i>
Market segment	Response to market research questionnaire (yes/no)
Concept of a group of words	Key words appearing in documents
Group of moviegoers with similar tastes	Rating on movies
<i>Mixture</i>	
<i>Latent Variable (Discrete)</i>	<i>Observed Variable (Continuous)</i>
Type of customer (routine vs. random)	Shopping pattern
Components of a digit	Handwritten digit
DNA motifs	Biopolymer DNA sequences of alphabets
<i>Latent Trait</i>	
<i>Latent Variable (Continuous)</i>	<i>Observed Variable (Discrete)</i>
Proficiency in a subject matter	Response to questions of the subject matter
Seriousness along a specific dimension of a disability	Presence or absence of symptoms
Propensity to shop a brand	Purchases under various marketing conditions
<i>Factor Analytic</i>	
<i>Latent Variable (Continuous)</i>	<i>Observed Variable (Continuous)</i>
Athletic abilities	Scores in decathlon competition
Intelligence	Scores on tests
Characterizing features of stocks	Daily stock prices

The taxonomy in Table 9.1 provides a useful framework for understanding latent variable models. In the following we first describe the basic structure and learning scheme for each model in Table 9.1. Then we examine a number of important extensions of these models and investigate strategies for adapting them for data mining.

The remainder of this chapter is organized as follows. In the next section, after a brief introduction, the data-generative mechanism and learning scheme for each of the four model types is described. The section following outlines strategies for extending the models to handle data sets of increasing complexity and size. After that, we describe a real-world example to illustrate how latent variable models can be adapted for the mining of two transaction data sets. Finally, we provide some references, a brief review of relevant tools, and a summary.

BASIC LATENT VARIABLE MODELS

The Basic Latent Class Model

Introduction

The latent class model (LCM) is a class of generative models for unsupervised learning. In a basic LCM the data can be viewed as comprising individual records, each of which is a vector of binary features. For example, the presence/absence of specific items from a shopper's market basket can be encoded into a long string of 0's and 1's (e.g., purchase shampoo? yes = 1, no = 0). In LCM modeling, the entirety of the observed (manifest) vectors is envisioned as a

diverse collection of outputs that actually are generated by multiple groups of “latent classes” (e.g., heavy buyers versus light buyers, or buyers of personal care products versus buyers of food products). Learning in LCM aims to self-organize the collection of records into a predetermined number of homogeneous groups—that is, latent classes.

Generative Mechanism

The basic LCM is specified by the mixing proportion of the classes and the marginal distributions of output features within a specific class. Suppose that \mathbf{y}_i denotes a J -dimensional feature vector of binary values (0/1) from record i , $i = 1, \dots, n$. In the example of shoppers, \mathbf{y}_i is the string of 0’s and 1’s indicating the i th shopper’s purchase. We call each unit y_{ij} a coordinate of the feature vector. Let z_{ik} , $k = 1, \dots, K$, denote the unobserved indicator of the latent classes for record i : $z_{ik} = 1$ if the i th record belongs to class k , and 0 otherwise, and $\sum_k z_{ik} = 1$. Let the probability that the output is positive (i.e., equals 1) given that the record belongs to class k , be denoted by $\pi_{jk} = p(y_{ij} = 1 | z_{ik} = 1)$, and let the proportion of the k th class be given by w_k , $\sum_k w_k = 1$. Suppose the second latent class represents a group of personal care product buyers, then π_{12} denotes how likely it is this group would purchase product 1. If this group of buyers constitutes 30% of the population, then $w_2 = 0.3$.

Table 9.3 shows a sample of vectors simulated by the generative model using the following population values of the parameters: ($w_1 = 0.2$, $w_2 = 0.8$), and $\pi_1 = (0.2, 0.1, 0.3, 0.8, 0.9)$ for the probabilities within Class 1, and $\pi_2 = (0.8, 0.9, 0.7, 0.1, 0.2)$ within Class 2. Only the nonshaded quantities in Table 9.3 are actually observed.

The generative model can be summarized by the following procedure:

1. For record $i = 1$, sample a class label (say k) from the multinomial distribution (w_1, \dots, w_K) . For example, in Table 9.3 a class label 2 for record $i = 1$ is obtained from a random draw with probability 0.8 of being 2.
2. Given the class label k , generate the first coordinate of the output feature vector with parameter $p(y_{i1} = 1 | z_{1k} = 1) = \pi_{1k}$. For example, in Table 9.3 the first “1” in $y_{i1}, i = 1$, is the result of generating a Bernoulli output (0 or 1) with probability of $\pi_{12} = 0.8$ of taking the value 1. A way to generate this Bernoulli output is to draw a uniform random number x from the interval $[0, 1]$. If $x < 0.8$, then $y_{i1} = 1$, otherwise, $y_{i1} = 0$.
3. Repeat Step 2 independently for each of the remaining $(J - 1)$ coordinates to obtain the feature vector $\mathbf{y}_1 = (y_{i1}, \dots, y_{iJ})$. For example, in Table 9.3 for record $i = 1$, the remaining coordinates 1, 1, 0, 0 are generated.
4. Repeat the above generative steps independently for each record $i = 2, \dots, n$. For example, in Table 9.3 the record $\mathbf{y}_2 = (1, 1, 0, 0, 0)$ is generated for $i = 2$.

TABLE 9.3
A Sample of Vectors Simulated from a Two-Class Latent Class Model with
 $w_1 = 0.2$, $w_2 = 0.8$, $\pi_1 = (0.2, 0.1, 0.3, 0.8, 0.9)$,
 $\pi_2 = (0.8, 0.9, 0.7, 0.1, 0.2)$

i	<i>Class k</i>	y_{i1}	y_{i2}	y_{i3}	y_{i4}	y_{i5}
1	2	1	1	1	0	0
2	2	1	1	0	0	0
3	2	1	1	1	0	1
4	1	0	0	1	1	1
5	2	0	1	1	0	0

Important Assumptions

Given a specific class, LCM assumes that the coordinates of the output features are conditionally independent. This assumption, often called local independence in the psychometric literature, implies that class membership completely explains the correlation that arises between various binary coordinates.

The conditional assumption greatly simplifies the representation of the joint probability of the J -vector feature space. For binary outputs, to fully specify the probability distribution of the joint feature space would require $2^J - 1$ parameters. Under the assumption of a K -class LCM, the number of parameters is of the order $K \times J$, which can be a considerable reduction in the number of parameters when J is large, for example, J can be $O(10^5)$ for text data when each record represents a document and each feature represents whether a word or term is present (1) or absent (0).

Learning

Learning takes place through maximizing the likelihood of observing the collection of feature vectors \mathbf{y}_i , $i = 1, \dots, n$:

$$\begin{aligned} p(\mathbf{y}) &= \prod_{i=1}^n p(\mathbf{y}_i) = \prod_{i=1}^n \sum_{k=1}^K w_k p_k(\mathbf{y}_i) \\ &= \prod_{i=1}^n \sum_{k=1}^K w_k \prod_{j=1}^J [\pi_{jk}]^{y_{ij}} [1 - \pi_{jk}]^{1-y_{ij}}. \end{aligned} \quad (1)$$

For example, given a collected data record $(1, 1, 1, 0, 0)$, the likelihood function can be computed in the following way (assuming that the parameters are set at the population values of the example in Table 9.3):

$$\begin{aligned} p((1, 1, 1, 0, 0)) &= (0.2)\{(0.2)(0.1)(0.3)(0.2)(0.1)\} + (0.8)\{(0.8)(0.9)(0.7)(0.9)(0.8)\} \\ &= 0.29. \end{aligned} \quad (2)$$

Learning in LCM aims to maximize the function in Equation 1 over (π, w) . A naive way to do this would be to search for the best estimate for each parameter and cycle through the entire set of parameters one by one until some convergence criterion is met. This method, called cyclic coordinate descent (Luenberger, 1984), is not efficient for problems with more than about a dozen parameters.

There exist a variety of efficient methods, including the expectation-maximization (EM) algorithm. The EM algorithm is an iterative learning procedure that cycles through iterations in two steps—the expectation (E-) step and the maximization (M-) step. The EM algorithm is especially useful for maximizing a likelihood function when there exist missing values in the data. For LCM, EM treats the class memberships as missing data (shaded data in Table 9.3). At each iteration EM provides a soft classification of each record to each class, given a set of provisional values for the parameters (π, w) . That is, each record i is given a weight z_{ik}^* to indicate its expected likelihood of belonging to a specific class k (1 being highest and 0 lowest, and $\sum_k z_{ik}^* = 1$). This process is called the E step.

Given the estimated weights assigned for a class—that is, the likelihood that a particular record belongs to a specific class—learning about the parameters π and w can take place in a straightforward manner. The learning step maximizes the artificially completed likelihood function and updates (π, w) , which is why this step is called the maximization or M step.

Specifically, the two-step learning process is given by:

$$\text{E-step: } z_{ik}^* = \frac{w_k p_k(\mathbf{y}_i)}{\sum_{k=1}^K w_k p_k(\mathbf{y}_i)},$$

$$\text{M-step: } w_k = \frac{1}{n} \sum_{i=1}^n z_{ik}^*,$$

$$\pi_{jk} = \frac{\sum_{i=1}^n z_{ik}^* y_{ij}}{\sum_{i=1}^n z_{ik}^*}.$$

Continuing from Example 2, soft classification of the first record leads to (assuming parameters are given by their population values):

$$z_{11} = \frac{(0.2)(0.2)(0.1)(0.3)(0.2)(0.1)}{0.29} = 0.00008, \quad (3)$$

$$z_{12} = \frac{(0.8)(0.8)(0.9)(0.7)(0.9)(0.8)}{0.29} = 0.99992, \quad (4)$$

that is, according to the current parameter values, the first record has probability close to 1 of having been generated by component 2. The E and M steps iterate until a specific convergence criterion is met. The computational complexity per iteration is $O(n K J)$. The general setup for the EM algorithm is described in chapter 6 of this volume.

Geometric Interpretation

The EM algorithm for learning in LCM has an intuitive geometric interpretation. The E and M steps can be viewed as alternating projections between a data subspace and a model subspace. To set up notation, let the possible feature patterns \mathbf{x}_u observable in a sample be indexed by $u = 1, \dots, U, U = 2^J$. For example, in Equation 2, the output $(1, 1, 1, 0, 0)$ is one of the $2^5 = 32$ possible patterns. Suppose that all of the latent classes $c_k, k = 1, \dots, K$, are observable. Then the empirical distribution q of a sample of n data points across the classes and the marginal can be specified by Table 9.4. The empirical distribution q is in fact nothing more than the histogram that assigns a probability mass of $1/n$ for each observation.

When the latent classes are known, the maximum likelihood method of estimating the class probabilities amounts to projecting q , with respect to an entropy measure of distance [the Kullback-Leibler measure, of which the distance between points p and q is $E_q \log(q/p)$], onto a space of probability distributions $\Xi = \{p_\xi : \xi \in A\}$, where A is the domain of the parameters (π, w) (see Fig. 9.1a)].

However, because the latent classes are not known, the data for the number of records with feature pattern \mathbf{y}_i within latent class k is now implicit (shaded region in Table 9.4). Only

TABLE 9.4
Distribution across Ability Levels

		Latent Class				Cumulative Frequency
		c_1	c_2	\dots	c_K	
Pattern	X_1	n_{11}	n_{12}		n_{1K}	f_1
	X_2	n_{21}	n_{22}		n_{2K}	f_2
	\vdots					
	X_U	n_{U1}	n_{U2}		n_{UK}	f_U
	Total					n

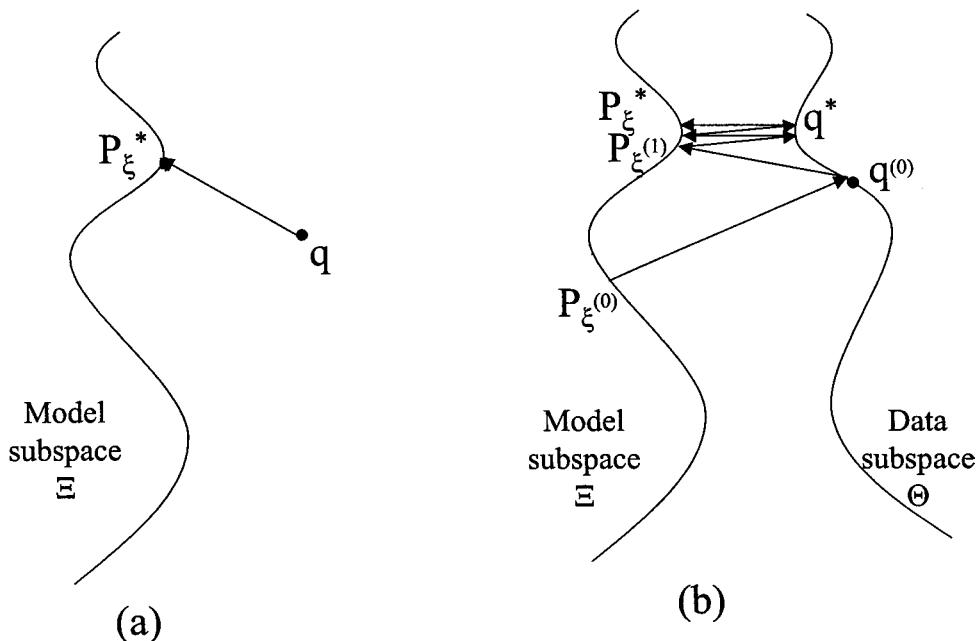


FIG. 9.1. Alternate projections of EM. The estimated model is represented by p_ξ^* . In (b), the superscript stands for iteration number.

marginal frequencies f_1, \dots, f_U are observed. Therefore, instead of a single element q in the data space, the empirical distribution of interest now includes all possible tables that give rise to the observed margins f_1, \dots, f_U . The collection of possible empirical distributions giving rise to the observed margins constitutes a data subspace Θ . To find the “shortest distance” between the two subspaces, EM alternately projects from one space to the other (see Fig. 9.1b). The E-step projects from a specific p_ξ to the data subspace, whereas the M-step uses a tentative estimate of the empirical distribution q (an artificially completed Table 9.4) to project on the model subspace. This method results in a pair of points (p_ξ^*, q^*) on the model subspace and the data subspace that are closest to each other under the entropy metric. The point p_ξ^* specifies the maximum likelihood estimate of LCM, whereas q^* specifies the expected distribution of latent classes.

The geometry of EM for other latent variable models such as mixtures is analogous to LCM.

The Basic Finite Mixture Model

Introduction

Finite mixture models are generative models that can be used to describe heterogeneity within a population when the features are continuous. Mixture models are known by different names, including unsupervised probabilistic learning in artificial intelligence, latent profile analysis in psychometrics, and intrinsic classification in philosophy. Although the term “mixture” has also been used to include LCM, here we use it only strictly to refer to models that have discrete latent variables and continuous observed features.

Perhaps the simplest case of a mixture distribution is a two-component mixture of univariate Gaussian distributions. An example of this mixture would be spending patterns on airline fares of a population of customers over specific long-distance airline routes. It is likely that one group or component of customers (say, first-class and business-class passengers) spends significantly

more than another group or component (say, economy-class passengers). Thus, the marginal distribution is one that exhibits two “bumps,” each having its own mean and variance. The “components,” analogous to the latent classes in LCM, are not directly observed.

Generative Mechanism

The formulation of the basic finite mixture model resembles Equation 1. Let \mathbf{y}_i denote a J -vector of continuous features for the i th record, and let $\mathbf{z}_i = (z_{i1}, \dots, z_{iK})$ denote the latent variable that is used to indicate which mixture component should be used to generate the observed vector. Further, let $\Phi(\mathbf{y} | k)$ denote the J -dimensional Gaussian distribution with mean μ_k and variance Σ_k for the distribution of the k th mixture component, $k = 1, \dots, K$. The generative mechanism takes place according to the following sequence:

1. For record $i = 1$, sample a class label, say k , from the multinomial distribution (w_1, \dots, w_K) .
2. Given the class label k , draw a sample \mathbf{y}_i , $i = 1$, from the multivariate distribution $\Phi(\mu_k, \Sigma_k)$.
3. Repeat the above generative steps independently for each record $i = 2, \dots, n$.

Figure 9.2 shows the distribution of a mixture of two bivariate Gaussian distributions with distinct means and covariance structures. The parameters used in the mixture are, respectively, $\mu_1 = (0, 0)$ and $\mu_2 = (2, 2)$, and the covariances Σ are diagonal matrices with 1’s and 2’s, respectively, on the diagonal. A generative sample is represented by circles (Class 1) and crosses (Class 2) in Fig. 9.2. Of course, in reality the class identities are not known a priori.

Learning

The likelihood equation of the observed data $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_n)$ is given by:

$$\begin{aligned} p(\mathbf{y}) &= \prod_{i=1}^n p(\mathbf{y}_i) \\ &= \prod_{i=1}^n \sum_{k=1}^K w_k p(\mathbf{y}_i | k), \end{aligned} \quad (5)$$

where $p(\mathbf{y}_i | k) = p_k(\mathbf{y}_i)$ is the Gaussian density $\Phi(\mu_k, \Sigma_k)$.

The goal of mixture learning is to arrive at accurate estimates of the parameters (μ_k, Σ_k) , and w_k , $k = 1, \dots, K$. We again can apply EM for learning in basic finite mixture models. The updating equations for solving the maximum of Equation 5 are specified by:

$$\begin{aligned} \text{E-step } z_{ik}^* &= \frac{w_k p_k(\mathbf{y}_i)}{\sum_{k=1}^K w_k p_k(\mathbf{y}_i)}, \\ \text{M-step } w_k &= \frac{1}{n} \sum_{i=1}^n z_{ik}^*, \\ \mu_k &= \sum_{i=1}^n z_{ik}^* \mathbf{y}_i \Bigg/ \sum_{i=1}^n z_{ik}^*, \\ \Sigma_k &= \sum_{i=1}^n z_{ik}^* (\mathbf{y}_i - \mu_k)(\mathbf{y}_i - \mu_k)^T \Bigg/ \sum_{i=1}^n z_{ik}^*, \end{aligned}$$

where z_{ik}^* denotes the expected value of z_{ik} .

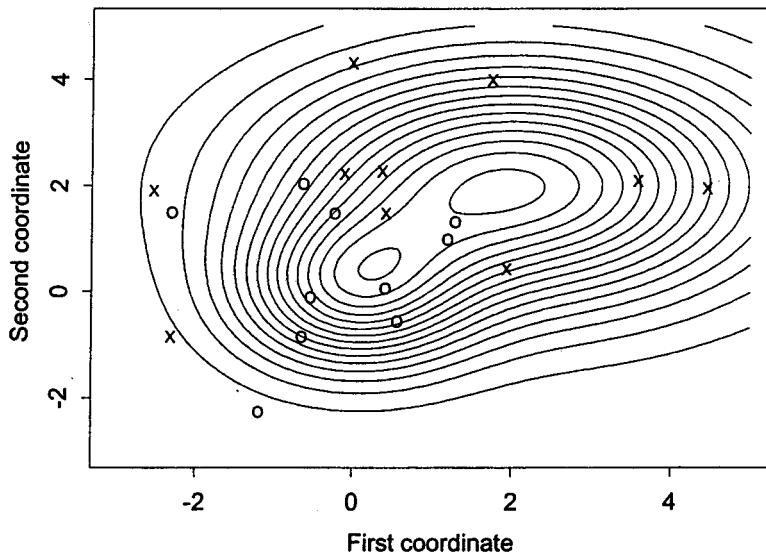


FIG. 9.2. Contour plot of the mixture of two bivariate normal distributions.

As a simple numerical example, suppose there are five data points: $\mathbf{y}_1 = (7, 8)$, $\mathbf{y}_2 = (5, 7)$, $\mathbf{y}_3 = (-1, -2)$, $\mathbf{y}_4 = (0, 1)$, $\mathbf{y}_5 = (-3, -1)$, and there are two mixture components, that is, $K = 2$. Further suppose in a certain E-step, $z_{i1}^* = 0.9, 0.8, 0.1, 0.3, 0.05$, for $i = 1, \dots, 5$. Then in the M-step,

$$w_1 = \frac{0.9 + 0.8 + \dots + 0.05}{5} = 0.43, \quad w_2 = 0.57,$$

$$\begin{aligned} \mu_1 &= \left(\frac{(0.9)(7) + (0.8)(5) + \dots + (0.05)(-3)}{0.9 + \dots + 0.05}, \frac{(0.9)(8) + (0.8)(7) + \dots + (0.05)(-1)}{0.9 + \dots + 0.05} \right) \\ &= (4.67, 5.98) \end{aligned}$$

$$\begin{aligned} \mu_2 &= \left(\frac{(0.1)(7) + (0.2)(5) + \dots + (0.95)(-3)}{0.1 + \dots + 0.95}, \frac{(0.1)(8) + (0.2)(7) + \dots + (0.95)(-1)}{0.1 + \dots + 0.95} \right) \\ &= (-0.72, 0.05) \end{aligned}$$

$$\begin{aligned} \Sigma_1 &= \frac{1}{0.9 + \dots + 0.05} \left[(0.9) \begin{pmatrix} 7 - 4.67 \\ 8 - 5.98 \end{pmatrix} \begin{pmatrix} 7 - 4.67 \\ 8 - 5.98 \end{pmatrix}^T + \dots \right. \\ &\quad \left. + (0.05) \begin{pmatrix} -3 - 4.67 \\ -1 - 5.98 \end{pmatrix} \begin{pmatrix} -3 - 4.67 \\ -1 - 5.98 \end{pmatrix}^T \right] = \begin{pmatrix} 8.22 & 8.69 \\ 8.69 & 9.65 \end{pmatrix}, \end{aligned}$$

$$\begin{aligned} \Sigma_2 &= \frac{1}{0.1 + \dots + 0.95} \left[(0.1) \begin{pmatrix} 7 + 0.72 \\ 8 - 0.05 \end{pmatrix} \begin{pmatrix} 7 + 0.72 \\ 8 - 0.05 \end{pmatrix}^T + \dots \right. \\ &\quad \left. + (0.95) \begin{pmatrix} -3 + 0.72 \\ -1 - 0.05 \end{pmatrix} \begin{pmatrix} -3 + 0.72 \\ -1 - 0.05 \end{pmatrix}^T \right] = \begin{pmatrix} 6.27 & 6.09 \\ 6.09 & 7.52 \end{pmatrix}. \end{aligned}$$

Given μ_k and Σ_k , $k = 1, 2$, one can compute the bivariate Gaussian density $p_k(\mathbf{y}_i)$, which is necessary for the next E-step.

The EM learning scheme seems deceptively simple. However, in practice finite mixture learning often has to deal with problems such as local maxima, class label switching, and the fact that certain class sizes converge to zero. Some of these issues are discussed in the references given in the References and Tools section of this chapter.

Important Assumptions

Unlike categorical output features, for which the number of parameters grows as $O(2^J)$, the number of parameters in a Gaussian distribution grows only as $O(J^2)$, and the computational complexity per EM step is $O(KnJ^2)$. The conditional independence assumption (equivalently $\Sigma_k = D_J$ for all k , where D_J is a diagonal matrix of dimension J) is sometimes evoked in Gaussian mixture models particularly when J is large.

The Basic Latent Trait Model

Introduction

Latent trait theory is mainly developed under the name of item response theory (IRT) in psychometrics. Applications of IRT models primarily have occurred in the fields of educational and psychological testing. In IRT the latent variable Z (assumed to be univariate in the basic model) could be a continuous rating scale that indicates the extent to which a person exhibits a psychological trait, or it could be a continuous score that suggests the proficiency of a student in a particular content area. The latent variable, or trait, drives multiple responses (coordinates of the feature vector \mathbf{y}_i) from a subject on a collection of “items.” An item could be a question for which the response value is scored either as correct (1) or incorrect (0), or as the presence (1) or absence (0) of an observable physiological feature. In the basic model the outcome of response to an item is binary. If items are questions from a cognitive test, the latent variable Z can be thought of as a general measure of “smartness.” Thus, the “smarter” a student (higher Z), the more likely it is that the student would respond correctly to every item. This notion of a continuum of “smartness” is typically captured in IRT by the following equation:

$$p_{ij}(z) = p(y_{ij} = 1 | z_i) = \frac{\exp(\alpha_{j1} + \alpha_{j2}z_i)}{1 + \exp(\alpha_{j1} + \alpha_{j2}z_i)}, \quad (6)$$

where y_{ij} indicates the response of the i th subject to the j th item (or the j th coordinate of the feature vector), α_{j1} is an item-specific parameter that governs how “easy” it is to get a positive response on the item, α_{j2} indicates how the slope of the response curve changes with Z , and z_i is the individual-specific measure of “smartness.” The function $y = \exp(x)/(1 + \exp(x))$ used in Equation 6 that links the probability of positive outcome and the latent variable is the well-known logistic (sigmoid) function. Figure 9.3 shows a graph of two logistic functions with distinct values of α . These curves are often called item response functions (IRF).

Generative Mechanism

Typically, the continuous latent trait variable Z is assumed to be generated from a Gaussian distribution with mean 0 and variance 1. Setting the mean to 0 and the variance to 1 avoids an identifiability problem in the location and scale of the latent trait.

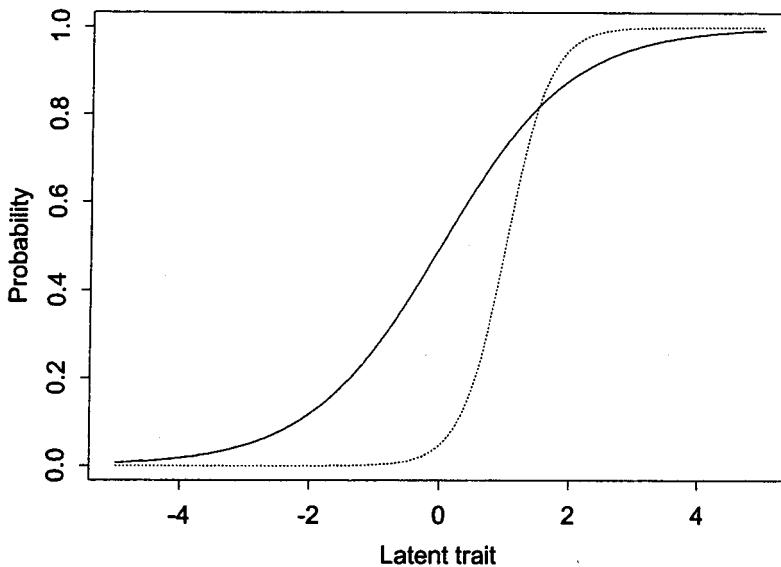


FIG. 9.3. Item response functions of two different items.

The generative mechanism takes place according to the following sequence:

1. For record $i = 1$, draw a sample z_i from the normal distribution $N(0, 1)$.
2. Given value of z_i and α_j , $j = 1$, compute the probability in Equation 6 and let the value be denoted by p .
3. Simulate a uniform number x from the interval $[0, 1]$. If $x < p$, then $y_{i1} = 1$, otherwise $y_{i1} = 0$.
4. Repeat the above step independently to generate y_{ij} for $j = 2, \dots, J$, $i = 1$.
5. Repeat the above generative steps independently for each record $i = 2, \dots, n$.

Table 9.5 shows a sample of three-item responses from a generative model with $z \sim N(0, 1)$, $\alpha_1 = (0, 1)$, $\alpha_2 = (-1, 1.5)$, $\alpha_3 = (1, 0.5)$. Only the unshaded quantities are observed.

Important Assumptions

The conditional (local) independence in Step 4, reminiscent of the local independence assumption in LCM, again plays a critical role in simplifying the joint distribution of the responses (features). The latent trait model assumes that, given Z , item responses are all conditionally independent. In other words, the “smartness” factor is assumed to completely explain how a student responds to questions.

TABLE 9.5
A Sample of Responses Generated under the Latent Trait Model with
 $\alpha_1 = (0, 1)$, $\alpha_2 = (-1, 1.5)$, $\alpha_3 = (1, 0.5)$, $P_i = P(y_i = 1 | z)$, $i = 1, 2, 3$

i	Z	P_1	Y_1	P_2	Y_2	P_3	Y_3
1	-0.76	0.32	0	0.11	0	0.65	1
2	0.71	0.67	1	0.52	1	0.89	1
3	0.14	0.53	0	0.31	0	0.75	1
4	-0.69	0.33	0	0.12	0	0.66	0
5	0.22	0.56	1	0.34	0	0.75	1

There are two other implicit assumptions that are often critical in the formulation of IRT models. The first is the monotonicity of the IRF. This assumption, of course, is automatically satisfied by the logistic function in Equation 6. For other forms of IRF this assumption requires that the probability of response increases or decreases with the latent trait. Another implicit assumption about the IRT model is the unidimensionality of the latent trait Z . Various tests have been devised to assess the validity of these assumptions by examining the fit of a model to the response data. The section on extending the basic model describes how some of these assumptions can be relaxed.

Learning

Learning in latent trait models can be based on the following likelihood function:

$$L = \prod_{i=1}^n \int_{-\infty}^{\infty} \prod_{j=1}^J [p_{ij}(z)]^{y_{ij}} [1 - p_{ij}(z)]^{1-y_{ij}} \phi(z) dz, \quad (7)$$

where $\phi(z)$ is the standard Gaussian density.

The EM paradigm is well suited for the learning of item-specific parameters α . The continuous latent variable is often discretized into a number of “bins.” The computational complexity per EM step is $O(BnJ)$, where B is the number of “bins.”

In the psychometric literature, learning about individual latent traits Z is often as important as learning about the item-specific parameters. The individual-specific latent traits can be learned from Bayes rule once the item-specific parameters are known. For example, the updating equation for the expected aposterior distribution of Z , given Y , is given by:

$$E(z_i | y_i, \alpha) = \frac{\int_{-\infty}^{\infty} z \prod_{j=1}^J [p_{ij}(z)]^{y_{ij}} [1 - p_{ij}(z)]^{1-y_{ij}} \phi(z) dz}{\int_{-\infty}^{\infty} \prod_{j=1}^J [p_{ij}(z)]^{y_{ij}} [1 - p_{ij}(z)]^{1-y_{ij}} \phi(z) dz}. \quad (8)$$

The Basic Factor Analytic Model

Introduction

Factor analysis is perhaps the class of latent variable models that has the longest history. It dates back to the work of Karl Pearson, Charles Spearman, and L. L. Thurstone for defining and measuring human intelligence (e.g., Thurstone, 1947). In basic factor analysis both the latent variable and the manifest variable are continuous. In most practical applications both of these variables are also multivariate in nature.

The basic factor analytic model is essentially a data reduction technique for continuous data. Its objective is to explain common elements within multivariate data by reducing the large amounts of interrelated information into a representation that is manageable both in size and complexity. For example, we may want to reduce the 10 measurements in decathlon events into two or three “latent” athletic abilities. In addition to direct measurement of features, factor analysis can be applied to complex correlation matrices between features, in which case its aim is to find a parsimonious representation of the correlation matrix.

Generative Mechanism

The generative mechanism takes place according to the following sequence:

1. Draw for the first record ($i = 1$) the K -vector latent variable \mathbf{z}_i from the Gaussian distribution $N(0, I_K)$, where I_K is the identity matrix of dimension K .

2. Form the J -vector $A\mathbf{z}_i$ for a specified $J \times K$ factor loading matrix A .
3. Draw a random variate e_{i1} from a Gaussian distribution $N(0, d_1)$.
4. Repeat Step 3 independently from $N(0, d_j)$ to obtain e_{ij} , $j = 2, \dots, J$.
5. Form the J -vector output feature $\mathbf{y}_i = A\mathbf{z}_i + \mathbf{e}_i$, $i = 1$.
6. Repeat the above procedures independently to obtain \mathbf{y}_i , $i = 2, \dots, n$.

Important Assumptions

There are three independence assumptions in the basic model: given \mathbf{z}_i , the coordinates of \mathbf{y}_i are conditionally independent (specifically, $\mathbf{y}_i | \mathbf{z}_i \sim N(A\mathbf{z}_i, D)$, $D = \text{diag}(d_1, \dots, d_J)$); the components of \mathbf{z} are independent (specifically, $\mathbf{z}_i \sim N(0, I_K)$); and \mathbf{z}_i is independent of \mathbf{e}_i . The three independence assumptions imply that the marginal distribution of \mathbf{y} is given by $\mathbf{y} \sim N(0, AA^T + D)$. The vector \mathbf{y}_i can be pooled to form the matrix Y , and so can \mathbf{z}_i and \mathbf{e}_i , to form, respectively, matrices Z and E . Hence, the factor analytic model can be expressed in the succinct form: $Y = AZ + E$.

Learning

The statistical learning problem is to find optimal solutions for factor loadings A and the variance structure D . When A and D are estimated, Z can be inferred by Bayes rule.

The learning problem can be solved in a number of ways. Among them, spectral decomposition (a special case of singular value decomposition, or SVD) seems most promising and has the potential to scale up. Consider for the time being the case in which the random component \mathbf{e} in the model is absent. Then the problem becomes one of finding A and Z so that $Y \approx AZ$. This is a problem of evaluating the principal components for Y , and it can proceed in the following way using spectral decomposition. Let Σ denote the covariance matrix of Y . Use the sample covariance matrix $S = (1/(n-1)) \sum_i (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^T$ in place of Σ , where $\bar{\mathbf{y}}$ is the mean of \mathbf{y} . Next, decompose S to yield eigenvectors and values of dimension J . Then truncate the components corresponding to the smallest $(J-K)$ eigenvalues by hard thresholding. This leads to a K -dimensional representation of the original J -dimensional data space. Specifically, let $\Sigma = \lambda_1 a_1 a_1^T + \dots + \lambda_J a_J a_J^T$ be the spectral decomposition of Σ , where a is a system of orthonormal eigenvectors, and $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_J \geq 0$ be the system of eigenvalues. The approximate representation of Σ using only the first largest K eigenvalues is given by:

$$\Sigma \approx \hat{A}\hat{A}^T = (\sqrt{\lambda_1}a_1 \dots \sqrt{\lambda_K}a_K) \begin{pmatrix} \sqrt{\lambda_1}a_1^T \\ \vdots \\ \sqrt{\lambda_K}a_K^T \end{pmatrix}. \quad (9)$$

When the random component is present, we allow for the variance of the additive noise in the learning process:

$$\Sigma \approx \hat{A}\hat{A}^T + D \quad (10)$$

$$= (\sqrt{\lambda_1}a_1 \dots \sqrt{\lambda_K}a_K) \begin{pmatrix} \sqrt{\lambda_1}a_1^T \\ \vdots \\ \sqrt{\lambda_K}a_K^T \end{pmatrix} + \begin{pmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ 0 & 0 & \dots & d_J \end{pmatrix} \quad (11)$$

The variance matrix D is estimated by the residual $S - \hat{A}\hat{A}^T$. The spectral decomposition method often is termed the principal factor method because of its close connection with principal component analysis.

Numerical Example

Consider the following correlation matrix R that is obtained from a sample of marketing survey data. The matrix R indicates the sample correlation between three attributes—taste, flavor, and perceived value of a new product line of snacks:

$$R = \begin{matrix} & \text{Taste} & \text{Flavor} & \text{Value} \\ \text{Taste} & 1 & 0.9 & 0.3 \\ \text{Flavor} & 0.9 & 1 & 0.2 \\ \text{Value} & 0.3 & 0.2 & 1 \end{matrix}. \quad (12)$$

The eigenvalues of the correlation matrix R in Equation 12 are 2.02, 0.88, and 0.09, and their corresponding eigenvectors are respectively $(0.68, 0.66, 0.33)^T$, $(-0.17, -0.29, 0.94)^T$, and $(0.72, -0.69, -0.08)^T$. Suppose we use only one factor in approximating the correlation matrix R , then

$$\begin{aligned} R \approx \hat{A}\hat{A}^T &= \begin{pmatrix} \sqrt{2.02} & 0.68 \\ \sqrt{2.02} & 0.66 \\ \sqrt{2.02} & 0.33 \end{pmatrix} \begin{pmatrix} \sqrt{2.02} & 0.68 & \sqrt{2.02} & 0.66 & \sqrt{2.02} & 0.33 \end{pmatrix} \\ &= \begin{pmatrix} 0.926 & 0.903 & 0.448 \\ 0.903 & 0.879 & 0.437 \\ 0.448 & 0.437 & 0.217 \end{pmatrix}. \end{aligned} \quad (13)$$

When the random component D is included, we have

$$\begin{aligned} R \approx \hat{A}\hat{A}^T + D &= \begin{pmatrix} 0.926 & 0.903 & 0.448 \\ 0.903 & 0.879 & 0.437 \\ 0.448 & 0.437 & 0.217 \end{pmatrix} + \begin{pmatrix} 0.074 & 0 & 0 \\ 0 & 0.121 & 0 \\ 0 & 0 & 0.783 \end{pmatrix} \\ &= \begin{pmatrix} 1.0 & 0.903 & 0.448 \\ 0.903 & 1.0 & 0.437 \\ 0.448 & 0.437 & 1.0 \end{pmatrix}. \end{aligned} \quad (14)$$

Now suppose a two-factor solution is used, then

$$\begin{aligned} R \approx \hat{A}\hat{A}^T &= \begin{pmatrix} \sqrt{2.02} & 0.68 & \sqrt{0.88}(-0.17) \\ \sqrt{2.02} & 0.66 & \sqrt{0.88}(-0.29) \\ \sqrt{2.02} & 0.33 & \sqrt{0.88}0.94 \end{pmatrix} \begin{pmatrix} \sqrt{2.02} & 0.68 & \sqrt{2.02} & 0.66 & \sqrt{2.02} & 0.33 \\ \sqrt{0.88}(-0.17) & \sqrt{0.88}(-0.29) & \sqrt{0.88}0.94 \end{pmatrix} \\ &= \begin{pmatrix} 0.952 & 0.947 & 0.306 \\ 0.947 & 0.965 & 0.195 \\ 0.306 & 0.195 & 0.999 \end{pmatrix}. \end{aligned} \quad (15)$$

Clearly, the last matrix obtained from using the two-factor model has significantly improved the approximation from using the one-factor model. The two-factor case when the noise component is present is analogous to Equation 13.

Factor loadings A are determined only up to an orthogonal matrix Γ . In other words, if \hat{A} is a solution to Equation 11, then $\Gamma\hat{A}$ is also a solution. One can therefore choose an orthogonal

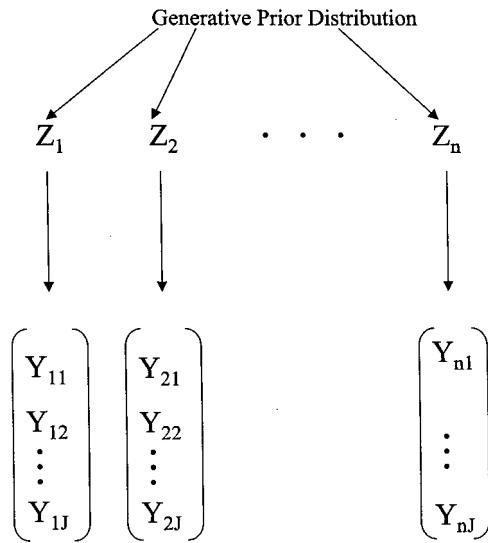


FIG. 9.4. Structure of a basic latent variable model.

matrix for rotating A to produce interpretable results. For example, a pattern of loading in which each variable loads highly on a single factor and has small loadings on the other factors would be ideal. Thus, the interpretation of factor loadings can be difficult and has been controversial in the past.

A maximum likelihood method such as the EM algorithm can be applied to solve for A and D under the Gaussian generative model. In maximum likelihood estimation, it is common to impose a uniqueness condition such that $A^T D^{-1} A$ is diagonal. The maximum likelihood methods are not described in this chapter.

Common Structure

The common structure among the four basic models can be summarized by Fig. 9.4. The generative prior distribution for both the latent class and mixture models is the multinomial distribution, whereas for latent trait and factor analytic models it is the Gaussian distribution.

EXTENSION FOR DATA MINING

This section consists of four subsections. The subsections describe how each of the four basic latent variable models can be expanded for data mining.

Extending the Basic Latent Class Model

Traditional LCMs usually are restricted to a few latent classes for feature vectors of relatively low dimension in data sets of small size. Newer applications can feature hundreds of latent classes, and output vectors of relatively high dimension. For example, Hofmann (2001) applies LCM modeling to the well-known EachMovie data, a benchmark test set used in collaborative filtering. The data set consists of 1,623 items, more than 60,000 individuals, and a total of over 2.8 million movie ratings.

To enhance the utility of LCM modeling to handle complex and massive data sets, the basic LCM needs to be expanded. One problem that prohibits LCMs from being useful for large data sets in which a high level of heterogeneity exists within the data is the potentially large number of parameters required in learning. The number of parameters in a J -vector K -class model is of the order $J \times K$, which can be substantial if either J or K is sizeable. Using a smaller number of classes K often impedes the performance of LCM, making the basic model rather limited for handling large real-world data sets. Determining the number of classes is yet another challenging issue of its own, especially when the number of classes is beyond a handful. A third problem often encountered in LCM learning is that some probability estimates of certain classes tend to converge to 0 or 1. Yet another problem with the basic LCM is its lack of utility in dealing with auxillary data such as feature attributes (e.g., movie profiles) that can provide additional predictive power if they can be incorporated into the model. Finally, alternative discrete forms of data other than binary also need to be accommodated.

The following bullet points summarize strategies for expanding the LCM to cope with the above challenges:

- Constrain the basic LCM by using equality and inequality constraints on the parameters. Alternatively, a Bayesian paradigm can be applied to LCM to regularize parameter estimates.
- Use predictor variables in the mixing weights and the population probabilities within each class to increase predictive power.
- Use alternative discrete distributions such as Poisson and multinomial in describing the feature space to handle nonbinary data.
- Relax the conditional independence assumption by introducing dependence models into the conditional joint distribution of features given latent class.
- Employ early stopping in learning to avoid overfitting and the so-called terminal problem of class probabilities converging to zero or one.
- Determine the appropriate number of latent classes, for example, using selection criteria based on information, bootstrapping (a resampling technique), or Bayesian models.
- Expand the scope of LCM to include latent classes for general “objects” such as decision trees.

Various constrained LCM models have been proposed to reduce the number of parameters needed to describe the features within each latent class. Linear equality constraints or fixing some parameters at prespecified values allow parsimonious models to be fitted to complex data sets. For example, one may require that the probabilities of positive response across some items be fixed to the same value. More complex constraints such as ordered constraints on probabilities across classes also can be imposed.

In the presence of auxiliary predictors such as contextual information on the feature vector, we can expand the basic LCM to include the predictors in a linear regression function. The generalized linear model (McCullagh & Nelder, 1989) is a powerful tool to accomplish this goal. For example, in Bandeen-Roche, Miglioretti, Zeger, and Rathouz (1997), an LCM is fitted to a data set concerning disability measures of a sample of community-dwelling women age 65 years and older. The output features are measures such as “ability to lift up to 10 pounds,” “using the telephone,” and so on. Age and self-report of arthritis history are used as predictors in the vector \mathbf{u} in the following equation:

$$\pi_{jk} = \exp(\alpha_k^T \mathbf{u}) / (1 + \exp(\alpha_k^T \mathbf{u})). \quad (16)$$

Basically, Equation 16 posits different regression coefficients in different latent classes. Simple constraints (e.g., setting the coefficients for age to be constant across classes) can be used to reduce the total number of parameters.

To broaden the scope of LCM for handling complex data sets, distributions other than the conditionally independent, multivariate binary distribution have been used in the LCM literature. Because the observed variable is discrete, and in some cases multivariate in nature (i.e., $J > 1$), we can resort to a range of parametric distributions for multivariate discrete data in modeling the feature space. For example, a multinomial distribution is often used to model sparse count data for a large number of possible categories. Nigam, McCallum, Thrun, and Mitchell (1999) used an LCM based on a multinomial distribution for document classification and clustering. Each document was represented by a vector of word counts for a selected collection of key words. If the value of the count is ignored but instead recorded as either 1 or 0, the basic LCM can be applied by evoking the conditional independence assumption that the presence or absence of a specific word is conditionally independent of any other word, given its class (e.g., medical-type document). McCallum and Nigam (1998) compared the use of multinomial and conditional independence models. They found that the multinomial model provided more accurate classification, although the difference in accuracies between the two models was not significant. The Poisson distribution is another common choice for modeling count data. For example, the number of calls received by various specialists at a call center may be modeled as comprising several classes, each having its own rate λ of contacting the call center.

When the conditional independence assumption is not viable, we may consider using a local dependence model such as a log-linear model in place of conditional independence. However, the complexity of the overall model substantially increases, and its computational overhead increases accordingly. We are not aware of any large-scale implementation of locally dependent LCM modeling. For further reference on local dependence LCM, please refer to the References and Tools section.

Some authors have proposed methods of regularization for the EM algorithm to avoid probabilities converging to 0 or 1 and also to avoid overfitting. For example, Hofmann (1999) proposed the tempered EM for LCM in automated document indexing, and Schneider (2001) used ridge regression with general cross-validation to regularize EM.

Determining the number of latent classes is a special case for a more general class of problems called model selection (e.g., see chap. 5 of this volume). Traditionally, the determination of the appropriate number of latent classes has been through the use of chi square-type hypothesis testing, which only works under limited conditions (McLachlan & Basford, 1988). Other criteria, such as the Akaike information criterion (AIC; Akaike, 1973) or Schwarz Bayesian information criterion (BIC; Schwarz, 1978), are also commonly used but have their own limitations as well. These criteria are based on fit statistics, such as the log-likelihood, plus a penalty term for model complexity based on the total number of parameters used. However, these criteria may give rather different results for large data sets.

An alternative approach to determining the number of latent classes is to use external validation. In this case a separate validation data set is used to score models with different numbers of latent classes, and the model with the best score is chosen as the winner. For applications in which the number of latent classes is large and interpreting the meaning of each class is of less importance, we can visually locate the point of diminishing returns on performance (e.g., goodness-of-fit) on a plot of performance against number of classes. This helps determine an approximate optimal point for the number of classes required. Finally, Bayesian approximation methods also can be used to compare the posterior distributions for various values of K given the observed data (see chap. 5 of this volume).

LCM can be applied to other “objects” besides multivariate binary vectors. This type of generalization is often discussed in the literature under the notion of mixture models.

Extending the Basic Mixture Model

The strategies for extending the basic Gaussian mixture model can be summarized as follows:

- Use alternative continuous distributions, possibly multivariate, in the place of the Gaussian features wherever appropriate.
- Exploit the mixture framework for innovative applications such as detection of outliers and fraudulent cases.
- Extend the mixture concept to data that are not necessarily in a J -vector form. Examples include mixtures of regressions, mixtures of trees, mixtures of sequences, mixtures of hierarchical models, and so on.

In addition to the Gaussian distribution, the mixture model may include parametric distributions such as exponential (for between-event time), Weibull, and gamma. The framework of estimation remains essentially the same as in the Gaussian case.

The mixture model is a powerful and flexible tool to describe heterogeneous populations. Because the underlying idea is very general, mixture modeling can include complex models in cases where there appears to be overdispersion (more variance than expected from a single model) that is not captured by a single model. For example, when a single logistic regression line does not seem to fit a data set well and we suspect that there is a subpopulation that exhibits very different characteristics (e.g., different slopes for predictors), then one may consider using a mixture of logistic regression functions. It is well known that logistic regression can be used in the context of supervised learning for classifying binary outputs. When the population consists of heterogeneous data, using a mixture of multiple components of logistic functions, each having its own regression coefficients, could improve overall predictive performance.

By the same token, almost any model can be extended by the mixture concept. For example, mixtures of ultrametric trees, a type of clustering for dissimilarity judgment data (e.g., how dissimilar do you think Camry is from Accord versus Camry from Taurus), can be used to describe diverse groups of consumer preferences. Another example of mixtures that arises in the psychometric literature is the mixture of multidimensional scaling (MDS) models.

To illustrate how the basic mixture model can be extended for analyzing large-scale transaction data, we use a real-world example that contains a data set extracted from a national retail chain store in Japan. Shoppers of chain stores are known to be heterogeneous in terms of their shopping regularity as well as their frequency of shopping. One can envision distinct latent classes of “random” and “regular” shoppers. Random shoppers typically do not follow a consistent pattern of shopping and are often impulsive. Regular shoppers, on the other hand, tend to have consistent and routine patterns of shopping.

The intershopping time of the first class of shoppers—the random shoppers—can be modeled using an exponential distribution, in which each random shopper shops with an individual rate parameter λ_i : $f_i(t) = \lambda_i \exp(-\lambda_i t)$. As a result of the well-known memoryless property of the exponential distribution, each shopping trip is independent of the time elapsed from the last trip. On the other hand, each consistent shopper is modeled with a scale parameter τ that governs an Erlang-2 distribution, which has the following form: $g_i(t) = \tau_i^2 t \exp(-\tau_i t)$.

The rate parameter λ_i is assumed to follow a gamma distribution with scale parameter γ_1^R , and shape parameter γ_2^R . That is, $\lambda_i \sim \text{Gamma}(\gamma_1^R, \gamma_2^R)$, where the superscript R indicates random shoppers. For a consistent shopper the shape parameter τ_i is indicative of the average

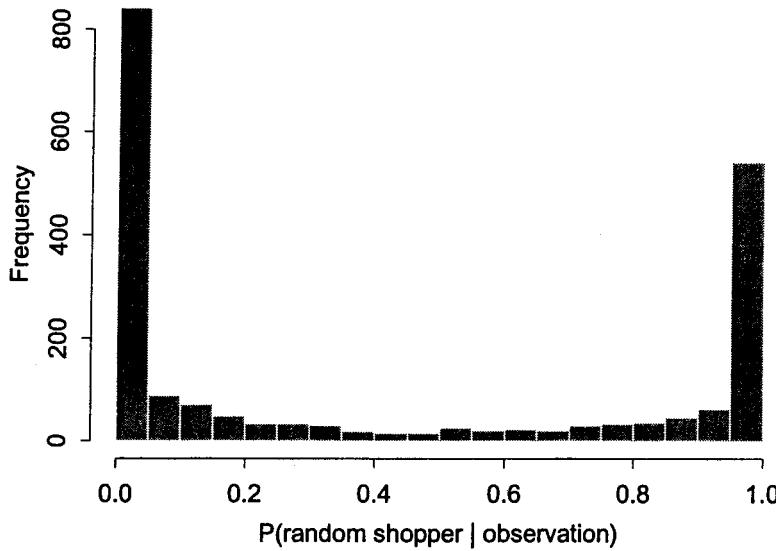


FIG. 9.5. Histogram of shoppers sorted by posterior probability of belonging to the class of random shoppers.

length of the shopper's intershopping times. Much like λ_i , the parameter $\tau_i \sim \text{Gamma}(\gamma_1^C, \gamma_2^C)$, where the superscript C indicates consistent shoppers.

Figure 9.5 shows the result of applying EM to learn the posterior probability of belonging to the class of consistent shoppers for a sample of customers. The sample consists of a sample of 67,587 transactions generated by 2,053 customers. It can be seen from the graph that most shoppers fall into one of the two mixture components with rather high probabilities, thus providing some supporting evidence for heterogeneity of shopping behavior. In the learning process we also factored in the fact that the last observation was censored (for details, see Kim & Park, 1997). The EM algorithm for this model took less than 20 iterations in learning and provided the following estimates: $(\gamma_1^R, \gamma_2^R) = (54.71, 1.791)$, $(\gamma_1^C, \gamma_2^C) = (13.42, 0.736)$. For a gamma distribution with scale and shape parameters (γ_1, γ_2) , its mean is γ_2/γ_1 and its variance is γ_2/γ_1^2 . The mean intershopping rate of the random and consistent groups are therefore, respectively, $1.79/54.7 = 0.033$ and $0.736/13 = 0.055$, suggesting, for example, that the consistent shoppers have on average a shopping rate of approximately 1 in 18 days.

Extending the Latent Trait Model

Latent trait models are flexible tools for measuring individual-specific traits, and they have the potential for predictive modeling of large transaction data sets. For example, a latent trait can be posited for the propensity to purchase across various products of a brand for a consumer, or for the inclination to browse materials from a specific category for a WWW user. In both cases the feature variables would contain inputs derived from observable behaviors (e.g., transaction, click stream).

A large psychometric literature exists on extending the basic latent trait and IRT models. The following points summarize strategies for these extensions:

- Generalize the parametric form (Equation 6). This includes adding parameters to the basic logistic function, changing the form of the logistic function to handle multiple

ordered categories and nominal data of multiple categories, or completely abandoning the parametric form for spline-like smooth functions.

- Use auxiliary predictors in the parameters and also in the distribution of the latent trait.
- Relax model assumptions imposed by the basic model. For example, replace the conditional independence models by conditional dependence models, replace monotonicity by unfolding models that feature a bell-shaped IRF, and replace unidimensional models with multidimensional models.
- Cross-breed IRT models with other types of models to provide innovative applications. For example, in marketing research different IRT models are calibrated for different segments of respondents under a mixture framework.

One important development in IRT is the generalization of the basic latent trait model to handle features of ordinal nature (e.g., y can take any of the values 1, 2, 3, 4). One idea for the generalization of Equation 6 is the partial credit model (Masters, 1982), which exponentiates the linear function for each category and then normalizes each term by dividing it by the sum of all involved terms. The probability of observing y_j taking the value in the h th category is:

$$p(y_j = h | z) = \frac{\exp\left(\sum_{v=1}^h s_{jv}(z)\right)}{\sum_{c=1}^H \exp\left(\sum_{v=1}^c s_{jv}(z)\right)}, \quad (17)$$

$h = 1, \dots, H - 1$, where $s_{jv}(z) = \alpha_{j0v} + \alpha_{j1v}z$. One of the categories (e.g., $h = 0$) is typically used as a baseline in such a way that $s_{j0} \equiv 1$. The denominator in Equation 17 is the normalizing constant; it guarantees that the probabilities across categories sum to one.

Learning about the latent trait is often as important as learning about the item-specific parameters. The parametric form of IRF constitutes an efficient basis for evaluating response probabilities. One strategy for efficiently learning the latent trait of a large number of individuals is to first learn the item-specific parameters α through a random sample of the data, using methods such as EM. The continuum of the z space can then be discretized into a number of “bins” for fast updating. Then each individual is “classified” into one of the bins using naive Bayes (see chap. 5 this volume).

Extending the Factor Analytic Model

A significant limitation of the classic factor analysis model is the assumption that the vector of latent variables \mathbf{z} has an uncorrelated multivariate Gaussian distribution. The technique of independent component analysis (ICA) retains the same functional form as the standard factor analysis model, $\mathbf{y} = A\mathbf{z} + \mathbf{e}$, but relaxes the Gaussian assumption and instead seeks a representation for the observed measurement vector \mathbf{y} as a linear combination of non-Gaussian variables \mathbf{z} that are statistically independent from each other.

Because the Gaussian distribution has the property of having maximum entropy over all possible distributions with the same fixed variance, a widely used technique to determine non-Gaussian latent variables \mathbf{z} in ICA is to seek independent variables that have minimum entropy. The entropy for different projections of the data is estimated empirically, and a variety of iterative algorithms can be used to greedily search through the space of possible projections to find a solution \mathbf{z} that maximizes the likelihood of the observed data. Unlike the SVD decomposition methodology for classical factor analysis (described earlier under The Basic Factor Analytic Model), there is no analytic solution; however, in ICA it is often useful first to decorrelate \mathbf{y} using the SVD method so that $\text{cov}(\mathbf{Y}) = I$ so that ICA can be viewed as extending the factor analysis solution.

Thus, ICA can be viewed as a useful alternative to classical factor analysis, and the two techniques are closely related. As pointed out by Hastie, Tibshirani, and Friedman (2001, p. 498), “ICA starts from essentially a factor analysis solution, and looks for rotations that lead to independent components. From this point of view, ICA is just another factor rotation method.”

One can also combine the general concept of mixture models with factor analysis by defining mixtures of factor analyzers. The generative model consists of a discrete-valued latent variable (call it C , with values c_k , $1 \leq k \leq K$); each data point \mathbf{y}_i is assumed to be generated by one of the discrete components c_k . Unlike the classical mixture model, however, where $p(\mathbf{y} | c_k)$ is assumed to be multivariate Gaussian, in the mixture of factor analyzers model we have that $p(\mathbf{y} | c_k)$ corresponds to a factor model, $\mathbf{y} = A_k \mathbf{z} + \mathbf{e}$, where the factor loading matrix A_k can be different for different values of the discrete latent variable C . In the simplest version of this model each of the “component” factor models has the same dimensionality. One can think of this model either as a modified mixture of Gaussian models in which each Gaussian is modeled by a factor model (clearly this is potentially useful for high-dimensional mixture modeling problems). Equivalently, one can think of “covering” the Y space by several different “local” factor models. The EM algorithm can be used to fit this type of model by extending the maximum-likelihood version of factor analysis to incorporate the latent variable C .

Another generalization of the factor analysis model results by introducing dependence (e.g., over time) in the latent variable \mathbf{z} . Specifically,

$$\mathbf{y}_i = A\mathbf{z}_i + \mathbf{e}_i, \quad (18)$$

$$\mathbf{z}_i = B\mathbf{z}_{i-1} + \mathbf{f}_{i-1}, \quad i = 2, \dots, n \quad (19)$$

where A and \mathbf{e}_i are the factor loading matrix A and vector of independent random variates as defined in the basic factor analytic model, and B is a $K \times K$ vector that couples the vector of latent variables \mathbf{z}_i at time i with the latent variables \mathbf{z}_{i-1} at time $i - 1$, and \mathbf{f}_{i-1} is another K -dimensional vector of independent random variates, often assumed to be Gaussian. This model is widely used in engineering under the name “Kalman filter,” where the latent vector \mathbf{z} often corresponds to a physical (but unobservable) *state* of a system, such as a vector containing estimates of the position, velocity, and acceleration of an object in motion, and where \mathbf{y} would represent observed measurements on the object (e.g., from a camera or radar instrument tracking an aircraft). Given a set of Y measurements, the Kalman filter methodology uses Equations 18 and 19 to recursively derive estimates of the Z variables given the Y ’s. Traditionally the noise terms \mathbf{f} and \mathbf{e} are both assumed to be zero-mean Gaussian with specific covariance matrices, leading to direct analytic solutions for estimating the expected values of the \mathbf{z}_i ’s given the \mathbf{y}_i ’s.

Unlike the standard factor analysis approach, in traditional Kalman filtering the parameter matrices A and B are defined based on prior knowledge of the physical system being observed. However, one can also learn the parameters of the model from observed data Y using the EM algorithm. This opens up the possibility of defining more general models. For example, one can define a mixture of Kalman filters (analogous to a mixture of factor analyzers, but with time-dependence added) in which the dynamics of the system (as captured by B) are now allowed to switch between K different dynamical systems characterized by B_1, \dots, B_K and in which the switching is controlled by a latent variable C taking K values c_1, \dots, c_K that itself typically has Markov dependence.

Further recent extensions of the Kalman filter model have relaxed the linear-Gaussian assumptions in various ways, typically motivated by the desire to model nonlinear, non-Gaussian behavior in real systems. Unfortunately, such extensions typically lose the nice analytical closed-form recursive solutions that characterize the standard linear-Gaussian model, and

instead produce computations that are at first glance intractable. One useful way to get around this intractability is to adopt a Bayesian approach to both inference of the state variables Z and estimation of the model parameters (A , B , etc.), where Monte Carlo sampling techniques (such as so-called particle filters) can be used to approximate posterior distributions on quantities of interest in a relatively efficient manner.

The extensions to factor analyzers described above have seen relatively few applications to date in the context of large-scale data mining problems. One reason is that many of these techniques are relatively new and there has been relatively little experience with using these models to date on real-world data. Another reason is that these models are relatively complex (involving multiple levels of hidden variables in some cases) and, thus, require more statistical knowledge and “know-how” to implement successfully. Nonetheless, given that modeling data in high dimensions (large J) is a more fundamentally challenging problem than modeling with large data sets (large n), ideas such as ICA and mixtures of factor analyzers would appear to have significant potential promise for data mining problems with high-dimensional, heterogeneous, continuous data. Similarly, for modeling of individuals over time (for example, tracking the state of a customer over many months based on spending patterns), models such as Kalman filters and their generalizations would appear to provide a useful general framework.

AN ILLUSTRATIVE EXAMPLE

In this section we provide an illustrative example of an extension of the mixture models described earlier in this chapter. First, we describe a hierarchical data structure commonly encountered in analysis of transaction data and show a natural generalization of mixture models to account for this additional structure. Next, we describe two real-life data sets. Finally, we show an illustrative example of the information discovered by mining the two data sets, and we demonstrate intuitively and experimentally how the new model outperforms traditional models.

Hierarchical Structure in Transaction Data

Consider a realistic retail corporation with stores in several locations, many customers, and a variety of intuitively related (e.g., beer and wine, bread and butter, shirts and ties) and unrelated products (e.g., power tools and cosmetics) offered at each store. At the highest level, stores at various locations attract customers with different social backgrounds and needs, different purchasing power and habits, and so forth. Also, customers within each store purchase different groups of items, with occasional overlap between groups (e.g., customers might often buy milk and bread, but occasionally they buy milk, bread, and kitchen cleaners).

When there is no means to track individual customers, a simple mixture model may be appropriate to describe the overall observed shopping behavior. For example, each mixture component might correspond to a shopping behavior in which a customer buys products from a group of tightly related items. The mixture model data mining can recover the group structure of items by finding products that often are purchased together and provide an estimate of how often customers engage in a specific shopping behavior. Similarly, the data from each store location can be analyzed separately and compared with the overall corporate data to discover trends and location-specific behavior.

However, contemporary lifestyle involves predominantly cashless transactions in which purchases are made by credit cards and checks. Combined with very cheap digital storage, cashless transactions give retail corporations the capability to keep additional information for

each transaction and customer almost for free. Individual transactions are no longer independent, but are linked through the specific customer who made them. Consequently, the data becomes hierarchical, where at the lower level there are individual transactions, but at the next level the sets of transactions are grouped together based on the specific individual who generated them.

Individualized Mixture Models

In this subsection we describe a natural generalization of the standard mixture model that can be used to analyze hierarchical data. Our aim is to describe one plausible scenario and demonstrate that the resulting *individualized model* outperforms the basic mixture model.

The individualized model we describe here is a combination of the individual-specific behavior and the overall population properties. Specifically, the overall (global) population model—a standard mixture model—defines a set of K general shopping behaviors, whereas an individualized model describes a specific individual's propensity to engage in each of the K behaviors on various visits to the store. We assume that the global mixture weights specify the relative frequency of each shopping behavior in the population as a whole, whereas each individual has his or her own set of individualized mixture weights that describe that individual's frequency of engaging in each of the shopping behaviors.

A simple model is to assume that the individual-specific weights are a realization of random draw from the multinomial with global weights as parameters. Intuitively, individuals with very few transactions are a priori considered to have behavior similar to that of the whole population; that is, their frequency of engaging in each behavior is that of the population as a whole. As the number of transactions generated by an individual grows, so does our knowledge about the individual's behavior. These additional transactions define an individual's specific frequency of engaging in each shopping behavior. We formalize the individualized model as a generative data model in the following manner.

Individualized Generative Model

- An individual $i = 1$ is randomly drawn from the overall population.
- The individual's propensity of engaging in each shopping behavior is a random draw from a Dirichlet, the mean of which is a multinomial with the global mixture weights as parameters.
- From the individualized frequency of each shopping behavior (the individualized weights), a specific shopping behavior (say, the k th component mixture) is drawn randomly.
- For the given shopping behavior, an item is drawn randomly.
- The item selection process is repeated for a prespecified number of items in the transaction.
- The shopping behavior selection process is repeated for a prespecified number of transactions J_i for the individual.
- The overall process is repeated for the prespecified number of individuals $i = 2, \dots, n$.

Mathematically, we can write down the log-likelihood associated with the individualized model as:

$$\log P(\Theta | D) = \sum_{i=1}^n \sum_{j=1}^{J_i} \log \left[\sum_{k=1}^K \alpha_{ik} P(\mathbf{y}_{ij} | \boldsymbol{\theta}_k) \right] + \sum_{i=1}^n \log P(\boldsymbol{\alpha}_i | \boldsymbol{\xi}). \quad (20)$$

In this equation D represents the whole data set, y_{ij} represents the j th transaction of individual i , α_{ik} represent the individual-specific weights (the frequency of individual i engaging in shopping behavior k), θ_k are parameters of the k th shopping behavior, ξ are parameters of the Dirichlet prior for the individualized weights (i.e., proportional to the global weights), and Θ represents all the model parameters. The learning task consist of finding the optimal set of parameters Θ . We use a version of the EM algorithm to find the optimal parameters Θ (Cadez, Smyth, & Mannila, 2001).

Data Sets

We use two large real-world retail transaction data sets—a retail clothing data set, and a retail drugstore data set—in our experiments. When a basket (transaction) is paid for, each item in the basket is recorded, along with the identification of the individual making the purchase, and the time and day of the purchase. For both data sets the items being purchased are categorized according to a product hierarchy. At the top of the hierarchy are broad categories such as departments within the store. At the bottom of this hierarchy are very specific product descriptions. For example, in the retail clothing data set there are 53 departments at the top level and approximately 50,000 specific item descriptions at the bottom level of the hierarchy. The categories at each level of this hierarchy are mutually exclusive and exhaustive. In this chapter we demonstrate making predictions at the second level of this product hierarchy.

The first data set consists of purchases over a 2-year period at a set of stores for a retail clothing chain in the United States. There are approximately 1.2 million items purchased during 500,000 separate transactions by approximately 200,000 different individuals. Items are categorized into 53 departments at the top level and 409 products at the second level of the hierarchy.

The second data set contains transaction records collected during the period of 1996–1999 from a national drugstore retail chain in Japan. The data set consists of approximately 15.6 million items purchased during 2.5 million separate transactions by about 300,000 individuals over approximately 1,000 stores across Japan. The department level (Level 1) of the product hierarchy comprises 21 categories (such as medical devices, baby products, etc.) and the second level contains 151 more detailed categories.

Experimental Results

In this section we show two high-level results of the proposed individualized model. First, we show that the out-of-sample performance is consistently improved by taking into account individual-specific frequency of engaging in various shopping behaviors, then we show an illustrative example of the global shopping behaviors discovered in the retail drugstore data.

To measure performance of the models we split the data into two disjoint sets: the first 70% of the data for training and the remaining 30% of the data for scoring. The out-of-sample (predictive) log-likelihood of a model is a measure of how well the model can predict future purchases. When properly scaled (e.g., negative total out-of-sample log-likelihood divided by the total number of items purchased in the out-of-sample period), the out-of-sample log-likelihood is converted into more intuitive units of “bits per item,” which can be treated as a predictive score. For example, if we had 256 available items, then a random model would have the score of approximately 8 bits per item; that is, the model would provide no information about the future purchases, and we would need 8 additional bits per purchased item to make perfect predictions. If the model had any predictive power, its predictive score would be lower; that is, we would need less than 8 additional bits to make perfect predictions. Finally, a perfect

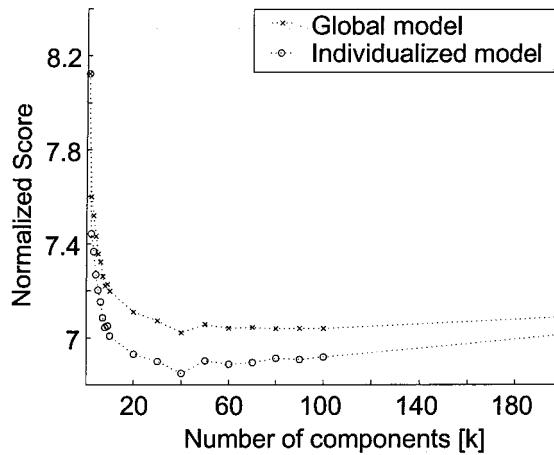


FIG. 9.6. Negative out-of-sample log-likelihood normalized to bits per item for individuals with at least 10 transactions in the training period for retail clothing data.

model would have score of zero, because we would need no additional information to make perfect predictions. Figures 9.6 and 9.7 show predictive scores for a variety of models with the number of mixture components (prototypical shopping behaviors) ranging from 1 to 200 for retail clothing data (Fig. 9.6) and drugstore data (Fig. 9.7). The data sets in both figures consist of individuals who made at least 10 transactions during the training period. Figures 9.6 and 9.7 show that for both real-life data sets we observe a systematic improvement when we take into account individualized information; that is, the individualized model makes more accurate predictions (fewer bits) than the standard global mixture model.

Table 9.6 describes the estimated shopping behavior models at Level 2 for the drugstore retail data. We list only high-frequency items (or categories) that have lift ratios significantly greater than 1. A lift ratio is defined as the probability of an item being purchased given the component compared with the probability of an item being purchased in general. The

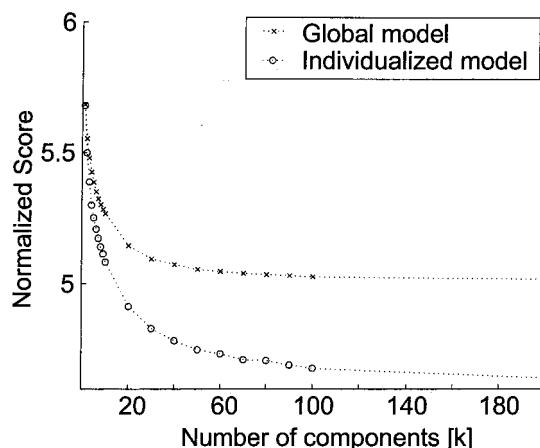


FIG. 9.7. Negative out-of-sample log-likelihood normalized to bits per item for individuals with at least 10 transactions in the training period for drugstore data.

TABLE 9.6
High Lift Items and Associated Probabilities for the k = 20 Components (Shopping Behaviors)
in the Drugstore Data, Level 2

<i>Cluster k = 1, Weight = 0.102 (194760 Baskets)</i>				<i>Cluster k = 11, Weight = 0.045 (85330 Baskets)</i>			
<i>Item</i>	<i>P(Item k)</i>	<i>P(Item)</i>	<i>Lift</i>	<i>Item</i>	<i>P(Item k)</i>	<i>P(Item)</i>	<i>Lift</i>
External pain killers	0.0744	0.0110	6.8	Bottled strengthening drinks	0.6075	0.0363	16.8
Eye drops	0.0627	0.0111	5.6	Cold medicine	0.0325	0.0147	2.2
Vitamins	0.0588	0.0088	6.7	<i>Cluster k = 12, Weight = 0.042 (79862 Baskets)</i>			
Stomach and intestinal medicine	0.0553	0.0090	6.2	Cold medicine	0.1930	0.0147	13.1
Skin medicine	0.0503	0.0111	4.5	Internal pain killer	0.1244	0.0087	14.3
<i>Cluster k = 2, Weight = 0.090 (171222 Baskets)</i>				Cough medicine	0.0643	0.0036	17.8
Basic cosmetics	0.1513	0.0224	6.8	Throat drops	0.0417	0.0027	15.7
Makeup cosmetics	0.0940	0.0103	9.1	Regulated medicine	0.0399	0.0034	11.7
Hair care products	0.0929	0.0190	4.9	<i>Cluster k = 13, Weight = 0.037 (70780 Baskets)</i>			
Cosmetic products	0.0603	0.0086	7.0	Light medical treatment products	0.2945	0.0156	18.9
<i>Cluster k = 3, Weight = 0.087 (166125 Baskets)</i>				Nursing care suppl.	0.2069	0.0096	21.5
Kitchen cleaner	0.1956	0.0468	4.2	Bandages	0.0536	0.0079	6.6
Fabric softener	0.1910	0.0392	4.9	Skin medicine	0.0408	0.0111	3.7
House cleaner	0.1425	0.0354	4.0	<i>Cluster k = 14, Weight = 0.033 (61848 Baskets)</i>			
Laundry detergent	0.0803	0.0389	2.1	Cleaning tools	0.4491	0.0319	14.1
<i>Cluster k = 4, Weight = 0.084 (159132 Baskets)</i>				<i>Cluster k = 15, Weight = 0.029 (56060 Baskets)</i>			
Baby diaper	0.3097	0.0360	8.6	Baby food	0.5105	0.0241	21.2
Milk powder	0.1144	0.0130	8.8	Baby diaper	0.0822	0.0360	2.3
Lactation and weaning products	0.0467	0.0050	9.4	Milk powder	0.0818	0.0130	6.3
Baby skin care products	0.0288	0.0044	6.6	Lactation and weaning products	0.0301	0.0050	6.1
<i>Cluster k = 5, Weight = 0.081 (154382 Baskets)</i>				<i>Cluster k = 16, Weight = 0.027 (51802 Baskets)</i>			
Shampoo and conditioner	0.2413	0.0406	5.9	Insecticides	0.4291	0.0163	26.3
Laundry detergent	0.1461	0.0389	3.8	Anti-itch cream for mosquito bites	0.0690	0.0063	11.0
Soap	0.0762	0.0174	4.4	<i>Cluster k = 17, Weight = 0.022 (42083 Baskets)</i>			
<i>Cluster k = 6, Weight = 0.061 (115911 Baskets)</i>				Body warmers	0.5204	0.0170	30.7
Paper products	0.6183	0.0929	6.7	<i>Cluster k = 18, Weight = 0.018 (34849 Baskets)</i>			
<i>Cluster k = 7, Weight = 0.57 (108696 Baskets)</i>				Mothballs	0.2895	0.0093	31.2
Toothbrush	0.3200	0.0327	9.8	Kitchen gloves	0.0917	0.0047	19.5
Toothpaste	0.0873	0.0282	3.1	Dehumidifiers	0.0729	0.0027	26.7
<i>Cluster k = 8, Weight = 0.54 (102972 Baskets)</i>				<i>Cluster k = 19, Weight = 0.017 (31477 Baskets)</i>			
Air freshener	0.1235	0.0142	8.7	Snacks	0.2271	0.0077	29.6
Deodorizer	0.0756	0.0089	8.5	Constipation medic.	0.2157	0.0070	31.0
Table seasoning	0.0572	0.0055	10.4	Alcoholic drinks	0.0484	0.0008	58.6
<i>Cluster k = 9, Weight = 0.53 (100269 Baskets)</i>				Processed foods	0.0276	0.0012	22.6
Sanitary napkins	0.3142	0.0286	11.0	<i>Cluster k = 20, Weight = 0.014 (26907 Baskets)</i>			
Tampons	0.0415	0.0029	14.1	Beverages	0.3511	0.0075	46.7
<i>Cluster k = 10, Weight = 0.045 (86276 Baskets)</i>				Contact lens cleans	0.1357	0.0066	20.5
Food wrappers	0.3641	0.0345	10.6	Shaving products	0.0576	0.0045	12.8

components are intuitive—similar items tend to appear together in the same component. For example, the first component can be characterized as generic medicine, Components 2 and 9 as feminine personal care products, Component 3 as household products, and Component 4 as baby products. Component 19 provides an interesting example. It consists of 31,000 transactions and suggests an association between alcohol, junk food (snack foods and processed foods), and constipation medicine. For example, if an individual has purchased alcohol, the empirical evidence suggests that he or she is 20 times more likely to have purchased processed foods, than if alcohol was not purchased.

The results presented in this section represent an introductory review of the a mixture-based individualized model for transaction data. An in-depth analysis, plus a description of the full model and learning and extensive comments and comparison to related data mining techniques can be found in Cadez, Smyth, Ip, and Mannila (2002).

REFERENCES AND TOOLS

References

References for LCM

A classic book on LCM is Lazarsfeld and Henry (1968), and Rost and Langeheine (1997) provide a comprehensive review. Parameter fixing and imposition of equality constraints on LCM can be found in the work of Goodman (1974). Other constrained LCM models, such as those using a predictor, can be found in the work of Formann (1982) and Heijden, Dressens, and Böckenholt (1996). Constrained models have been used extensively in psychological scaling models (e.g., Dayton & Macready, 1976; Böckenholt & Böckenholt, 1991). A dependence model is discussed in Uebersax (1999).

Muthén (2001) provided an overview of recent developments of LCM. Some examples of extensions of LCM for the analyses of marketing data can be found in the book edited by Bagozzi (1994). Other recent applications of LCM include modeling latent trajectory of criminal cases (Roeder, Lynch, & Nagin, 1999), document indexing (Hofmann, 1999), and collaborative filtering for a recommender system (Hofmann, 2001).

The framework of EM is established in Dempster, Laird, and Rubin (1977). McLachlan and Krishnan (1997) provided an extensive review. Redner and Walker (1984) reviewed computational issues related to the learning of finite mixtures via EM. Neal and Hinton (1998) gave online versions of EM for mining massive data sets. A theoretical discussion of alternate projection of EM is discussed in Csiszar and Tusnady (1984), and the geometric interpretation for LCM and IRT is given in Ip and Lalwani (2000).

References for Mixtures

General reviews of mixture models can be found in Titterington, Smith, and Markov (1985); McLachlan and Basford (1988); and more recently McLachlan and Peel (2000). Dunmur and Titterington (1998) discuss issues related to EM and related methods in mixture model learning. A recent comprehensive review of linear Gaussian models including mixture and factor analysis is provided by Roweis and Ghahramani (1999).

Meila and Jordan (2000) used a mixture of trees for classification. Jordan and Jacobs (1994) provided a general framework for function approximation based on mixtures of “experts” (generalized linear regressions). Lenk and DeSarbo (2000) discussed mixture of generalized models with random effects. Wedel and DeSarbo (1998) applied mixtures to ultrametric trees,

and Wedel and Bijmolt (1998) used mixtures of multidimensional scaling models in market segmentation. Quandt and Ramsey (1978) described mixtures of curves for parametric models, and Gaffney and Smyth (1999) generalized the notion for the nonparametric case. Examples of the application of mixtures to business transaction data, in particular modeling brand choice and purchase incidence, include Green, Carmone, and Wachpress (1976) and Böckenholt (1993).

Because the mixture model is such a general concept, there has been growing interest in its innovative applications. A sample of recently published innovative applications of mixtures for large-scale data sets include classifying pixel images of digits (Saul & Lee, 2001), classifying documents (Li & Yamanishi, 1997), and microarray classification (McLachlan, 2001).

References for Latent Trait Models

The classic references for IRT are Lord (1980) and papers collected in Lord and Novick (1968). An overview of the modern extensions is provided in Van der Linden and Hambleton (1997). Estimation of IRT models using EM is discussed in Bock and Aitkin (1981). Baker (1992) provided technical descriptions of various estimation methods. Examples of extending IRT include multivariate IRT (Anderson, 1980; Stout, 1990); locally dependent models (Fischer & Formann, 1982; Hosken & De Boeck, 1997; Ip, 2002); random effects for clustered responses (Bradlow, Wainer, & Wang, 1999); latent Markov models (Vermunt, Langeheine, & Böckenholt, 1999); incorporating auxiliary information (Mislevy, 1988); integrating various aspects such as local dependence and auxillary information (Scott & Ip, 2002); and the unfolding model for nonmonotone IRF (Andrich, 1988; Roberts, Donoghue, & Laughlin, 2000; De Sarbo & Hoffman, 1987). In the marketing literature, Balasubramanian and Kamakura (1989) used a mixture of IRT for defining specific market segments.

References for Factor Analysis

Harman (1976) and Gorsuch (1983) provided overviews of factor analysis with an emphasis on interpretation and psychological application. A deeper treatment of factor analysis and other latent variable models can be found in Bartholomew and Knott (1999).

Some recent interesting developments of factor analysis in the psychometric literature include mixtures of factor models (Longford & Muthén, 1992; Yung, 1997), models for handling outliers (Yuan, Marshall, & Bentler, 2002), heterogeneous models for both mean and covariances (Ansari, Jedidi, & Dube, 2002), and advances in rotation methods (e.g., Jennrich, 2002).

ICA was proposed in the signal processing literature by Comon (1994). Hyvarinen and Oja (2000) provided a general overview of different variants of ICA and their applications. Hastie, Tibshirani, and Friedman (2001) discussed the connections between ICA and factor analysis. A special issue of the Journal of Machine Learning Research in 2003 is dedicated to ICA.

Rubin and Thayer (1982) demonstrate how factor analyzers could be estimated via the EM algorithm. In the machine learning literature, mixtures of factor analyzers were introduced by Ghahramani and Hinton (1997), and an EM algorithm was proposed for learning such models. Related models have been proposed by Jedidi, Jagpai, and DeSarbo (1997; mixtures of structural equation models) and by Attias (1999; independent factor analysis). Tipping and Bishop (1999) demonstrated how the relationship (and differences) between factor analysis and principal component analysis can be understood via generative probabilistic modeling, and then went on to propose mixtures of principal component models, which are closely related to mixtures of factor analyzers. McLachlan and Peel (2000, chap. 8) provided a detailed review of many of these ideas.

Kalman filtering has a long history in engineering (e.g., Rauch, 1963), having been widely used in aerospace and military applications for tracking of airplanes, spacecraft, and missiles. Shumway and Stoffer (1982) introduced an EM algorithm for estimating the parameters of Kalman filter models. Roweis and Ghahramani (1999) provided a unifying review of factor analyzers, mixture models, Kalman filters, and hidden Markov models (which can be viewed as mixture models or latent class models in which the latent variable has Markov dependence. See also chapter 6 on hidden Markov model this volume.) Shumway and Stoffer (1991) described “switching mixtures” of Kalman filters. The edited collection of Doucet, de Freitas, and Gordon (2001) provides an introduction to sequential Monte Carlo and particle filter methods as well as several applications in areas such as signal processing, computer vision, and robot localization.

Tools

Most of the standard packages such as SAS, SPSS, and SYSTAT do not include programs for LCM or mixture models. Programming EM and SVD using development tools such as MATLAB, SPLUS, GENSTAT, or in specific programming languages such as C is an option for data mining applications with specific needs. An alternative option is to use existing public domain and commercial software that has been developed for LCM and mixtures. This type of software is not necessarily as robust as standard packages and is often applicable only to relatively small data sets. For LCM, examples include Mplus (Muthén & Muthén, 1998), a Windows-based product that has the capacity to analyze a diverse range of latent variable models from the perspective of structural equation models. Latent GOLD (Vermunt & Magidson, 2000) is a tool that performs simple LCM and latent class of regressions. The three commonly used structural equation modeling packages LISREL, EQS, and AMOS can also perform some limited versions of LCM.

For mixtures, there is range of freely available software that can be downloaded from the Internet. They are mostly developed by researchers in machine learning. This category of software includes EMMIX, developed by Geoff McLachlan; MCLUS, by Chris Fraley and Adrian Raftery; AUTOCLASS, by Peter Cheeseman; toolboxes for Kalman filters and mixture related applications developed in MATLAB, by Kevin Murphy; and mixtures and factor analysis software in MATLAB, by Zoubin Ghahramani and Sam Roweis. WEKA is an open-source Java-based software that contains some basic algorithms such as EM for mixtures. Its versatility and rather broad coverage allows a user to adapt and extend the program for new applications.

Other packages for mixture modeling developed in psychometrics and marketing include PANMARK (Van De Pol, Langeheine, & De Jong, 1991), GLMMIX (Wedel, 1997), and MULTIMIX, developed by Murray Jorgensen. Some of the software can handle only limited amount of data.

Several latent trait software programs that have been developed under an IRT paradigm are now distributed by Assessment System Inc. They include BILOG (Mislevy & Bock, 1982), one of the earliest programs developed for IRT estimation, and MULTILOG (Thissen, 1991) for multicategory responses.

Major standard packages such as SAS and SPSS contain modules for factor analysis. PROC FACTOR in SAS, for example, can perform computations for point estimates, standard error, and confidence interval using maximum likelihood method. It also contains a wide range of rotation methods. The free software Comprehensive Exploratory Factor Analysis (CEFA) is developed by Michael Browne and can be downloaded from his website. In SPSS, factor analysis can be conducted through principal component analysis (SVD method).

SUMMARY

Historically, latent variable models that arise from the psychometric literature place strong emphasis on the interpretation of the latent variables (e.g., the psychological construct a factor represents in factor analysis). For modern data mining the emphasis has shifted to using latent variable models as a tool for describing high-dimensional data sets with a parsimonious representation. The integration of latent variable modeling into the data mining literature shows that methods from other disciplines can provide powerful tools for analyzing large-scale data.

REFERENCES

- Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov & F. Csaki (Eds.), *Second international symposium on information theory* (pp. 267–281). Budapest: academiai Kiado.
- Anderson, E. B. (1980). *Discrete statistical models with social science applications*. New York: North Holland.
- Andrich, D. (1988). The application of an unfolding model of the PIRT type to the measurement of attitude. *Applied Psychological Measurement*, 12, 33–51.
- Ansari, A., Jedidi, K., & Dube, L. (2002). Heterogeneous factor analysis models: A Bayesian approach. *Psychometrika*, 67, 49–78.
- Attias, H. (1999). Independent factor analysis. *Neural Computation*, 11, 803–851.
- Bagozzi, R. P. (Ed.). (1994). *Advanced methods of marketing research*. Cambridge, U.K.: Blackwell Publishers.
- Baker, F. (1992). *Item Response Theory: Parameter estimation techniques*. New York: Marcel Dekker.
- Bartholomew, D. J., & Knott, M. (1999). *Latent variable models and factor analysis* (2nd ed.). London: Hodder Arnold.
- Bandeen-Roche, K., Miglioretti, D., Zeger, S., & Rathouz, P. (1997). Latent variable regression for multiple discrete outcomes. *Journal of the American Statistical Association*, 92, 1375–1386.
- Balasubramanian, S. K., & Kamakura, W. A. (1989). Measuring consumer attitudes toward the marketplace with tailored interviews. *Journal of Marketing Research*, 26, 311–326.
- Bock, R. D., & Aitkin, N. (1981). Marginal maximum likelihood estimation of item parameters: An application of an EM algorithm. *Psychometrika*, 46, 443–459.
- Böckenholt, U. (1993). Estimating latent distribution in recurrent choice data. *Psychometrika*, 58, 489–509.
- Böckenholt, U., & Böckenholt, I. (1991). Constrained latent class analysis: Simultaneous classification and scaling of discrete choice data. *Psychometrika*, 56, 699–716.
- Bradlow, E., Wainer, H., & Wang, X. (1999). A Bayesian random effects model for testlets. *Psychometrika*, 64, 153–168.
- Cadez, I., Smyth, P., Ip, E., & Mannila, H. (2002). *Predictive profiles for transaction data using mixtures of baskets*. (Technical Report 01-67). Irvine: University of California. (Conditionally accepted for publication by the *Journal of Machine Learning Research*.)
- Cadez, I., Smyth, P., & Mannila, H. (2001). Probabilistic modeling of transaction data with applications to profiling, visualization, and prediction. In *Proceedings of the seventh ACM SIGKDD international conference of knowledge discovery and data mining, 2001* (pp. 37–46). New York: Association for Computing Machinery Press.
- Comon, P. (1994). Independent component analysis—a new concept? *Signal Processing*, 36, 287–314.
- Csiszar, I., & Tusnady, G. (1984). Information geometry and alternating minimization procedures. *Statistics and Decisions Supplement Issue*, 1, 205–237.
- Dayton, C. M., & Macready, G. B. (1976). A probabilistic model for the validation of behavioral hierarchies. *Psychometrika*, 41, 189–204.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society B*, 39, 1–38.
- DeSarbo, W. S., & Hoffman, D. L. (1987). Constructing MDS joint spaces from binary choice data: A multidimensional unfolding threshold model for marketing research. *Journal of Marketing Research*, 24, 40–54.
- Doucet, A., de Freitas, N., & Gordon, N. (Eds.). (2001). *Sequential Monte carlo methods in practice*. New York: Springer.
- Dunmur, A. P., & Titterington, D. M. (1998). Parameter estimation in latent profile models. *Computational Statistics and Data Analysis*, 27, 371–388.

- Fisher, G., & Formann, A. K. (1982). Some applications of logistic latent trait models with linear constraints on the parameter. *Applied Psychological Measurement*, 6, 397–416.
- Formann, A. K. (1982). Linear logistic latent class analysis. *Biometrics*, 24, 71–190.
- Gaffney, S., & Smyth, P. (1999). Trajectory clustering with mixtures of regression models. In *Proceedings of the ACM 1999 conference on knowledge discovery and data mining* (pp. 63–72). New York: ACM Press.
- Goodman, L. A. (1974). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, 61, 215–231.
- Gorsuch, R. L. (1983). *Factor analysis*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Green, P. E., Carmone, F. J., & Wachpress, D. P. (1976). Consumer segmentation via latent class analysis. *Journal of Consumer Research*, 3, 170–174.
- Harman, H. H. (1976). *Modern factor analysis* (3rd ed.). Chicago: University of Chicago Press.
- Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning: Data mining, inference, and prediction*. Berlin: Springer-Verlag.
- Heijden, P. G. M., Dressens, J., & Böckenholt, U. (1996). Estimating the concomitant-variable latent-class model with the EM algorithm. *Journal of Educational and Behavioral Statistics*, 21, 215–229.
- Hofmann, T. (1999). Probabilistic latent semantic indexing. In *Proceedings of the ACM SIGIR conference* (pp. 50–57). New York: ACM Press.
- Hofmann, T. (2001). Learning what people (don't) want. In *Proceedings of the European conference in machine learning* (pp. 214–225). Berlin: Springer-Verlag.
- Hosken, M., & De Boeck, P. (1997). A parametric model for local dependence among test items. *Psychological Methods*, 2, 261–277.
- Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, 13, 411–430.
- Ip, E., & Lalwani, N. (2000). A note on the geometric interpretation of the EM algorithm in estimating item characteristics and student abilities. *Psychometrika*, 65, 533–537.
- Ip, E. (2002). Locally dependent latent trait model and the Dutch identity revisited. *Psychometrika*, 67, 367–386.
- Jedidi, K., Jagpal, H. S., & DeSarbo, W. S. (1997). STEMM: A general finite mixture structural equation model. *Journal of Classification*, 14, 12–50.
- Jennrich, R. I. (2002). A simple general method for oblique rotation. *Psychometrika*, 67, 7–20.
- Jordan, M. I., & Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6, 181–214.
- Kim, B., & Park, K. (1997). Studying patterns of consumer's grocery shopping trip. *Journal of Retailing*, 73, 501–517.
- Lazarsfeld, P. F., & Henry, N. W. (1968). *Latent structure analysis*. New York: Houghton Mifflin.
- Lenk, P., & DeSarbo, W. (2000). Bayesian inference for finite mixtures of generalized linear models with random effects. *Psychometrika*, 65, 93–119.
- Li, H., & Yamanishi, K. (1997). Document classification using a finite mixture model. In *Proceedings of the 35th annual meeting of the Association for Computational Linguistics* (pp. 39–47). New Brunswick, NJ: Association for Computational Linguistics.
- Longford, N. T., & Muthén, B. (1992). Factor analysis for clustered observations. *Psychometrika*, 57, 581–597.
- Lord, F. M. (1980). *Applications of item response theory to practical testing problems*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Lord, F. M., & Novick, M. R. (1968). *Statistical theories of mental test scores*. Reading, MA: Addison-Wesley.
- Luenberger, D. G. (1984). *Linear and nonlinear programming*. Reading, MA: Addison-Wesley.
- Masters, G. N. (1982). A Rasch model for partial credit scoring. *Psychometrika*, 47, 149–174.
- McCallum, A., & Nigam, K. (1998). A comparison of event models for Naïve Bayes text classification. *AAAI-98 Workshop on learning for text categorization*. Menlo Park, CA: American Association for Artificial Intelligence.
- McCullagh, P., & Nelder, J. A. (1989). *Generalized linear models*. New York: Chapman and Hall.
- McLachlan, G. J. (2001, December). Mixture model-based clustering of microarray expression data. Paper presented at the Australian Biometrics and New Zealand Statistical Association Joint Conference. Christchurch, New Zealand.
- McLachlan, G. J., & Basford, K. E. (1988). *Mixture models: Inference and applications to clustering*. New York: Marcel Dekker.
- McLachlan, G. J., & Krishnan, T. (1997). *The EM algorithm and extensions*. New York: Wiley.
- McLachlan, G., & Peel, D. (2000). *Finite mixture models*. New York: Wiley.
- Meila, M., & Jordan, M. (2000). Learning with mixtures of trees. *Journal of Machine Learning Research*, 1, 1–48.
- Mislevy, R. (1988). Exploiting auxiliary information about items in the estimation of the Rasch item difficulty parameters. *Applied Psychological Measurement*, 12, 281–296.
- Mislevy, R., & Bock, D. (1982). *BILOG: Item analysis & test scoring with binary logistic models*. Mooresville, IN: Scientific Software.

- Müthen, B. (2001). Latent variable mixture modeling. In G. A. Marcoulides & R. E. Schumacker (Eds.), *New developments and techniques in structural equation modeling* (pp. 1–33). Mahwah, NJ: Lawrence Erlbaum Associates.
- Müthen, L., & Müthen, B. (1998). *Mplus user's guide*. Los Angeles: Müthen & Müthen.
- Neal, R., & Hinton, G. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in graphical models* (pp. 355–371). Cambridge, MA: MIT Press.
- Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39, 103–134.
- Quandt, R. E., & Ramsey, J. B. (1978). Estimating mixtures of normal distributions and switching regressions. *Journal of the American Statistical Association*, 73, 703–738.
- Rauch, H. E. (1963). Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*, 8, 371–372.
- Redner, R. A., & Walker, H. F. (1984). Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26, 195–239.
- Roberts, J. S., Donoghue, J. R., & Laughlin, J. E. (2000). A general item response theory model for unfolding unidimensional polytomous responses. *Applied Psychological Measurement*, 24, 3–32.
- Roeder, K., Lynch, K., & Nagin, D. (1999). Modeling uncertainty in latent class membership: A case study from criminology. *Journal of the American Statistical Association*, 93, 766–777.
- Rost, J., & Langeheine, R. (1997). A guide through latent structure models for categorical data. In J. Rost & R. Langeheine (Eds.), *Applications of latent trait and latent class models in the social sciences* (pp. 13–37). New York: Waxmann.
- Roweis, S., & Ghahramani, Z. (1999). A unifying review of linear Gaussian models. *Neural Computation*, 11, 305–345.
- Rubin, D., & Thayer, D. (1982). EM algorithms for ML factor analysis. *Psychometrika*, 47, 69–76.
- Saul, L., & Lee, D. (2001). Multiplicative updates for classification by mixture models. In T. G. Kiefferich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems 14*. Cambridge, MA: MIT Press.
- Schneider, T. (2001). Analysis of incomplete climate data: Estimation of mean values and covariance matrices and imputation of missing values. *Journal of Climate*, 14, 853–871.
- Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, 6, 461–464.
- Scott, S., & Ip, E. (2002). Empirical Bayes and item clustering effects in a latent variable hierarchical model: A case study from the National Assessment of Educational Progress. *Journal of the American Statistical Association*, 97, 409–419.
- Shumway, R. H., & Stoffer, D. S. (1982). An approach to time-series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3, 254–264.
- Shumway, R. H., & Stoffer, D. S. (1991). Dynamical linear models with switching. *Journal of the American Statistical Association*, 86, 763–769.
- Stout, W. (1990). A new item response theory modeling approach with application to unidimensional assessment and ability estimation. *Psychometrika*, 54, 661–679.
- Thissen, D. (1991). *MULTILOG version 6 user's guide*. Mooresville, IN: Scientific Software.
- Tipping, M. E., & Bishop, C. E. (1999). Mixtures of probabilistic principle component analyzers. *Neural Computation*, 11, 443–482.
- Titterington, D. M., Smith, A. F. M., & Makov, U. E. (1985). *Statistical analysis of finite mixture distributions*. Chichester, U.K.: Wiley.
- Thurstone, L. L. (1947). *Multiple factor analysis*. Chicago: University of Chicago Press.
- Uebersax, J. S. (1999). Probit latent class analysis: Conditional independence and conditional dependence models. *Applied Psychological Measurement*, 23, 283–297.
- Van de Pol, F., Langeheine, R., & De Jong, W. (1991). *PANMARK user's Manual* (Report number 8493-89-M1-2). Netherlands Central Bureau of Statistics, Department of Statistical Methods.
- Van der Linden, W. J., & Hambleton, R. (1997). *Handbook of modern item response theory*. Berlin: Springer-Verlag.
- Vermunt, J. K., Langeheine, R., & Böckenholz, U. (1999). Latent Markov models with time-contant and time-varying covariates. *Journal of Educational and Behavioral Statistics*, 24, 178–205.
- Vermunt, J. K., & Magidson, J. (2000). *Latent Gold 2.0 User's Guide*. Belmont, MA: Statistical Innovation Inc.
- Wedel, M. (1997). *GLIMMIX user manual*. Groningen, Netherlands: Programma.
- Wedel, M., & Bijmolt, H. A. (1998). *Mixed tree and spatial representations of dissimilarity judgements* (Center Working Paper). Tilburg University, Netherlands.
- Wedel, M., & DeSarbo, W. S. (1998). Mixtures of constrained ultrametric trees. *Psychometrika*, 63, 419–444.
- Yuan, K. H., Marshall, L. L., & Bentler, P. M. (2002). A unified approach to exploratory factor analysis with missing data, nonnormal data, and in the presence of outliers. *Psychometrika*, 67, 95–122.
- Yung, Y. F. (1997). Finite mixtures in confirmatory factor-analysis models. *Psychometrika*, 62, 297–330.

10

Scalable Clustering

Joydeep Ghosh

University of Texas at Austin

Introduction	248
Clustering Techniques: A Brief Survey	249
Partitional Methods	250
Hierarchical Methods	255
Discriminative versus Generative Models	256
Assessment of Results	256
Visualization of Results	258
Clustering Challenges in Data Mining	259
Transactional Data Analysis	259
Next Generation Clickstream Clustering	260
Clustering Coupled Sequences	261
Large Scale Remote Sensing	261
Scalable Clustering for Data Mining	262
Scalability to Large Number of Records or Patterns, N	262
Scalability to Large Number of Attributes or Dimensions, d	264
Balanced Clustering	266
Sequence Clustering Techniques	266
Case Study: Similarity Based Clustering of Market Baskets and Web Logs	267
Case Study: Impact of Similarity Measures on Web Document Clustering	270
Similarity Measures: A Sampler	270
Clustering Algorithms and Text Data Sets	272
Comparative Results	273

This research was supported in part by the NSF under grant ECS-9900353 and by awards from Intel and IBM/Tivoli.

Clustering Software	274
Summary	274
Acknowledgments	274
References	275

INTRODUCTION

Clustering can be defined as the process of grouping a collection of N patterns into distinct segments or clusters based on a suitable notion of closeness or similarity among these patterns. Good clusters show high similarity within a group and low similarity between patterns belonging to two different groups. For applications such as customer or product segmentation, clustering is the primary goal. But more often it is a crucial intermediate step needed for further data analysis, a view underscored by the placement of the cluster utility in the “exploration” stage in Enterprise Miner, a leading data mining software from SAS Institute.

As an applied statistical technique, cluster analysis has been studied extensively for more than 40 years and across many disciplines, including the social sciences (Hartigan, 1975; Jain & Dubes, 1988). This is because clustering is a fundamental data analysis step and a pattern is a very general concept. One may desire to group similar species, sounds, gene sequences, images, signals, or database records, among other possibilities. Thus, in this chapter we use the words *pattern*, *data-point*, *record*, and *object* interchangeably or as appropriate to denote the entities that are clustered. Clustering also has been studied in the fields of machine learning and statistical pattern recognition as a type of unsupervised learning because it does not rely on predefined class-labeled training examples (Duda, Hart, & Stork, 2001). However, serious efforts to perform effective and efficient clustering on large data sets started only in recent years with the emergence of data mining.

Classic approaches to clustering include partitional methods such as k -means, hierarchical agglomerative clustering, unsupervised Bayes, mode finding, or density based approaches. There are also a variety of soft clustering techniques, such as those based on fuzzy logic or statistical mechanics, wherein a data point may belong to multiple clusters with different degrees of membership. The main reason there is such a diversity of techniques is that although the clustering problem is easy to conceptualize, it may be quite difficult to solve in specific instances. Moreover the quality of clustering obtained by a given method is very data dependent, and although some methods are uniformly inferior, there is no method that works best over all types of data sets. Indeed, each approach has certain underlying assumptions about the data and thus is most suitable for specific scenarios, and there is no method that is “best” across all situations. For example, given the correct guess for k and good initialization, k -means clustering works fine if the data is well captured by a mixture of identical, isotropic Gaussians (or Gaussians with identical diagonal covariances if Mahalonobis distance is used), but can degrade rapidly if one deviates from these assumptions. For non-Gaussian distributions in very high dimensional space, k -means clustering is often no better than random partitioning and also regularly turns up near-empty clusters. Conceptual clustering, which maximizes category utility, a measure of predictability improvement in attribute values given a clustering, is popular in the machine learning community, but is most applicable only when data is binary or categorical with small numerosity values.

Because there are many nice books and survey articles on classical clustering methods, including very recent ones (Jain, Murty, & Flynn, 1999), we will only briefly mention them in the following section. Instead, the intent of this chapter is to highlight clustering issues

and techniques most relevant to data mining applications. We illustrate some of the new challenges posed by certain large data sets acquired recently from scientific domains as well as the World Wide Web (WWW), explain some newer requirements such as scalability and balancing, and then describe some of the state-of-the-art techniques developed recently to address these challenges.

CLUSTERING TECHNIQUES: A BRIEF SURVEY

A typical pattern clustering activity involves the following five steps (Jain & Dubes, 1988):

1. Suitable pattern representation
2. Definition of proximity (via a suitable distance or similarity measure) between objects
3. Clustering
4. Data abstraction
5. Assessment of output

The first three steps are unavoidable, whereas the last two are optional and depend on the specific application.

The first step pertains to obtaining features or attributes to appropriately represent each pattern. Features can be binary, quantitative (taking continuous or discrete numeric values), or qualitative (unordered attributes such as color, or ordered ones such as small, medium, and large). If each pattern is a record in a database, the features are already provided as the attributes of the records. In other situations the patterns are in more raw form, for example, as speech signals or images, and one also has to decide on appropriate preprocessing steps such as filtering and representational form such as autocorrelation coefficients and Fourier coefficients. Also, although one may try to use all the available features during the clustering process, it is often prudent to either use a subset of the original features, or use a smaller number of derived features. These processes are called feature selection and feature extraction respectively, and an array of methods are available for both approaches (Duda et al., 2001).

In some cases the features are not made explicit, and a human judgment is relied on to determine the similarity between two patterns. For example, one may ask a human subject to rate the similarity between two smells on a scale between 0, signifying (subjectively) totally different smells, and 1, signifying identical smells. In such situations one is restricted to methods such as multidimensional scaling (Torgerson, 1952), that can work solely from pairwise similarity values. In the most common case, though, each pattern is represented as a point in d -dimensional space. In this case there is an obvious candidate for the choice in Step 2, namely, use the Euclidean distance between two points as a measure inversely related to proximity. In general, distances take values between 0 and infinity, similarity ranges from 0 to 1, and the two concepts are related by a smooth, monotonically decreasing function, such as $\text{similarity} = e^{-(\text{distance})^2}$.

Figure 10.1 shows seven patterns, each represented by two numeric values. Using Euclidean distance to determine similarity, a clustering of these seven points into three groups, $\{1,2\}$, $\{3,4\}$, and $\{5,6,7\}$ seems reasonable. However, if the two coordinates measure qualitatively different features, say age and salary, it is more difficult to convert these measurements to a common d -dimensional vector space of reals, R^d . For example, in Fig. 10.1 one may later realize that one should have compressed the vertical axis by 10 as compared with the horizontal axis. In that case the Patterns 5, 6, and 7 will belong to the same cluster as 1 and 2. Thus, even when each pattern is represented as a d -dimensional point, the Euclidean distance might not be the most suitable. If the data has different spreads along different attributes, one may

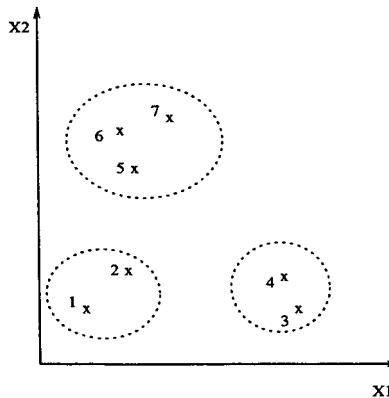


FIG. 10.1. Grouping of seven objects represented in a two-dimensional space of numeric features, into three clusters.

prefer to compensate by using the Mahalanobis distance instead, which normalizes the spread along each attribute to unit variance. If only the direction of the data-points is important, then the cosine of the angle between the two data-vectors may be most appropriate. When the patterns are more complex, such as sequences or trees, determining similarity becomes even more difficult. Because the choice of a distance or similarity measure can profoundly impact cluster quality, one needs to take care in Step 2. To concretize the effect of the proximity measure on cluster quality, we examine this issue in detail in later in the chapter for clustering web documents represented as high dimensional, sparse vectors, and also consider it while clustering sequences.

Let us now move to Step 3, the actual clustering process. There are broadly two types of approaches:

1. *Partitional methods* partition the data set into k clusters. An immediate problem is how to determine the “best” value for k . This is done either by guesswork, by application requirements, or by trying running the clustering algorithm several times with different values for k , and then selecting one of the solutions based on a suitable evaluation criterion.
2. *Hierarchical methods* give a range of nested clusterings obtained through an agglomerative (bottom-up) or divisive (top-down) procedure.

In practical situations one sometimes postprocesses the output of a partitional method by merging or splitting clusters, as in done in ISODATA (Jain et al., 1999). In such methods a flavor of both approaches is found. Let us now look at clustering techniques in a bit more detail.

Partitional Methods

Square-Error Based Hard Clustering Through Iterative Relocation

These algorithms were popular long before the emergence of data mining. Given a set of n objects in a d -dimensional space and an input parameter k , a partitioning algorithm organizes the objects into k clusters such that the total deviation of each object from its cluster representative is minimized.

The two most popular representatives are the k -means algorithm and the k -median algorithm. The k -means algorithm is an iterative algorithm to minimize the least squares error criterion.

A cluster \mathcal{C}_ℓ is represented by its center μ_ℓ , the mean of all samples in \mathcal{C}_ℓ . The centers or cluster representatives are initialized typically with a random selection of k data objects or are the solutions of a k -clustering algorithm on a small subset of the data. Each sample is then labeled with the index ℓ of the *nearest* or *most similar* center. In classical k -means, “nearest” is determined based on the smallest Euclidean distance. Subsequent recomputing of the mean for each cluster (by moving it to the geometric center of all the data points assigned to that cluster in the previous step) and reassigning the cluster labels is iterated until convergence to a fixed labeling after m iterations. The complexity of this algorithm is $O(n \cdot d \cdot k \cdot m)$. The k -median algorithm is very similar except that the cluster representative is constrained to be one of the actual data points rather than the geometric center. This involves extra computations.

The data model underlying the k -means algorithm is a mixture of Gaussians with identical and spherical covariances. Thus, it does best when one makes the correct guess for k and the data is naturally clumped into spherical regions with comparable spreads. If the data actually deviates significantly from this scenario, the cluster quality can be very poor. Still, it remains one of the most popular partitional methods.

The self-organizing map (Kohonen, 1995) is a popular topology combining the clustering algorithm with nice visualization properties. It is related to competitive learning, an on-line version of k -means, in which the input patterns are presented one at a time, and at each step the closest mean to the input under consideration is moved a bit closer to this input. However, the cluster representatives are also *logically* arranged as a simple (typically two-dimensional) topology. This topology induces a neighborhood function on these representatives, and when the cluster representative is adjusted to move it closer to the input currently being considered, it also makes its logical neighbors move, to a lesser extent, in this direction. This mechanism causes logically nearby cluster representatives to cover nearby areas in the input space as well. The resulting mapping from inputs to cluster means is akin to projections onto a principal surface (Chang & Ghosh, 2001). Once the map is formed, each input can be projected onto the nearest representative, and then conveniently visualized in the logical space.

Density Based Methods

Some clustering methods have been developed based on the notion of *density*. These typically regard clusters as dense regions of objects in the feature space that are separated by regions of relatively low density. Density based methods can be used to filter out noise and discover clusters of arbitrary shape. These methods are also called mode-seeking methods, when each cluster is located at a local mode or maxima of the data density function. They are particularly popular for low dimensional applications such as spatial data clustering. Unfortunately, in general the amount of data required to estimate the local density at a given level of accuracy increases exponentially with the number of data dimensions, a manifestation of the famous “curse-of-dimensionality” (Friedman, 1994). So unless a simple¹ parametric form of the data density is known a priori, it is impractical to apply density based methods on data with hundreds of dimensions.

The key role of *scale* is evident in density based methods. Given a finite number of patterns, one can try to model the continuous density function from which these patterns have been drawn at various degrees of smoothing or scale. Cluster centers can then be located at the local maxima of the estimated density function. As the amount of smoothing increases, the number of maxima will decrease. In fact, one can obtain a whole range of clusters, from each point being its own cluster (no smoothing) to one big cluster for the entire data (infinite smoothing).

¹That is, involving a small number of parameters that can be estimated easily.

Scale-space theory says that the salient clusters are those that appear over a wide range of scales or smoothing values.

Many of the clustering algorithms proposed in the 1990s for data mining, largely from the database community, were density based or closely related grid-based approaches. Let us take a quick look at some representative proposals. The reader is referred to chapter 8 of Han and Kamber (2001) and references therein for details. The DBSCAN algorithm judges the density around the neighborhood of an object to be sufficiently high if the number of points within a distance ϵ of an object is greater than $MinPts$ number of points. If this condition is satisfied, all these points are considered to belong to the same cluster. In addition, the region is grown so that points in adjacent regions that also have sufficient density are also considered part of that cluster. Thus, unlike k -means, the clusters are not required to be spherical, but can take arbitrary shapes as long as contiguous regions of adequately high density exist. This property is very useful in low-dimensional applications such as spatial clustering. Unfortunately, the number of clusters discovered depends on the parameters ϵ and $MinPts$ and cannot be determined a priori. So DBSCAN relies on the user's ability to select a good set of parameters. More critically, if the density of data varies significantly from region to region, it can miss natural clusters in a sparser area (even though the density in such clusters is much higher than in the immediate surroundings) while merging clusters in regions of higher density. This is actually a tough problem to solve as it involves looking at the data using a locally adaptive scale. Such a dynamic adaptation capability is provided by the Chameleon algorithm described later.

A cluster ordering method called OPTICS recognizes and leverages the issue of scale in DBSCAN. Rather than producing a data set clustering explicitly, OPTICS computes an augmented *cluster ordering* for automatic and interactive cluster analysis by essentially providing clusters at multiple scales. This is achieved by noting that denser clusters satisfying the $MinPts$ criterion at smaller value of ϵ are subclusters of the clusters obtained using a larger ϵ value. Therefore, one can concurrently obtain and present results to the user at multiple ϵ values and use his feedback to determine the proper threshold levels. Both DBSCAN and OPTICS are $O(N^2)$ algorithms, which can be reduced to $O(N \log N)$ if spatial index structures are used.

A different density based approach is found in DENCLUE, which models the overall density of a point analytically as the sum of the influence functions of data points around it. The influence function is essentially the shape of the window in the classical Parzen windows technique for density estimation. If the influence function chosen is a good model of the data, then somewhat higher dimensional data can be handled. To compute the sum of influence functions efficiently, a grid structure is utilized. Though DENCLUE seems to perform experimentally better than DBSCAN, a careful selection of clustering parameters such as density and noise thresholds is required in addition to suitable choices for the influence functions.

A grid based clustering approach tries to make a density based method more efficient by imposing a grid data structure over the data space. This quantizes the data space into a finite number of cells. As the records are scanned, counts in each cell are updated, and at the end, cells with sufficiently high counts indicate clusters. Both the quality of clustering and the amount of computation required increase with the number of cells in the grid. Often a multiresolution grid is employed because it improves efficiency, particularly in merging high-density cells at a finer level of granularity into a cell at a coarser level. Note that once again there are problems in higher dimensions because, given a fixed number of bins per dimension, the total number of cells increases exponentially with the number of dimensions of the input.

Some examples of the grid based approaches proposed for data mining include STING, which explores statistical information stored in the grid cells; WaveCluster, which clusters objects using a wavelet transform method; and CLIQUE, which represents a grid and density based approach in high-dimensional data space. A grid based approach is usually more efficient

than a direct density based approach, but at the cost of quality loss if the grid structure does not fit the distribution of data.

Graph Based Methods

Graph based methods transform the clustering problem into a combinatorial optimization problem that is solved using graph algorithms and related heuristics. Typically, if N objects are to be clustered, an undirected weighted graph $G = (V, E)$ is constructed the vertices of which are the N points so that $|V| = N$. An edge connecting any two vertices has a weight equal to a suitable similarity measure between the corresponding objects. The choice of the similarity measure quite often depends on the problem domain, for example, Jaccard coefficients for market baskets, normalized dot products for text, and so forth. A set of edges the removal of which partitions a graph into k pair-wise disjoint subgraphs is called an edge separator. The objective in graph partitioning is to find such a separator with a minimum sum of edge weights. Although one strives for the minimum cut objective, the number of objects in each cluster typically is kept approximately equal, so that reasonably balanced (equal sized) clusters are obtained. For $k > 2$, the min-cut problem is NP-complete, and so most of the graph based techniques are approximate solutions or good heuristics. In fact, the most expensive step in clustering via graph partitioning is usually the computation of the $N \times N$ similarity matrix, because very efficient heuristics exist for obtaining the edge separators. When the attributes are sparse, such as the words in a text document, it can be more efficient to compute similarity between each object and its attributes, rather than between all pairs of objects. Then a more efficient coclustering can be achieved by solving the corresponding bipartite graph partitioning problem (Dhillon, 2001).

Some of the well known graph based clustering algorithms include spectral bisectioning, ROCK, and Chameleon (Karypis, Han, & Kumar, 1999). Spectral bisectioning leverages the fact that the second lowest eigenvector of the Laplacian of a graph provides a good indication of the min-cut for a connected graph. It can be recursively used to obtain more than two partitions. ROCK is an agglomerative hierarchical clustering technique for categorical attributes. It uses the binary Jaccard coefficient and a thresholding criterion to form unweighted edges connecting the data points. A key idea in ROCK is to define transitive neighbor relationship, that is, in addition to using simple neighbors (according to the adjacency matrix \mathbf{A}), all pairs having common neighbors (adjacency according to the matrix $\mathbf{A}^T \mathbf{A}$) are also considered neighbors. The common neighbors are used to define interconnectivity between clusters, which is used to merge clusters. Thus, it is akin to the classic shared neighbor clustering concept in pattern recognition. Chameleon starts with partitioning the data into a large number of small clusters by partitioning the v -nearest neighbor graph. In the subsequent stages clusters are merged based on a closeness that is dynamically adjusted to measure localized distances as well as interconnectivity.

A subclass of graph-partition algorithms are those based on *hypergraph partitioning*. A hypergraph is a graph the edges of which can connect more than two vertices (hyperedges). The clustering problem is then formulated as finding the minimum-cut of a hypergraph. A minimum-cut is the removal of the set of hyperedges (with minimum edge weight) that separates the hypergraph into k unconnected components. Again, an object \mathbf{x}_j maps to a vertex v_j . Each feature maps to a hyperedge connecting all vertices with nonzero frequency count of this feature. The weight of this hyperedge is chosen to be the total number of occurrences in the data set. Hence, the importance of a hyperedge during partitioning is proportional to the occurrence of the corresponding feature. The minimum-cut of this hypergraph into k unconnected components gives the desired clustering. Efficient packages such as hMetis exist

(Han, Karypis, Kumar, & Mobasher, 1998) for partitioning. An advantage of this approach is that the clustering problem can be mapped to a graph problem without the explicit computation of similarity, which makes this approach computationally efficient with $O(n \cdot d \cdot k)$ assuming a (close to) linear performing hypergraph partitioner.

Soft Clustering

In soft clustering, a pattern can belong to multiple clusters with different degrees of “association” (Kumar & Ghosh, 1999). This is the natural viewpoint in fuzzy clustering where the degree of membership of a pattern in a given cluster decreases as its distance from that cluster’s center increases, leading to the fuzzy c-means algorithm that has been widely applied.

The expectation-maximization (EM) framework (Dempster, Laird, & Rubin, 1977) as applied to clustering also yields soft partitions. Assume that the n observed points $\{x_i\}_{i=1}^n$ actually come from a underlying parametric distribution that is a mixture of k probability density functions so that for any observed point x ,

$$P(x | \Theta) = \sum_{i=1}^k \alpha_i p_i(x | \theta_i) \quad (1)$$

where the parameter vector $\Theta = (\alpha_1, \dots, \alpha_k, \theta_1, \dots, \theta_k)$ such that $\sum_{i=1}^k \alpha_i = 1$ and each p_i is a density function parameterized by θ_i . The most commonly used component density function is the d -dimensional Gaussian for which $\theta_i = (\mu_i, \Sigma_i)$, where μ_i is the mean and Σ_i is the covariance of the Gaussian representing the i th component, and

$$p_i(x | \theta_i) = \frac{1}{(2\pi)^{d/2} |\Sigma_i|^{1/2}} e^{-\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1} (x - \mu_i)} \quad (2)$$

Each of these components represents a cluster C_i , $i = 1, \dots, k$ and each of the n points is assumed to come from exactly one of the components and hence belong to the corresponding cluster. The clustering problem is to make the best possible estimate of the component densities under certain assumptions and assign a cluster label to each of the data points based on the most likely component to have generated it. Let the set of random variables Z be the indicator functions denoting which point comes from which cluster. The clustering algorithm tries to find the values of the model parameters so as to maximize the log-likelihood of the data, given by

$$\mathcal{L}(\mathcal{X}, Z | \Theta) = \sum_{i=1}^n \log P(x_i, Z | \Theta) \quad (3)$$

Note that unless Z is known, one cannot directly solve the maximum likelihood estimation problem, and, for clustering, Z is obviously not known. The classical way to address this problem is to find the parameters so that the expected value of the log-likelihood is maximized where the expectation is computed over the conditional distribution $p(Z | \mathcal{X}, \Theta)$ of Z given \mathcal{X} and Θ (Dempster et al., 1977). The expected log-likelihood is given by

$$\mathbf{E}\mathcal{L}(\mathcal{X}, Z | \Theta) = \mathbf{E}_{Z|\mathcal{X},\Theta} \mathcal{L}(\mathcal{X}, Z | \Theta) \quad (4)$$

Again, this cannot be directly solved because $p(Z | \mathcal{X}, \Theta)$ is not known. This problem is solved by the (EM) technique that starts from an arbitrary choice of Θ and iteratively improves the estimates of Θ and $p(Z | \mathcal{X}, \Theta)$ while trying to maximize the current estimate of the expected log-likelihood. The technique is guaranteed to converge to a local maxima of the expected log-likelihood. This is a soft clustering method because $p(Z | \mathcal{X}, \Theta)$, gives the soft memberships of points among the clusters.

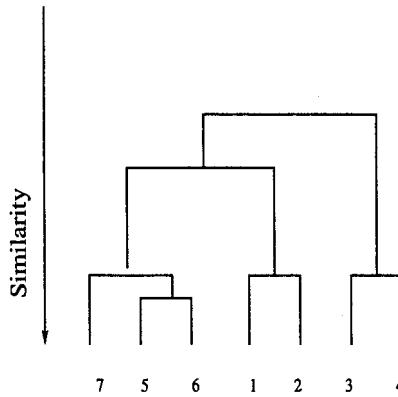


FIG. 10.2. Dendrogram obtained from single link-based agglomerative clustering of the seven objects described in Fig. 1.

Hierarchical Methods

A hierarchical method creates a hierarchical decomposition of the given data objects. The results are often displayed via a dendrogram as in Fig. 10.2. At the leaves each object is a cluster by itself, and the height at which two subclusters are merged signifies the distance between these two groups. So if a cluster does not merge with others over a considerable length along the vertical axis, it is relatively stable. The dendrogram can be formed in two ways: *bottom-up* and *top-down*. The bottom-up approach, also called the *agglomerative* approach, starts with each object forming a separate group. It successively merges the objects or groups that are closest according to some distance measure, until a termination condition is satisfied. Depending on whether the distance between two clusters is measured as the distance between the closest pair (one from each cluster), the farthest pair, or an average among all pairs, one obtains the single link, complete link, and average link variants of agglomerative clustering. Single link is most efficient, because an approximate algorithm is given by the $O(N^2)$ minimum spanning tree solution, however, this tends to give elongated clusters and is sensitive to noise. Complete link and average link methods are more computationally expensive, but yield more compact clusters. The dendrogram shown in Fig. 10.2 is for the single link algorithm. If complete link had been used, the group {3,4} would have fused with {1,2} before the group {5,6,7}.

The top-down approach, also called the *divisive* approach, starts with all the objects in the same cluster. In each successive iteration a cluster is split into smaller clusters according to some measure until a termination condition is satisfied. For example, in bisecting k -means (Steinbach, Karypis, & Kumar, 2000), the whole data set is first partitioned into two groups using 2-means algorithm. Then 2-means is applied to further split the larger of the two groups, and so on until an adequate number of clusters are obtained. Divisive methods are less popular, even though algorithms such as bisecting k -means are quite efficient when the number of clusters required is small, and they often give good results as well.

Earlier hierarchical clustering methods sometimes suffered from the use of oversimplified measures for splitting and merging. Also, the merge or split operation was done irreversibly. This simple rigid approach could result in erroneous clusters being found. To enhance the effectiveness of hierarchical clustering algorithms, some recent methods such as Chameleon (Karypis et al., 1999), utilize more complex and robust principles when splitting or merging the clusters.

Discriminative versus Generative Models

Almost all clustering algorithms essentially try to solve a properly formulated optimization problem. Conceptually speaking, it is helpful to categorize clustering algorithms into *generative* and *discriminative* approaches, based on how the optimization problem is approached.

The EM based soft clustering algorithm described earlier is a prototypical example of the generative approach. This approach assumes that the patterns belonging to the various clusters have been generated by a probabilistic pattern generation process that is dependent on certain parameters. Quite often there is a well-defined mapping between these parameters and the data clusters, though in some cases there is no such explicit mapping. In the soft clustering via EM example, it is assumed that the data points are produced by a mixture of Gaussians, and the natural choice for cluster means are the centers of the component Gaussians. A generative clustering algorithm tries to make the best estimate of the parameters of the underlying generative and then obtains the data clusters using these estimates. Such a framework corresponds to the model based or parametric approaches in classical pattern recognition. Parameter estimation can be done using maximum likelihood (ML), maximum a posteriori (MAP), or mean posterior techniques. It is interesting to note that all three of these estimates converge to the same value of Θ in the limit of large data sets, so this distinction is not terribly important in the data mining context.

In a discriminative approach no assumption is made regarding the source or method of generation of the patterns. However, it is assumed that the patterns exist in a space that has a well-defined distance or similarity measure between any pair of patterns. The patterns are placed into separate groups so that patterns within a cluster have high similarity with one another but are dissimilar to patterns in other clusters, as indicated by a suitable objective function. This roughly corresponds to the memory based or nonparametric approaches in classical pattern recognition. Classical agglomerative approaches are good examples of the discriminative model of clustering. Graph partitioning for clustering also follows this paradigm, trying to minimize the objective

$$\mathcal{J}(U, \bar{U}) = \frac{\text{cut}(U, \bar{U})}{W(U)} + \frac{\text{cut}(U, \bar{U})}{W(\bar{U})}, \quad (5)$$

where $W(U)$ and $W(\bar{U})$ are the sums of the weights of vertices in partition U and \bar{U} , respectively. Depending on how vertex weights are specified, several versions of this objective such as ratio-cut, normalized-cut, and so forth are obtained. Moreover, there are extensions to this formulation for the k -way partitioning case, and some simple variants of this problem can be solved by using max-flow min-cut theorems (Leighton & Rao, 1999). However, for problems of practical interest it is quite often extremely difficult to get exact theoretical solutions. As a result, the algorithms for such formulations often rely on approximations, including intelligent heuristics based on empirical evidence (Karypis & Kumar, 1998).

Assessment of Results

There are two fundamentally different ways of evaluating the quality of results delivered by a clustering algorithm. *Internal* criteria formulate quality as a function of the given data and/or similarities. Typically the quality is measured by the compactness of the clusters, and in some cases by the separation between different clusters as well. For an internal criterion to be valid, the choice of features and the associated similarity measure should be appropriate for the problem being considered. In fact, the ugly duckling theorem (Watanabe, 1969) states the somewhat “unintuitive” fact that there is no way to distinguish between two different classes of

objects, when they are compared over all possible features. As a consequence, any two arbitrary objects are equally similar unless we use domain knowledge. In contrast, *external* criteria judge quality using additional, external information not given to the clusterer, such as class labels.

Internal (Model Based, Unsupervised) Quality

Sum of Squared Errors (SSE). The most popular internal quality measure is SSE, in which error is simply the distance between each data point and the representative (center) of the cluster to which it is assigned. This directly measures the average compactness of the clusters. It can be shown easily that the k -means algorithm tries to greedily optimize this objective and arrives at a local optimum. Note that one can use the transform $e^{-\text{SSE}}$ if a quality measure ranging from 0 to 1 is desired, where 1 indicates a perfect clustering.

The SSE objective also can be viewed from a perspective of a generative model. Assuming the data is generated by a mixture of multivariate Gaussians with identical, diagonal, covariance matrices, the SSE objective is equivalent to maximizing the likelihood of observing the data by adjusting the centers and minimizing weights of the Gaussian mixture.

Edge Cut. When clustering is posed as a graph partitioning problem, the objective is to minimize edge cut. Formulated as a [0, 1]-quality maximization problem, the objective is the ratio of remaining edge weights to total precut edge weights. Note that this quality measure can be trivially maximized when there are no restrictions on the sizes of clusters. Because the edge-cut measure is fair only when the compared clusterings are well-balanced, the min-cut objective is typically modified to penalize highly imbalanced solutions, as done in Equation 5.

Category Utility (Gluck & Corter, 1985). The category utility function measures quality as the increase in predictability of attributes given a clustering. Category utility is defined as the increase in the expected number of attribute values that can be correctly guessed given a partitioning, over the expected number of correct guesses with no such knowledge. A weighted average over categories allows comparison of different sized partitions. Category utility is defined to maximize predictability of attributes for a clustering and is most appropriate for low-dimensional clustering problems, preferably with each dimension being a categorical variable with small cardinality.

Combined Measures. One may employ a combined measure that looks at both compactness within clusters and separation among clusters. Thus, these measures are more comprehensive. A variety of such combined measures is evaluated in Zhao and Karypis (2001).

External (Model Free, Semisupervised) Quality

One drawback of using an internal quality measure is that fair comparisons can be made only among clusterings with the same choices of vector representation and similarity/distance measure. For example, using edge-cut in cosine based similarity would not be meaningful for an evaluation of Euclidean k -means. So in many applications a consensus on the internal quality measure for clustering is not found. However, when the patterns are categorized (labeled) by an external source, an external quality measure can be used. This class of evaluation measures can be used to compare start-to-end performance of any kind of clustering regardless of the models or similarities used and is particularly suitable if the goal of clustering is to indicate the underlying classes. Note that when class conditional densities are multimodal and/or have a high degree of overlaps, the class labels may not represent the natural clusters in the data

well. Also, remember that unlike in classification, the “ground truth” indicated by class labels is not available to the clustering algorithm.

The simplest external evaluation measure is *purity*, interpreted as the classification accuracy under the assumption that all objects of a cluster are classified to be members of the dominant class for that cluster.

A more comprehensive measure than purity is *entropy*. Rather than just considering the number of objects “in” and “not in” the dominant class as in purity, entropy takes the entire distribution into account. However, both purity and entropy are biased to favor a large number of clusters. In fact, for both these criteria, the globally optimal value is trivially reached when each cluster is a single object!

Another alternative is to use precision and recall, which are standard measures in the information retrieval community. If a cluster is viewed as the results of a query for a particular category, then precision is the fraction of correctly retrieved objects, whereas recall is the fraction of correctly retrieved objects out of all matching objects in the database. Moreover, the F-measure can be employed to combine precision and recall into a single number (Baeza-Yates & Ribeiro-Neto, 1999). Unlike purity and entropy, the F-measure is not biased toward a larger number of clusters. In fact, it favors coarser clusterings.

Mutual Information (Cover & Thomas, 1991). A symmetrically defined mutual information between the given set of class labels and cluster labels is a superior measure of external quality. Let the n objects be classified into g categories, with $n^{(h)}$ being the number of objects in category h . Let these objects also be clustered into k groups, and n_ℓ be the number of objects in cluster ℓ . Also let $n_\ell^{(h)}$ denote the number of objects that are in cluster ℓ as well as in category h . Then, a [0, 1]-normalized mutual information-based quality measure is given by Strehl, Ghosh, and Mooney (2000):

$$\Lambda^{(\text{MI})} = \frac{2}{n} \sum_{\ell=1}^k \sum_{h=1}^g n_\ell^{(h)} \log_{k \cdot g} \left(\frac{n_\ell^{(h)} n}{n^{(h)} n_\ell} \right). \quad (6)$$

Mutual information does not suffer from biases like purity, entropy, and the F-measure. Singletons are not evaluated as perfect. Random clustering has mutual information of 0 in the limit. However, the best possible labeling evaluates to less than 1, unless classes are balanced.

External criteria enable us to compare different clustering methods fairly provided the external ground truth is of good quality. For the case study on document clustering later in this chapter, normalized mutual information is our preferred choice of evaluation, because it is an unbiased measure for the usefulness of the knowledge captured in clustering in predicting category labels.

Visualization of Results

One or two dimensional data can be readily visualized in the original feature space. Visualization of higher dimensional data clusters can be largely divided into three popular approaches:

1. Dimensionality reduction by selection of two or three dimensions, or, more generally, projecting the data down to two or three dimensions. Often these dimensions correspond to principal components or an scalable approximation thereof (e.g., Fastmap [Faloutsos & Lin, 1995]). Another noteworthy method is CViz (Dhillon, Modha, & Spangler, 1998), which projects all data points onto the plane that passes through three selected cluster centroids to yield a “discrimination optimal” two-dimensional projection. These projections are useful for

a medium number of dimensions, that is, if d is not too large (<100). An immediate question concerns how much degradation occurs because of the data projection. This is highly data and domain dependent. For text mining, linearly projecting down to about 20–50 dimensions has little effect on results (e.g., latent semantic indexing); projection to lower dimensions leads to substantial degradation, and three-dimensional projections are of very limited utility. Nonlinear projections also have been studied (Chang & Ghosh, 2001). Recreating a two- or three-dimensional space from a similarity graph also can be done through multidimensional scaling (Torgerson, 1952).

2. Parallel axis plots show each object as a line along d parallel axis. However, this technique is rendered ineffective if the number of dimensions d or the number of objects gets too high.

3. Kohonen's self organizing map (SOM) (Kohonen, 1995) provides an innovative and powerful way of clustering while enforcing constraints on a logical topology imposed on the cluster centers. If this topology is two-dimensional, one can readily “visualize” the clustering of data. Essentially, a two-dimensional manifold is mapped onto the (typically higher dimensional) feature space, trying to approximate data density while maintaining topological constraints. Because the mapping is not bijective, the quality can degrade very rapidly with increasing dimensionality of feature space, unless the data is largely confined to a much lower order manifold within this space (Chang & Ghosh, 2001). Multidimensional scaling (MDS) and associated methods also face similar issues.

Several visualization techniques relevant to data mining can be found in chapter 14 of this book. The visualization technique used to display the results of the first case study discussed later in this chapter involves a smart reordering of the similarity matrix. Reordering of data points for visualization also has been used in other contexts, for example, cluster analysis of genome data (Eisen, Spellman, Brown, & Botstein, 1998).

CLUSTERING CHALLENGES IN DATA MINING

In spite of the rich literature, tradition, and broad range of tools, existing methods for clustering are severely challenged by applications involving certain large data sets that are being acquired from scientific domains as well as the WWW.

In addition to scalability, problems may arise because of the difficult nature of the data. For example, data may be very high dimensional, highly noisy, have many outliers, and/or exhibit broad regions where there are no good clusters. Data also may not be readily amenable to a vector representation, for example, sequence data such as web logs, or structured data such as XML documents. Moreover, the clustering applications often pose additional requirements for evaluation, visualization, and actionability of results.

To illustrate the nature of some of these complex characteristics and new requirements, here are four concrete data mining application scenarios:

Transactional Data Analysis

A large market-basket database involves thousands of customers and product lines. Each record corresponds to a store visit by a customer, so a customer can have multiple entries over time. The whole transactional database can be conceptually viewed as a product (feature or attribute) by customer (object) matrix, with the (i, j) -th entry being nonzero only if customer j bought product i in that transaction. In that case the entry represents pertinent information such as

quantity bought or total amount paid. Because most customers buy only a small subset of these products during any given visit, the corresponding feature vector (column) describing such a transaction is high-dimensional (large number of products), but sparse (most features are zero). Also, transactional data is typically highly non-Gaussian. Big customers show up as outliers that are significant and should not be filtered out.

One way to reduce the feature space is to consider only the most dominant products (attribute selection); but in practice this still leaves hundreds of products to be considered. Also, product popularity tends to follow a Zipf distribution with a heavy tail of the less popular products that are not readily ignored because they typically have higher profit margins! A “roll-up” operation to reduce the number of products results in a corresponding loss in resolution or granularity. Feature extraction or transformation typically is not carried out, as derived features lose the semantics of the original ones as well as the sparsity property. Thus, even after preprocessing one is left with clustering non-Gaussian vectors in hundreds of dimensions. Note that the majority of data mining clustering algorithms either approximate k -means or local data densities, and both these approaches are severely hampered by the curse of dimensionality in such spaces.

Current solutions avoid the curse by grossly simplifying the data view. Two years back, when I talked to several large retail stores on behalf of a data mining company (Knowledge Discovery One, now part of Netperceptions), the standard approach was to segment customers based on a macrolevel view, for example, by modeling each person by up to five numbers (recency, frequency, monetary value, variety, and tenure) plus demographics information. Such approaches fail to capture actual purchasing behavior in more complex ways such as taste/brand preferences or price sensitivity. In academia the most popular approach was (and still is) to binarize the data so that each transaction is reduced to an unordered set of the purchased items. Thus, one can use techniques such as the a priori algorithm or its many variants and improvements to determine associations or rules. Unfortunately, this results in loss of vital information: one cannot differentiate between buying 1 gallon of milk and 100 gallons of milk, or one cannot weight importance between buying an apple versus buying a car, though clearly these are very different situations from a business perspective. In fact, none of the clients interviewed found this approach satisfactory!

Clustering transactional data also introduces three new requirements. First, the clusters should be *balanced*, that is, of comparable size according to some measure of importance (number of customers/products, revenue represented, etc.), so that comparable amounts of resources (number of sales teams or marketing dollars, shelf/floor space) can then be allocated to them (Strehl and Ghosh, in press). Because balancing is a global attribute, it is difficult to achieve by locally iterative, greedy methods. Second, each cluster should be easily characterized and the overall results visualized and interpreted by a nontechnical person such as the store manager. This eliminates several clustering and postprocessing choices. Seasonality effects also need to be considered, for example, clustering annual data instead of separately looking at summer and winter patterns can mask important seasonal associations (Gupta & Ghosh, 2001). As a final note, text documents represented as bag-of-words have some similar characteristics of high dimensionality and sparsity, though the clustering problem here is somewhat less severe because vector normalization is effective in this domain, and Latent Semantic Indexing (LSI) can project documents to about 100 dimensions without much degradation.

Next Generation Clickstream Clustering

A visitor to a Web site is characterized by a set of one or more sessions, each session comprising the *sequence* of pages visited, time spent on each page, information typed in, and so forth. Clustering visitors based on such clickstream information helps a Web site to provide customized

content for the users, thereby making it more *sticky* and enhancing user experience. But how can one cluster sets of sequences of entries having both symbolic and numeric attributes? Current commercial solutions (NetGenesis, Tivoli WebLog Analyzer, etc.), store preprocessed log data (and for app-servers, supplementary information such as search queries entered) into a data warehouse where online analytical processing (OLAP) tools are used to look at summary information—essentially first order statistics and breakouts—to answer questions such as how many visitors came to this page, where they came from, and where they went on the next click. Typically, detailed sequence and time information is simply ignored, even though recent studies (Banerjee & Ghosh, 2001) show that these ignored quantities make a substantial difference in cluster quality for certain Web sites. Some research papers also gloss over such details, whereas others are more sophisticated in that they deal with the sequential nature of data through frequent path or Markov model analysis, but do not scale well. So clustering visitors based on rich sequence representations, while being to scale and incrementally adapting to newly acquired data at rates of gigabytes per day, remains an open challenge.

Clustering Coupled Sequences

Tools such as hidden Markov models (HMMs) are widely used for clustering or classifying sequences of both symbolic and numeric data. A fundamental assumption underlying such tools is that the sequences are independently generated. But in several applications sequences are actually quite coupled. For example, consider the different electroencephalogram channels being measured simultaneously from a patient, or the two sets of movement sequences obtained from a dancing (or fighting) pair of humans (Brand, Oliver, & Pentland, 1997; Zhong & Ghosh, 2002). Researchers have started to describe such data types through a variety of coupled HMM formulations. Can one develop scalable algorithms for clustering coupled sequence data? Applications abound, especially in bioinformatics and biomedical processing.

Large Scale Remote Sensing

Determining the type of ground cover based on airborne sensors is a prototypical remote sensing application that has received a new life with the wider and cheaper availability of high resolution hyperspectral data. The inputs to this classification problem are vectors of about 200 dimensions representing energy measured in contiguous, narrow bands measured from each “pixel” (Kumar, Ghosh, & Crawford, 2001). Although exploiting spatial, spectral, and temporal correlations in such high-dimensional data is challenging in itself, the real problem is that providing ground truth requires actual field work, is expensive, and often highly subjective as well. By clustering the data measured from a nearby region and relating these clusters to those obtained from an already labeled region, one can eliminate some of the easier pixels and reduce the costs of incremental ground truthing. Effectively, this translates to a problem of combining multiple clusters without having a guaranteed one-to-one correspondence between clusters or even the underlying ground-cover classes. This problem is related to both active learning and semisupervised learning but is even more difficult because of its dynamic nature: Class distributions and proportions change over time and space.

The four application areas described involve complex data (sets of sequences, coupled sequences, high-dimensional non-Gaussian data) as well as additional requirements or constraints such as balancing, large number of clusters, distributed processing, and presence of legacy clusters. Some other challenges often encountered in data mining applications include presence of mixed attribute types (numerical, binary, categorical, ordinal) within the same

record, high amounts of noise, and domain constraints. Moreover, one may need to employ anytime algorithms that provide a range of cluster quality versus clustering time tradeoffs, or incrementally adaptable algorithms that can quickly modify an existing clustering in response to additional data. All these issues demand more powerful or general approaches to the clustering problem.

SCALABLE CLUSTERING FOR DATA MINING

Scalability in the sense of computational tractability is a central issue for data mining. There are three main aspects of computational scalability.

1. Scalability to large number of records or patterns, N . One would like algorithms that are near-linear in N . If the records are kept in secondary memory, it is also desirable to minimize the number of disk accesses or scans.
2. Scalability to large number of attributes or dimensions, d . High dimensional spaces have peculiar properties that may severely handicap methods that work well in much lower dimensions.
3. Scalable to large number of processors, P . This is relevant if one wants to implement the algorithm on parallel or distributed machines. Ideally, the total computation involved can be split into P near-equal parts that can be executed in parallel with very little communication or other overheads. In such cases near linear speed-ups can be obtained. Fortunately, many clustering algorithms can be readily parallelized in efficient ways (Dhillon & Modha, 1999), so we just focus on the first two aspects.

Scalability to Large Number of Records or Patterns, N

Scalability with respect to number of patterns (records) N , was not an important issue in classical works on clustering but serves as a key distinguishing feature of several recent data mining oriented proposals (Rastogi & Shim, 1999). The holy grail here is to achieve high-quality clustering with a near-linear algorithm that involves only a small number of passes through the database. One can improve scalability from four directions: (a) sequential building, (b) space or data partitioning, (c) sampling, and (d) limiting comparisons.

Sequential Building

These approaches start with a “core” clustering using a small number of records. Then the rest of the data is examined one by one, and in response to each record the cores are either adjusted or new clusters are formed. The leader-clustering algorithm (Jain et al., 1999) is a classical example of the *sequential building* approach. Sequential building is specially popular for out-of-core methods, the idea being to scan the database once to form a summarized model in main memory. Subsequent refinement based on summarized information is then restricted to main memory operations without resorting to further disk scans. In general, sequential building approaches are very suitable for online or streaming data, but the results are typically weaker and also quite sensitive to the sequence of incoming data points.

A nice illustration of how the k -means algorithm can be approximated in this way is given in (Fayyad, Reina, & Bradley, 1998). Recollect that the underlying generative model for k -means is a mixture of Gaussians with the same spherical covariance. Also, the sufficient statistics for

a single Gaussian are given by its mean and variance, both of which can be estimated readily given a finite sample from it, if one knows the number of data points in the sample, the sum of their values, and the sum of the squared values. Finally, this triplet of values can be updated when a new point is added by simply incrementing the number count and adding the value and value-squared of this point to the running sum and sum-squared values, respectively. The scalable method in (Fayyad et al., 1998) exploits all three observations made above. First, as in regular k -means, k seeds are chosen, say by sampling from the database, and the number of points (initially equal to one per cluster), sum, and sum-squared values are initialized for these k “clusters.” As the rest of the data is scanned, if a point is close enough to one of the cluster centers, then the triplet of statistics for that cluster is updated. Otherwise, the data is copied into main memory. So at the end of the scan, one has k sets of triplets, each representing a cluster, as well as some other points that represent outliers or clusters not well captured by the initialization. The hope is that such information can be easily contained in main memory and subsequently refined to yield k clusters. In fact, there may be adequate space to simultaneously update several k -means solutions, each stemming from a different set of initializations, as we scan the data. Subsequently, these solutions can be merged to obtain a single solution that is more robust to initialization. The same authors later also showed how the method could be applied to soft k -means that use the EM algorithm. Further refinements are discussed in (Farnstrom, Lewis, & Elkan, 2000). Note that the goal of having a single pass approximation of the multipass k -means is achieved admirably, but the methods cannot escape the inherent limitations of k -means—that it best applies to numeric data mostly distributed as spherical clouds of about the same size.

Another noteworthy sequential building approach is BIRCH, which stands for balanced, iterative reducing and clustering using hierarchies (Zhang, Ramakrishnan, & Livny, 1997). It uses an index tree that is incrementally built as the data is scanned. With each node of the tree is a triplet of number, sum, and sum-squared values of all the records under that node, which is updated whenever a new record is passed to that node or to its children. Balancing of the tree can be done by splitting a node if it represents too many records. Once again, a single-disk scan and thus highly scalable method is obtained with a desirable incremental update capability, but retaining the limitations of k -means. In addition, the number of clusters cannot be predetermined, so multiple runs or postprocessing is required.

Space or Data Partitioning

Suppose one is given a partitioning of the input space into (nearly) disjoint regions. These regions can be viewed as bins, and the count of the number of records that falls into each bin can be determined with a single database scan. After merging adjacent dense bins, contiguous regions (each comprised of one or more bins) of relatively high density are obtained, and each such region can be viewed as a cluster. Thus, binning itself is a crude clustering technique. The key issue is how to specify the bins. There is a wide variety of choices, but primarily one can try to have bins of equal volumes in feature space (space partitioning) or bins representing roughly equal amounts of data (data partitioning). Note that in databases a variety of mechanisms for efficient indexing based on space *partitioning* (e.g., B-trees, R-trees, and variants) and data partitioning (such as KDB-trees) already exists. In fact, one can view a lower node of such trees as representing a cluster of all the records under it. Thus, it may make sense to use such partitioning structures for clustering as well. In addition, rolling up the data along concept hierarchies also provides natural aggregating mechanisms.

Prior domain knowledge is very helpful for presegmenting the data using indices or other partitioning methods. Data and space partitioning methods are typically tractable for up to

10- to 15-dimensional data, and by a judicious hybrid of these two approaches data with tens of attributes may be partitioned (Chakrabarti & Mehrotra, 1999). Estimating, predetermining, and populating the partitions become increasingly difficult in higher dimensions, where significant overlaps among the hyper-rectangles and/or the occurrences of several empty areas become increasingly common.

Sampling

Sampling can be applied to make a slower algorithm “scalable,” at a possible cost in quality. The idea is to cluster only a subset of the records and then use a quick heuristic to allocate the remaining records to the clusters obtained in the first step. In applications such as document browsing, quality takes a back seat to speed, and sampling based algorithms such as Buckshot for the scatter/gather approach (Cutting, Karger, Pedersen, & Tukey, 1992), have been very useful. Note that if the sample is $O(\sqrt{N})$, then it can be clustered by a quadratic algorithm in $O(N)$ time. Then, if the “nearest cluster center” is used to allocate the nonsampled points, one obtains an overall $O(kN)$ time. Also related are randomized approaches that can partition a set of points into two clusters of comparable size in sublinear time, producing a $(1 + \epsilon)$ solution with high probability (Indyk, 1999). In the data mining context, sampling has been suggested as the way to scalability for CLARANS, CURE, ROCK, and so forth. Sampling is particularly effective for scaling up balanced clustering approaches, as explained in a following section. If the underlying clusters have a comparable number of patterns, then even a modest sample has a high probability of containing representatives from each cluster, thus ensuring that the quality of the clusters obtained using only the sampled data is reasonable (Banerjee & Ghosh, 2002).

Limited Comparisons

Clustering is a global procedure that is, in its common formulations, at least NP-complete if more than two clusters are sought. Iterative heuristics such as k -means and k -medoids are linear in N , because they compare each data point only with cluster representatives rather than with all other points. Consequently, they are very popular but are vulnerable to the initial choices of the cluster representatives. Also, a method based on limited comparisons is handicapped if a global metric such as balancing is also desired.

Scalability to Large Number of Attributes or Dimensions, d

Because dealing with a large number of dimensions is difficult, this aspect is typically addressed by reducing the number of attributes in a preprocessing phase rather than applying the clustering algorithm directly to the high-dimensional data. So let us briefly examine ways of obtaining a derived feature space of reduced dimensionality before looking at the impact of d on different classes of clustering algorithms.

Reduction can be done by either selection of a subset based on a suitable criteria, or by transforming the original set of attributes into a smaller one using linear projections (e.g., principal component analysis [PCA]) or through nonlinear means. Extensive approaches for feature selection or extraction have long been studied, particularly in the pattern recognition community (Duda et al., 2001; Mao & Jain, 1995). If these techniques succeed in reducing the number of (derived) features to the tens or less without much loss of information, then a variety of clustering and visualization methods can be applied to this reduced dimensionality feature space. Otherwise, the problem may still be tractable if the data distribution has a simple

structure that can be well captured by a generative approach. For example, if the features are approximately cluster-conditionally independent, then the data can be reasonably characterized by the superposition of a *small* number of Gaussian components with identical *and* isotropic covariances, in which case k -means can be directly applied to a high-dimensional feature space with good results. If the components have different covariance matrices that are still diagonal (or else the number of parameters will grow quadratically), unsupervised Bayes or mixture-density modeling with EM can be fruitfully applied. Recent results on random projections of mixtures of Gaussians indicate that one can project such distributions to $O(\log k)$ space, estimate the parameters in this projected space, and then project them back to the original space, obtaining excellent results in the process (Dasgupta, 2000).

Another tractable solution is when most of the data can be accounted for by a two- or three-dimensional manifold within the high-dimensional embedding space. Then principal surfaces or nonlinear PCA, self-organizing map (SOM), multidimensional scaling (MDS), or more efficient custom formulations such as FASTMAP (Faloutsos & Lin, 1995), can be applied effectively. A different approach is to assume that the clusters of interest are in regions specified by a small subset of the attributes. This leads to subspace clustering approaches, such as CLIQUE (Agrawal, Gehrke, Gunopulos, & Raghavan, 1998) and CACTUS (Ganti, Gehrke, & Ramakrishnan, 1999). In CLIQUE each dimension is either binary/categorical, or already discretized into bins. The algorithm is based on noting that dense axis-parallel regions in p dimensions have dense projections along any $p - 1$ sized subset of these dimensions. Dense clusters are first found by projecting the data onto the individual dimensions. Then the cartesian product of one-dimensional clusters is examined to identify dense regions in two-dimensional space. Such an approach has the flavor of the Apriori algorithm for finding association rules in its goal of pruning the search space. The analogy of a k item set here is a dense region specified by choices in k attributes. Because the clusters are “rectangular” in nature, they are easy to specify as a rule. However, clearly the algorithm is based on simplistic assumptions on the data, most significant of which are that the clusters of interest are axis-parallel and that the data is either all categorical or is readily discretized. CACTUS is another interesting algorithm specialized for categorical data.

Suppose one already has performed feasible dimensionality reduction at the preprocessing stage such that further reduction will incur significant information loss. If this reduced space still has hundreds of dimensions, the curse of dimensionality issue is formidable. (Friedman, 1994). The vast majority of data mining oriented clustering proposals for numerical data focus on an efficient approximation of k -means, k -medoids, or density estimation. Thus, these approaches inherit the fundamental limitations of the underlying generative data models, and in particular will have difficulty with highly non-Gaussian data in spaces of such dimensionality. Indeed, data mining papers on “scalable, high-dimensional clustering” only report results on simple data types with only up to a few tens of attributes (Agrawal et al., 1998). To be fair, algorithms such as DBSCAN were intended to be used for low-dimensional problems such as spatial clustering and so are fine within their scope.

To avoid clustering in high-dimensional spaces, it is often more tractable to work in a *similarity space* that consists of a similarity value (typically a number between 0 and 1) for each pair of objects, instead of the original feature space. Thus, both graph partitioning and hierarchical clustering can be applied to high-dimensional data as long as a suitable (domain-specific) similarity measure can be found. Another advantage of working in similarity space is that one can more easily cater to more complex data types such as variable-length sequences, records with mixed attributes, and structured data such as XML files, once again assuming that a suitable measure of similarity between pairs of such complex objects can be found. The case studies presented later give specific examples of this approach.

Balanced Clustering

Balancing is usually not a criteria in clustering algorithms, even though several applications prefer this quality. In fact, algorithms such as k -means and single-link agglomerative clustering quite often give clusters with widely varying sizes. But certain approaches tend to give more balanced clusters than others. For example, bisecting k -means strives at balancing by partitioning the larger clusters, whereas the complete and average link variants of agglomerative clustering in general give more balanced results than the single-link algorithm. Approaches such as BIRCH, which limit the diameter of a cluster, and recombination methods such as ISODATA also provide some amount of balancing. Also, as mentioned previously, constrained partitioning of a similarity graph automatically incorporates balancing in the objective function.

Among online methods the most notable approach to balancing is frequency sensitive competitive learning (FSCL), which originally was formulated to remedy the problem of underutilization of parts of a code book in vector quantization (Ahalt, Krishnamurthy, Chen, & Melton, 1990). FSCL introduces a “conscience mechanism” that multiplicatively scales the distortion (distance of the code book vector or cluster representative from the input) by the number of times that exemplar was a winner in the past. Thus, highly winning representatives were discouraged from attracting new inputs. This scheme is quite efficient, and we recently adapted it for clustering documents represented by unit-length bag-of-words vectors, with promising results (Banerjee & Ghosh, 2002).

Balancing is a global attribute, and so conflicts with scalability demands. Moreover, the natural clusters present in the data may not be of equal size. Thus, one should be open to relaxed balancing even when the application demands balanced results. A middle ground between no constraints on balancing (e.g., k -means) and tight balancing (e.g., graph partitioning) can be achieved by overclustering using a balanced algorithm and then merging clusters subsequently.

SEQUENCE CLUSTERING TECHNIQUES

The analysis of variable length sequential patterns is different from analysis of many other data types because they cannot be expressed efficiently as “points” in a finite dimensional vector space. Elements of a sequence can be (a) discrete symbols such as web page accesses and protein and gene sequences, or (b) continuous numeric values such as biological time series (e.g., EEG data), speech signals, and gene expression data. Complex sequence data include those that involve significant long-term temporal dependencies, noise-corrupted measurements, or multiple channels/sequences coupled together (Zhong & Ghosh, 2002). Clustering of sequences is relatively less explored but is becoming increasingly important in data mining applications such as web usage mining and bioinformatics.

Existing sequence clustering approaches for both types of data are best viewed under the discriminative versus generative paradigm explained earlier. *Discriminative approaches* determine a distance or similarity function between sequence pairs and then resort to traditional clustering. For sequences of symbols drawn from a given alphabet, the most commonly used distance functions measure the cost of converting one sequence to the other via elementary operations: *insertion*, in which a particular alphabet is inserted; *deletion*, in which an alphabet is deleted; *substitution*, in which an alphabet is replaced by another; and *transposition*, when two different alphabets are swapped. A popular distance function using these basic operations is the *Levenshtein or edit distance* (Levenshtein, 1965), which allows insertions, deletions, and replacements. In the simplified definition, all operations cost 1. This can be rephrased as “the minimal number of insertions, deletions, and replacements to make two strings equal.” In the literature this problem is in many cases called “string matching with k differences” (Mannila

& Ronakainen, 1997). Other measures include the well-known Hamming distance, the *episode distance*, which allows only insertions, and the *longest common subsequence distance*, which allows only insertions and deletions and will be used in the next section for a case study on clustering Web site visitors based on their weblogs.

For numeric sequences, expert knowledge-based heuristics or dynamic programming techniques such as dynamic time warping often are used. In general, determining a good similarity measure from the continuous sequences is highly domain-dependent and nontrivial. Calculating the similarity measure between each pair of sequences is also computationally inefficient.

An alternative way to cluster sequences of symbols is to pad the sequences such that they all are of equal size. Then each sequence can be viewed as a vector of length equal to the (padded) sequence length, and the components of which are obtained simply from the corresponding positions in the sequence. Then any suitable existing vector clustering algorithm can be applied. Note that padding may introduce artifacts that severely compromise clustering results.

Parametric, model based approaches, on the other hand, attempt to learn *generative models* from the data, with each model corresponding to one particular cluster. The type of model often is specified a priori, such as the mixture of Gaussians (MOGs) or HMMs. The model structure (e.g., number of Gaussians in a MOG model or number of hidden states in an HMM) can be determined by model selection techniques (Li & Biswas, 2000) and parameters estimated using the EM algorithm that optimizes a likelihood criterion. For sequence data, generative model-based clustering (typically invoking HMMs) is a natural fit (Cadez, Gaffney, & Smith, 2000; Smyth, 1997), given the difficulty of constructing feature vectors or otherwise directly finding good similarity measures between sequences.

CASE STUDY: SIMILARITY BASED CLUSTERING OF MARKET BASKETS AND WEB LOGS

In this section I present a case study on clustering customers and products based on transactional data, summarizing the OPOSSUM (optimal partitioning of sparse similarities using Metis) framework described in Strehl and Ghosh, in press). The issues and challenges with this task were discussed under the heading Clustering Challenges in Data Mining. The solution is in the form of a relationship based approach that tries to sidestep the curse of dimensionality by working in a suitable similarity space instead of the original high-dimensional attribute space. This intermediary similarity space can be suitably tailored to satisfy business criteria such as requiring customer clusters to represent comparable amounts of revenue. Efficient and scalable graph-partitioning based clustering techniques are then applied in this space. The output from the clustering algorithm is used to reorder the data points so that the resulting permuted similarity matrix can be readily visualized in two dimensions, with clusters showing up as bands.

The first step is to determine a suitable measure of similarity between two market baskets. The *extended Jaccard* similarity measure between transactions \mathbf{x}_a and \mathbf{x}_b is given by

$$s^{(J)}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^T \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2 - \mathbf{x}_a^T \mathbf{x}_b}, \quad (7)$$

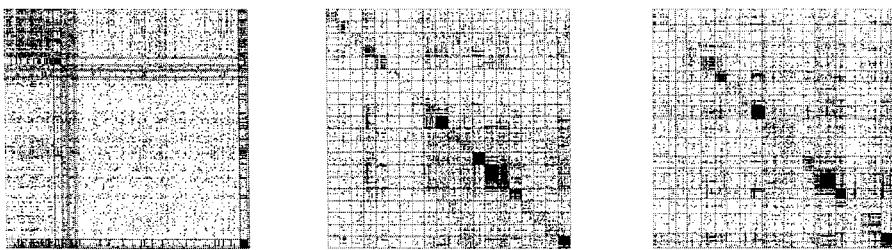
where ^T denotes transpose and the subscript 2 indicates that the L_2 norm is being considered. This turns out to be a very suitable measure for such transactional data. Note that if features are all binary, so that both \mathbf{x}_a and \mathbf{x}_b are reduced to sets of items, the Jaccard coefficient measures the ratio of the cardinalities of the intersection to the union of these two sets.

Given the similarity measures between each pair of transactions, one then forms a similarity graph that is to be partitioned into k pieces as described under the heading Partitional Methods.

Because we want each cluster to be of comparable importance, while striving to find an edge separator with a minimum sum of edge weights that partitions the graph into k disjoint pieces, a balancing constraint is imposed that sets a bound on the size of the largest cluster, where the size can be measured by the number of customers in a cluster or by the amount of revenue corresponding to these customers.

The resultant constrained multiobjective graph partitioning problem is ably handled by the software package Metis (Karypis & Kumar, 1998). It operates in three phases: (a) coarsening, (b) initial partitioning, and (c) refining. First, a sequence of successively smaller and therefore coarser graphs is constructed through heavy-edge matching. Second, the initial partitioning is constructed using one of four heuristic algorithms (three based on graph growing and one based on spectral bisection). In the third phase the coarsened partitioned graph undergoes boundary Kernighan-Lin refinement. In this last phase, vertices are swapped only among neighboring partitions (boundaries). This ensures that neighboring clusters are more related than nonneighboring clusters. Metis is extremely fast and efficient. It has an added benefit of imposing a seriation or ordering of the clusters, which can be exploited to visualize the clustering results in terms of the reordered similarity matrix rendered as a grayscale image.

Figure 10.3 compares the results of (a) k -means with OPOSSUM, which provides clusters of similar size (b) or revenue (c), for $k = 20$. The symmetric customer \times customer matrix is reordered so that customers from the same cluster are contiguous. Cluster boundaries are marked horizontally and vertically with red lines dividing the matrix into k^2 rectangular regions. The gray level at entry (i, j) is proportional to the behavioral similarity between customer i and customer j . Hence, the average amount of grayness in the diagonal and off-diagonal rectangles represents the degree of similarity within and between the corresponding clusters. This information can be used to interactively make postprocessing cluster split/merge decisions; please see www.strehl.com for demonstrations. The numbers below the figures show the ranges of the number of customers/cluster and the total revenue per cluster obtained by the corresponding algorithm. For example, k -means results in one singleton, several very small clusters, and one big cluster of 1,645 customers out of a total of 2,466 customers. OPOSSUM's customer balanced solution (b) yields cluster sizes of 122 to 125, and revenue balanced version (c), obtained by changing only the weight formula, has sizes ranging from 28 to 203 but is very comparable based on total dollar amounts. In fact, Figs. 10.3(b) and (c) indicate superior results to (a) on all three indicators: compactness, isolation, and balancing, even though there is the additional constraint for balancing. Favorable comparative results also are obtained against some other popular clustering methods (Strehl and Ghosh, in press).



(a) [1 – 1645], [\$52 – \$78480] (b) [122 – 125], [\$1624 – \$14381] (c) [28 – 203], [\$6187 – \$6609]

FIG. 10.3. Visualizing partitioning of drugstore customers into 20 clusters based on purchase of 762 products, using k -means (a), customer-balanced OPOS-SUM (b), and OPOSSUM with revenue-balancing (c).

A very compact and useful way of profiling a cluster of products is to look at their most *descriptive* and their most *discriminative* features. For market-basket data this can be done by looking at a cluster's highest revenue products and the most unusual revenue drivers (e.g., products with highest revenue lift). Revenue lift is the ratio of the average spending on a product in a particular cluster to the average spending in the entire dataset.

In Table 10.1 the top three descriptive and discriminative products for the customers in the 20 revenue balanced clusters are shown [see also Fig. 10.3(c)]. Customers in cluster \mathcal{C}_2 , for example, mostly spent their money on smoking cessation gum (\$10.15 on average). Interestingly, although this is a 35-fold average spending on smoking cessation gum, these customers also

TABLE 10.1
List of *Descriptive* (Top) and *Discriminative* (Bottom) Products Dominant in each of the 20 Value
Balanced Clusters Obtained from the Drugstore Data

\mathcal{C}_ℓ	<i>Top Product</i>	\$	<i>Lift</i>	<i>Sec. Product</i>	\$	<i>Lift</i>	<i>Third Product</i>	\$	<i>Lift</i>
1	bath gift packs	3.44	7.69	hair growth m	0.90	9.73	boutique island	0.81	2.61
2	smoking cessati	10.15	34.73	tp canning item	2.04	18.74	blood pressure	1.69	34.73
3	vitamins other	3.56	12.57	tp coffee maker	1.46	10.90	underpads hea	1.31	16.52
4	games items 180	3.10	7.32	facial moisturi	1.80	6.04	tp wine jug ite	1.25	8.01
5	batt alkaline i	4.37	7.27	appliances item	3.65	11.99	appliances appl	2.00	9.12
6	christmas light	8.11	12.22	appliances hair	1.61	7.23	tp toaster/oven	0.67	4.03
7	christmas food	3.42	7.35	christmas cards	1.99	6.19	cold bronchial	1.91	12.02
8	girl toys/dolls	4.13	12.51	boy toys items	3.42	8.20	everyday girls	1.85	6.46
9	christmas giftw	12.51	12.99	christmas home	1.24	3.92	christmas food	0.97	2.07
10	christmas giftw	19.94	20.71	christmas light	5.63	8.49	pers cd player	4.28	70.46
11	tp laundry soap	1.20	5.17	facial cleanser	1.11	4.15	hand&body thera	0.76	5.55
12	film cameras it	1.64	5.20	planners/calend	0.94	5.02	antacid h2 bloc	0.69	3.85
13	tools/accessori	4.46	11.17	binders items 2	3.59	10.16	drawing supplie	1.96	7.71
14	american greeti	4.42	5.34	paperback items	2.69	11.04	fragrances op	2.66	12.27
15	american greeti	5.56	6.72	christmas cards	0.45	2.12	basket candy it	0.44	1.45
16	tp seasonal boo	10.78	15.49	american greeti	0.98	1.18	valentine box c	0.71	4.08
17	vitamins e item	1.76	6.79	group stationer	1.01	11.55	tp seasonal boo	0.99	1.42
18	halloween bag c	2.11	6.06	basket candy it	1.23	4.07	cold cold items	1.17	4.24
19	hair clr perman	12.00	16.76	american greeti	1.11	1.34	revlon cls face	0.83	3.07
20	revlon cls face	7.05	26.06	hair clr perman	4.14	5.77	headache ibupro	2.37	12.65
\mathcal{C}_ℓ	<i>Top Product</i>	\$	<i>Lift</i>	<i>Sec. Product</i>	\$	<i>Lift</i>	<i>Third Product</i>	\$	<i>Lift</i>
1	action items 30	0.26	15.13	tp video comedy	0.19	15.13	family items 30	0.14	11.41
2	smoking cessati	10.15	34.73	blood pressure	1.69	34.73	snacks/pnts nut	0.44	34.73
3	underpads hea	1.31	16.52	miscellaneous k	0.53	15.59	tp irons items	0.47	14.28
4	acrylics/gels/w	0.19	11.22	tp exercise ite	0.15	11.20	dental applianc	0.81	9.50
5	appliances item	3.65	11.99	housewares peg	0.13	9.92	tp tarps items	0.22	9.58
6	multiples packs	0.17	13.87	christmas light	8.11	12.22	tv's items 6	0.44	8.32
7	sleep aids item	0.31	14.61	kava kava items	0.51	14.21	tp beer super p	0.14	12.44
8	batt rechargeab	0.34	21.82	tp razors items	0.28	21.82	tp metal cookwa	0.39	12.77
9	tp furniture it	0.45	22.42	tp art&craft al	0.19	13.77	tp family plan,	0.15	13.76
10	pers cd player	4.28	70.46	tp plumbing ite	1.71	56.24	umbrellas adult	0.89	48.92
11	cat litter scoo	0.10	8.70	child acetamino	0.12	7.25	pro treatment i	0.07	6.78
12	heaters items 8	0.16	12.91	lavenderie ca	0.14	10.49	ginseng items 4	0.20	6.10
13	mop/broom lint	0.17	13.73	halloween cards	0.30	12.39	tools/accessori	4.46	11.17
14	dental repair k	0.80	38.17	tp lawn seed it	0.44	35.88	tp telephones/a	2.20	31.73
15	gift boxes item	0.10	8.18	hearing aid bat	0.08	7.25	american greeti	5.56	6.72
16	economy diapers	0.21	17.50	tp seasonal boo	10.78	15.49	girls socks ite	0.16	12.20
17	tp wine 1.5l va	0.17	15.91	group stationer	1.01	11.55	stereos items 2	0.13	10.61
18	tp med oint liq	0.10	8.22	tp dinnerware i	0.32	7.70	tp bath towels	0.12	7.28
19	hair clr perman	12.00	16.76	covergirl imple	0.14	11.83	tp power tools	0.25	10.89
20	revlon cls face	7.05	26.06	telephones cord	0.56	25.92	ardell lashes i	0.59	21.87

Note: (see also Fig. 10.3(c)). For each item the average number of dollars spent in this cluster and the corresponding lift is given.

spend 35 times more on blood pressure related items, peanuts, and snacks. Do these customers lead an unhealthy lifestyle and are eager to change? Cluster \mathcal{C}_{15} is a highly compact cluster of Christmas shoppers characterized by greeting card and candy purchases. Note that OPOSSUM had an extra constraint that clusters should be of comparable value. This may force a larger natural cluster to split, as may be the case causing the similar clusters \mathcal{C}_9 and \mathcal{C}_{10} . Both are Christmas gift shoppers (Table 10.1[top]), cluster \mathcal{C}_9 are the moderate spenders and cluster \mathcal{C}_{10} are the big spenders, as cluster \mathcal{C}_{10} is much smaller with equal revenue contribution. Our hunch is reinforced by looking at Fig. 10.3(c).

To show its versatility, the above methodology also was successfully used to cluster web documents and web usage sessions, both of which also are represented by sparse, high-dimensional vectors (Strehl & Ghosh, in press; Strehl et al., 2000). However, the overall method is $O(N^2)$, which is a key drawback but can be addressed by sampling and bootstrapping.

CASE STUDY: IMPACT OF SIMILARITY MEASURES ON WEB DOCUMENT CLUSTERING

Document clusters can provide a structure for organizing large bodies of text for efficient browsing and searching. For example, recent advances in Internet search engines (e.g., www.vivisimo.com, www.metacrawler.com) exploit document cluster analysis. For this purpose a document is commonly represented as a vector consisting of the suitably normalized frequency counts of words or terms. This is the “bag-of-words” representation, because the ordering of the words is not remembered. Each document typically contains only a small percentage of all the words ever used. If we consider each document as a multidimensional vector, then the dimension of a document is the size of the vocabulary, often in the tens of thousands. Moreover, the vectors are sparse and have positive ordinal attribute values. Thus, the data set looks similar to the market basket data described in the previous section, with two important differences: (a) The document vectors typically are normalized to unit length, so that longer documents do not dominate short articles; and (b) an external class label often is available for each document and can be used to judge the quality of the clusters. So for such data, what are suitable similarity measures, and how do they affect clustering results for different types of clustering approaches? I now summarize the results of Strehl et al. (2000) to address this question.

Similarity Measures: A Sampler

In this section I introduce several similarity measures relevant to describing the relationship between two documents and illustrate some of their properties. Then I summarize how several types of algorithms perform when using the different measures to see if the chosen measures affect cluster quality. First, consider the following choices for obtaining a measure of similarity between 0 and 1.

Conversion From a Distance Metric

The Minkowski distances $L_p(\mathbf{x}_a, \mathbf{x}_b) = \left(\sum_{i=1}^d |\mathbf{x}_{i,a} - \mathbf{x}_{i,b}|^p \right)^{1/p}$ are the standard metrics for geometrical problems. For $p = 2$ we obtain the Euclidean distance. There are several possibilities for converting such a distance metric (in $[0, \inf]$, with 0 closest) into a similarity measure (in $[0, 1]$, with 1 closest) by a monotonic decreasing function. For Euclidean space, I chose

to relate distances d and similarities s using $s = e^{-d^2}$. Consequently, we define Euclidean [0, 1]-normalized similarity as

$$s^{(E)}(\mathbf{x}_a, \mathbf{x}_b) = e^{-\|\mathbf{x}_a - \mathbf{x}_b\|_2^2} \quad (8)$$

which has important desirable properties (as we will see in the discussion) that the more commonly adopted $s(\mathbf{x}_a, \mathbf{x}_b) = 1/(1 + \|\mathbf{x}_a - \mathbf{x}_b\|_2)$ lacks. Other distance functions can be used as well. The Mahalanobis distance normalizes the features using the covariance matrix. Due to the high-dimensional nature of text data, covariance estimation is inaccurate and often computationally intractable, and normalization is done if need be at the document representation stage itself, typically by applying Term Frequency-Inverse Document Frequency (TF-IDF).

Cosine Measure

A popular measure of similarity for text (which normalizes the features by the covariance matrix), clustering is the cosine of the angle between two vectors. The cosine measure is given by

$$s^{(C)}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^T \mathbf{x}_b}{\|\mathbf{x}_a\|_2 \cdot \|\mathbf{x}_b\|_2} \quad (9)$$

and captures a scale invariant understanding of similarity. An even stronger property is that the cosine similarity does not depend on the length: $s^{(C)}(\alpha \mathbf{x}_a, \mathbf{x}_b) = s^{(C)}(\mathbf{x}_a, \mathbf{x}_b)$ for $\alpha > 0$. This allows documents with the same composition but different totals to be treated identically, which makes this the most popular measure for text documents. Also, due to this property, samples can be normalized to the unit sphere for more efficient processing.

Pearson Correlation

In collaborative filtering, correlation is often used to predict a feature from a highly similar mentor group of objects the features of which are known. The [0, 1] normalized Pearson correlation is defined as

$$s^{(P)}(\mathbf{x}_a, \mathbf{x}_b) = \frac{1}{2} \left(\frac{(\mathbf{x}_a - \bar{\mathbf{x}}_a)^T (\mathbf{x}_b - \bar{\mathbf{x}}_b)}{\|\mathbf{x}_a - \bar{\mathbf{x}}_a\|_2 \cdot \|\mathbf{x}_b - \bar{\mathbf{x}}_b\|_2} + 1 \right), \quad (10)$$

where $\bar{\mathbf{x}}$ denotes the average feature value of \mathbf{x} over all dimensions. Note that this definition of Pearson correlation tends to give a full matrix. Other important correlations have been proposed, such as Spearman correlation [Spearman, 1906], which works well on rank orders.

Extended Jaccard Similarity

The binary Jaccard coefficient measures the degree of overlap between two sets and is computed as the ratio of the number of shared attributes (words) of \mathbf{x}_a AND \mathbf{x}_b to the number possessed by \mathbf{x}_a OR \mathbf{x}_b . For example, given two sets' binary indicator vectors $\mathbf{x}_a = (0, 1, 1, 0)^T$ and $\mathbf{x}_b = (1, 1, 0, 0)^T$, the cardinality of their intersect is 1 and the cardinality of their union is 3, rendering their Jaccard coefficient 1/3. The binary Jaccard coefficient is often used in retail market-basket applications. In Strehl and Ghosh (2000), the authors extended the binary definition of Jaccard coefficient to continuous or discrete nonnegative features. The extended

Jaccard is computed as

$$s^{(J)}(\mathbf{x}_a, \mathbf{x}_b) = \frac{\mathbf{x}_a^T \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2 - \mathbf{x}_a^T \mathbf{x}_b}, \quad (11)$$

which is equivalent to the binary version when the feature vector entries are binary. Extended Jaccard similarity (Strehl & Ghosh, (2000) retains the sparsity property of the cosine while allowing discrimination of collinear vectors as is shown in the following subsection. Another similarity measure highly related to the extended Jaccard is the Dice coefficient ($s^{(D)}(\mathbf{x}_a, \mathbf{x}_b) = \frac{2\mathbf{x}_a^T \mathbf{x}_b}{\|\mathbf{x}_a\|_2^2 + \|\mathbf{x}_b\|_2^2}$). The Dice coefficient can be obtained from the extended Jaccard coefficient by adding $\mathbf{x}_a^T \mathbf{x}_b$ to both the numerator and denominator. It is omitted here because it behaves very similarly to the extended Jaccard coefficient.

Other (Dis-)Similarity Measures

Many other (dis-)similarity measures, such as mutual neighbor or edit distance, are possible (Jain et al., 1999). The similarity measures discussed previously are the ones deemed pertinent to text documents in previous studies (Salton, 1989).

Clustering Algorithms and Text Data Sets

For a comprehensive comparison, one needs to examine several clustering algorithms over multiple data sets. The following results compare four approaches: k -means (KM), graph partitioning (GP), hypergraph partitioning (HGP), and a one-dimensional SOM (see the section titled Partitional Methods for details), with four variants involving different similarity measures each for k -means and graph partitioning. A random partitioning was added as a baseline, thus yielding 11 solutions in total. Four data sets were chosen:

- YAH. This data was parsed from Yahoo! news Web pages (Boley et al., 1999) and can be downloaded from <ftp://ftp.cs.umn.edu/dept/users/boley/> (K1 series). The pages are from 20 categories with highly uneven distributions.
- N20. The data contains roughly 1,000 postings each from 20 newsgroup topics (Lang, 1995). Although these groups are balanced, some of the newsgroups are highly related, whereas others are not. The data can be found for example, at <http://www.csail.mit.edu/~jrennie/20Newsgroups/>.
- WKB. From the CMU Web KB Project (Craven et al., 1998), Web pages from 10 industry sectors according to Yahoo! were selected. Each industry contributes about 10% of the pages.
- REU. The Reuters-21578, Distribution 1.0 is available from Lewis at <http://www.research.att.com/~lewis>. The primary topic keyword was used as the category. There were 82 unique primary topics in the data. The categories are highly imbalanced.

The data sets encompass a large variety of text styles. For example, in WKB documents vary significantly in length, some are in the wrong category, and some are dead links or have little content (e.g., are mostly images). Also, the hub pages that Yahoo! refers to usually are top-level branch pages. These tend to have more similar bag-of-words content across different classes (e.g., contact information, search windows, welcome messages) than news content-oriented pages. In contrast, REU contains well-written news agency messages. However, they often belong to more than one category.

Words were stemmed using Porter's suffix stripping algorithm (Frakes, 1992) in YAH and REU. For all data sets, words occurring on average between 0.01 and 0.1 times per document were counted to yield the term-document matrix. This excludes stop words such as *a*, and very generic words such as *new*, as well as very rare words such as *haruspex*.

Comparative Results

Clustering quality, as measured by mutual information and balance, was used to compare the results of the 11 approaches on the four document data sets. For each data set the number of clusters k was set to be twice the number of categories g , except for the REU data set, where $k = 40$ was used because there are many small categories. Using a greater number of clusters than classes allows multimodal distributions for each class. For example, in an XOR-like problem, there are two classes but four clusters.

For each data set, results were averaged over 10 trials. Every experiment involved a randomly chosen sample of 800 documents from the entire corpus. Figure 10.4 averages the four sets of averaged results to indicate mean performance with standard variation bars. This high-level summary is shown for brevity because the trends are very clear even at this macrolevel. In Fig. 10.4(a), the y-axis indicates quality in terms of mutual information, whereas the x-axis lists the clustering method used, with C, P, J, and E indicating the similarity measure used (cosine, Pearson correlation, Jaccard, or Euclidean). It is clear that cosine, correlation, and Jaccard based graph partitioning approaches work best on text data followed by nonmetric k -means approaches. Clearly, a nonmetric, for example, dot-product based similarity measure, is necessary for good quality. Due to the conservative normalization, depending on the given data set the maximum obtainable mutual information (for a perfect classifier!) tends to be around 0.8 to 0.9. A mutual information-based quality around 0.4 and 0.5, which is approximately 0.3 to 0.4 better than random, is an excellent result. Hypergraph partitioning constitutes the third tier. Euclidean techniques including SOM perform rather poorly. Surprisingly, SOM still delivers significantly better than random results despite the limited expressiveness of the implicitly used Euclidean distances. The success of SOM is explained by the fact that the

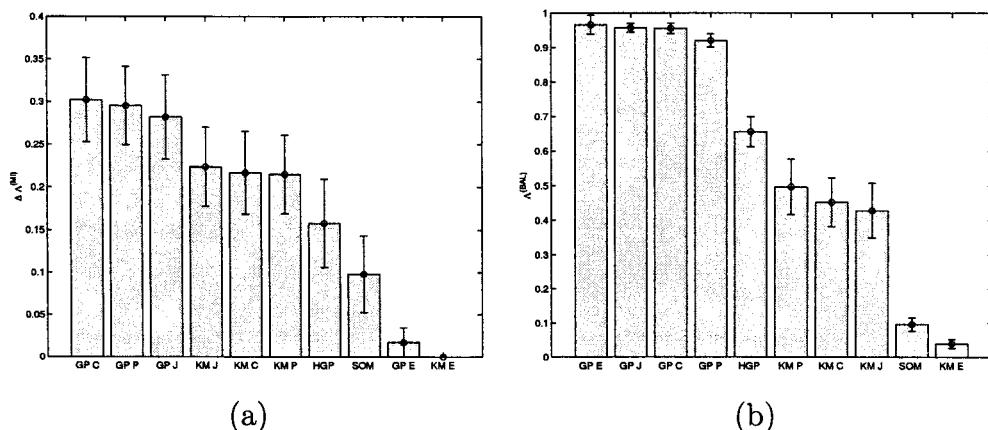


FIG. 10.4. Comparison of cluster quality in terms of (a) mutual information and (b) balance on average over four data sets with 10 trials each at 800 samples. Error-bars indicate ± 1 standard deviation. Graph partitioning is significantly better in terms of mutual information as well as in balance. Euclidean distance based approaches do not perform better than random clustering.

Euclidean distance becomes locally meaningful once the cell-centroids are locked onto a good cluster.

All approaches behaved consistently over the four data sets with only slightly different scale caused by the different data sets' complexities. The performance was best on YAH and WKB followed by N20 and REU. Interestingly, the gap between GP and KM techniques was wider on YAH than on WKB. The low performance on REU is probably due to the high number of classes (82) and their widely varying sizes.

To compare the amount of balancing, the metric used was:

$$\Lambda^{(\text{BAL})} = \frac{n/k}{\max_{\ell \in \{1, \dots, k\}} n_\ell}. \quad (12)$$

Thus, a balance of 1 (highest value possible) indicates that all clusters have the same size. From Fig. 10.4(b), the advantages of graph partitioning for getting balanced results are clear. Graph partitioning explicitly tries to achieve balanced clusters. The second tier is hypergraph partitioning, which is also a balanced technique, followed by nonmetric k -means approaches. Poor balancing is shown by SOM and Euclidean k -means. Interestingly, balance does not change significantly for the k -means based approaches as the number of samples N increases. Graph partitioning based approaches quickly approach perfect balancing as would be expected because they are explicitly designed to do so.

The case study shows that in a specific domain, certain similarity measures can significantly outperform others even when used with the same clustering algorithm. For text clustering, a suitable normalization of the document vectors is very helpful.

CLUSTERING SOFTWARE

Because clustering is a fundamental and widely used routine, it is not surprising that a wide variety of clustering software is available, both commercially and as freeware. Any reasonable statistical or data mining package will have a clustering module. The most common implementations are some variant of k -means and of hierarchical agglomerative clustering. A comprehensive list of public domain online software for clustering can be found at <http://www.pitt.edu/csna/software.html>; <http://www.kdnuggets.com/software/clustering.html> lists some clustering software under the data mining umbrella.

SUMMARY

Clustering is a fundamental data analysis step and has been studied widely for decades in many disciplines. Data mining applications and the associated data sets have brought new clustering challenges such as scalability, working with data in secondary memory, and obtaining comparably sized clusters. This chapter highlighted some of these issues and also described recent advances for successfully addressing them.

ACKNOWLEDGMENTS

I would like to thank Alexander Strehl and Arindam Banerjee for their inputs to this chapter.

REFERENCES

- Agrawal, R., Gehrke, J., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 SIGMOD Conference* (pp. 94–105).
- Ahalt, S. C., Krishnamurthy, A. K., Chen, P., & Melton, D. E. (1990). Competitive learning algorithms for vector quantization. *Neural Networks*, 3, 277–290.
- Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval*. New York: Addison-Wesley.
- Banerjee, A., & Ghosh, J. (2001). Click-stream clustering using weighted longest common subsequences. In *Workshop on web mining: First SIAM conference on data mining* (pp. 33–40). Philadelphia: SIAM.
- Banerjee, A., & Ghosh, J. (2002). On scaling balanced clustering. In *Proceedings of the Second SIAM International Conference on Data Mining* (pp. 333–349). Philadelphia: SIAM.
- Boley, D., Gini, M., Gross, R., Han, E., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., & Moore, J. (1999). Partitioning-based clustering for web document categorization. *Decision Support Systems*, 27, 329–341.
- Brand, M., Oliver, N., & Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 994–999). Los Alamitos, CA: IEEE Press.
- Cadez, I. V., Gaffney, S., & Smyth, P. (2000). A general probabilistic framework for clustering individuals and objects. In *Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining (KDD 2000)* (pp. 140–149). New York: ACM Press.
- Chakrabarti, K., & Mehrotra, S. (1999). The hybrid tree: An index structure for high dimensional feature spaces. In *Proceedings of the 15th International Conference on Data Engineering* (pp. 440–447). New York: ACM Press.
- Chang, K., & Ghosh, J. (2001). A unified model for probabilistic principal surfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23, 22–41.
- Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: Wiley.
- Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the American Association for Artificial Intelligence (AAAI98)* (pp. 509–516). New York: ACM Press.
- Cutting, D. R., Karger, D. R., Pedersen, J. O., & Tukey, J. W. (1992). Scatter/gather: A cluster-based approach to browsing large document collection. In *Proceedings of the 15th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 318–329). New York: ACM Press.
- Dasgupta, S. (2000). Experiments with random projection. In *Proceedings of the 16th Conference on Uncertainty in AI* (pp. 143–151). Menlo Park, CA: AAAI Press.
- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39, 1–38.
- Dhillon, I., Modha, D., & Spangler, W. (1998). Visualizing class structure of multidimensional data. In S. Weisberg (Ed.), *Proceedings of the 30th Symposium on the Interface: Computing Science and Statistics*.
- Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. (Tech. Rep. No. 2001-05). Austin: University of Texas.
- Dhillon, I. S., & Modha, D. S. (1999). A data-clustering algorithm on distributed memory multiprocessors. In *Proceedings of the Large-Scale Parallel Knowledge Systems and Data Mining Workshop, ACM SIGKDD*. New York: ACM Press.
- Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). New York: Wiley.
- Eisen, M. B., Spellman, P. T., Brown, P. O., & Botstein, D. (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Science (USA)*, 95, 14863–14868.
- Faloutsos, C., & Lin, K. (1995). Fastmap: A fast algorithm for indexing, data mining and visualization of traditional and multimedia datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 163–174). San Jose, CA: ACM.
- Farnstrom, F., Lewis, J., & Elkan, C. (2000). Scalability for clustering algorithms revisited. *SIGKDD Explorations*, 2, 1–7.
- Fayyad, U. M., Reina, C., & Bradley, P. S. (1998). Initialization of iterative refinement clustering algorithms. In *Proceedings of the 14th International Conference on Machine Learning (ICML)* (pp. 194–198). San Francisco: Morgan Kaufmann.
- Frakes, W. (1992). Stemming algorithms. In W. Frakes & R. Baeza-Yates (Eds.), *Information Retrieval: Data Structures and Algorithms* (pp. 131–160). Englewood Cliffs, NJ: Prentice Hall.
- Friedman, J. H. (1994). An overview of predictive learning and function approximation. In V. Cherkassky, J. Friedman, & H. Wechsler (Eds.), *From Statistics to Neural Networks, Proceedings of the NATO/ASI Workshop* (pp. 1–61). Springer-Verlag.

- Ganti, V., Gehrke, J., & Ramakrishnan, R. (1999). CACTUS—clustering categorical data using summaries. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining* (pp. 73–83). New York: ACM Press.
- Gluck, M. A., & Corter, J. E. (1985). Information, uncertainty, and the utility of categories. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum Associates.
- Gupta, G. K., & Ghosh, J. (2001). Detecting seasonal trends and cluster motion visualization for very high dimensional transactional data. In *Proceedings of the First SIAM Conference on Data Mining (SDM 2001)* (pp. 115–129). Philadelphia: SIAM Press.
- Han, E.-H., Karypis, G., Kumar, V., & Mobasher, B. (1998). Hypergraph based clustering in high-dimensional data sets: A summary of results. *Data Engineering Bulletin*, 21, 15–22.
- Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.
- Hartigan, J. A. (1975). *Clustering algorithms*. New York: Wiley.
- Indyk, P. (1999). A sublinear-time approximation scheme for clustering in metric spaces. In *Proceedings of the 40th Symposium on Foundations of Computer Science* (pp. 154–159). New York: ACM Press.
- Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Englewood Cliffs, NJ: Prentice Hall.
- Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31, 264–323.
- Karypis, G., Han, E.-H., & Kumar, V. (1999). Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8), 68–75.
- Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20, 359–392.
- Kohonen, T. (1995). *Self-organizing maps*. Berlin: Springer.
- Kumar, S., & Ghosh, J. (1999). GAMLS: A generalized framework for associative modular learning systems. In *SPIE Proceedings Vol. 3722* (pp. 24–34). Bellingham, WA: SPIE.
- Kumar, S., Ghosh, J., & Crawford, M. M. (2001). Best-bases feature extraction algorithms for classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 39, 1368–1379.
- Lang, K. (1995). Newsweeder: Learning to filter netnews. In *International Conference on Machine Learning* (pp. 331–339).
- Leighton, T., & Rao, S. (1999). Muticommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46, 787–832. New York: ACM Press.
- Levenshtein, V. (1965). Binary codes capable of correcting deletions, insertions and reversals. *Problems of Information Transmission*, 1, 8–17.
- Li, C., & Biswas, G. (2000). Improving HMM clustering with Bayesian model selection. In *IEEE International Conference on Systems, Man and Cybernetics*. Piscataway, NJ: IEEE Press.
- Mannila, H., & Ronakainen, P. (1997). Similarity of event sequences. In *Proceedings of the Fourth International Workshop on Temporal Representation and Reasoning* (pp. 136–139).
- Mao, J., & Jain, A. (1995). Artificial neural networks for feature extraction and multivariate data projection. *IEEE Transactions on Neural Networks*, 6, 296–317.
- Rastogi, R., & Shim, K. (1999). Scalable algorithms for mining large databases. In J. Han (Ed.), *KDD '99 Tutorial Notes*. New York: ACM.
- Salton, G. (1989). *Automatic text processing: The transformation, analysis, and retrieval of information by computer*. Reading, MA: Addison-Wesley.
- Smyth, P. (1997). Clustering sequences with hidden Markov models. In M. J. Mozer & T. Petsche (Eds.), *Advances in Neural Information Processing Systems—9* (pp. 648–654). Cambridge, MA: MIT Press.
- Spearman, C. (1906). Footrule for measuring correlations. *British Journal of Psychology*, 2, 89–108.
- Steinbach, M., Karypis, G., & Kumar, V. (2000). A comparison of document clustering techniques. In *KDD Workshop on Text Mining*. New York: ACM Press.
- Strehl, A., & Ghosh, J. (2000). A scalable approach to balanced, high-dimensional clustering of market-baskets. In S. Vajapeyam, M. Valero, & V. Prasanna (Eds.), *Proceedings of the Seventh International Conference on High Performance Computing 2000: Vol. 1970. Lecture Notes in Computer Science (LNCS)* (pp. 525–536). Heidelberg, German, Springer.
- Strehl, A., & Ghosh, J. (in press). Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*.
- Strehl, A., Ghosh, J., & Mooney, R. (2000). Impact of similarity measures on Web-page clustering. In *Proceedings of Seventeenth National Conference on Artificial Intelligence: Workshop of Artificial Intelligence for Web Search* (pp. 58–64). Menlo Park, CA: AAAI.
- Torgerson, W. (1952). Multidimensional scaling, i: Theory and method. *Psychometrika*, 17, 401–419.
- Watanabe, S. (1969). *Knowing and guessing—A formal and quantitative study*. New York: Wiley.

- Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1, 141–182.
- Zhao, Y., & Karypis, G. (2001). *Criterion functions for document clustering: Experiments and analysis* (Tech. Rep. 01-40). University of Minnesota, Minneapolis, MN.
- Zhong, S., & Ghosh, J. (2002). HMMs and coupled HMMs for multi-channel EEG classification. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN '02)* (pp. 1154–1159). Piscataway, NJ: IEEE Press.

11

Time Series Similarity and Indexing

Gautam Das

Microsoft Research

Dimitrios Gunopulos

University of California, Riverside

Introduction	279
Time Series Similarity Measures	281
Euclidean Distances and L_p Norms	281
Normalization Transformations	282
General Transformations	282
Dynamic Time Warping	283
Longest Common Subsequence Similarity	284
Piecewise Linear Representations	287
Probabilistic Methods	288
Other Similarity Measures	288
Indexing Techniques for Time Series	289
Indexing Time Series When the Distance Function Is a Metric	290
A Survey of Dimensionality Reduction Techniques	292
Similar Time-Series Retrieval When the Distance Function Is Not a Metric	299
Subsequence Retrieval	301
Summary	302
References	302

INTRODUCTION

A time series is the simplest form of temporal data. It is a sequence of real numbers collected regularly in time, in which each number represents a value of an observed variable. Time-series data come up in a variety of domains including stock market analysis and environmental,

telecommunications, medical, and financial data. Web data that record the usage (e.g., number of clicks) of different Web sites also are modeled as time series. In fact, time series account for a large fraction of the data stored in commercial databases. In recognition of this fact, the support for time series as a different data type in commercial database management systems has been increasing recently.

The pervasiveness and importance of time-series data has sparked a lot of research on the analysis of such data. Statistics literature on time series is vast and has mainly been focused on problems such as identifying patterns and trend analysis (such as a company's linear growth in sales over the years), seasonality (e.g., winter sales are approximately twice summer sales), and forecasting (e.g., what is the expected sales figure for the next quarter?). These classical topics are discussed in a number of books (e.g., Hamilton, 1994). However, statisticians have not studied methods that would be appropriate for the time series *similarity* and *indexing* problems we discuss here; much of the relevant work on these problems has been done by the computer science community. It is the latter class of problems that this chapter focuses on.

One interesting problem with time-series data is finding whether different time series display similar behavior. More formally, the problem can be stated as follows: given two time series X and Y , determine whether they are similar; in other words, define and compute a similarity function $\text{Sim}(X, Y)$, or equivalently, a distance function $\text{Dist}(X, Y)$. Typically, each time series describes the evolution of an object (for example, the price of a stock) as a function of time at a given data collection station. The objective can be to cluster the different objects to similar groups (e.g., group stocks that have similar price movements) or to classify an object based on a set of known object examples. The problem is hard because the similarity model should allow for imprecise matches. An interesting variation of the similarity problem is the *subsequence similarity* problem, in which for a given a time series X and another shorter *pattern* time series Y , we have to determine those subsequences of X that are similar to pattern Y .

To answer these questions, different notions of similarity between time series have been proposed in data mining research. In this chapter we examine the different time series similarity models that have been proposed and evaluate them based on efficiency and accuracy. The solutions encompass techniques from a wide variety of disciplines, such as databases, signal processing, speech recognition, pattern matching, combinatorics, and statistics. Examples of similarity measures we discuss are those based on Euclidean norms, piecewise linear approximations, dynamic time warping, longest common subsequences, and probabilistic similarity models.

Another problem that is important in applications is the time series *indexing/retrieval* problem: Given a set of time series $\mathcal{S} = \{Y_1, \dots, Y_N\}$ and a query series X , find the time series in \mathcal{S} that are most similar to the query X . As an example of an application, we may wish to find out all the days of the year in which a particular stock had similar movements to those of today. A variation is the *subsequence indexing* problem: Given a set of sequences \mathcal{S} , and a query sequence (pattern) X , find the sequences in \mathcal{S} that contain subsequences that are similar to X . To solve these problems efficiently, appropriate indexing techniques have to be used. The similarity problem is related to the indexing problem: Simple (and possibly inaccurate) similarity measures are usually easy to build indexes for, whereas more sophisticated similarity measures make the indexing problem hard and interesting.

A time series of length n can be considered as a tuple in an n -dimensional space. Indexing this space directly is inefficient because of the very high dimensionality. The main idea is to use a dimensionality reduction technique that takes any n -item time series X , extracts a few key "features," and maps it to a point $f(X)$ in the lower dimensional feature space with k dimensions (hopefully, $k \ll n$), such that the (dis)similarity between X and Y is approximately equal to

the *Euclidean distance* between the two points $f(X)$ and $f(Y)$. One can then use well-known spatial access methods for indexing the lower dimensional feature space (including R-trees [Guttman, 1984], kd-trees [Bentley, 1979], or VP-trees [Yianilos, 1993]).

There are several advantages to dimensionality reduction. First, indexing structures work much better in lower dimensions. The distance computations become much faster because the time to compute the distance is proportional to the number of dimensions. Finally, the overall size of the database becomes smaller, and this can lead to better performance. Clearly, for such techniques to work effectively, the distance computed in the reduced dimensionality space should not diverge too much from the real distance.

Scalability is an important issue, because if similarity measures become more sophisticated, then the indexing problem becomes prohibitive to solve. The research challenge is to design solutions that attempt to strike a balance between accuracy and efficiency. In this chapter we examine the various indexing techniques that can be used for different similarity models and the dimensionality reduction techniques that have been proposed to improve indexing performance. We also give descriptions of the most important techniques used for dimensionality reduction. These include the singular value decomposition, Fourier transform (and the discrete cosine transform), Wavelet decomposition, multidimensional scaling, random projection techniques, FastMap (and variants), and linear partitioning. These techniques have specific strengths and weaknesses, making some of them better suited for specific applications and settings. We also consider the subsequence indexing problem and describe the techniques that have been developed to address it.

The rest of this chapter is organized as follows. In the second section we discuss various similarity measures that have been developed for time series data. In the third section we discuss solutions to the indexing problem.

The reader should note that this chapter is not meant to be a comprehensive survey of time-series similarity. Our emphasis is on discussing some of the important developments and research trends in the area. For more details of time-series similarity, see tutorials by Gunopulos and Das (2000, 2001).

TIME SERIES SIMILARITY MEASURES

Euclidean Distances and L_p Norms

One of the simplest similarity measures for time series is the *Euclidean distance measure*. Assume that both time sequences are of the same length n . We view each sequence as a point in n -dimensional Euclidean space and define the dissimilarity between sequences X and Y as $L_p(X, Y)$, that is, the distance between the two points measured by the L_p norm (when $p = 2$, this reduces to the familiar Euclidean distance).

There are several advantages of such a measure. It is simple to understand and easy to compute. As we shall see in a later section, it also allows scalable solutions for the other problems, such as time-series indexing and clustering. However, there are several crucial disadvantages that make it inappropriate in numerous applications. One major disadvantage is that it does not allow for different *baselines* in the time sequences. For example, consider stock X , which fluctuates around \$100, and stock Y , which fluctuates around \$30. Even though the shapes of both time sequences may be very similar, the Euclidean distance between them will be large. In addition, this measure does not allow for different *scales*. For example, stock X may fluctuate with a small amplitude (between \$95 and \$105), whereas stock Y may fluctuate with a larger amplitude (between \$20 and \$40).

Normalization Transformations

In Goldin and Kanellakis (1995), the authors described a method in which the sequences are *normalized* in an effort to address the disadvantages of the L_p norm as a similarity measure. Let $\mu(X)$ and $\sigma(X)$ be the mean and variance of sequence $X = \{x_1, \dots, x_n\}$. The sequence X is replaced by the normalized sequence X' , where

$$x'_i = (x_i - \mu(X))/\sigma(X)$$

Likewise, the sequence Y is replaced by the normalized sequence Y' . Finally, dissimilarity between X and Y is defined as $L_p(X', Y')$.

This similarity definition resolves the disadvantages of using the L_p norm directly on the unnormalized sequences. For example, consider the two stocks X and Y discussed previously. After normalization both will have the same baseline (because both the means are normalized to 0), and have the same amplitude (because both the variances are also normalized).

However, this normalization process has several disadvantages of its own. For example, it is very sensitive to *phase shifts* in time. As an example, consider two sequences X and Y , where X resembles a sine wave, whereas Y resembles a cosine wave. Both have essentially the same shape except that one is shifted in phase compared to the other. However, the Euclidean distance between the two sequences will be significant (with or without normalization).

Euclidean distances also do not allow for *acceleration and deceleration* along the time axis. For example, suppose two sequences X and Y resemble sine waves, except that the period of X is twice as long than the period of Y . Even with normalization, the Euclidean distance will fail to detect the similarity between the two signals.

General Transformations

Recognizing the importance of the notion of “shape” in similarity computations, an alternate approach was undertaken by Jagadish, Mendelzon, and Milo (1995). In this paper the authors described a general similarity framework involving a transformation rules language. Each rule in the transformation language takes an input sequence and produces an output sequence at a cost that is associated with the rule. The similarity of sequence X to sequence Y is the minimum cost of transforming X to Y by applying a sequence of such rules. The actual rules language is application specific.

For example, consider Fig. 11.1 in which sequences are shown as piecewise linear curves. An example of a transformation rule may be: *collapse adjacent segments into one segment*. The associated cost of this rule may be a function of the lengths and slopes of the new segment and the original segments. Likewise, another rule may exist that replaces a single segment with a pair of adjacent segments.

In Rafiei and Mendelzon (1998) the authors described applications with transformation rules such as *moving averages*, *scales*, and *shifts*. Moving averages are well-known techniques

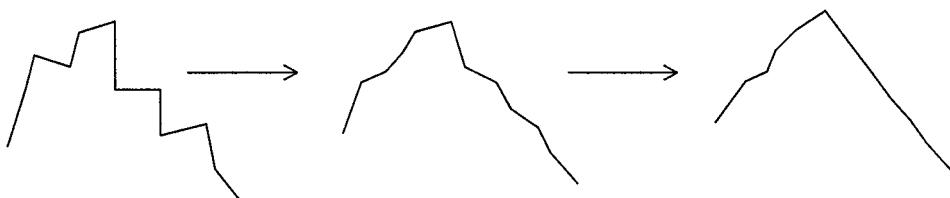


FIG. 11.1. Transformation rules. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

for smoothening time sequences. For example, consider a time sequence X that represents the daily closing price of a stock over a certain number of days. Applying a *3-day moving average* transformation to X results in a sequence X' , where $x'_i = (x_{i-1} + x_i + x_{i+1})/3$. A scale transformation multiplies each element by a scaling constant (i.e., each x_i is replaced by cx_i , where c is a constant). A shift transformation shifts each element by a constant number of positions either to the left or right (i.e., each x_i is replaced by x_{i+c} , where c is an integer constant).

Transformation rules offer a very general mechanism for defining similarity that can be tailored to specific applications. However, they too suffer from some disadvantages. Subsequent computations (such as the indexing problem) become difficult, because extracting features from a particular sequence X becomes complicated, especially if the rules to apply become dependent on particular pairs of sequences X and Y . Also, Euclidean distances in the feature space may not be good approximations of the original sequence dissimilarity.

Dynamic Time Warping

In Berndt and Clifford (1996), the authors introduce the technique of *dynamic time warping* (DTW) to time-series similarity. DTW is an extensively used technique in speech recognition and allows acceleration–deceleration of signals along the time dimension. We describe the basic idea in the following paragraph.

Consider two sequences (of possibly different lengths) $X = \{x_1, \dots, x_m\}$ and $Y = \{y_1, \dots, y_n\}$. We are allowed to extend each sequence by repeating elements. Thus, for example, we may create new sequences $X' = \{x_1, x_2, x_2, x_3, x_4, x_4, x_4, \dots\}$ and $Y' = \{y_1, y_1, y_2, y_2, y_2, y_2, y_3, \dots\}$. The DTW distance, $DTW(X, Y)$, is defined as:

$$DTW(X, Y) = \min_{\forall X', Y' \text{ s.t. } |X'|=|Y'|} L_1(X', Y')$$

Consider Fig. 11.2. It illustrates a *warping path*, which is a pictorial description of the manner in which the two sequences are extended by repeating elements. Thus, the warping path in the example in Fig. 11.2 shows that $X' = \{x_1, x_2, x_2, x_3, x_3, \dots\}$ and $Y' = \{y_1, y_2, y_3, y_3, y_4, \dots\}$.

There are several restrictions on the topology of warping paths. They have to be *monotonic*; that is, they cannot go down or to the left. They are also *continuous*; that is, no elements may be skipped in a sequence. Sometimes, in the interest of computational efficiency, a bounded *warping window* of size w may be defined; that is, for any grid point (i, j) on the warping path, $|i - j| \leq w$.

The algorithmic problem is thus to find the minimum cost warping path. We first give a recursive formulation for DTW. Let $D(i, j)$ refer to the DTW distance between the subsequences

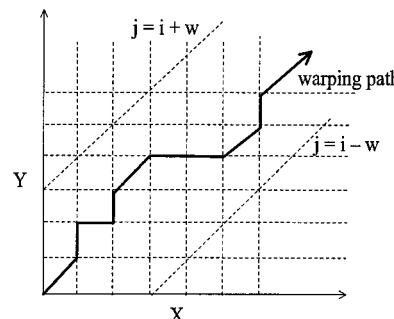


FIG. 11.2. Dynamic time warping. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

$\{x_1, \dots, x_i\}$ and $\{y_1, \dots, y_j\}$. It is easy to see that

$$D(i, j) = |x_i - y_j| + \min\{D(i - 1, j), D(i - 1, j - 1), D(i, j - 1)\}$$

A straightforward algorithm uses a bottom-up *dynamic programming* approach, in which the smaller subproblems $D(i, j)$ are solved first and are then used to solve the larger subproblems, until finally $D(m, n)$ is computed (for a general discussion of dynamic programming, see Cormen, Leiserson, & Rivest, 1990). A basic implementation runs in $O(mn)$ time. If a warping window w is specified, then the running time reduces to $O(nw)$.

The DTW similarity measure suffers from shortcomings such as sensitivity to different baselines and scales of the sequences. These shortcomings can be reduced by normalizing the sequences first as described earlier and then computing $DTW(X, Y)$. However, there are further disadvantages of the *DTW* measure that we discuss next.

Longest Common Subsequence Similarity

The *longest common subsequence* (LCSS) *similarity measure* is a variation of *edit distance* used in speech recognition and text pattern matching. The basic idea is to match two sequences by allowing some elements to be unmatched. The advantages of the LCSS method are twofold: (a) some elements may be unmatched (e.g., *outliers*), where in Euclidean and DTW all elements must be matched, even the outliers; and (b) the LCSS measure allows a more efficient approximate computation, as will be shown later.

A more concrete example is as follows. Consider two sequences:

$$X = \{3, 2, 5, 7, 4, 8, 10, 7\} \quad \text{and} \quad Y = \{2, 5, 4, 7, 3, 10, 8\}.$$

Figure 11.3 illustrates the basic idea of the LCSS measure. Note that certain parts of each sequence are dropped from the matching process (e.g., bursts of noise or outliers). The LCSS of X and Y is $\{2, 5, 7, 10\}$.

Formally, let X and Y be two sequences of length m and n , respectively. As was done with dynamic time warping, we give a recursive definition of the length of the LCSS of X and Y . Let $L(i, j)$ denote the LCSS of the subsequences $\{x_1, \dots, x_i\}$ and $\{y_1, \dots, y_j\}$. $L(i, j)$ may be recursively defined as follows:

$$\begin{aligned} \text{If } x_i = y_j \quad \text{then } L(i, j) &= 1 + L(i - 1, j - 1) \\ \text{else } L(i, j) &= \max\{D(i - 1, j), D(i, j - 1)\} \end{aligned}$$

We define the dissimilarity between X and Y as $LCSS(X, Y) = (m + n - 2l)/(m + n)$, where l is the length of the LCSS. Intuitively, this quantity determines the minimum (normalized) number of elements that should be removed from and inserted into X to transform X to Y . As with DTW, the LCSS measure can be computed by dynamic programming in $O(mn)$ time. This can be improved to $O(nw)$ time if a *matching window* of length w is specified (i.e., where $|i - j|$ is allowed to be at most w).

$$\begin{array}{lll} X & = & \mathbf{3, 2, 5, 7, 4, 8, 10, 7} \\ Y & = & \mathbf{2, 5, 4, 7, 3, 10, 8, 6} \\ \text{LCS} & = & \mathbf{2, 5, 7, 10} \end{array}$$

FIG. 11.3. Longest common subsequence. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

With time-series data the requirement that the corresponding elements in the common subsequence should match exactly is rather rigid. This problem is addressed by allowing some tolerance (say $\epsilon > 0$) when comparing elements. Thus, two elements a and b (from sequences X and Y , respectively) are said to match if

$$a(1 - \epsilon) < b < a(1 + \epsilon)$$

Instead of relative tolerance, we may also define *absolute tolerance*, where b should be within $a - \epsilon$ and $a + \epsilon$.

An LCSS-like measure for time series very similar to the above formulation was discussed in Yazdani and Ozsoyoglu (1996). However, this measure had the usual shortcomings of sensitivity to different scaling factors and baselines. In the next subsections we discuss several other approaches that address these problems.

In general, string edit distances have numerous other variations that are interesting, but have not been adopted in time-series similarity research. For example, instead of single element inserts and deletes, one may even count *block* insert and delete operations (i.e., when entire contiguous subsequences may be inserted or deleted in one step). For a general discussion of string edit distances, see Kruskal and Sankoff (1983).

In the next two subsections we discuss approaches that try to incorporate scaling functions in the basic LCSS similarity measure. There are essentially two types of approaches: those that use local scaling functions (i.e., several different scaling functions, each valid only on certain portions of the sequences), and those that use a single global scaling function.

Using Local Scaling Functions

In Agrawal, Lin, Sawhney, and Shim (1995) the authors developed a similarity measure that resembles LCSS-like similarity with local scaling functions. Here we give only an intuitive outline of the complex algorithm; further details may be found in the paper.

The basic idea is that two sequences are similar if they have enough nonoverlapping time-ordered pairs of contiguous subsequences that are similar. A pair of contiguous subsequences are similar if one can be scaled and translated appropriately to approximately resemble the other. The scaling and translation function is *local*; that is, it may be different for other pairs of subsequences.

Consider Fig. 11.4. Each sequence has been cut into five contiguous subsequences. The first, third, and fifth pairs of corresponding subsequences are similar to each other except that the scaling and translation functions needed to demonstrate their similarity are different for each pair. The second and fourth subsequences of each sequence are considered outliers and eliminated from further consideration.

The algorithmic challenge is to determine how/where to cut the original sequences into subsequences so that the overall similarity is minimized. We describe it in brief here (for further details, refer to Agrawal et al., 1995). The first step is to find all pairs of *atomic subsequences* in the original sequences X and Y that are similar (atomic implies subsequences of a certain small size, say a parameter w). This step is done by a spatial self-join (using a spatial access structure such as an R-tree) over the set of all atomic subsequences. The next step is to “stitch” similar atomic subsequences to form pairs of larger similar subsequences. The last step is to find a nonoverlapping ordering of subsequence matches having the longest match length. The stitching and subsequence ordering steps can be reduced to finding longest paths in a directed acyclic graph, where vertices are pairs of similar subsequences and a directed edge denotes their ordering along the original sequences.

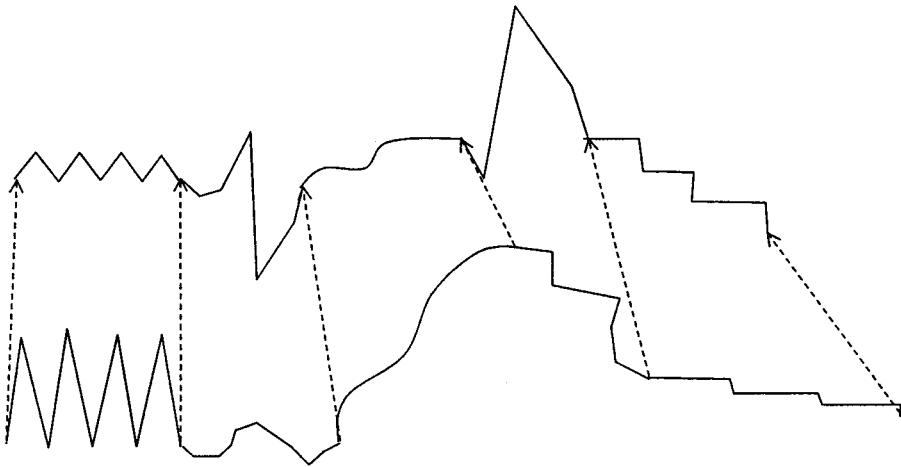


FIG. 11.4. LCSS similarity with local scaling. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

Using a Global Scaling Function

Instead of different local scaling functions that apply to different portions of the sequences, a simpler approach is to try and incorporate a single global scaling function with the LCSS similarity measure. An obvious method is to first normalize both sequences as described earlier, and then apply LCSS similarity to the normalized sequences. However, the disadvantage of this approach is that the normalization function is derived from all data points, including outliers. This defeats the very objective of the LCSS approach, which is to ignore outliers in the similarity calculations.

In Das, Gunopulos, and Mannila (1997) and Bollobas, Das, Gunopulos, and Mannila (2001), an LCSS-like similarity measure is described that derives a global scaling and translation function that is independent of outliers in the data. The basic idea is that two sequences, X and Y , are similar if there exist constants a and b and long common subsequence X' and Y' such that Y' is approximately equal to $aX' + b$. The scale + translation linear function (i.e., the constants a and b) is derived from the subsequences and not from the original sequences. Thus, outliers cannot taint the scale + translation function.

Let us define this measure more formally. Let $\delta > 0$ be an integer constant, $0 < \epsilon < 1$ a real constant, and f a linear function (of the form $f : y = ax + b$) belonging to the (infinite) family of linear functions \mathcal{L} . Given two sequences $X = x_1, \dots, x_n$ and $Y = y_1, \dots, y_n$ of real numbers, let $X' = (x_{i_1}, \dots, x_{i_l})$ and $Y' = (y_{j_1}, \dots, y_{j_l})$ be the longest subsequences in X and Y , respectively, such that

1. for $1 \leq k \leq l - 1$, $i_k < i_{k+1}$ and $j_k < j_{k+1}$,
2. for $1 \leq k \leq l$, $|i_k - j_k| \leq \delta$, and
3. for $1 \leq k \leq l$, $y_{j_k}/(1 + \epsilon) \leq f(x_{i_k}) \leq y_{j_k}(1 + \epsilon)$.

Let $\text{LCSS}_{f, \epsilon, \delta}(X, Y)$ be defined as l/n . Then the final similarity function, $\text{LCSS}_{\epsilon, \delta}(X, Y)$ is defined as

$$\text{LCSS}_{\epsilon, \delta}(X, Y) = \max_{f \in \mathcal{L}} \{\text{LCSS}_{f, \epsilon, \delta}(X, Y)\}$$

Thus, when $\text{LCSS}_{\epsilon, \delta}(X, Y)$ is close to 1, the two sequences are considered to be very similar. The constant δ ensures that the positions of each matched pair of elements are not

too far apart. In practice this is a reasonable assumption, and it also helps in designing very efficient algorithms. Note that an algorithm trying to compute the similarity will have to find out which linear function to use (i.e., the values of a and b) that maximizes l .

Given two sequences of length n and a fixed linear transformation f , $\text{LCSS}_{f,\epsilon,\delta}(X, Y)$ can be computed in $O(n\delta)$ time by an adaptation of the dynamic programming algorithm described earlier. Note that this is essentially linear time. To design algorithms to compute $\text{LCSS}_{\epsilon,\delta}(X, Y)$, the main task is to locate a finite set of all fundamentally different linear transformations f and run $\text{LCSS}_{f,\epsilon,\delta}(X, Y)$ on each one.

Although it appears that the space of all linear transformations is infinite, Das et al. (1997) and Bollobas et al. (2002) show that the number of different unique linear transformations is $O(n^2)$. A naive implementation would be to run $\text{LCSS}_{f,\epsilon,\delta}(X, Y)$ on all transformations, which would lead to an algorithm that takes $O(n^3)$ time. Instead, in Bollobas et al. (2002) an efficient randomized approximation algorithm is proposed to compute this similarity. Let $0 < \beta < 1$ be any desired small constant. The authors first show that a linear transformation f and the corresponding $\text{LCSS}_{f,\epsilon,\delta}(X, Y)$ can be computed in $O(n\delta^3/\beta^2)$ expected time such that $\text{LCSS}_{\epsilon,\delta}(X, Y) - \text{LCSS}_{f,\epsilon,\delta}(X, Y) \leq \beta$. Although the algorithm is extremely efficient and easy to implement, the analysis of its performance is involved, and we refer the reader to the original paper for further details. The authors showed that, of the total $O(n^2)$ possible transformations, a constant fraction of them is almost as good as the optimal transformation. The algorithm picks just a few (constant) number of transformations at random and tries them out. Thus, overall expected running time to get an approximate answer is $O(n)$.

Piecewise Linear Representations

In this subsection we discuss a class of approaches to time-series similarity that are based on first approximating each sequence as a piecewise linear function and then computing a distance between the two respective functions.

Consider a time series $X = \{x_1, x_2, \dots, x_n\}$. It may be regarded as a discrete function $X(t)$, with the function taking values for $t = \{0, 1, \dots, n\}$. We approximate $X(t)$ by a piece-wise linear function $X'(t)$ defined over the continuous time domain $t = [0, n]$, where the function is composed of a small (say k) number of linear segments.

Approximating a time series by a piecewise linear function is a form of data compression because several contiguous data points may be represented by a single line segment. There are several critical issues in selecting a suitable piecewise linear approximation. One is the selection of the parameter k , that is, the number of linear pieces in the function. If k is too small, then many features of the time series are lost. If k is too large, then much redundant information is retained. The second issue is, given k , how to select the best fitting segments? Typically, this is framed as an optimization problem in which the objective is to select the segments such that some error function, for example, L_1 or L_2 distance between X and X' (computed over the discrete time domain) is minimized. Efficient algorithms for these problems were pioneered in Pavlidis (1973) and Horowitz and Pavlidis (1976) and further studied in Keogh, Chu, Hart, and Pazzani (2001).

Once the piecewise linear approximations for each time sequence have been derived, the similarity between them can be defined in a number of ways. A straightforward measure of dissimilarity is to compute the L_1 or L_2 distance between the two approximations. In Keogh and Pazzani (1998), a more refined approach is investigated, in which different parts of the functions are weighed differently in the similarity calculations.

A special case of piecewise linear approximations is the notion of piecewise *constant* approximations. Here a time series $X(t)$ is approximated by a piecewise linear function $X'(t)$

where each segment is horizontal; thus, there are discontinuities at the end points of these segments. These approaches have been investigated very recently in Chakrabarti and Merhotra (2000); Keogh, Chakrabarti, Pazzani, and Merhotra (2000, 2001); and Yi and Faloutsos (2000). One of the advantages of these approaches is that the indexing problem for time series can be solved very efficiently.

Probabilistic Methods

A radically different class of approaches to time-series similarity is the class of *probabilistic similarity measures*. These measures have been studied in Keogh and Smyth (1997) and Ge and Smyth (2000). Although previous methods were “distance” based, some of these methods are “model” based. Because time-series similarity is inherently a fuzzy problem, probabilistic methods are well-suited for handling noise and uncertainty. They are also suitable for handling scaling and offset translations. Finally, they provide the ability to incorporate prior knowledge into the similarity measure. However, it is not clear whether other problems such as time-series indexing, retrieval, and clustering can be efficiently accomplished under probabilistic similarity measures.

Here we briefly describe the approach in Ge and Smyth (2000) (see original paper for further details). Given a sequence X , the basic idea is to construct a probabilistic generative model M_X , that is, a probability distribution on waveforms. Consider a time series X , which is first approximated as a piecewise linear function as described earlier. A discrete-time Markov model is one in which each segment in the data corresponds to a state of the model. For each state (i.e., segment), data is typically generated by a regression curve. A state-to-state transition matrix is provided. On entering state i , a duration t is drawn from a state-duration distribution $p(t)$. The process remains in state i for time t , after which the process transits to another state according to the state transition matrix. For further details of modeling of the distribution $p(t)$, please refer to Ge and Smyth (2000).

Figure 11.5 shows a two-state Markov model. The solid line segments represent the two states, whereas the dashed lines represent the actual noisy observations.

Once a model M_X as been constructed for a sequence X , we can compute similarity as follows. Given a new sequence pattern Y , similarity is measured by computing $p(Y | M_X)$, that is, the likelihood that M_X generates Y .

Other Similarity Measures

In this subsection we briefly summarize other similarity measures that have been investigated. In Keogh and Pazzani (1999b) the technique of *relevance feedback* is employed to incorporate a

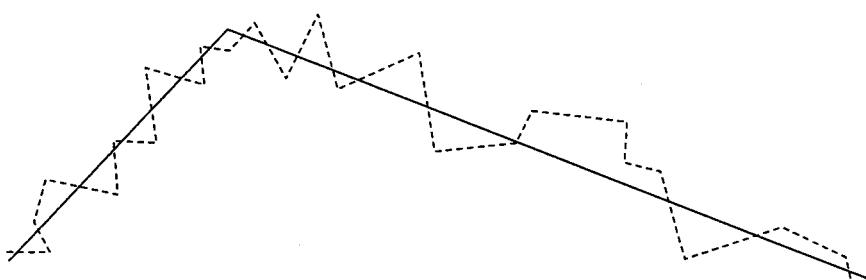


FIG. 11.5. Output of Markov model. (Gunopulos, D., & Das, G. (2000, August). Time series similarity measures. Paper presented at the ACM SIGKDD Conference, Boston.)

user's subjective notion of similarity. This similarity notion can be learned continually through user interaction. The paper describes a piecewise linear partitioning time series representation technique, develops a *merge* operation on these time-series representations, and uses relevance feedback to refine query shapes.

In the *landmarks* approach in Perng, Wang, Zhang, and Parker (2000) the authors observed a similarity definition that is closer to human perception than the Euclidean distance. In this approach a time series is viewed as a curve, and a point on the curve is an n th order landmark if the n th derivative is 0. Thus, local maxima and minima are first order landmarks. Landmark distances are tuples (e.g., in time and amplitude) that satisfy the triangle inequality. Several transformations are defined, such as shifting, amplitude scaling, and time warping.

INDEXING TECHNIQUES FOR TIME SERIES

In many domains it is important to retrieve quickly the most similar time series to a query time series. The problem comes up in many applications, such as finding stocks that behave similarly to a given stock, finding products with similar demand cycles, or finding gene expression patterns that are similar to the expression pattern of a given gene. The applications may need the retrieval mechanism for classifying the given time series using the nearest neighbor classification technique, for improving the running time performance of clustering algorithms, or for exploratory analysis of the data.

The problem of indexing of time series has attracted great attention in the research community, especially in the last decade, and this can be attributed partly to the explosion of database sizes. For example, consider environmental data collected daily or satellite image databases of the earth (www.terraserver.com). If we would like to allow the user to explore these vast databases, we should organize the data in such a way that one can retrieve accurately and efficiently the data of interest.

Formally, we can define the the *time-series retrieval problem* as follows. Given a database \mathcal{D} of time series, describe a preprocessing technique that allows the user to find efficiently the sequence $X \in \mathcal{D}$ that is closest to a given query sequence Q (not necessarily in the database).

To address the problem, we need to define the following:

1. A distance function, which will match the user's perception of what is considered similar.
2. An efficient indexing scheme, which will speed up the user's queries.

In the previous section we discussed different approaches for defining the similarity (or, equivalently, the distance) between two time series. The simplest approach is to define the distance between two sequences by mapping each sequence into a vector and then use an L_p -norm to calculate their distance. The L_p -norm distance between two n -dimensional vectors \bar{x} and \bar{y} is defined as: $L_p(\bar{x}, \bar{y}) = (\sum_{i=1}^n (x_i - y_i)^p)^{\frac{1}{p}}$. For $p = 2$ it is the well-known Euclidean distance and for $p = 1$ the Manhattan distance. Although such a distance measure can be calculated efficiently, it is very sensitive to small variations in the time axis and does not perform well in the presence of noisy data. As indicated, more flexible ways to describe similar sequences include DTW and the LCSS model.

There are several variations of the basic problem; for example, the user may be interested in either whole sequence matching or in subsequence matching. If, for example, the user is looking for appearances of a relatively short pattern, then the subsequence matching is of

interest. Different distance measures may be needed in different applications. Finally, the user may be interested in K -nearest queries (find the K -most similar time series), in range queries (find the time series that are within ϵ distance from the query), or in joins (for each time series find the most similar series).

In the following, we mainly consider the following situation:

1. The dissimilarity (or distance) function D between time series obeys the triangle inequality: $D(A, B) < D(A, C) + D(C, B)$ for time series A, B, C .
2. The query time series Q is a full-length time series.
3. The user asks for the nearest neighbors (the most similar time series to the query time series Q).

We also briefly examine the case in which the distance function is not a metric.

Indexing Time Series When the Distance Function Is a Metric

Indexing refers to the organization of data in a way that not only groups together similar items, but also facilitates the pruning of irrelevant data. The second part is greatly affected by whether our distance function is metric.

Definition 1. A distance function $d(X, Y)$ between two objects X and Y is metric when it complies to the following properties:

1. *Positivity*, $d(X, Y) \geq 0$ and $d(X, Y) = 0$ iff $X = Y$
2. *Symmetry*, $d(X, Y) = d(Y, X)$
3. *Triangle Inequality*, $d(X, Y) + d(Y, Z) \geq d(X, Z)$

The Euclidean distance is a metric distance function. The pruning power of such a function can be shown in the following example (Fig. 11.6).

Example. Assume that we are given the set of sequences $\mathcal{S} = \{\text{Seq1}, \text{Seq2}, \text{Seq3}\}$. Let us also assume that we have precomputed the pairwise distances between the three sequences and have stored them in a table.

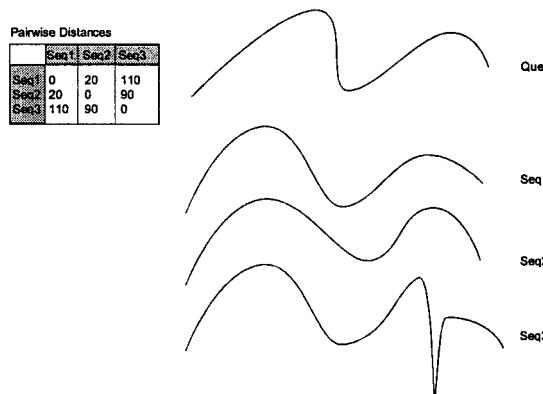


FIG. 11.6. Pruning power of triangle inequality.

Given a new query Q , the most straightforward approach to find the most similar of the three sequences is a sequential scan, that is, to compute the distance of Q with each one of the three sequences and select the one with the shortest distance. If the distance function obeys the triangle inequality, this approach can be improved by observing that if the distance $D(Q, Seq1) = 20$ and $D(Q, Seq2) = 130$; then, because $D(Q, Seq3) \geq D(Q, Seq2) - D(Seq2, Seq3) \Rightarrow D(Q, Seq3) \geq 130 - 90 = 40$, we can safely prune Seq3, because it is not going to offer a better solution.

To further improve the performance of the search, we must use a multidimensional index. Multidimensional indices, and the problem of efficiently finding nearest neighbors in high-dimensional spaces, has attracted a lot of attention in theoretical computer science (see, for example, the survey of Agrawal and Ericson, 1998, and Indyk and Motwani, 1998, for approximate nearest neighbor techniques) as well as in database research. Many index structures have been proposed for disk-resident data. These include R-trees (Guttman, 1984) and their variants such as R+-trees (Sellis, Roussopoulos, & Faloutsos, 1987) and R*-trees (Beckmann, Kriegel, Schneider, & Seeger, 1990); kd-trees (Bentley, 1979), and K-D-B-trees (Robinson, 1981); TV-trees (Lin, Jagadish, & Faloutsos, 1994); SS-trees (White & Jain, 1996); VP-trees (Yianilos, 1993) and their variants, metric-trees (Traina, Traina, Seeger, & Faloutsos, 2000). Such indexing structures store multidimensional points in external memory and are optimized mainly for answering range queries. Roussopoulos, Kelley, and Vincent (1995) and Hjaltason and Samet (1999) showed how to answer nearest neighbor queries with an R-tree or similar structure.

One naive indexing approach is to consider a time series of length n as an n -dimensional point. We can then store each time series as points in an n dimensional R-tree. To answer nearest neighbor queries, we also consider the query time series as an n -dimensional point, and use the index structure (R-tree, for example) to find those time series that are close to the query.

Unfortunately, this idea does not work very well in practice because due to their length time series typically make very high-dimensional points. The performance of the various index structures degrades quickly when the dimensionality increases (Weber, Schek, & Blott, 1998). As a result, this approach may be even slower than sequential scan.

Using Dimensionality Reduction

The key idea for efficient indexing of time series is to reduce the dimensionality of the space. To achieve that, we project the n -dimensional tuples that represent the time series in a k -dimensional space (with $k \ll n$) so that the distances are preserved as well as possible. We then can use an indexing technique on the new (smaller dimensionality) space.

A general framework, GEMINI, for indexing time series using dimensionality reduction techniques is presented in Faloutsos (1996). It uses the following steps:

1. Map the set of time series into a reduced dimensionality space.
2. Use an indexing technique to index the new space.
3. For a given query time series Q (a) map the query Q to the new space; (b) find nearest neighbors to Q in the new space using the index; and (c) compute the actual distances and keep the closest.

This approach is illustrated in Fig. 11.7.

To make sure that we will not actually miss the nearest time series (i.e., avoid false dismissals) we must be able to bound how much error the dimensionality reduction technique

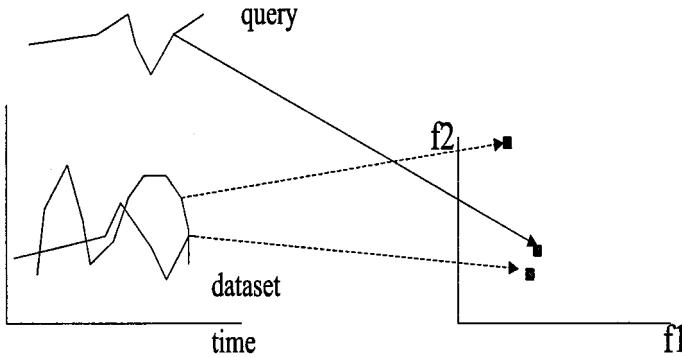


FIG. 11.7. A dimensionality reduction example: using two features to represent a time series. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

may introduce. In other words, we must be able to prove that for a given dimensionality reduction technique F , and for time series S, T : $D(F(S), F(T)) < a \cdot D(S, T)$ for some positive constant a . Here $D(F(S), F(T))$ is the distance in the new space and $D(S, T)$ is the original distance. Note that having a small rate of false positives (that is, time series that seem to have small distance in the reduced dimensionality space but are in fact dissimilar in the original space) is desirable for the efficiency of the technique, but not essential for the accuracy.

The advantages of the approach are that indexing structures work much better in lower dimensionality spaces, the distance computations run faster in the reduced dimensionality space, and the size of the data set is reduced, further improving performance.

In the next section we give a survey of dimensionality reduction techniques that can be applied to the general GEMINI framework described above.

A Survey of Dimensionality Reduction Techniques

Dimensionality reduction techniques compute a smaller representation of the initial data set. Although this representation is typically “lossy” because some information is lost when a smaller representation is chosen, dimensionality reduction techniques attempt to retain as much of the original structure as possible.

We could distinguish two general categories:

- *local or shape preserving*
- *global or topology preserving*

The first category includes methods that do not try to exploit the global properties of the data set but rather attempt to “simplify” the representation of each sequence regardless of the rest of the data set.

The selection of the k features should be such that the selected features retain most of the information of the original signal. For example, these features could be either the first coefficients of the Fourier decomposition, or the wavelet decomposition, or a piecewise constant approximation of the sequence.

The second category of methods has been used mostly for visualization purposes (without, of course, limiting its usefulness to only this application), and their primary purpose is to discover a parsimonious spatial representation of the objects. They differ from the previous

approaches because they aim to discover k features that minimize a *global* objective function. A typical problem addressed by this category is the following:

Suppose we have a table that contains the pairwise distances between major U.S. cities. Using only this information, can we place the cities as points on a two-dimensional map so the distances are preserved as closely as possible?

This problem can be addressed using multidimensional scaling (Kruskal & Wish, 1978), although the result is not going to be exactly like a U.S. map, because the points might have an arbitrary orientation.

Most of the original applications of this category studied the dimensions of interpersonal perception. They have originated from the behavioral sciences while attempting to visualize and analyze data from subjective responses (e.g., examining the perceived similarity of English letters or Morse code symbols, after analyzing how similar people considered each pair of observations on some scale).

Other global techniques include the SVD/Karhunen-Lowe transform, FastMap and its variants, recently proposed nonlinear techniques such as LLE, Isomap, and random projection.

Singular Value Decomposition

Among the most widely used dimensionality reduction methods is the Karhunen-Loëve (KL) transformation. The KL transformation is the optimal way to project n -dimensional points to k -dimensional points such that the error of the projections (the sum of the squared distances) is minimal. Duda and Hart (1973) describes the KL transform (see also Jolliffe, 1989), and Faloutsos (1996) describes how to use it for indexing time series.

The KL transformation gives a new set of orthogonal axes, each a linear combination of the original ones, sorted by the degree by which they preserve the distances of the points in the original space.

For a given set of M points in n dimensions, finding the set of axes in the KL transformation is equivalent to solving the singular value decomposition (SVD) problem in an $M \times n$ matrix N , each row of which represents a data point. The SVD of the matrix N is the decomposition into $N = U \times \Lambda \times V^t$, where U is an $M \times r$ matrix, Λ a diagonal $r \times r$ matrix, and V a column orthonormal $n \times r$ matrix. The matrix V represents the axes of the KL-decomposition (they are also the eigenvectors of the matrix $N \times N^t$), ordered by the respective values in the matrix Λ . Note that $r \leq n$, so the new space has potentially lower dimensionality. In addition, for each small entry in the matrix Λ the corresponding vectors may be eliminated and a lower dimensionality space obtained.

The SVD can be performed efficiently using linear algebra techniques. The eigenvectors of the covariance matrix define the new space, and the eigenvalues sort them in goodness order. The eigenvector that corresponds to the largest eigenvalue is the axis of the largest variation in the data. The next largest eigenvalue corresponds to the axis that is orthogonal to the first and has the largest variation, and so on (Fig. 11.8).

To approximate the time series, we use only the k eigenvectors that correspond to the k largest eigenvalues.

The SVD has the advantage—that is, the optimal dimensionality reduction (for linear projections)—in the sense that it best preserves the mean square error between the original images and the approximate images. However, it is computationally hard, compared to the other methods, especially if the time series are very long. In addition, it does not work for subsequence indexing.

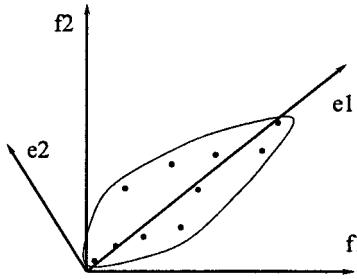


FIG. 11.8. The SVD decomposition identifies the axis of the highest data variation. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

Fourier and Discrete Cosine Transform

The discrete fourier transform (DFT) analyzes the frequency spectrum of a one-dimensional signal. For a signal $S = (S_0, \dots, S_{n-1})$, the DFT is:

$$S_f = 1/\sqrt{n} \sum_{i=0, \dots, n-1} S_i e^{-j2\pi f i / n}$$

where $f = 0, 1, \dots, n - 1$, and $j^2 = -1$. An efficient $O(n \log n)$ algorithm makes DFT a practical method.

DFT was proposed as a dimensionality reduction technique for time series by Agrawal, Faloutsos, and Swami (1993) and Rafiei and Mendelzon (1998). To approximate the time series, we keep the k first Fourier coefficients only. Parseval's theorem states that

$$\sum_{i=0, \dots, n-1} S_i^2 = \sum_{f=0, \dots, n-1} S_f^2$$

It follows that computing the distances using k coefficients provides a lower bound on the Euclidean distance of the original sequences.

The DFT has the advantage that there is an efficient algorithm for computing it. In many applications it works well as a dimensionality reduction technique because most of the energy is concentrated in the lower frequencies (first coefficients, Fig. 11.9). On the other hand, to

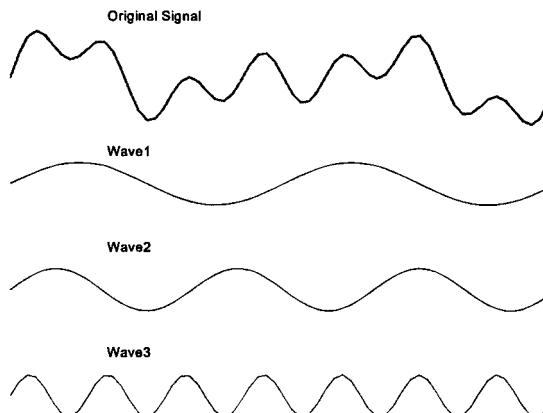


FIG. 11.9. Fourier decomposition. Top: original signal. Bottom: sinusoid waves that reconstruct the original signal.

project the n -dimensional time series into a k -dimensional space, the same k Fourier coefficients must be stored for all series. This may not be optimal for all sequences. To find the k optimal coefficients for M time series, we have to compute the average energy for each coefficient.

Wavelet Decomposition

The wavelet decomposition (WD) represents the time series as a sum of prototype functions, similar to the DFT (see, for example, the Imager Wavelet Library, www.cs.ubc.ca/nest/imager/contributions/bobl/wvlt/top.html). The simplest form of base functions are the Haar wavelets used by Struzik and Siebes (1999), but many bases have been proposed (Chan & Fu, 1999). It differs from the DFT because the effect of different coefficients is localized in time rather than frequency.

Following is an example of the wavelet decomposition (using the Haar wavelet basis):

Let $S = \{3, 1, 9, 3\}$ be the original time series.

To find the wavelet decomposition of S , we first find the average of consecutive pairs of points. This creates a sequence S_1 of half the length of S , that is, an approximation to the original time series S : The i th number of the new sequence, $S_1[i]$, is the average of the $2i$ th and the $(2i + 1)$ -th numbers in S ; that is, $S_1[i] = \frac{S[2i] + S[2i+1]}{2}$. In the example, we have:

$$S_1 = \{2, 6\}$$

To recover the original time series, we also need to store the differences from the average for each pair. Note that only one number needs to be stored, because the first number of the pair will be the average minus the difference, and the second will be the average plus the difference. We append these differences to the approximation S_1 to create the series:

$$S'_1 = (2, 6, -1, -3)$$

The new series contains an approximation of the original time series in its first half and information to reconstruct the original from the approximation in the second half.

In the next step we repeat this process for the first half (the current approximation) only. The final result:

$$S' = (4, 2, -1, -2)$$

is the wavelet decomposition of S . The original sequence can be reconstructed using the wavelet coefficients:

$$\begin{aligned} S[0] &= S'[0] - S'[1] - S'[2] \\ S[1] &= S'[0] - S'[1] + S'[2] \\ S[2] &= S'[0] + S'[1] - S'[3] \\ S[3] &= S'[0] + S'[1] + S'[3] \end{aligned}$$

To use wavelets for approximating sequences, we keep only k coefficients, and we approximate the rest with 0. If the the first k coefficients are kept, this operation is equivalent to low pass filtering. A more accurate representation would keep the largest (or more important) k coefficients. This approach is difficult to use for indexing, because we have to make sure that the same k coefficients are kept for all sequences. The advantages of the WD are that the transformed time series remains in the same (temporal) domain and that there exists an efficient $O(n)$ algorithm to compute the transformation. The disadvantages are similar to DFT.

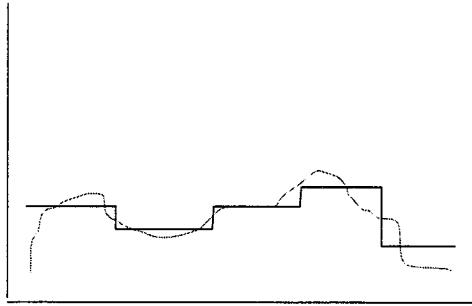


FIG. 11.10. An example of partitioning a time series to six constant pieces. (Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston.)

Line Segment Approximation

Instead of using DFT to approximate a time series, one could use a piecewise polynomial approximation, as proposed by Keogh and Pazzani (1998). Although using a piecewise linear approach has been a popular representation for researchers to define non-Euclidean distance measures (see, e.g., Shatkay & Zdonik, 1996), no general admissible (i.e., guaranteeing no false dismissals) indexing technique is known. Recent work, however, has shown that it is possible to index such a representation if the time series is approximate by a piecewise constant function (Fig. 11.10). The idea is to do dimensionality reduction by dividing the time series into k equal sized “frames.” The mean value of the data falling within a frame is calculated, and a vector of these values becomes the data reduced representation. This representation was introduced independently by Yi and Faloutsos (2000) and by Keogh, Chakrabarti, Pazzani, and Merhotra (2000), who called it the piecewise aggregate approximation (PAA).

Although the PAA representation approximates a time series with an equal-sized piecewise constant approximation, many real-world sequences have areas with varying levels of detail. It is clear that such signals could benefit from an adaptive representation, which assigns more approximating segments to areas of high activity and fewer segments to the areas of the signal that are relatively stationary. This approach was proposed by Pavlidis (1973). Recent work by Keogh et al. (2001) demonstrated a technique for indexing this representation, which the authors called adaptive piecewise constant approximations (APCA). The representation can be indexed with classic multidimensional index structures such as the R-tree after some modifications of the internal data structures.

This representation has some advantages over the DFT method, namely that the transformation can be performed in linear time. More important, PAA supports a variety of distance measures, including arbitrary L_p norms (Yi & Faloutsos, 2000), DTW (Keogh, 2002), and weighted Euclidean queries (Keogh et al., 2000). The technique is equivalent to the WD when k is a power of 2 and the mean is used to approximate the subsequences; however, it is more efficient to compute.

Random Projection

This is a general dimensionality reduction technique that was proposed by Indyk and Motwani (1998) (see also Achlioptas, 2001). It was applied to the time-series domain by Bingham and Mannila (2001). The technique uses the Johnson-Lindenstrauss lemma:

Lemma 1 (Johnson & Lindenstrauss, 1984). *For any $0 < \epsilon < 1/2$, and any (sufficiently large) set S of M points in R_n , and $k = O(\epsilon^{-2} \ln M)$, there exists a linear map $f : S \rightarrow R_k$, such that $(1 - \epsilon)d(A, B) < d(f(A), f(B)) < (1 + \epsilon)d(A, B)$ for $A, B \in S$.*

To apply the lemma to the time series domain, assume we have M time series, each of length n . The dimensionality reduction technique works as follows:

1. Set $k = O(e^{-2} \ln M)$
2. Select k random n -dimensional vectors
3. Project the time series into the k vectors.

The resulting k -dimensional space approximately preserves the distances with high probability. However, it is a Monte-Carlo algorithm, so the result may not be correct. Note that the dimensionality of the new space, k , depends on the number of time series and on the accuracy we want to achieve, not on the length of the individual time series.

This is a very useful technique, especially when used in conjunction with another technique. For example, we can use random projection to reduce the dimensionality from thousands to hundreds, then apply SVD to reduce dimensionality further, as suggested by Papadimitriou, Raghavan, Tamaki, and Vempala (1998).

Multidimensional Scaling

Multidimensional scaling (MDS) is a technique that focuses on the preservation of the original high-dimensional distances for a k -dimensional representation of the objects (Kruskal & Wish, 1978).

The input to the algorithm is a set of N objects, their pairwise distances, and the desirable dimensionality k . The only assumption that is made by MDS is that there is a monotonic relationship between the original and projected pairwise distances (i.e., if we plot the original distance d_{ij} and the projected distance d'_{ij} of all objects in a two-dimensional diagram, then the points should lie on a smooth line or curve that behaves monotonically).

The algorithm attempts to map each object into a k -dimensional space, so that a *stress* function:

$$\text{stress} = \sqrt{\frac{\sum_{i,j} (d'_{ij} - d_{ij})^2}{\sum_{i,j} d_{ij}^2}}$$

is minimized, where d_{ij} is the original distance between objects i , j , and d'_{ij} is the Euclidean distance between the k -dimensional points to which these objects are mapped.

Typical algorithms for optimizing this function work iteratively by picking an initial assignment and then trying to improve the solution by picking a point and moving it to the location that would minimize the *stress* function.

Although MDS performs very effective dimensionality reduction and has many applications in visualization as well as in the analysis of large data sets, it is difficult to use this technique for indexing time series. MDS can be used to map a set of time series to k -dimensional points (for some small k). However, to be able to answer similarity queries, we must be able to map the query time series to the same k -dimensional space. Because of the way the MDS algorithm works, to find the representation of the query we must find (or at least estimate) the distances of the query to all the time series in the set, which is a linear operation already.

FastMap

An approximation of multidimensional scaling is FastMap, given by Faloutsos and Lin (1995), which maps objects to k -dimensional points so that distances are preserved well. As with MDS, it works even when only distances are known. It is also efficient and allows efficient

query transformation. However, it offers no guarantees of optimality. Intuitively the algorithm operates by assuming that the objects are embedded in a k -dimensional space, and works as follows:

1. Find two objects that are far away.
2. Project all points on the line the two objects define to get the first coordinate.
3. Project all objects on a hyperplane perpendicular to the line the two objects define and find the pairwise distances of the projections.
4. Repeat $k - 1$ times to get a total of k coordinates.

One advantage of FastMap is that it needs only distances between objects and not the coordinates of the objects themselves. In addition, it allows the user to map the query to the new space in $O(k)$ time by following the same process that created the mapping. Therefore, it can be used for time-series indexing.

Isomap and LLE

Lately, another category of dimensionality reduction techniques has appeared, namely Isomap (Tenenbaum, de Silva, & Langford, 2000) and locally linear embedding (LLE, Roweis & Saul, 2000). These methods attempt to preserve as well as possible the local neighborhood of each object, while preserving the global distances through the rest of the objects (by means of a minimum spanning tree).

Isomap is a method that tries to map high-dimensional objects into a lower dimensional space (usually two to three for visualization purposes), preserving at the same time as well as possible the neighborhood of each object and also the “geodesic” distances between all pairs of objects. Assuming M time series of length n each, Isomap works as follows:

1. Calculate the K closest neighbors of each object (this takes $O(M \log M)$ or $O(M^2)$ for high dimensions)
2. Create the minimum spanning tree (MST) distances of the updated distance matrix (this takes $O(M^3)$ -Floyd or $O((aM + M^2)\log M)$ -Dijkstra, where a is the number of edges).
3. Run MDS on the new distance matrix.
4. Depict points on some lower dimension.

LLE also attempts to reconstruct as nearly as possible the neighborhood of each object, from some high dimension (q) into a lower dimension. The steps of the algorithm are as follows:

1. Calculate the K closest neighbors of each object.
2. Write each object as a linear combination of each K neighbors (this takes $O(qMK^3)$).
3. Compute the low dimensional vector \mathbf{Y} that best reconstructs the linear combination of each object from its neighbors. For this step the eigenvectors corresponding to the smaller eigenvalues are utilized (this takes $O(qM^2)$).
4. Show low-dimensional points.

Both algorithms attempt to achieve the same goal. Isomap tries to minimize the least square error of the geodesic distances, whereas LLE aims at minimizing the least squares error in the low dimension of the neighbors’ weights for every object.

We depict the potential power of the above methods with an example. Suppose that we have data that lie on a manifold in three dimensions (Fig. 11.11). For visualization purposes

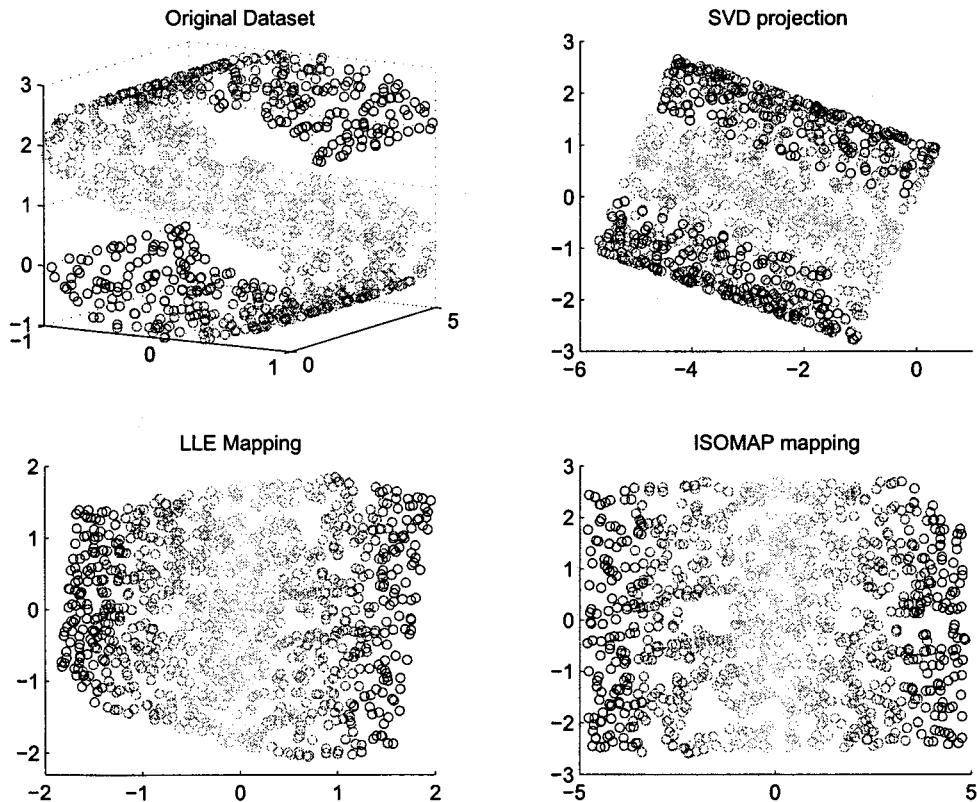


FIG. 11.11. Mapping in two dimensions of the SCURVE data set using SVD, LLE, and Isomap. (Vlachos, M., Domeniconi, C., Gunopulos, D., Kollios, G., & Koudas, N. (2002). Non-linear dimensionality reduction techniques for classification and visualization. In *Proceedings of the ACM SIGKDD 2002 Conference*. Included here by permission, © ACM, Inc.)

we would like to identify the fact that in two dimensions, the data could be placed on a plane, by “unfolding” or “stretching” the manifold. Locally linear methods provide us with this ability. However, by using some global method, such as SVD, the results are nonintuitive, and neighboring points get projected on top of each other (Fig. 11.11).

Similar Time-Series Retrieval When the Distance Function Is Not a Metric

Distance functions that are robust to extremely noisy data will typically violate the triangular inequality. This happens because such functions do not consider equally all parts of the time series. However, this behavior is useful because it represents a more accurate model of the human perception: When people compare any kind of data (images, time series, etc.), they focus mostly the portions that are similar and are willing to pay less attention to regions of dissimilarity.

Nonmetric distances are used nowadays in many domains, such as string (DNA) matching, collaborative filtering (when customers are matched with stored prototypical customers), and retrieval of similar images from databases. Furthermore, psychology research suggests that human similarity judgments are also nonmetric.

In addition, nonmetric distance functions allow us to address the following scenarios:

- *The time series were collected at different sampling rates or different speeds.* The time series that we obtain are not guaranteed to be the outcome of sampling at fixed time intervals. The sensors collecting the data may fail for some period of time, leading to inconsistent sampling rates. Moreover, two time series moving in exactly the same way, but one moving at twice the speed of the other, will result (most probably) in a very large Euclidean distance.
- *The time series contain outliers.* Outliers may be introduced due to an anomaly in the sensor collecting the data or can be attributed to human “failure” (e.g., jerky movement during a tracking process).
- *The time series have different lengths.* Euclidean distance deals with time series of equal length. In the case of different lengths we have to decide whether to truncate the longer series or pad with zeros the shorter, and so forth.

Examples of such distance metrics include the DTW and the LCSS distance functions. It is easy to see that neither of the two is a metric.

Example. Consider the DTW distance of the following sequences:

$$\text{Seq1} = (0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0)$$

$$\text{Seq2} = (0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1)$$

$$\text{Seq3} = (1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1)$$

Then $\text{DTW}(\text{Seq1}, \text{Seq2}) = 1$, $\text{DTW}(\text{Seq2}, \text{Seq3}) = 1$, and $\text{DTW}(\text{Seq1}, \text{Seq3}) = \sqrt{10}$.

Similarly, consider the LCSS_ϵ distance for the following sequences, if we set $\epsilon = 1$:

$$\text{Seq1} = (0, 0, 0, 0, 0, 0), \text{Seq2} = (1, 1, 1, 1, 1, 1), \text{Seq3} = (2, 2, 2, 2, 2, 2)$$

Then $\text{LCSS}_\epsilon(\text{Seq1}, \text{Seq2}) = 0$, $\text{LCSS}_\epsilon(\text{Seq2}, \text{Seq3}) = 0$, and $\text{LCSS}_\epsilon(\text{Seq1}, \text{Seq3}) = 1$.

To see that such distance functions nevertheless are very useful in practice, in Fig. 11.12 we show clustering results produced using the DTW and the LCSS distance metrics. The sequences

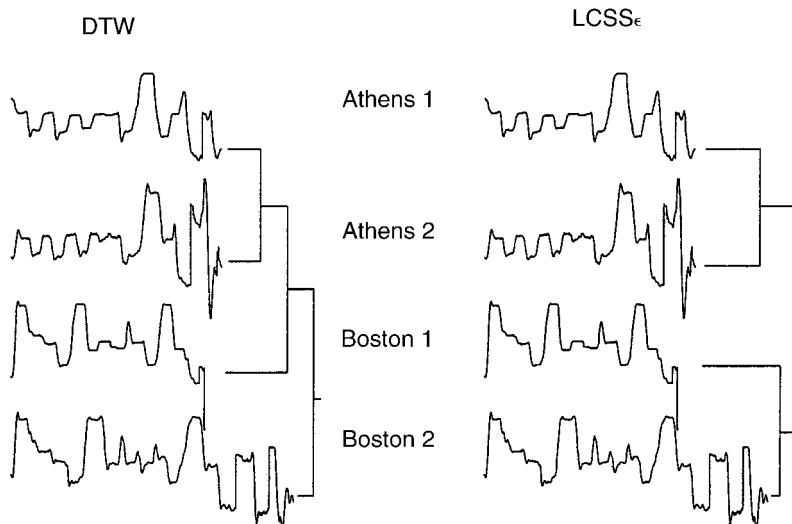


FIG. 11.12. Different distance metrics lead to different classifications of time series. (Reprinted with permission from Vlachos, M., Kolliss, G., & Gunopulos, D. (2002). Discovering similar trajectories. In *Proceedings of the IEEE International Conference on Data Engineering*. IEEE Computer Society Press. © 2002, IEEE.)

represent data collected through a video tracking process. The DTW fails to distinguish the two classes of words due to the large number of outliers, especially in the beginning and in the end of the trajectories. Using the Euclidean distance, we obtain even worse results. In this case the LCSS produces the most intuitive clustering, as shown in the same figure.

When the distance functions are not metrics, indexing techniques generally do not work (except for sequential scan). Most techniques try to speed up the sequential scan by bounding the distance from below. Yi and Faloutsos (1998) propose the following general technique:

1. Use a dimensionality reduction technique that needs only distances (FastMap in this case)
2. Use a pessimistic estimate to bound the actual distance (and possibly accept a number of false dismissals)
3. Index the time-series data set using the reduced dimensionality space.

Recent approaches focus more on approximating the DTW or the LCSS distance. For example, Keogh (2002) shows how to index time series using the DTW distance function. The idea is based on a novel, linear time, lower-bound of the DTW distance. The intuition behind the approach is the construction a special “envelope” around the query. It can be shown that the Euclidean distance between a potential match and the nearest orthogonal point on the envelope lower bounds the DTW distance. To index this representation, a slightly modified version of PAA is used to create an approximate bounding envelope.

Other techniques for lower bounding the DTW distance were proposed by Yi and Faloutsos (2000) and by Kim, Park, and Chu (2001).

A different approach is to use the LCSS model. Although it is not a metric, it can be shown that it obeys a weaker form of the triangle inequality (Vlachos, Kollios, & Gunopulos, 2002). As a result, efficient index structures can be designed for this distance model.

Subsequence Retrieval

There is considerably less work in this area, and the problem is more general and difficult. Most of the previous dimensionality reduction techniques cannot be extended to handle the subsequence matching problem. However, if the length of the subsequence is known, two general techniques can be applied:

1. Index all possible subsequences of given length k (which results in $n - k + 1$ subsequences of length k for each time series of length n)
2. Partition each time series into fewer subsequences and use an approximate matching retrieval mechanism.

Recent work on subsequence matching includes the subsequence matching technique of Faloutsos, Raganathan, and Manolopoulos (1994), which uses the DFT; the subsequence matching technique of Park, Chu, Yoon, and Hsu (2000), which uses the DTW distance metric; the technique of Qu, Wang, Gao, and Wang (in press), which allows the user to specify the parameters of the matching function; and the technique of Gao and Wang (2002), which continuously matches the most recent part of a streaming time series against a given query pattern.

Other recent work extends similarity measures to higher dimensional sequences, such as sequences of video frames, or sequences of locations in space (see Vlachos, Kollios, & Gunopulos, 2002; and Yazdani & Ozsoyoglu, 1996).

SUMMARY

There is a lot of work in the database community on time-series similarity measures and indexing techniques. The motivation comes mainly from the clustering/unsupervised learning problem and the nearest neighbor classification problem.

We gave an overview of simple similarity models that allow efficient indexing and of more realistic similarity models in which the indexing problem is not fully solved yet.

Open problems in the area include:

1. indexing nonmetric distance functions,
2. similarity models and indexing techniques for higher-dimensional time series, and
3. efficient trend detection/subsequence matching algorithms.

REFERENCES

- Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the 2001 ACM PODS Symposium* (pp. 274–281). New York: ACM Press.
- Erickson, J., & Agrawal, P. (1998). Geometric range searching and its relatives. In B. Chazelle, J. Goodman, & R. Pollack (Eds.), *Advances in discrete and comput. geom.* (pp. 1–56). Providence, RI: American Mathematical Society.
- Agrawal, R., Faloutsos, C., & Swami, A. (1993). Efficient similarity search in sequence databases. In D. Lomet (Ed.), *In Proceedings of Fourth International Conference on Foundations of Data Organization* (pp. 69–84). Heidelberg, Germany: Springer-Verlag.
- Agrawal, R., Lin, K.-I., Sawhney, H. S., & Shim, K. (1995). Fast similarity search in the presence of noise, scaling, and translation in time-series databases. In *Proceedings of the 21st International Conference on Very Large Databases* (pp. 490–501). San Francisco: Morgan Kaufmann.
- Beckmann, N., Kriegel, H.-P., Schneider, R., & Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. *SIGMOD Conference* (pp. 322–331).
- Bentley, J. L. (1979). Multidimensional binary search trees in database applications. *IEEE Transaction on Software Engineering*, 5, 333–340.
- Berndt, D. J., & Clifford, J. (1994, July). Using dynamic time warping to find patterns in time series. Paper presented at the KDD Workshop, Seattle, WA.
- Berndt, D. J., & Clifford, J. (1996). Finding patterns in time series: A dynamic programming approach. *Advances in Knowledge Discovery and Data Mining* (pp. 229–248). Menlo Park, CA: AAAI Press.
- Bingham, E., & Mannila, H. (2001). Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the ACM SIGKDD* (pp. 245–250). New York: ACM Press.
- Bollobas, B., Das, G., Gunopulos, D., & Mannila, H. (2001). Time-series similarity problems and well-separated geometric sets. *Nordic Journal of Computing*, 8, 409–423.
- Chakrabarti, K., & Mehrotra, S. (2000). Local dimensionality reduction: A new approach to indexing high dimensional spaces. *Proceedings of the Twenty-Sixth International Conference on Very Large Data Bases* (pp. 89–100). San Francisco: Morgan Kaufmann.
- Chan, K., & Fu, W. (1999). Efficient time series matching by wavelets. In M. Kitsuregawa (Ed.), *Proceedings of the IEEE, International Conference on Data Engineering* (pp. 126–135). Los Alamitos, CA: IEEE Computer Society Press.
- Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge, MA: MIT Press.
- Das, G., Gunopulos, D., & Mannila, H. (1997). Finding similar time series. In *Principles of data mining and knowledge discovery in databases* (pp. 88–100). Heidelberg, Germany: Springer-Verlag.
- Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.
- Faloutsos, C. (1996). *Searching multimedia databases by content*. Norwell, MA: Kluwer Academic Publishers.
- Faloutsos, C., & Lin, K.-I. (1995). FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM SIGMOD Conference* (pp. 163–174).
- Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1996). Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data* (pp. 419–429). New York: ACM Press.

- Gao, L., & Wang, X. (2002). Continually evaluating similarity-based pattern queries on a streaming time series. In *Proceedings ACM SIGMOD* (pp. 370–381). New York: ACM Press.
- Ge, X., & Smyth, P. (2000). Deformable Markov model templates for time-series pattern matching. In *Proceedings ACM SIGKDD* (pp. 81–90). New York: ACM Press.
- Goldin, D. Q., & Kanellakis, P. C. (1995). On similarity queries for time-series data: Constraint specification and implementation. In *Proceedings of the 1995 International Conference on the Principles and Practice of Constraint Programming* (pp. 137–153). Berlin: Springer.
- Gunopulos, D., & Das, G. (2001). *Time series similarity measures and time series indexing*. Paper presented at the ACM SIGMOD Conference, Santa Barbara, CA.
- Gunopulos, D., & Das, G. (2000, August). *Time series similarity measures*. Paper presented at the ACM SIGKDD Conference, Boston, MA.
- Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings ACM SIGMOD* (pp. 47–57). New York: ACM Press.
- Hamilton, J. (1994). Time series analysis. Princeton, NJ: Princeton University Press.
- Hjaltason, G. R., & Samet, H. (1999). Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24, 265–318.
- Horowitz, S. L., & Pavlidis, T. (1976). Picture segmentation by a tree traversal algorithm. *Journal of the Association of Computing Machinery*, 23, 368–388.
- Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of Thirtieth Annual ACM Symposium on Theory of Computing* (pp. 604–613). New York: ACM Press.
- Jagadish, H. V., Mendelzon, A. O., & Milo, T. (1995). Similarity-based queries. In *Proceedings of ACM Symposium on Principles of Database Systems* (pp. 36–45). New York: ACM Press.
- Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipshitz mapping into Hilbert space. *Contemporary Mathematics*, 26, 189–206.
- Jolliffe, I. T. (1989). *Principal component analysis*. New York: Springer-Verlag.
- Keogh, E. (2002). Exact indexing of dynamic time warping. In *Proceedings of Twenty-Eighth International Conference on Very Large Data Bases*. San Francisco: Morgan Kaufmann.
- Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2000). Dimensionality reduction for fast similarity search in large time series databases. *Journal of Knowledge and Information Systems*, 263–286.
- Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Locally adaptive dimensionality reduction for indexing large time series databases. In *Proceedings of the 2001 ACM SIGMOD Conference on Management of Data* (pp. 151–162). New York: ACM Press.
- Keogh, E. J., & Pazzani, M. J. (1998). An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 239–241). New York: ACM Press.
- Keogh, E., Chu, S., Hart, D., & Pazzani, M. (2001). An online algorithm for segmenting time series. In *Proceedings of the IEEE International Conference on Data Mining* (pp. 289–296). Los Alamitos, CA: IEEE Computer Society Press.
- Keogh, E., & Pazzani, M. J. (1999b). Relevance feedback retrieval of time series. The 22th International ACM-SIGIR Conference on Research and Development in Information Retrieval (pp. 183–190). New York: ACM Press.
- Keogh, E. J., & Pazzani, M. J. (1999a). An indexing scheme for fast similarity search in large time series databases. In *Proceedings of Eleventh International Conference on Scientific and Statistical Database Management* (pp. 56–67). Los Alamitos, CA: IEEE Computer Society Press.
- Keogh, E. J., & Smyth, P. (1997). A probabilistic approach to fast pattern matching in time series databases. In *Proceedings of Third International Conference of Knowledge Discovery and Data Mining* (pp. 24–30). Menlo Park, CA: AAAI Press.
- Kim, S., Park, S., & Chu, W. (2001). An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of International Conference on Data Engineering* (pp. 609–614). Los Alamitos, CA: IEEE Computer Society Press.
- Kruskal, J., & Wish, M. (1978). *Multidimensional scaling*. Thousand Oaks, CA: Sage Publications.
- Kruskal, J. B., & Sankoff, D. (Eds.). (1983). *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison*. Reading, MA: Addison-Wesley.
- Lin, K.-I., Jagadish, H. V., & Faloutsos, C. (1994). The TV-tree: An index structure for high-dimensional data. *VLDB Journal*, 3, 517–542.
- Papadimitriou, C., Raghavan, P., Tamaki, H., & Vempala, S. (1998). Latent semantic indexing: A probabilistic analysis. In *Proceedings of the 17th ACM Principles of Database Systems Conference* (pp. 159–168). New York: ACM Press.

- Park, S., Chu, W., Yoon, J., & Hsu, C. (2000). Efficient similarity searches for time-warped subsequences in sequence databases. In *Proceedings of Sixteenth International Conference on Database Engineering* (pp. 23–32). Los Alamitos, CA: IEEE Computer Society Press.
- Pavlidis, T. (1973). Waveform segmentation through functional approximation. *IEEE Transactions on Computers*, C-22, 689–697.
- Peng, C.-S., Wang, H., Zhang, S. R., & Stott Parker, D. (2000). Landmarks: a new model for similarity-based pattern querying in time series databases. In *Proceedings of Sixteenth International Conference on Database Engineering* (pp. 33–42). Los Alamitos, CA: IEEE Computer Society Press.
- Qu, Y., Wang, C., Gao, L., & Wang, X. (in press). Supporting movement pattern queries in user-specified scales. *IEEE Transactions on Knowledge and Data Engineering*.
- Rafiei, D., & Mendelzon, A. (1998, November). *Efficient retrieval of similar time sequences using DFT*. Paper presented at the FODO '98 Conference, Kobe, Japan.
- Robinson, J. T. (1981, October). The K-D-B-tree: A search structure for large multidimensional dynamic indexes. In *Proceedings of the ACM SIGMOD Conference* (pp. 10–18). New York: ACM Press.
- Roussopoulos, N., Kelley, S., & Vincent, R. (1995). Nearest neighbor queries. In *Proceedings of the ACM SIGMOD Conference* (pp. 71–79). New York: ACM Press.
- Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290, 2323–2326.
- Sellis, T. K., Roussopoulos, N., & Faloutsos, C. (2000). The R+tree: A dynamic index for multi-dimensional objects. In *Proceedings of Thirteenth International Conference on Very Large Data Bases* (pp. 507–518). San Francisco: Morgan Kaufmann.
- Shatkay, H., & Zdonik, S. (1996). Approximate queries and representations for large data sequences. In *Proceedings of the 12th IEEE International Conference on Data Engineering* (pp. 536–545). Los Alamitos, CA: IEEE Computer Society Press.
- Struzik, Z. R., & Siebes, A. (1999). The Haar wavelet transform in the time series similarity paradigm. In *Proceedings of the Third European Conference on Principles and Practice of Knowledge Discovery in Databases* (pp. 12–22). Heidelberg, Germany: Springer-Verlag.
- Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290, 2319–2323.
- Traina, Jr., C., Traina, A. J. M., Seeger, B., & Faloutsos, C. (2000). Slim-trees: High performance metric trees minimizing overlap between nodes. In *Proceedings of Seventh International Conference on Extending Database Technology* (pp. 51–65). Berlin: Springer-Verlag.
- Vlachos, M., Kollios, G., & Gunopulos, D. (2002). Discovering similar trajectories. In *Proceedings of the IEEE International Conference on Data Engineering*. Los Alamitos, CA: IEEE Computer Society Press.
- Weber, R., Schek, H.-J., & Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In A. Gupta, O. Schmueli, J. Widom (Eds.), *Proceedings of the Twenty-Fourth International Conference on Very Large Data Bases* (pp. 194–205). San Francisco: Morgan Kaufmann.
- White, D., & Jain, R. (1996). Similarity indexing with the SS-tree. In *Proceedings of the 12th IEEE International Conference on Data Engineering* (pp. 516–523).
- Yazdani, N., & Ozsoyoglu, Z. M. (1996). Sequence matching of images. In *Proceedings of the 8th International Conference on Scientific and Statistical Databases* (pp. 53–62). Los Alamitos, CA: IEEE Computer Society Press.
- Yi, B., Jagadish, H. V., & Faloutsos, C. (1998). Efficient retrieval of similar time series under time warping. In *Proceedings of Fourteenth International Conference in Data Engineering* (pp. 201–208). Los Alamitos, CA: IEEE Computer Society Press.
- Yi, B.-K., & Faloutsos, C. (2000). Fast time sequence indexing for arbitrary L_p norms. In *Proceedings of the Twenty-Sixth International Conference on Very Large Data Bases* (pp. 385–394). San Francisco: Morgan Kaufmann.
- Yianilos, P. N. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 311–321). New York: ACM Press.

12

Nonlinear Time Series Analysis

Ying-Cheng Lai, Zonghua Liu, Nong Ye
Arizona State University

Tolga Yalcinkaya
University of Kansas

Introduction	305
Embedding Method for Chaotic Time Series Analysis	307
Reconstruction of Phase Space	307
Computation of Dimension	309
Detection of Unstable Periodic Orbits	311
Computing Lyapunov Exponents from Time Series	317
Time-Frequency Analysis of Time Series	323
Analytic Signals and Hilbert Transform	324
Method of EMD	331
Summary	338
Acknowledgment	338
References	338

INTRODUCTION

Parallel to the rapid development of nonlinear dynamics since the late seventies, there has been a tremendous amount of effort on data analysis. Suppose an experiment is conducted and one or a few time series are measured. Such a time series can be, for instance, a voltage signal from a physical or biological experiment, or the concentration of a substance in a chemical reaction, or the amount of instantaneous traffic at a point in the Internet, and so on. The general question is: What can we say about the underlying dynamical system that generates the time series, if

This work was supported by Air Force Office of Scientific Research under grant no. F49620-01-1-0317.

the equations governing the time evolution of the system are unknown and the only available information about the system is the measured time series?

This chapter describes two approaches in nonlinear data analysis: (1) the delay-coordinate embedding technique (see Takens, 1981) that has been proven useful, particularly for time series from low-dimensional, deterministic, or mostly deterministic¹ dynamical systems; and (2) the time-frequency method based on the Hilbert transform (see Huang et al., 1998), which can be powerful for nonlinear and/or nonstationary time series if the quantity of primary interest is the characteristics of the instantaneous frequency spectrum. The delay-coordinate embedding method has been extremely powerful when in the underlying system, the dynamical invariant set responsible for the behavior of the measured time series is low dimensional and the influence of noise is relatively small. The method has been applied to many disciplines of science and engineering (see Kantz & Schreiber, 1997), and it is also the key to controlling chaos (see Ott, Grebogi & Yorker, 1990), one of the most fruitful areas in applicable chaotic dynamics in the last decade (see Chen, 1999). It should be kept in mind that, despite its demonstrated usefulness for many chaotic systems, the delay-coordinate embedding technique is not particularly appealing when the dynamical invariant set is high-dimensional and/or when there is a significant amount of noise or stochasticity in the system. The time-frequency method described here can in principle be applied to random and nonstationary time series, and it can be useful if one is interested in analyzing the system in terms of the distribution of the instantaneous physical frequencies.² In certain circumstances the time-frequency method can be powerful for identifying the physical mechanisms responsible for the characteristics of the instantaneous-frequency spectrum of the data (see Huang et al., 1998).

The mathematical foundation of the delay-coordinate embedding technique was laid by Takens (1981) in his seminal paper. He proved that, under fairly general conditions, the underlying dynamical system can be faithfully reconstructed from time series in the sense that a one-to-one correspondence can be established between the reconstructed and the true but unknown dynamical systems. Based on the reconstruction, quantities of importance for understanding the system can be estimated, such as the relative weights of determinism and stochasticity embedded in the time series, the dimensionality of the underlying dynamical system, the degree of sensitivity on initial conditions as characterized by the Lyapunov exponents, and unstable periodic orbits that constitute the skeleton of the invariant set responsible for the observed dynamics. The following section is devoted to the essential features of the delay-coordinate embedding technique and various techniques for computing the dynamical invariants of the system.

The time-frequency method described in this chapter aims to extract the fundamental physical frequencies from the time series. Roughly, a random but bounded time series implies recurrence in time of the measured physical quantity. Conceptually, the recurrence can be regarded as being composed of rotations in a physical or mathematical space. The questions are: How many principal rotations exist, and what are the frequency characteristics of these rotations? The recent groundbreaking idea by Huang et al. (1998) represents a powerful technique that provides answers to these questions. It should be stressed that the frequency components here correspond to rotations and, hence, they are more physical and fundamentally different from those in the traditional Fourier analysis or in the wavelet analysis. In many situations the frequency components of the rotations contain much more information than the Fourier or wavelet frequency components that are associated with simple mathematical functions such as

¹In realistic situations environmental noise is inevitable. Here “mostly deterministic” means that the system evolves according to a set of deterministic rules, under the influence of little noise.

²These are the traditional Fourier frequencies, as described in this chapter.

the harmonics. Although Fourier and wavelet analyses are well-developed methods for signals from linear systems, Huang et al. (1998) argue that the Hilbert analysis is fundamentally superior to both the Fourier and wavelet analyses for nonstationary and nonlinear time series. The third section of this chapter details the essential concepts such as rotations and instantaneous frequencies, analytic signals, and the Hilbert transform, and describes the corresponding analysis technique developed by Huang et al. (1998). For completeness, at the beginning of that section we briefly review the Fourier spectral and wavelet methods.

EMBEDDING METHOD FOR CHAOTIC TIME SERIES ANALYSIS

Reconstruction of Phase Space

Let $u_i(t)$ ($i = 1, \dots, l$) be a set of l measurements, which can be, for instance, a set of voltage signals recorded from a physical experiment or electroencephalogram signals from a patient with epileptic seizures. In principle, the measured time series come from an underlying dynamical system that evolves the state variable in time according to a set of deterministic rules, generally represented by a set of differential equations, with or without the influence of noise. Mathematically, any such set of differential equations can be easily converted to a set of first-order, autonomous equations. The dynamical variables from all the first-order equations constitute the *phase space*, and the number of such variables is the *dimension* of the phase space, which we denote by M . The phase-space dimension can in general be quite large. For instance, in a fluid experiment, the governing equation is the Navier-Stokes equation, which is a nonlinear partial differential equation. To represent the system by first-order ordinary differential equations by, say, the procedure of spatial discretization, the number of required equations is infinite. The phase-space dimension in this case is thus infinite. However, it often occurs that the asymptotic evolution of the system lives on a dynamical invariant set of only finite dimension. The assumption here is that the details of the system equations in the phase space and the asymptotic invariant set that determines what can be observed through experimental probes are unknown. The task is to estimate, based solely on one or few time series, practically useful quantities characterizing the invariant set, such as its dimension, its dynamical skeleton, and its degree of sensitivity on initial conditions. The delay-coordinate embedding technique established by Takens provides a practical solution to this task. In particular, Takens' embedding theorem guarantees that a topological equivalence of the phase space of the underlying dynamical system can be reconstructed from time series, based on which characteristics of the dynamical invariant set can be estimated using a variety of techniques.

Takens' delay-coordinate method goes as follows. From each measured time series $u_i(t)$ ($i = 1, \dots, l$), the following vector quantity of q components is constructed:

$$\mathbf{u}_i(t) = \{u_i(t), u_i(t + \tau), \dots, u_i[t + (q - 1)\tau]\},$$

where τ is the *delay time*. Because there are l time series, a vector with $m \equiv ql$ components can be constructed as follows:

$$\begin{aligned} \mathbf{x}(t) &= \{\mathbf{u}_1(t), \mathbf{u}_2(t), \dots, \mathbf{u}_l(t)\} \\ &= \{u_1(t), u_1(t + \tau), \dots, u_1[t + (q - 1)\tau], \\ &\quad u_2(t), u_2(t + \tau), \dots, u_2[t + (q - 1)\tau], \\ &\quad \dots, u_l(t), u_l(t + \tau), \dots, u_l[t + (q - 1)\tau]\}, \end{aligned} \tag{1}$$

where m is the embedding dimension. Clearly, the delay time τ and the embedding dimension m are the two important parameters in the time-series analysis based on delay-coordinate embedding. In the sequel, we discuss the criteria to determine these two parameters.

Delay Time τ . For the time-delayed components $u_i(t + j\tau)$ ($j = 0, \dots, q - 1$) to serve as independent variables, the delay time τ needs to be chosen carefully. Roughly, if τ is too small, then adjacent components, say $u_i(t)$ and $u_i(t + \tau)$, will be too correlated for them to serve as independent coordinates. If, on the other hand, τ is too large, then neighboring components are too uncorrelated for the purpose. Empirically, one can examine the autocorrelation function of $u_i(t)$ and decide a proper delay time (see Theiler, 1986). In particular, one computes:

$$c(\tau') \equiv \frac{\langle u_i(t)u_i(t + \tau') \rangle}{\langle u_i^2(t) \rangle},$$

where $\langle \cdot \rangle$ stands for time average. The delay time τ can be chosen to be the value of τ' such that $c(\tau')/c(0) \approx e^{-1}$.

Embedding Dimension m . To have a faithful representation of the true dynamical system, the embedding dimension m should be large enough. Takens' theorem provides a lower bound for m . In particular, suppose the dynamical invariant set lies in a d -dimensional manifold (or subspace) in phase space. Then, if $m > 2d$, the m -dimensional reconstructed vectors $\mathbf{x}(t)$ have a one-to-one correspondence to the vectors of the true underlying dynamical system. This result can be understood by the following simple mathematical argument. Consider two smooth surfaces of dimension d_1 and d_2 in an M -dimensional space and examine the set of their intersection. The question is whether they intersect generically in the sense that the intersection cannot be removed by small perturbations of either surface. The natural approach is then to look at the dimension d_I of the intersecting set. An elementary mathematical fact states that:

$$d_I = d_1 + d_2 - M.$$

If $d_I \geq 0$, the intersection is generic. For example, consider the intersection of two one-dimensional curves in a two-dimensional plane: $d_1 = d_2 = 1$ and $M = 2$. We obtain: $d_I = 0$, which means that the intersecting set consists of points, and the intersections are generic because small perturbations cannot remove, for instance, the intersecting points between two lines in the plane. If, however, $M = 3$, then $d_I < 0$, which means that two one-dimensional curves do not intersect generically in a three-dimensional space. These two cases, together with an additional one— $d_1 = 1$, $d_2 = 2$, and $M = 3$ —are illustrated in Fig. 12.1. In our case of embedding we can ask whether the dynamical invariant set would intersect itself in the reconstructed phase space. To obtain a one-to-one correspondence between points on the invariant sets in the actual and reconstructed phase spaces, self-intersection must not occur. Otherwise, there will be two directions at the self-intersection, which will destroy the one-to-one correspondence. Thus, taking $d_1 = d_2 = d$ and $M = m$, no self-intersection requires: $d_I < 0$, which means that $m > 2d$.

Although Takens' theorem assumes that the relevant dimension d of the set is that of the manifold in which the set lies, this dimension can be quite different from the dimension of the set itself, which is physically more relevant. Work by Sauer, Yorke, and Casdagli (1991) extends Takens's theorem to relax the dimension requirement: The dimension d can in fact be the box-counting dimension D_0 (Farmer, Ott & Yorke, 1983) of the invariant set.

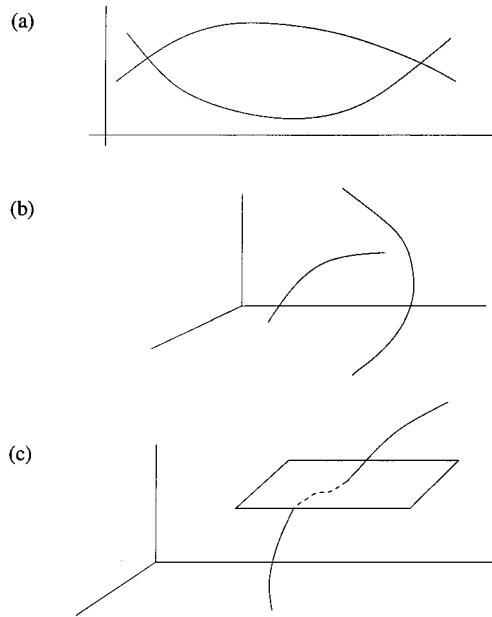


FIG. 12.1. Illustration of generic and nongeneric intersections of simple geometric sets: (a) $d_1 = d_2 = 1$ and $M = 2$ (generic intersection), (b) $d_1 = d_2 = 1$ and $M = 3$ (nongeneric intersection), and (c) $d_1 = 1$, $d_2 = 2$, and $M = 3$ (generic intersection).

Computation of Dimension

Basic Concepts. An often computed dimension in nonlinear time series analysis is the correlation dimension D_2 , which is a good approximation of the box-counting dimension D_0 . Grassberger and Procaccia (1983b) show that D_2 can be evaluated by using the correlation integral $C(\epsilon)$, which is the probability that a pair of points, chosen randomly in the reconstructed phase space, is separated by a distance less than ϵ . Let N be the number of points in the reconstructed vector time series $\mathbf{x}(t)$. The correlation integral can be approximated by the following correlation sum:

$$C_N(\epsilon) = \frac{2}{N(N-1)} \sum_{j=1}^N \sum_{i=j+1}^N \Theta(\epsilon - |\mathbf{x}_i - \mathbf{x}_j|), \quad (2)$$

where $\Theta(\cdot)$ is the Heaviside function, $\Theta(x) = 1$ for $x \geq 0$ and 0 otherwise, and $|\mathbf{x}_i - \mathbf{x}_j|$ stands for the distance between points \mathbf{x}_i and \mathbf{x}_j . Grassberger and Procaccia (1983b) argue that the correlation dimension is given by:

$$D_2 = \lim_{\epsilon \rightarrow 0} \lim_{N \rightarrow \infty} \frac{\log C_N(\epsilon)}{\log \epsilon}. \quad (3)$$

In practice, for a time series of finite length, the sum in Equation 2 also depends on the embedding dimension m as $\mathbf{x}(t)$ depends on the embedding space. Due to such dependencies the correlation dimension D_2 usually is estimated by examining the slope of the linear portion of the plot of $\log C_N(\epsilon)$ versus $\log \epsilon$ for a series of increasing values of m . For $m < D_2$, the

dimension of the reconstructed phase space is not high enough to resolve the structure of the dynamical state and, hence, the slope approximates the embedding dimension. As m increases, the resolution of the dynamical state in the reconstructed phase space is improved. Typically, the slope in the plot of $\log C_N(\epsilon)$ versus $\log \epsilon$ increases with m until it reaches a plateau; its value at the plateau is then taken as the estimate of D_2 (Ding, Grebogi, Ott, Jauery Yorke, 1993; Grassberger & Procaccia, 1983b). For an infinite and noiseless time series the value of m at which this plateau begins satisfies $m = \text{Ceil}(D_2)$, where $\text{Ceil}(D_2)$ is the smallest integer greater than or equal to D_2 (Ding et al., 1993). In any realistic situation short data sets and observational noise can cause the plateau onset to occur at a value of m , which can be considerably larger than $\text{Ceil}(D_2)$. Even so, the embedding dimension at which the plateau is reached still provides a reasonably sharp upper bound for the true correlation dimension D_2 . Dependencies of the length of the linear scaling region on fundamental parameters such as m , τ , and $m\tau$ have been analyzed systematically in Lai, Lerner & Hayden (1996) and Lai & Lerner (1998).

A Numerical Example. For illustrative purposes we present numerical results with the following Hénon (1976) map:

$$\begin{aligned} x_{n+1} &= a - x_n^2 + b y_n, \\ y_{n+1} &= x_n, \end{aligned} \quad (4)$$

where the parameters are chosen to be: $a = 1.4$ and $b = 0.3$ (the standard parameter value). The map is believed to possess a chaotic attractor, as shown in Fig. 12.2(a) in the two-dimensional phase space (x, y) . Numerically, the chaotic attractor can be generated by iterating the map Equation 4 from any randomly chosen initial condition in the region $-2 \leq (x, y) \leq 2$. The theoretical value of the correlation dimension of the chaotic attractor is $D_2 \approx 1.2$ (Lai et al., 1996; Lai & Lerner, 1998). A typical time series $\{x_n\}$ is shown in Fig. 12.2(b).

The GP algorithm can be applied to estimating the correlation dimension from the time series $\{x_n\}$. To select the delay time τ , note that any discrete-time map can be regarded as arising from a Poincaré surface of section of a continuous-time flow (Ott, 1993). Thus, one iteration of the map corresponds to, roughly, one period of oscillation of the continuous-time

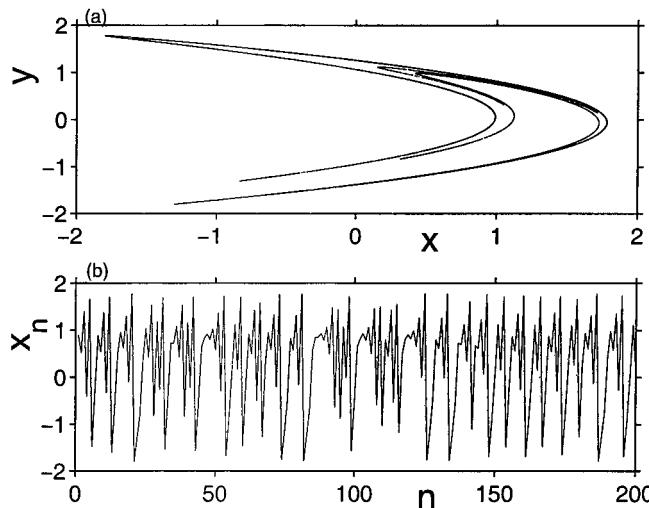


FIG. 12.2. (a) The Hénon chaotic attractor and (b) a typical chaotic time series.

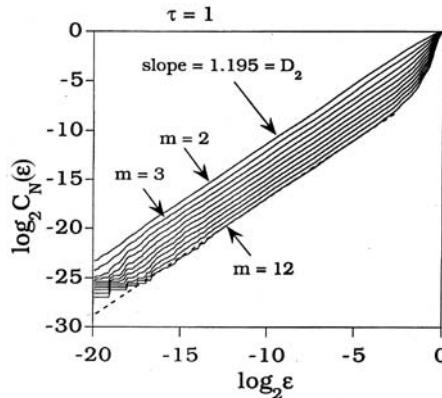


FIG. 12.3. Plots of the correlation integral on a logarithmic scale for $m = 1, \dots, 8$. Least squares fits give $D_2 \approx 1.2$ for $m \geq 2$.

signal $x(t)$, which for chaotic systems is approximately the decay time of the autocorrelation of $x(t)$. As an empirical rule, the delay time can be chosen to be $\tau = 1$ for chaotic time series from discrete-time maps. Equivalently, for chaotic signal $x(t)$ from a continuous-time flow, the delay time should be chosen approximately as the average period of oscillation.

After the delay time τ is chosen, the next step is to compute the correlation integral $C_N(\epsilon)$ for a set of systematically increasing values of the embedding dimension m . Figure 12.3 shows, for $N = 20,000$, the plots of $C_N(\epsilon)$ versus ϵ on the base-2 logarithmic scale for $m = 1, \dots, 8$. The lines are approximately linear, and they are parallel for $m \geq 2$. Least squares fits give $D_2 \approx 1.2$ for $m \geq 2$, indicating that the correlation dimension can be estimated reliably from a time series by utilizing the GP algorithm. A constant slope occurs at $m = 2$, which is the smallest integer above the value of D_2 . Recall that the embedding theorem requires a minimum embedding dimension of $2D_0 + 1$, which is 4 for the Hénon chaotic attractor. This is because the task here is to estimate the dimension only, whereas the embedding theorem guarantees a one-to-one correspondence between the reconstructed and the true chaotic attractors. To estimate the dimension does not necessarily require such a one-to-one correspondence. For instance, consider a two-dimensional surface in a three-dimensional space. The projection of this surface onto a two-dimensional plane is still a two-dimensional region. Thus, its dimension can be estimated even in a two-dimensional subspace. The general rule in dimension computation is that (Ding et al., 1993), for a chaotic set with correlation dimension D_2 , if the time series is noiseless and sufficiently long, then the embedding dimension required is $\text{Ceil}(D_2)$, the smallest integer above the value of D_2 . For short and/or noisy time series, higher values of the embedding dimension may be necessary (Ding, 1993).

Detection of Unstable Periodic Orbits

A fundamental feature that differentiates a deterministic chaotic system from a stochastic one is the existence of an infinite number of unstable periodic orbits that constitute the skeleton of the chaotic invariant set. At a fundamental level, unstable periodic orbits are related to the natural measure of any chaotic set (Grebogi, Ott & Yorke, 1988; Lai, Nagai, & Grebogi, 1997; and Lai, 1997), which is the base for defining quantities that are physically important such as the fractal dimensions and Lyapunov exponents. At a practical level, successful detection of unstable periodic orbits provides unequivocal evidence for the deterministic origin of the underlying dynamical process. In what follows we review the basic concepts of the natural

measure and unstable periodic orbits, describe an algorithm for their detection from time series, and provide numerical examples.

Unstable Periodic Orbits and Natural Measure in Chaotic Systems.

One of the most important problems in dealing with a chaotic system is to compute long-term statistics such as averages of physical quantities, Lyapunov exponents, dimensions, and other invariants of the probability density or the measure. The interest in the statistics springs from the fact that trajectories of deterministic chaotic systems are apparently random and ergodic. These statistical quantities, however, are *physically meaningful* only when the measure being considered is the one generated by a typical trajectory in phase space. This measure is called the natural measure, and it is invariant under the evolution of the dynamics (Grebogi et al., 1988).

The importance of the natural measure can be seen by examining how trajectories behave in a chaotic system. Due to ergodicity, trajectories on a chaotic set exhibit sensitive dependence on initial conditions. Moreover, the long-term probability distribution generated by a typical trajectory on the chaotic set is generally highly singular. For a chaotic attractor a trajectory originated from a random initial condition in the basin of attraction visits different parts of the attractor with drastically different probabilities. Call regions with high probabilities “hot” spots and regions with low probabilities “cold” spots. Such hot and cold spots in the attractor can in general be interwoven on arbitrarily fine scales. In this sense chaotic attractors are said to possess a multifractal structure. Due to this singular behavior, one utilizes the concept of the natural measure to characterize chaotic attractors (Ott, 1993). To obtain the natural measure, one covers the chaotic attractor with a grid of cells and examines the frequency with which a typical trajectory visits these cells in the limit that the length of the trajectory goes to infinity and the size of the grid goes to zero (Farmer et al., 1983). Except for an initial condition set of Lebesgue measure zero in the basin of attraction, these frequencies in the cells are the natural measure. Specifically, let $f(\mathbf{x}_0, T, \epsilon_i)$ be the amount of time that a trajectory from a random initial condition \mathbf{x}_0 in the basin of attraction spends in the i th covering cell C_i of edge length ϵ_i in a time T . The probability measure of the attractor in the cell C_i is,

$$\mu_i = \lim_{\epsilon_i \rightarrow 0} \lim_{T \rightarrow \infty} \frac{f(\mathbf{x}_0, T, \epsilon_i)}{T}. \quad (5)$$

The measure is called *natural* if it is the same for all randomly chosen initial conditions, that is, for all initial conditions in the basin of attraction except for a set of Lebesgue measure zero. The spectrum of an infinite number of fractal dimensions quantifies the behavior of the natural measure (Grassberger & Procaccia, 1983a).

The above description can be seen by considering the following physical example, the forced damped pendulum:

$$\begin{aligned} \frac{dx}{dt} &= y, \\ \frac{dy}{dt} &= -0.05y - \sin x + 2.5 \sin t. \end{aligned} \quad (6)$$

Figure 12.4(a) shows (Lai, 1997), on the stroboscopic surface of the section defined at discrete times $t_n = 2\pi n$, $n = 1, \dots$, a trajectory of 1.5×10^5 points on the chaotic attractor, where the abscissa and the ordinate are the angle $x(t_n)$ and the angular velocity $y(t_n) \equiv dx/dt|_{t_n}$ of the pendulum, respectively. Figure 12.4(b) shows the one-dimensional probability distribution on the attractor at $y = -2$. To obtain Fig. 12.4(b), a one-dimensional array of 1,000 rectangular cells is defined in the x -direction at $y = -2$. The size of each cell is $2\pi/1,000 \times 0.06$. The

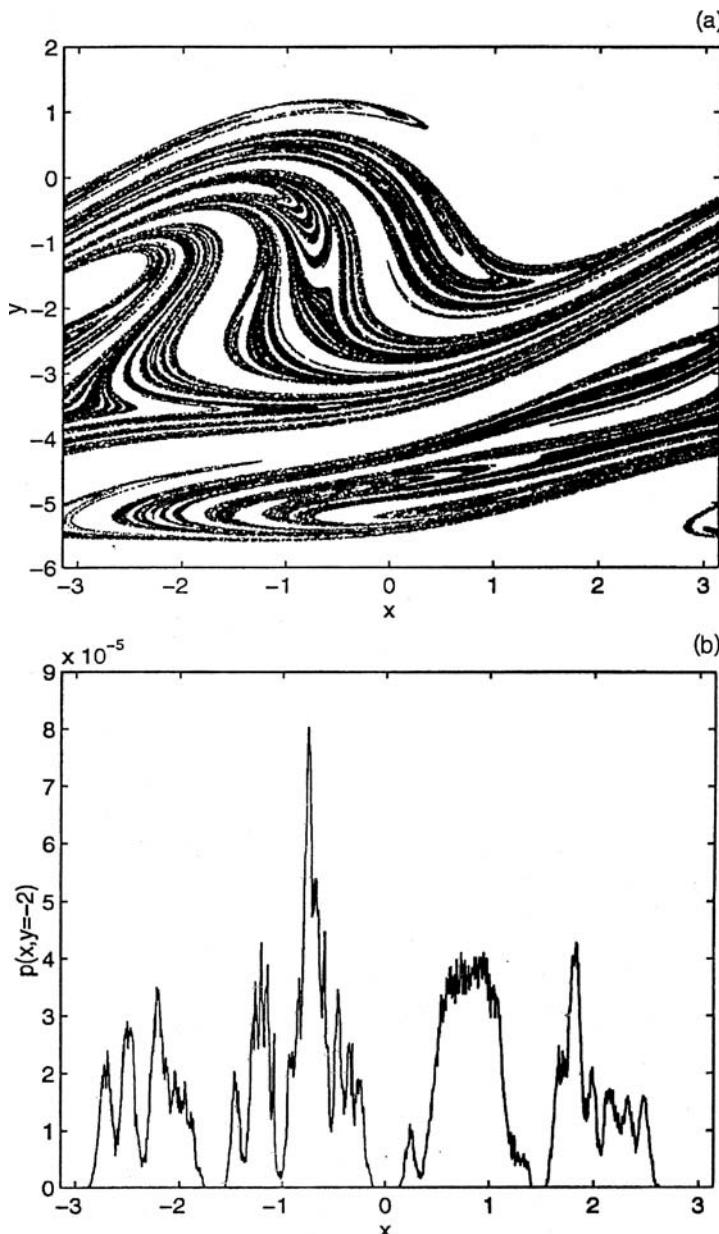


FIG. 12.4. For the forced damped pendulum system Equation 6, (a) a trajectory of 1.5×10^5 points on the chaotic attractor on the stroboscopic surface of section, and (b) the distribution of the natural measure in a one-dimensional array of 1,000 rectangular cells in the x -direction at $y = 2$. The size of each cell is $2\pi/1,000 \times 0.06$. Numerically, the total measure contained in the attractor is normalized to unity. Apparently, the natural measure is singular.

frequency of the trajectory's visiting each cell is then computed by utilizing a trajectory of 10^7 points on the surface of section. In fact, probability distributions on any line intersecting the chaotic attractor exhibit similar behavior. These results suggest a highly singular probability distribution on the chaotic attractor.

Because of the physical importance of the natural measure, it is desirable be able to understand and characterize it in terms of the fundamental dynamical quantities of the chaotic set. There is nothing more fundamental than to express the natural measure in terms of the periodic orbits embedded in a chaotic attractor.

A chaotic set has embedded within itself an infinite number of unstable periodic orbits. These periodic orbits are *atypical* in the sense that they form a Lebesgue measure zero set. With probability one, randomly chosen initial conditions do not yield trajectories that live on unstable periodic orbits. Invariant measures produced by unstable periodic orbits are thus atypical, and there is an infinite number of such atypical invariant measures embedded in a chaotic attractor. The hot and cold spots are a reflection of these atypical measures. The natural measure, on the other hand, is typical in the sense that it is generated by a trajectory originated from any one of the randomly chosen initial conditions in the basin of attraction. A typical trajectory visits a fixed neighborhood of any one of the periodic orbits from time to time. Thus, chaos can be considered as being organized with respect to the unstable periodic orbits (Auerbach, Cvitanovic, Eckmann, Gunaratne & Procaccia 1987). An interesting question then is how the natural measure is related to the infinite number of atypical invariant measures embedded in the attractor.

This fundamental question was addressed by Grebogi et al., (1988). They derived, for the special case of hyperbolic chaotic systems,³ a formula relating the natural measure of the chaotic set in the phase space to the expanding eigenvalues of all the periodic orbits embedded in the set. Specifically, consider an N -dimensional map $\mathbf{M}(\mathbf{x})$. Let \mathbf{x}_{ip} be the i th fixed point of the p -times iterated map, that is, $\mathbf{M}^p(\mathbf{x}_{ip}) = \mathbf{x}_{ip}$. Thus, each \mathbf{x}_{ip} is on a periodic orbit whose period is either p or a factor of p . The natural measure of a chaotic attractor in a phase-space region S is given by,

$$\mu(S) = \lim_{p \rightarrow \infty} \sum_{\mathbf{x}_{ip} \in S} \frac{1}{L_1(\mathbf{x}_{ip}, p)}, \quad (7)$$

where $L_1(\mathbf{x}_{ip}, p)$ is the magnitude of the expanding eigenvalue of the Jacobian matrix $\mathbf{DM}^p(\mathbf{x}_{ip})$, and the summation is taken over all fixed points of $\mathbf{M}^p(\mathbf{x})$ in S . This formula can be derived under the assumption that the phase space can be divided into cells via a Markov partition, a condition that is generally satisfied by hyperbolic chaotic systems. Explicit verification of this formula was done for several analyzable hyperbolic maps (Grebogi, et al., 1988). Equation 7 is theoretically significant and interesting because it provides a fundamental link between the natural measure and various atypical invariant measures embedded in a chaotic attractor. The applicability of Equation 7 to nonhyperbolic chaotic systems was addressed in 1997 (Lai, 1997; Lai et al., 1997).

³The dynamic is hyperbolic on a chaotic set if at each point of the trajectory the phase space can be split into expanding and contracting subspaces and the angle between them is bounded away from zero. Furthermore, the expanding subspace evolves into the expanding one along the trajectory, and the same is true for the contracting subspace. Otherwise, the set is nonhyperbolic. In general, nonhyperbolicity is a complicating feature because it can cause fundamental difficulties in the study of the chaotic systems, a known one being the shadowability of numerical trajectories by true trajectories (Dawson, Grebogi, Sauer, & Yorke, 1994; Grebogi, Hammel, Yorke, & Sauer, 1990; Hammel, Yorke, & Grebogi, 1987, 1988; Lai, & Grebogi, 1999; Lai, Grebogi, & Kurths, 1999; Lai, Lerner, Williams, & Grebogi, 1999).

Computing Unstable Periodic Orbits From Time Series. A powerful algorithm to extract unstable periodic orbits from chaotic time series was devised by Lathrop and Kostelich (LK) (1989). The method is based on identifying sets of recurrent points in the reconstructed phase space. To do this, one first reconstructs a phase-space trajectory $\mathbf{x}(t)$ from a measured scalar time series $\{u(t)\}$ by using the delay-coordinate embedding method described previously. To identify unstable periodic orbits, one follows the images of $\mathbf{x}(t)$ under the dynamics until a value $t_1 > t$ is found such that $\|\mathbf{x}(t_1) - \mathbf{x}(t)\| < \epsilon$, where ϵ is a respecified small number that defines the size of the recurrent neighborhood at $\mathbf{x}(t)$. In this case $\mathbf{x}(t)$ is called an (T, ϵ) recurrent point, and $T = t_1 - t$ is the *recurrence time*. A recurrent point is not necessarily a component of a periodic orbit of period T . However, if a particular recurrence time T appears frequently in the reconstructed phase space, it is likely that the corresponding recurrent points are close to periodic orbits of period T . The idea is then to construct a histogram of the recurrence times and identify peaks in the histogram. Points that occur frequently are taken to be, approximately, components of the periodic orbits. The LK algorithm has been used to detect unstable periodic orbits, for instance, from measurements of chaotic chemical reactions (Lathrop & Kostelich, 1989).

More recently, the LK algorithm was adapted to detect unstable periodic orbits from short, transiently chaotic series by Dhamala, Lai, and Kostelich (2000). This is quite meaningful because in many situations the relevant measured data are represented by short time series. The reason that the LK algorithm is applicable to transient time series lies in the statistical nature of this method, as a histogram of recurrence times can be obtained even with short time series. Provided that there is a large number of such time series so that good statistics of the recurrence times can be obtained, unstable periodic orbits embedded in the underlying chaotic set can be identified. It is not necessary to concatenate many short time series to form a single long one (such concatenations are invariably problematic; Janosi & Tel, 1994). Intuitively, because the time series are short, periodic orbits of short periods can be detected.

To demonstrate the LK algorithm, here we take the numerical examples reported in (Dhamala, Lai & Kostelich, 2000) with the following chaotic Rössler system (Rössler, 1976):

$$\begin{aligned} dx/dt &= -y - z, \\ dy/dt &= x + ay, \\ dz/dt &= b + (x - c)z, \end{aligned} \tag{8}$$

where a , b , and c are parameters. There is transient chaos when the set of parameter values yields a periodic window in which a stable periodic attractor and a chaotic saddle coexist. For instance, for $a = b = 0.2$ and $c = 5.3$, the system falls in a periodic window of period 3. A typical measurement of a dynamical variable, say $x(t)$, exhibits chaotic behavior for a finite amount of time before settling in the period-3 attractor. In Dhamala et al. (2000), 10 such time series are generated by integrating the Rössler system from different initial conditions, and the corresponding time series $x(t)$ for $0 \leq t \leq 4$ are recorded, where the approximate lifetime of the chaotic transient is about 4. These time series are assumed to be the only available data from the system. For each time series a seven-dimensional vector space is reconstructed by using the delay time $\tau = 0.02$. To obtain recurrence times, it is necessary to determine ϵ , the size of the recurrent neighborhood. The value of ϵ must not be large such that many false positives are reported, but ϵ must not be so small that genuine recurrences are missed. Typically, it is found in numerical experiments that the number of recurrences $N(\epsilon)$ increases with the length and the number of the individual transient trajectories and with ϵ . It tends to saturate when ϵ is too large. The value of ϵ at which $N(\epsilon)$ saturates is taken to be an appropriate size of the recurrent

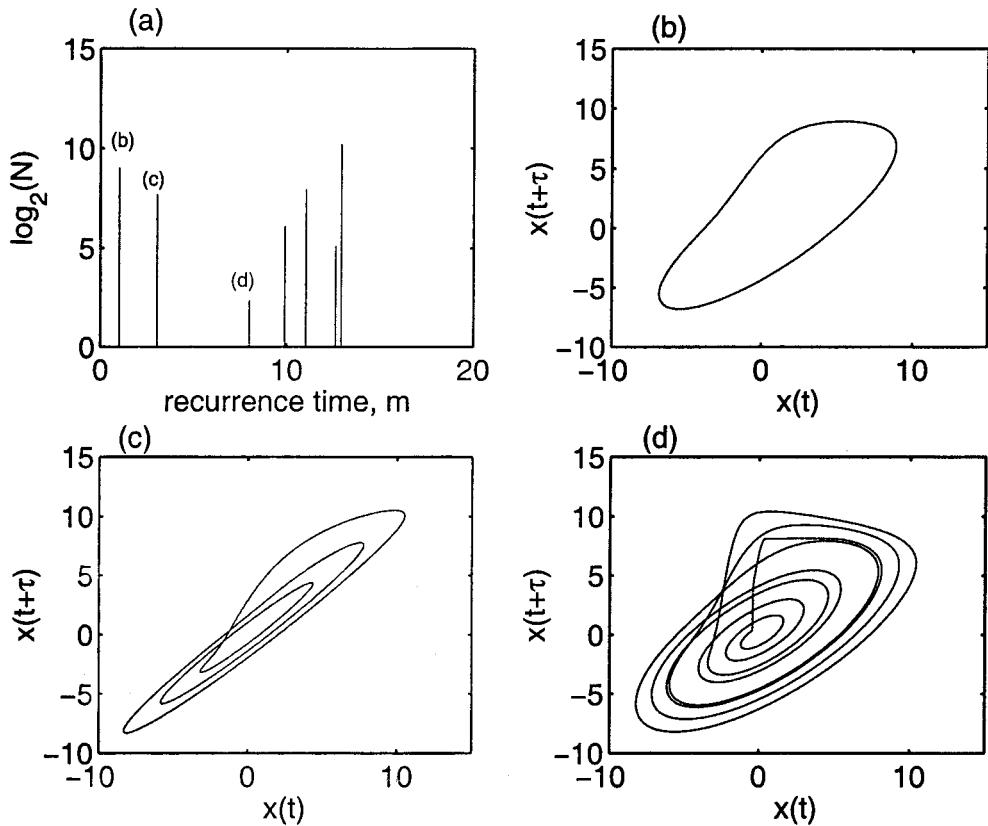


FIG. 12.5. For the Rössler system: (a) histogram of the recurrence time T , (b-d) period 1, period 3, and period 8 recurrent orbits extracted from the histogram in (a), respectively.

neighborhood. For the Rössler system, Dhamala et al. (2000) uses $\epsilon = 2\%$ of the root-mean-square (rms) value of the chaotic signal. Figure 12.5(a) shows the histogram of the recurrence times for the 10 transient chaotic time series from the period-3 window. Figures 12.5(b-d) show, in the plane of $x(t)$ versus $x(t + \tau)$, three recurrent orbits. The orbit in Fig. 12.5(b) has the shortest recurrence time, so we call it a “period-1” orbit. Figures 12.5(c) and (d) show a period-3 and a period-8 orbit. The orbits are selected from the set of recurrent points comprising the corresponding peak in the histogram. In general, it is found (Dhamala et al., 2000) that the LK algorithm is capable of yielding many periodic orbits of low periods (say, period < 10).

In an experimental setting, time series are contaminated by dynamical and/or observational noise. A question is whether periodic orbits can still be extracted from noisy transient chaotic time series. Qualitatively, under the influence of noise the effective volume of a recurrent region in the phase space decreases and, hence, a decrease in the number of recurrences is expected. Figures 12.6(a-d) show the number of recurrent points (a) and three periodic orbits extracted from 10 transient chaotic time series with additive noise of the form $G(0, 0.01)$, a normal (Gaussian) distribution centered at 0 with variance 0.01. This noise level represents an rms value that is approximately 0.5% of that of the chaotic signal. It can be seen that at this low noise level periodic orbits can still be reliably detected. It is found, however, that for the Rössler system at $\epsilon = 2\%$ of the rms value of the chaotic signal with noise beyond 1%, no periodic

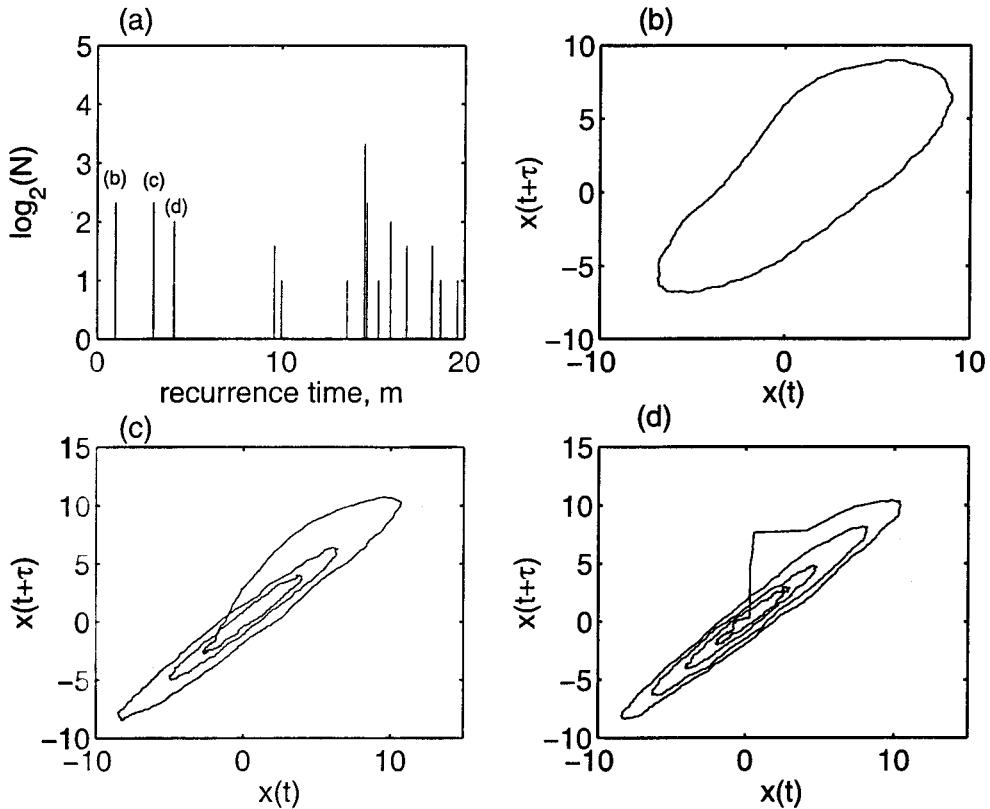


FIG. 12.6. For a noisy Rössler system: (a) histogram of the recurrence time T , (b-d) period 1, period 2, and period 4 recurrent orbits extracted from the histogram in (a), respectively, where $\epsilon = 6\%$ and the rms value of the noise is at about 0.5% of that of the chaotic signal.

orbits can be extracted from the histogram of recurrences. To be systematic Dhamala et al. (2000) computes, at several fixed values of ϵ , how the number of recurrent points decreases as the noise amplitude (η) is increased. Figures 12.7(a-b) show the result of such computations for $\epsilon = 2\%$ (a) and $\epsilon = 6\%$ (b) of the rms value of the signal. It can be seen that the number of recurrent points goes to zero at $\eta \approx \epsilon/2$, which can be understood as follows. Under noise of amplitude η , both the center and the boundary of the recurrent region are uncertain within η . Thus, the effective phase-space volume in d dimensions in which two points can still be considered within distance ϵ (recurrent) is proportional to $(\epsilon - \eta)^d - \eta^d$, which vanishes at $\eta = \epsilon/2$. Because ϵ should be small to guarantee recurrence, we see that the noise level that can be tolerated is also small.

Computing Lyapunov Exponents from Time Series

The Lyapunov exponents characterize how a set of orthonormal infinitesimal distances evolve under the dynamics. For a chaotic system there is at least one positive Lyapunov exponent—let $\lambda_1 > 0$ be the largest exponent. The defining property of chaos is sensitive dependence on initial conditions, in the following sense. Given an initial infinitesimal distance $\Delta x(0)$, its evolution

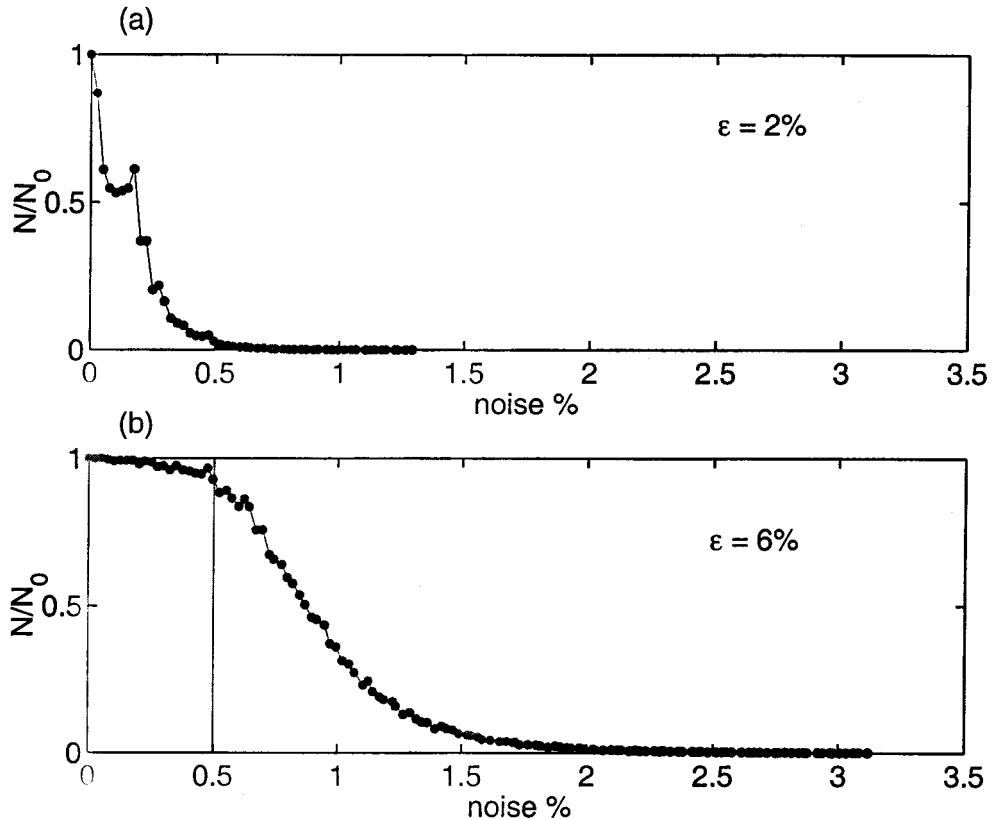


FIG. 12.7. For the noisy Rössler system, the relative number (N/N_0) of recurrent points versus the amplitude of noise for two values of the size of the recurrent neighborhood: (a) $\epsilon = 2\%$ and (b) $\epsilon = 6\%$ of the rms value of the signal, where N_0 is the number of recurrent points at zero amplitude of noise. The vertical line in (b) denotes the noise level at which periodic orbits in Fig. 12.6 are extracted.

obeys:

$$\Delta x(t) = \Delta x(0)e^{\lambda_1 t}.$$

For an M -dimensional dynamical system, there are M -Lyapunov exponents. Here we describe a procedure to compute all the exponents from time series (Eckmann, Kamphorst, Ruelle & Ciliberto, 1986).

Consider a dynamical system described by the following equation:

$$\frac{d\mathbf{x}}{dt} = \mathbf{F}(\mathbf{x}), \quad (9)$$

where $\mathbf{x} \in \mathbf{R}^M$ is a M -dimensional vector. Taking variation of both sides of Equation 9 yields the following equation governing the evolution of the infinitesimal vector $\delta\mathbf{x}$ in the tangent space at $\mathbf{x}(t)$:

$$\frac{d\delta\mathbf{x}}{dt} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \cdot \delta\mathbf{x}. \quad (10)$$

Solving for Equation 10 gives:

$$\delta\mathbf{x}(t) = \mathbf{A}'\delta\mathbf{x}(0), \quad (11)$$

where \mathbf{A}' is a linear operator that evolves an infinitesimal vector at time 0 to time t . The mean exponential rate of divergence of the tangent vector is then given by:

$$\lambda[\mathbf{x}(0), \delta\mathbf{x}(0)] = \lim_{t \rightarrow \infty} \frac{1}{t} \ln \frac{\|\delta\mathbf{x}(t)\|}{\|\delta\mathbf{x}(0)\|}, \quad (12)$$

where $\|\cdot\|$ denotes length of the vector inside with respect to a Riemannian metric. In typical situations there exists a d -dimensional basis vector $\{\mathbf{e}_i\}$, in the following sense:

$$\lambda_i \equiv \lambda[\mathbf{x}(0), \mathbf{e}_i]. \quad (13)$$

These λ_i 's define the Lyapunov spectrum, which can be ordered as follows:

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d. \quad (14)$$

For chaotic systems, values of λ_i do not depend on the choice of the initial condition $\mathbf{x}(0)$, as long as \mathbf{x}_0 is chosen randomly.

If the system Equation 9 is known, then λ_i 's can be computed using the standard procedure developed by Benettin, Galgani, Giorgilli, and strelcyn, (1980). For a chaotic time series, there exist several methods for computing the Lyapunov spectrum (see Brown, Bryant & Abarbanel, (1991); Eckmann & Ruelle, (1985); Sano & Sawada, (1985); Wolf, Swift & swinney (1985)). Although details of these methods are different, they share the same basic principle. Here we describe the one developed by Eckmann et al. (1986). The algorithm consists of three steps: (1) reconstruct the dynamics using delay-coordinate embedding and search for neighbors for each point in the embedding space, (2) compute the tangent maps at each point by least-squares fit, and (3) deduce the Lyapunov exponents from the tangent maps.

Searching for Neighbors in the Embedding Space. Consider a discrete time series $\{x_i\}$, $i = 1, \dots, N$, obtained by sampling a measured quantity $x(t)$, that is, $x_i = x(i\tau)$. The total recording time is then $T = N\tau$. In general, T should be sufficiently large so that the full dynamics are exhibited in the time series. Now choose an embedding dimension d_E to get a series of vectors:

$$\mathbf{x}_i = (x_i, x_{i+1}, \dots, x_{i+d_E-1}), \quad i = 1, 2, \dots, N - m + 1. \quad (15)$$

To determine tangent maps, it is necessary to search for neighbors, that is, search for \mathbf{x}_j such that:

$$\|\mathbf{x}_j - \mathbf{x}_i\| \leq r, \quad (16)$$

where, r is a small number, and $\|\cdot\|$ is defined as:

$$\|\mathbf{x}_j - \mathbf{x}_i\| = \max_{0 \leq \alpha \leq m-1} |x_{j+\alpha} - x_{i+\alpha}|. \quad (17)$$

Such a definition of the distance is only for the consideration of computational speed. If $m = 1$, the time series can be sorted to yield:

$$x_{\Pi(1)} \leq x_{\Pi(2)} \leq \dots \leq x_{\Pi(N)}, \quad (18)$$

where Π is the permutation which, together with its inverse Π^{-1} , are stored. The neighbors of x_i can then be obtained by looking at $k = \Pi^{-1}(i)$ and scanning the sorted time series $x_{\Pi(s)}$ for $s = k \pm 1, k \pm 2, \dots$ until $|x_{\Pi(s)} - x_i| > r$. When $m > 1$, values of s are first selected for which

$$|x_{\Pi(s)} - x_i| \leq r, \quad (19)$$

as for the case where $m = 1$. The following conditions are further imposed:

$$|x_{\Pi(s)+\alpha} - x_{i+\alpha}| \leq r, \quad \alpha = 1, 2, \dots, m-1, \quad (20)$$

resulting in a complete set of neighbors of \mathbf{x}_i within distance r .

Computing the Tangent Maps. The task is to determine the $m \times m$ matrix T_i which describes how the dynamics send small vectors around \mathbf{x}_i to small vectors around \mathbf{x}_{i+1} :

$$T_i(\mathbf{x}_j - \mathbf{x}_i) \approx \mathbf{x}_{j+1} - \mathbf{x}_{i+1}. \quad (21)$$

A serious problem is that T_i may not span \mathbf{R}^m because m is usually made much larger than the actual phase-space dimension of the system to guarantee a proper embedding. Eckmann et al. (1986) proposes a strategy that allows T_i to be a $d_M \times d_M$ matrix, where $d_M \leq m$. In such a case, T_i corresponds to the time evolution from \mathbf{x}_i to \mathbf{x}_{i+m} , and

$$m = (d_M - 1)l + 1, \quad l \geq 1. \quad (22)$$

A new set of embedding vectors can then be constructed:

$$\mathbf{y}_i = (x_i, x_{i+l}, \dots, x_{i+(d_M-1)l}). \quad (23)$$

The new vector \mathbf{y}_i is obtained by taking every m th element in the time series and, hence, T_i is defined in the new embedding space as follows:

$$T_i(\mathbf{y}_j - \mathbf{y}_i) \approx \mathbf{y}_{j+l} - \mathbf{y}_{i+l}. \quad (24)$$

Or,

$$T_i \begin{pmatrix} x_j - x_i \\ x_{j+l} - x_{i+l} \\ \dots \\ x_{j+(d_M-2)l} - x_{i+(d_M-2)l} \\ x_{j+(d_M-1)l} - x_{i+(d_M-1)l} \end{pmatrix} = \begin{pmatrix} x_{j+l} - x_{i+l} \\ x_{j+2l} - x_{i+2l} \\ \dots \\ x_{j+(d_M-1)l} - x_{i+(d_M-1)l} \\ x_{j+d_Ml} - x_{i+d_Ml} \end{pmatrix} \quad (25)$$

Therefore, T_i can be expressed as:

$$T_i = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 0 \\ a_1 & a_2 & a_3 & \dots & a_{d_M} \end{pmatrix} \quad (26)$$

The task of finding T_i then reduces to that of finding the set of d_M matrix elements $a_i (i = 1, 2, \dots, d_M)$. This can be accomplished by using a least squares fit. Let $S_i^E(r)$ be the set

of indices j of neighbors \mathbf{x}_j of \mathbf{x}_i within distance r . The procedure is to minimize the following quantity:

$$\sum_{j \in S_i^E(r)} \left[\sum_{k=0}^{d_M-1} a_{k+1}(x_{j+kl} - x_{i+kl}) - (x_{j+d_Ml} - x_{i+d_Ml}) \right]^2. \quad (27)$$

A critical quantity is $S_i^E(r)$. If $S_i^E(r)$ is large, the computation required is intensive. On the other hand, if $S_i^E(r)$ is too small, the least squares fit may fail. Generally, it is necessary to choose r sufficiently large so that $S_i^E(r)$ contains at least d_M elements. But r also needs to be small so that the linear dynamics approximation around every \mathbf{x}_i is valid. Eckmann et al. (1986) suggested the following empirical rule for choosing r : Count the number of neighbors of \mathbf{x}_i corresponding to increasing values of r from a preselected sequence of possible values, and stop when the number of neighbors exceeds $\min(2d_M, d_{M+4})$ for the first time. Increase r further if T_i is singular.

Computing the Exponents. To compute the Lyapunov exponents from the tangent maps is relatively straightforward. Eckmann and Ruelle (1985) suggested the following procedure. Starting from an identity matrix $Q_{(0)} \equiv I$, the following matrix decomposition (QR decomposition) can be done:

$$\begin{aligned} T_1 Q_{(0)} &= Q_{(1)} R_{(1)}, \\ T_{1+m} Q_{(1)} &= Q_{(2)} R_{(2)}, \\ &\dots, \\ T_{1+jm} Q_{(j)} &= Q_{(j+1)} R_{(j+1)}, \\ &\dots, \end{aligned} \quad (28)$$

where $Q_{(j)}$'s are orthogonal matrices, and $R_{(j)}$'s are upper triangular matrices with positive diagonal elements. The above decomposition is robust, an algorithm of which can be found in Press, Flannery, Teukolsky & Vetterling, (1986). The Lyapunov exponents are then given by:

$$\lambda_k = \frac{1}{l\tau k} \sum_{j=0}^{K-1} \ln R_{(j)kk}, \quad (29)$$

where $R_{(j)kk}$ is the k th diagonal element of the matrix $R_{(j)}$, and $K \leq (N - d_M l - 1)/l$ is the number of available matrices.

Remarks. The algorithm just described is robust. In particular, it can compute all the positive Lyapunov exponents reliably (Eckmann, 1986). Yet, it is necessary to stress three main points. First, d_M cannot be too large, otherwise spurious exponents may arise. Generally, d_M should be larger than the number of positive exponents. Second, the choice of r is critical, as discussed previously. In the presence of noise it may be good to replace the ball $\{\mathbf{x}_j : \|\mathbf{x}_j - \mathbf{x}_i\| \leq r\}$ by a shell $\{\mathbf{x}_j : r_{\min} < \|\mathbf{x}_j - \mathbf{x}_i\| \leq r\}$ when searching for neighbors. Third, increasing the number of points in the time series at fixed recording time is useless. To improve the computation, the total recording time T should be increased.

In brief summary, when computing the Lyapunov exponents from time series, the following rules should be followed:

1. Use long recording time T , but not very small time step τ .
2. Use large embedding dimension m .
3. Use a matrix dimension d_M somewhat larger than the expected number of positive Lyapunov exponents.
4. Choose r such that the number of neighbors is greater than the smaller of $2d_M$ and d_{M+4} .
5. Otherwise keep r as small as possible.
6. Take a product of as many matrices as possible to determine the Lyapunov exponents.

A Numerical Example. Here we discuss the computation of the Lyapunov exponents from an ensemble of transient chaotic time series using the procedure described above. Dhamala, Lai, and Kostelich (2001) consider the Hénon map in a parameter region where the map generates transient chaos. In particular, in Dhamala, et al. (2001) an ensemble of chaotic transients is generated from the Hénon map for the parameter pairs $(a, b) = (1.46, 0.3)$ and $(a, b) = (1.50, 0.3)$, and 21,000 points near the chaotic set are accumulated by using 300 random initial conditions in $[-2, 2] \times [-2, 2]$ for the case $a = 1.46$ and 700 random initial conditions for $a = 1.50$. (The average lifetime of the chaotic transients is about 70 iterates for $a = 1.46$ and 30 iterates for $a = 1.50$.) A two-dimensional embedding with a time delay of 1

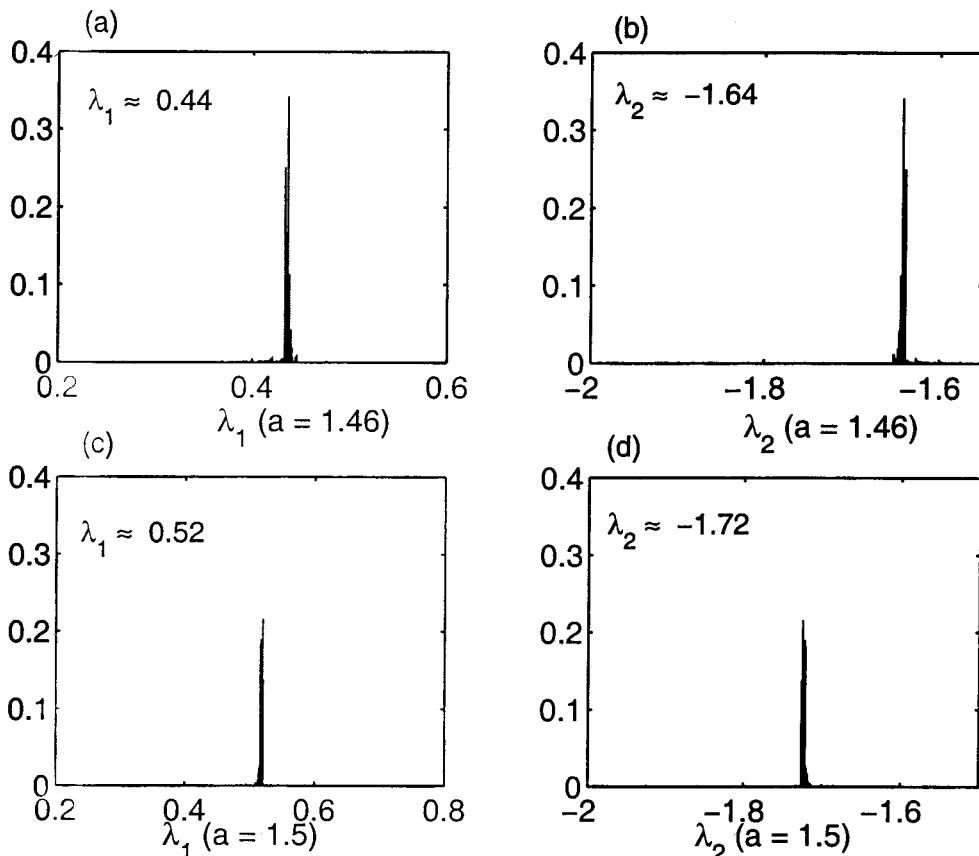


FIG. 12.8. (a-d) The distributions of Lyapunov exponents of the Hénon map for chaotic transients at the two parameter values of $a = 1.46$ and $a = 1.50$, respectively.

is constructed from each collection of time series. Local linear maps are computed using least squares for each neighborhood.

In the case $a = 1.46$, each transient time series consists of about 70 iterates. Thus, the “Lyapunov exponents” computed are actually finite-time approximations, where a suitable product of the 70 or so linear maps associated with points on the individual transient time series is considered. Similarly, when $a = 1.50$, it is necessary to consider products of 30 or so linear maps. Figure 12.8(a) shows the distribution of the largest Lyapunov exponent, λ_1 , computed from the ensembles of time series for $a = 1.46$, and Figure 12.8(b) shows the distribution of the values of λ_2 . It can be seen that $\lambda_1 = 0.44 \pm 0.05$ and $\lambda_2 = -1.72 \pm 0.06$. Similarly, for $a = 1.50$, the exponents are: $\lambda_1 = 0.54 \pm 0.06$ and $\lambda_2 = -1.77 \pm 0.08$. The estimated values of the exponents agree reasonably well with the theoretical ones (Dhamala et al., 2001).

TIME-FREQUENCY ANALYSIS OF TIME SERIES

The method of chaotic time series analysis described in the previous section is powerful for low-dimensional, low-noise deterministic dynamical systems. For more general and more complicated signals, time-frequency analyses are still commonly used. An excellent discussion and comparison of various time-frequency methods can be found in Huang et al. (1998). These are (1) the traditional Fourier method, (2) the wavelet method, and (3) the Hilbert transform method. In what follows we briefly discuss the Fourier and wavelet methods, following mainly the discussion in Huang (1998). We then focus on the Hilbert transform and give examples to illustrate its usefulness.

Fourier spectral analysis is traditionally the time series analysis method and can be found in many standard textbooks. A necessary condition for the Fourier analysis to be meaningful is that the time series should be piecewise stationary. Even then there are common situations in which the Fourier analysis cannot yield much information, such as deterministic chaotic time series with broad-band power spectra for which the Fourier spectrum gives absolutely no indication about the deterministic origin, let alone the fundamental invariants of the underlying dynamical system. There exist other technical difficulties associated with the Fourier analysis. For instance, when a set of windows is chosen from a piecewise stationary time series for Fourier analysis, how can one guarantee that the window width coincides with the stationary time scale? To detect events localized in time, the window width must be small, which usually results in poor resolution in the frequency domain. Therefore, in general the Fourier method is useful for stationary time series. It is particularly powerful if the underlying system is linear.

In recent years wavelet analysis has become more popular for the analysis of nonstationary time series, which is essentially an *adjustable* window Fourier spectral analysis. In general, for a time series $x(t)$, the wavelet transform is defined to be:

$$W(a, b) = \frac{1}{\sqrt{|a|}} \int_{-\infty}^{\infty} x(t) \psi^* \left(\frac{t-b}{a} \right) dt, \quad (30)$$

where $\psi^*(\cdot)$ is the basic wavelet function that satisfies certain general conditions, a is the dilation factor, and b is the translation parameter. Although time and frequency do not appear explicitly in $W(a, b)$, the variable $1/a$ gives the frequency scale and b indicates the temporal location of an event. Physically, $W(a, b)$ is the energy of $x(t)$ of scale a about $t = b$. For different problems the choices of the wavelet function $\psi^*(\cdot)$ are usually different. For instance, a commonly used function is the Morlet wavelet, which is essentially Gaussian enveloped sine and cosine waves (Chan, 1995). Fundamentally, the wavelet analysis is a linear method.

Although it provides a uniform energy resolution for all the scales, the resolution is usually poor because the size of the basic wavelet function is limited (Huang, 1998). In the past decade or so there has been a tremendous amount of work on wavelet analysis, particularly in the applied mathematics literature. Successful applications include edge detection and image compression (Chan, 1995). Here, we do not give a detailed discussion of the wavelet analysis, as there are chapters in this book dealing exclusively with it.

Instead, we describe a recent method based on the Hilbert transform, which is powerful for nonstationary, nonlinear, and/or random time series. The method was pioneered by Huang et al. (1998) and has been demonstrated to be useful for a number of applications. In what follows we describe the basic ideas of the method and give examples of applications in chaotic systems.

Analytic Signals and Hilbert Transform

Analytic Signals. Signals in the physical world are real. Nevertheless, in a variety of applications such as optics, it is convenient to represent real signals by complex ones. The concept of *analytic signals*, first proposed by Gabor in his study of optical holography, is a natural way to define a complex signal with a clear physical meaning. In particular, given a real signal $x(t)$, one performs a mathematical transform to obtain the corresponding imaginary part $\tilde{x}(t)$, yielding the following complex signal $\psi(t)$:

$$\psi(t) = x(t) + i\tilde{x}(t) = A(t) e^{i\phi(t)}.$$

Suppose that the imaginary part can be obtained uniquely through the mathematical transform in the sense that the analytic signal $\psi(t)$ corresponds geometrically to a rotation, the following amplitude and phase can then be defined:

$$A(t) = \sqrt{x(t)^2 + \tilde{x}(t)^2}, \quad \phi(t) = \arctan\left(\frac{\tilde{x}(t)}{x(t)}\right).$$

The phase variable $\phi(t)$ gives the following *instantaneous frequency* $\omega(t)$:

$$\omega(t) = \frac{d\phi(t)}{dt} = \frac{x(t)\dot{\tilde{x}}(t) - \dot{x}(t)\tilde{x}(t)}{A^2(t)}, \quad (31)$$

where $\dot{\tilde{x}}(t)$ and $\dot{x}(t)$ denote the derivatives of $\tilde{x}(t)$ and $x(t)$ to t , respectively. Note that the instantaneous frequency $\omega(t)$ is fundamentally different from the concept of frequency in the Fourier transform defined in the base of simple harmonic functions. Here the base is the physically meaningful concept of rotations. The instantaneous frequency $\omega(t)$ measures the rate of rotation in the complex plane of the corresponding analytic signal.

The main issue is then how to define the imaginary part of complex signal. Interest in the proper definition of the imaginary part first arose with the advent of frequency modulation for radio transmission in the 1920s. Carson and Fry and later Van der Pol approached the problem of formulating a definition for instantaneous frequency. They argued that the notion of instantaneous frequency is a generalization of the definition of frequency, and that instantaneous frequency would lead to a physically consistent picture of frequency modulation. The necessity for the notion of instantaneous frequency, which quantifies the rate of change of phase angle, was clearly identified and a general scheme was proposed, but a good definition of phase angle was missing at the time. It was Gabor who proposed a solution to the problem by inventing the analytic signal.

Gabor's approach is as follows. Observe that if the real signal $x(t)$ has a Fourier transform $S(\omega)$, then the complex signal, $\psi(t)$, the spectrum of which is composed of positive frequencies of $S(\omega)$ only, is given by the inverse transform of $S(\omega)$, where the integration goes only over the positive frequencies:

$$\psi(t) = \frac{2}{\sqrt{2\pi}} \int_0^\infty S(\omega) e^{i\omega t} d\omega$$

The factor of 2 is inserted so that the real part of the analytical signal is $x(t)$, not one half of that. The explicit form of $\psi(t)$ can then be obtained in terms of the real signal $x(t)$. Because

$$S(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^\infty x(t) e^{-i\omega t} dt,$$

the complex signal can be written as:

$$\begin{aligned} \psi(t) &= 2 \frac{1}{2\pi} \int_0^\infty \int_{-\infty}^\infty x(t') e^{-i\omega t'} e^{i\omega t} dt' d\omega \\ &= \frac{1}{\pi} \int_0^\infty \int_{-\infty}^\infty x(t') e^{-i\omega(t-t')} dt' d\omega. \end{aligned}$$

The following mathematical identity (Arfken, 1995):

$$\int_0^\infty e^{i\omega\tau} d\omega = \pi\delta(\tau) + \frac{i}{\tau}$$

gives:

$$\psi(t) = \frac{1}{\pi} \int_{-\infty}^\infty x(t') \left[\pi\delta(t-t') + \frac{i}{t-t'} \right] dt',$$

which yields (Hahn, 1996):

$$\psi(t) = x(t) + i \frac{1}{\pi} \int_{-\infty}^\infty \frac{x(t')}{t-t'} dt' \quad (32)$$

which is the analytic signal corresponding to the real signal $x(t)$. The imaginary part of Equation 32 is just the Hilbert transform $\tilde{x}(t)$ of the real signal $x(t)$:

$$\tilde{x}(t) = P.V. \left[\frac{1}{\pi} \int_{-\infty}^\infty \frac{x(t')}{t-t'} dt' \right], \quad (33)$$

where $P.V.$ stands for the Cauchy principal value for the integral. In principle, there are many ways to define a complex function from $x(t)$, but the Hilbert transform provides a unique way to define $\tilde{x}(t)$ (also known as the *quadrature signal*; (Okunev, 1997) so that $\psi(t)$ has an analytic continuation over the complex upper half-plane. From a physical standpoint $x(t)$ represents some physical measurement and also serves as a boundary condition for defining some analytic function.

The Hilbert Transformation. To better understand the meaning of the Hilbert transform, we examine some mathematical issues. Consider the following one-dimensional integral-transform pair:

$$u(t) \Leftrightarrow U(s)$$

where a time function $u(t)$ is transformed into a complex function of a real variable s . This notation may also be written in the form of the following pair of integrals (Hahn, 1996):

$$\begin{aligned} U(s) &= \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{u(t)}{s-t} dt \\ u(t) &= \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{U(s)}{t-s} ds. \end{aligned}$$

A careful look at these integrals reveals that the Hilbert transforms are defined using the kernel $\frac{1}{\pi(s-t)}$ and the conjugate kernel $\frac{1}{\pi(t-s)}$. In general, the Hilbert transforms are written in a more explicit form:

$$\tilde{u}(t) = P.V. \left[\frac{-1}{\pi} \int_{-\infty}^{\infty} \frac{u(\tau)}{\tau-t} d\tau \right], \quad (34)$$

$$u(t) = P.V. \left[\frac{1}{\pi} \int_{-\infty}^{\infty} \frac{\tilde{u}(\tau)}{\tau-t} d\tau \right]. \quad (35)$$

The integrals in these equations are improper integrals in the sense of Cauchy principal value. For Equation 35 the integral is defined by the limit (Hahn, 1996):

$$u(t) = \lim_{\epsilon \rightarrow 0; L \rightarrow \infty} \frac{1}{\pi} \left\{ \int_{-L}^{t-\epsilon} \frac{\tilde{u}(\tau)}{\tau-t} d\tau + \int_{t+\epsilon}^L \frac{\tilde{u}(\tau)}{\tau-t} d\tau \right\}.$$

Essentially, Equations (34) and (35) define convolution, which can be seen by applying the change of variable ($\tau \rightarrow t - \tau$) to the Hilbert transform given in Equation 34. This yields:

$$\tilde{u}(t) = \int_{-\infty}^{\infty} \frac{u(t-\tau)}{\pi\tau} d\tau, \quad (36)$$

where the integral again is to be taken as Cauchy principal value. This form of Hilbert transformation clearly shows that $\tilde{u}(t)$ is the convolution of $u(t)$ with $\frac{1}{\pi t}$, that is,

$$\begin{aligned} \tilde{u}(t) &= \frac{1}{\pi t} * u(t), \\ u(t) &= \frac{-1}{\pi t} * \tilde{u}(t), \end{aligned} \quad (37)$$

where $*$ denotes the convolution operation.

Numerical Computation of the Analytical Signal. The key to numerically computing the analytic signal is the relation between the Hilbert and the Fourier transforms. In particular, the Hilbert transform of $x(t)$ is the positive part of the Fourier transform of $x(t)$, multiplied by two (Frenking, 1994). This can be seen by examining the convolution given

in Equation 37. Recall that the convolution $g * h$ is one member of a simple transformation pair:

$$g * h \Leftrightarrow \mathcal{F}T(g)\mathcal{F}T(h),$$

where $\mathcal{F}T$ denotes the Fourier transformation, which is the convolution theorem. We thus have (Shenoi, 1995):

$$\begin{aligned} \frac{1}{\pi t} * u(t) &\Leftrightarrow \mathcal{F}T\left(\frac{1}{\pi t}\right)\mathcal{F}T(u(t)) \\ &\Leftrightarrow -i \operatorname{sgn}(\omega) U(\omega), \end{aligned}$$

where $U(\omega)$ is the complex Fourier transform of $u(t)$ and the signum function is defined as (Hahn, 1996):

$$\operatorname{sgn}(\omega) = \begin{cases} +1 & \text{for } \omega > 0 \\ 0 & \text{for } \omega = 0 \\ -1 & \text{for } \omega < 0 \end{cases}$$

Because $u(t)$ is a real function of time, we have $U(-\omega) = U^*(\omega)$. Then, the analytical signal can be written as

$$\psi(t) = u(t) + i\tilde{u}(t) = 2 \frac{1}{\sqrt{2\pi}} \int_0^\infty U(\omega) e^{i\omega t} d\omega. \quad (38)$$

This simple result is important because it means that, if the signal can be expressed in the Fourier frequency domain, then the analytic signal can be obtained by dropping negative-frequency terms, multiplying the positive-frequency terms by two, and transforming back to time domain.

Instantaneous Frequency. Instantaneous frequency, together with phase, is an intuitive concept. The exact mathematical description of the frequency modulation is quite intriguing, but the following simple argument suggests the physical meaning of the definition of the instantaneous frequency as the derivative of the phase variable.

Assume that $\psi(t)$ has the spectrum $S(\omega)$, then the mean frequency is given by Boashash, Powers, & Zoubir (1995):

$$\begin{aligned} \langle \omega \rangle &= \int \omega |S(\omega)|^2 d\omega = \frac{1}{2\pi} \iiint \omega \psi^*(t) \psi(t') e^{i\omega(t-t')} d\omega dt' dt \\ &= \frac{1}{2\pi i} \iiint \psi^*(t) \psi(t') \left(\frac{\partial}{\partial t} e^{i\omega(t-t')} \right) d\omega dt' dt \\ &= \frac{1}{i} \iint \psi^*(t) \left(\frac{\partial}{\partial t} \delta(t-t') \right) \psi(t') d\omega dt' dt \\ &= \int \psi^*(t) \frac{1}{i} \left(\frac{d}{dt} \psi(t) \right) dt, \end{aligned}$$

which can be written also as:

$$\langle \omega \rangle = \int \left[\frac{d}{dt} \phi(t) - \frac{i}{A(t)} \left(\frac{d}{dt} A(t) \right) \right] A^2(t) dt.$$

The second term is zero because that term is purely imaginary; it must be zero for $\langle \omega \rangle$ to be real, which gives (Huang, 1998):

$$\langle \omega \rangle = \int \left(\frac{d}{dt} \phi(t) \right) |S(\omega)|^2 d\omega = \int \left(\frac{d}{dt} \phi(t) \right) A^2(t) dt.$$

This is an interesting and important result because it says that the average frequency may be obtained by integrating the instantaneous frequency with the density over all time:

$$\omega(t) = \frac{d}{dt} \phi(t). \quad (39)$$

As an example, Fig. 12.9(a) shows the plot of $x(t)$ of a simple harmonic oscillator, versus $\tilde{x}(t)$, the Hilbert transform of $x(t)$. For $x(t) = A \sin(\omega t)$, where ω is a constant, the Hilbert transform is trivial and can be performed by shifting all frequency components of $x(t)$ by $\frac{\pi}{2}$, that is, $\tilde{x}(t) = -A \cos(\omega t)$. The unwrapped phase function corresponding to Fig. 12.9(a) is a monotonically increasing straight line as shown in Fig. 12.9(b), and the instantaneous frequency shown in Fig. 12.9(c) is constant, as expected.

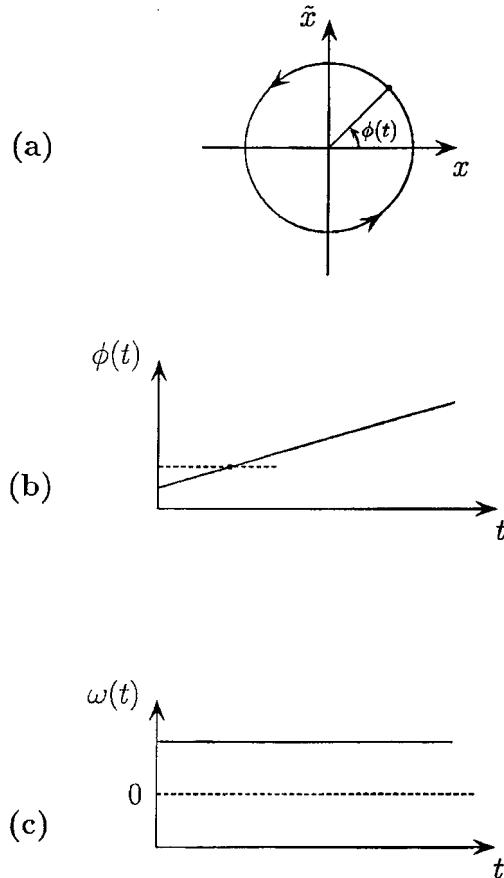


FIG. 12.9. Illustration of the Hilbert transform and instantaneous frequency associated with a simple harmonic oscillator: (a) the real signal $x(t)$, (b) its Hilbert transform, and (c) the constant frequency.

With these notations, the bandwidth associated with the spectrum of the instantaneous frequency can be defined as (Huang, 1998):

$$\begin{aligned} B^2 &= \sigma_{\omega}^2 = \int (\omega - \langle \omega \rangle)^2 |S(\omega)|^2 d\omega \\ &= \int \psi^*(t) \left(\frac{1}{i} \frac{d}{dt} - \langle \omega \rangle \right)^2 \psi(t) dt \\ &= \int \left| \left(\frac{1}{i} \frac{d}{dt} - \langle \omega \rangle \right) \psi(t) \right|^2 dt \\ &= \int \left| \frac{-i}{A(t)} \left(\frac{d}{dt} A(t) \right) + \frac{d}{dt} \phi(t) - \langle \omega \rangle \right|^2 A^2(t) dt, \end{aligned}$$

or

$$B^2 = \int \left(\frac{d}{dt} A(t) \right)^2 dt + \int \left[\frac{d}{dt} \phi(t) - \langle \omega \rangle \right]^2 A^2(t) dt.$$

For a *narrow band signal*, B^2 must be a small, that is, both the amplitude $A(t)$ and the phase $\phi(t)$ are slowly varying functions of time.

Even with the definition given in Equation 39 as the derivative of phase, there still is a considerable controversy over the meaning of instantaneous frequency, especially if the signal is not *monocomponent*, a term that was introduced to ensure $\omega(t)$ has a *narrow band spectrum*. Suppose $\phi(t)$ generated by Equation 32 can be represented by:

$$\phi_i(t) = \int_0^t \omega_i(\tau) d\tau.$$

Here, i indicates the different components having different oscillatory frequencies. Assume the original signal $x(t)$ can be expressed as (Boashash, 1992):

$$x(t) = \sum_{i=1}^N C_i(t) + \eta(t), \quad (40)$$

where $\eta(t)$ represents a residue term with negligible amplitude and N is some finite number. By construction, each C_i is an *intrinsic mode* of $x(t)$ with a simple oscillatory waveform described by the envelopes $A_i(t)$ and the instantaneous frequencies $\omega_i(t)$ such that the analytic signal $\psi_i(t)$ associated with $C_i(t)$ is

$$\psi_i(t) = A_i(t) e^{\phi_i(t)} \quad \text{that is,} \quad C_i(t) = A_i(t) \cos(\phi_i(t))$$

If $i = 1$, the signal is said to be a monocomponent signal; if, however, $i \geq 2$, then the signal is referred to as a multicomponent signal (Boashash, 1992).

Figure 12.10(a) shows an example of a multicomponent signal. The real part of the signal is $x(t) = A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t)$ with $\omega_1 = 1$, $\omega_2 = 3$, and $A_1 = 1$, $A_2 = 0.9$. The spectrum of this signal consists of two delta functions at ω_1 and ω_2 :

$$S(\omega) = A_1 \delta(\omega - \omega_1) + A_2 \delta(\omega - \omega_2).$$

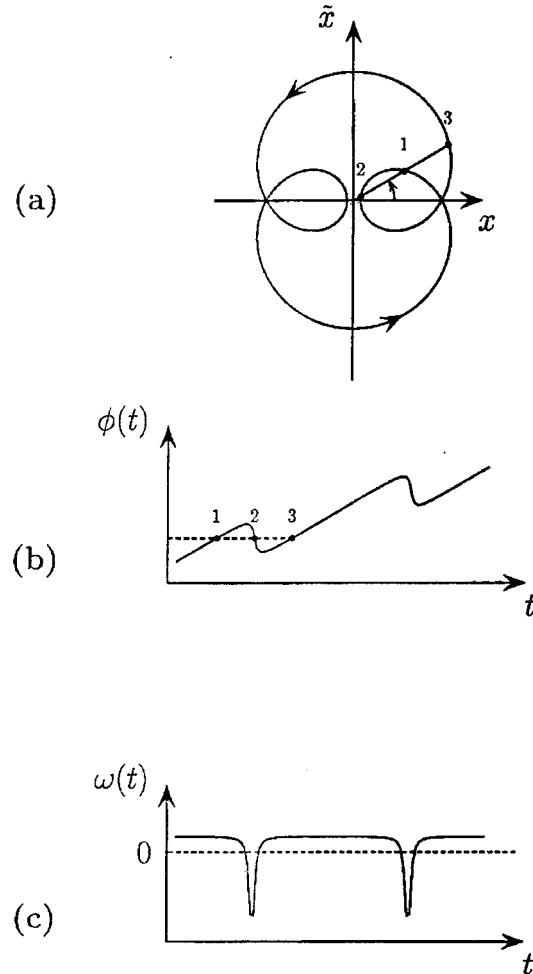


FIG. 12.10. (a) The plot of $x(t) = \sin(t) + 0.9 \sin(3t)$ in its complex plane of analytic signal. The Hilbert transform of the signal is $\tilde{x}(t) = -\cos(t) - 0.9 \cos(3t)$. Clearly, the rotation is not a proper one. The trajectory needs to change the direction of rotation with respect to the origin to pass through the points denoted by 1, 2, and 3. (b) Unwrapped phase function $\phi(t)$ obtained from (a). (c) Time derivative of the phase function in (a) has intervals with negative frequencies that are *unphysical*.

Because ω_1 and ω_2 are taken to be positive, the signal is analytic. Solving for the phase and amplitude:

$$\phi(t) = \arctan \frac{A_1 \sin(\omega_1 t) + A_2 \sin(\omega_2 t)}{A_1 \cos(\omega_1 t) + A_2 \cos(\omega_2 t)},$$

$$A^2(t) = A_1^2 + A_2^2 + 2A_1 A_2 \cos(\omega_2 - \omega_1)t$$

and taking the derivative of the phase yield the instantaneous frequency:

$$\omega(t) = \frac{d}{dt} \phi(t) = \frac{1}{2}(\omega_2 + \omega_1) + \frac{1}{2}(\omega_2 - \omega_1) \frac{A_1^2 - A_2^2}{A^2(t)}.$$

Notice that due to the last term in the above equation, the trajectory in Fig. 12.10(a) traces out a rotation with two separate centers. This is more apparent in Fig. 12.10(b), where the unwrapped phase function is no longer a monotonically increasing function of time. The result of this behavior is evident in the instantaneous frequency plot shown in Fig. 12.10(c). Clearly, the negative values assumed by the instantaneous frequency are *unphysical*. This example illustrates that even for a simple signal, a meaningful instantaneous frequency can be obtained only if some restrictive conditions are imposed on the data. However, almost all of the conditions discussed in several references (Bedrosian, 1963; Boashash, 1992) are global and do not provide a scheme as to how to obtain the decomposition given in Equation 40. Recently, Huang et al. (1998) introduced the empirical mode decomposition (EMD) method for generating intrinsic modes that are monocomponent.

Method of EMD

To define physically meaningful instantaneous frequencies, it is necessary that the analytic signal possess a proper structure of rotation. For instance, there should be a unique center of rotation. More precisely, the analytic signal $\psi(t)$ should satisfy the following two conditions in its own complex plane: (1) There is a preferred direction of rotation (e.g., either clockwise or counterclockwise) and (2) the rotation can be defined with respect to a unique center.

The first condition is self-explanatory, and the second condition ensures that the instantaneous frequency does not have any undesirable fluctuations induced by asymmetric waveforms (Huang, 1998). Physically, these two requirements amount to having a well-behaved rotation-like motion as shown in Fig. 12.9(a). It is natural to call a rotation satisfying these two conditions in the complex plane of analytic signal a *proper rotation*. For a complicated signal that does not satisfy the above conditions, it is necessary to express it by a sum of proper rotations (if possible). The EMD method developed by Huang et al. (1998) is such a method. The heart of the decomposition method is to identify the innate undulations belonging to different time scales and sift them out to obtain one intrinsic mode at a time. This can be achieved by making use of the envelopes defined by the local maxima and minima to discern waves riding on top of the others. Once the extrema of the data set are identified, all the local maxima and local minima are connected by a cubic spline to form an upper envelope $s_{\max}(t)$ and a lower envelope $s_{\min}(t)$. Their mean is denoted by:

$$m(t) = \frac{s_{\max}(t) + s_{\min}(t)}{2} \quad (41)$$

and it is subtracted (*sifted out*) from the original data $x(t)$ to yield $r(t)$:

$$r(t) = x(t) - m(t).$$

Ideally, $r(t)$ should be an intrinsic mode; however, due to the imperfections in the construction of the envelopes, overshoots and undershoots can generate new extrema or modify the existing ones. Therefore, $r(t)$ must be checked to see if it satisfies the two conditions specifying a proper rotation. If the conditions are not satisfied, the sifting procedure is carried out again by forming $s_{\max}(t)$ and $s_{\min}(t)$ but this time from $r(t)$. The mean $m(t)$ is calculated by Equation 41. That is, in the second sifting $r(t)$ is regarded as the data to be analyzed:

$$r(t) \rightarrow r(t) - m(t).$$

By sifting out the local mean $m(t)$ from $r(t)$, some riding waves can be eliminated, yielding wave profiles that are more symmetric. This process of sifting continues until $r(t)$ corresponds

to a proper rotation. Then, it is designated as:

$$C_1(t) = r(t),$$

which is the first intrinsic mode. In general, $C_1(t)$ contains the highest frequency oscillations of the original signal because the envelopes are formed using maxima of the fastest riding waves. This shortest time-scale component can be subtracted from the data:

$$x(t) \rightarrow x(t) - C_1(t).$$

The remainder can be treated as the new data subjected to the same sifting process described earlier. This procedure can be applied repeatedly to obtain subsequent intrinsic modes $C_i(t)$ belonging to different time scales. The sifting process is stopped when $r(t)$ shows no apparent variations (i.e., it has fewer than two local extrema) or the amplitude of the oscillations in the time domain becomes negligibly small compared with the amplitude of the original signal $x(t)$. The last component is denoted by $\epsilon(t)$, which is roughly a quadratic or a linear function of time with relatively small amplitude. Summarizing these steps, it can be seen that the original signal $x(t)$ is decomposed in the following manner:

$$x(t) = \sum_{i=1}^M C_i(t) + \epsilon(t),$$

as given in Equation 40.

To illustrate the decomposition procedure, we use the data collected from the chaotic attractor of the Lorenz system (Lorenz, 1963):

$$\begin{aligned} \frac{dx}{dt} &= 16(x - y), \\ \frac{dy}{dt} &= -xz + 45.92x - y, \\ \frac{dz}{dt} &= xy - 4z. \end{aligned} \quad (42)$$

Figure 12.11(a) shows the time series obtained from $y(t)$ for $\sigma = 16$, $\rho = 45.92$, and $\beta = 4$. The data appear to be quite complicated and, despite many local extrema, there are relatively fewer zero crossings. The corresponding trajectory in the complex plane of $\psi(t)$ is shown in Fig. 12.11(b), where it is apparent that the rotation is not proper as it has two centers and, hence, in this case no proper phase function $\phi(t)$ can be defined. Figure 12.11(c) shows the trajectory in the complex plain of the first intrinsic mode $C_1(t)$ from the Lorenz system. Now there is a unique center for which the phase of $C_1(t)$ can be properly defined.

The first component $C_1(t)$ shown in Fig. 12.11(c) is obtained as follows: We first start identifying the local extrema and obtain upper and lower envelopes. In practice, serious problems can occur near the end points of the original signal $x(t)$ during the spline fitting process. If the edges of the data are left untreated, the spline fittings introduce large perturbations into the data that can propagate inward and eventually corrupt the signal. An example of the edge effects is shown in Fig. 12.12(a). Here the thin line is the data to be sifted and the thick solid line is the upper envelope constructed by connecting all the local maxima by cubic splines. The original signal has about 64000 data points and lasts for 640 seconds; however, the figure shows only the first 6 seconds of the signal. At the beginning of the signal, one would expect to see an

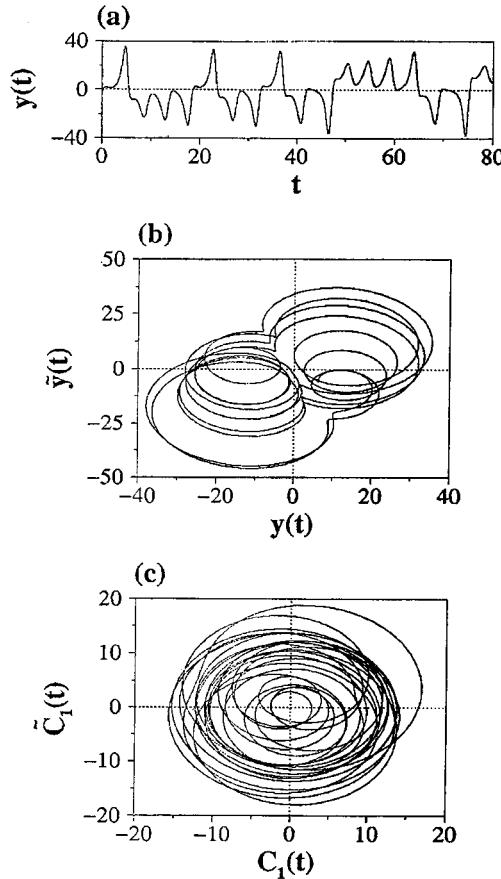


FIG. 12.11. (a) A trajectory from y -component of the Lorenz equation. (b) The same trajectory in the complex plain of the analytic signal. The motion exhibits multiple centers of rotation so that a proper phase cannot be defined. (c) A similar trajectory from the first intrinsic mode $C_1(t)$ of $y(t)$ of the Lorenz system. In this case, the phase $\phi_1(t)$ can be properly defined.

envelope function with zero slope. However, due to the way the cubic splines are constructed, the envelope function exhibits a large swing at the edge of the data. Combined with the similar behavior from the lower envelope, some artificial extrema with relatively large amplitudes are generated. As the sifting procedure is continued, these undesired edge effects can propagate toward the middle of the signal and eventually corrupt it. Here, a procedure similar to the one in Huang (1998) is utilized (see Yalcinkaya, 1998) to eliminate the edge effects. Specifically, a number of extra data points is introduced for both the beginning and the end of the data by repeating the local extrema of the typical waves. To preserve the original data length, they are located outside the given data interval. As a result, more pairs of additional maxima and minima are to be used by splines. Figure 12.12(b) shows the reduction in the artificial swing in the upper envelope function using six extra points at both the beginning and the end of the data.

The edge effects are not the only problems with spline fittings. An ideal envelope should enclose the data and should not interfere with the general flow of it. In reality, however, undershoots and overshoots are present even with very simple data. The example shown in Fig. 12.12 contains several locations where the envelope crosses the signal and exhibits uncharacteristic

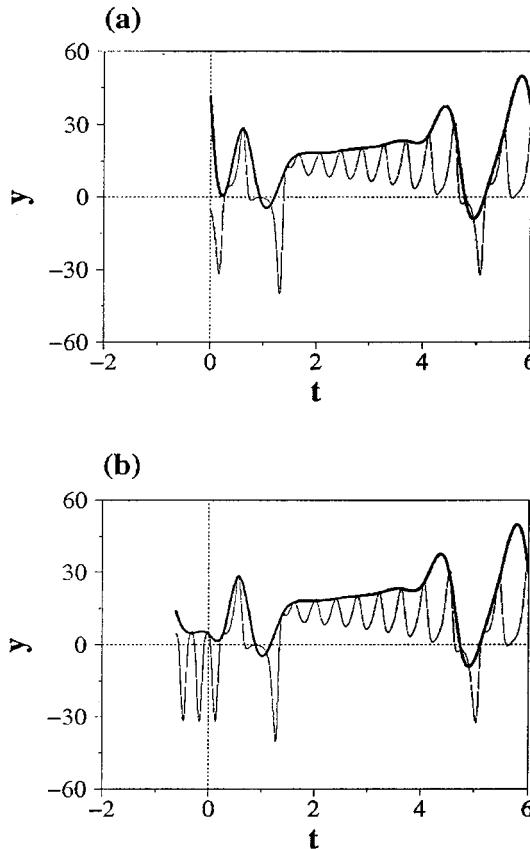


FIG. 12.12. (a) A time series collected from the chaotic attractor of the Lorenz system. The upper envelope exhibits a large swing at the beginning of the signal because the edge is not treated. (b) Same data with the edges are properly treated.

swings. It may seem that such imperfections in the envelope construction would have drastic effects in the final form of the intrinsic modes; however, that is not the case. In general, these artificial oscillations introduced by the envelopes belong to different time scales, and hence they are eliminated by the sifting procedure.

Figures 12.13 and 12.14 summarize the first seven intrinsic modes from this repeated sifting process with the edge effects taken care of as illustrated in Fig. 12.12(b). Although it appears that the improvement in the method of spline fitting is critical to minimizing unwanted undulations introduced by the numerical procedure, the choice of which spline to use does not have substantial influence on the obtained intrinsic modes (Yalcinkaya, 1998). That is, qualitatively similar intrinsic modes are obtained, provided that the splines are smooth functions of time. Therefore, the intrinsic modes, as exemplified in Figs. 12.13 and 12.14, are robust.

In general, the number of fundamental modes M required to capture the rotation-like motions in a chaotic signal is *small*. Figure 12.15 shows the phase functions $\phi_j(t)$ corresponding to the modes shown in Figs. 12.13 and 12.14 for the Lorenz flow. There appears to be a separation between the average rotation frequencies (the average slopes) of the various modes, with ω_1 being the largest. It can also be seen that $\omega_j \approx 0$ for $j \geq 7$, indicating that the eighth mode and up are insignificant.

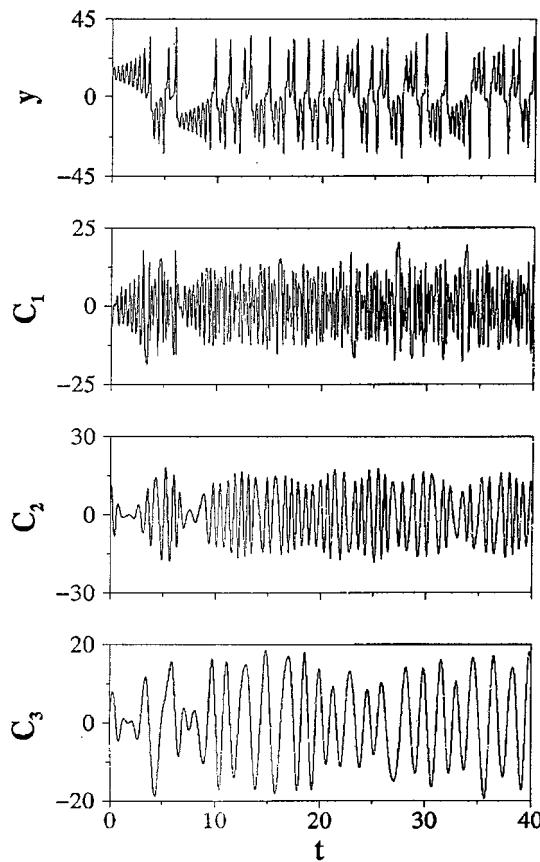


FIG. 12.13. A trajectory from the y component of the Lorenz equation and the three intrinsic modes.

What are the characteristics of the distribution of the mean instantaneous frequencies? Because the signal is chaotic and has a broadband Fourier spectrum, one may expect an unlocalized distribution of the average frequencies. Nonetheless, as described, the intrinsic modes, by construction, have narrow bandwidth. Therefore, the EMD method yields frequency components that are inherently different from those of Fourier transform. Figure 12.16 shows the histograms of the average instantaneous frequencies for all three state variables, $x(t)$, $y(t)$, and $z(t)$, of the Lorenz equations with $\sigma = 16$, $\rho = 45.92$, and $\beta = 4$. Each histogram is obtained using 5,000 samples with random initial conditions and calculating $\langle \omega_j \rangle$ for $(j = 1, \dots, 4)$. There is a clear separation among the distribution of the average frequencies for each intrinsic mode. In addition, a quick look at the average of the frequency distributions reveals something more interesting. Table 12.1 shows the mean of the instantaneous frequency distribution from the x -component of the Lorenz system, the associated standard deviation, and the relative ratios. Interestingly, if the mean instantaneous frequency of C_1 is regarded as the fundamental frequency, the remaining intrinsic modes exhibit roughly rational harmonics of this fundamental frequency. A similar relationship is also apparent in Tables 12.2 and 12.3. Furthermore, $x(t)$ and $y(t)$ seem to have approximately the same frequency distribution and same fundamental frequencies, that is, $\omega_x \cong \omega_y$. This shows that the phase dynamics of the Lorenz system has an unexpected simplicity. Roughly speaking, there are only two fundamental frequencies

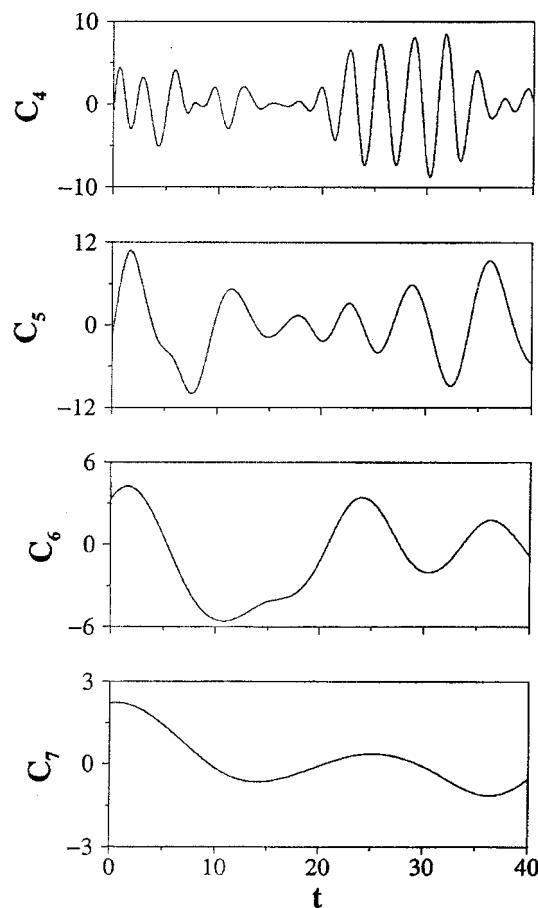


FIG. 12.14. Next four intrinsic modes from the data shown in Fig. 12.13.

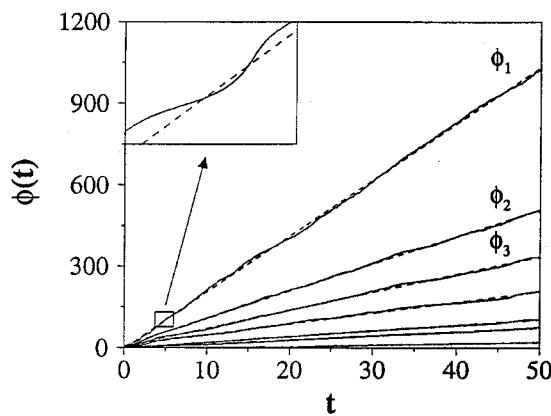


FIG. 12.15. $\phi_j(t)$ versus t for $j = 1, \dots, 8$. Note that the phase variation in $\phi_8(t)$ is already approximately zero, indicating that further components are insignificant.

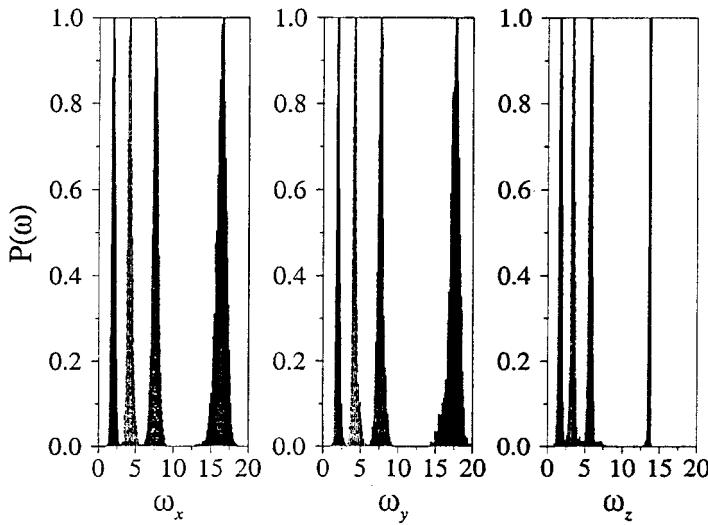


FIG. 12.16. The histograms obtained from Lorenz. Each distribution is obtained using 5,000 random initial condition each yielding a time series for $0 \leq t \leq 600$ (after a sufficiently long transient time) and calculating the mean frequencies for the first four intrinsic modes.

TABLE 12.1
Frequency Distribution of x -Component from
the Lorenz Chaotic Attractor

$x(t)$	$\langle \omega \rangle$	σ	Ratio
C_1	16.3	0.68	ω_x
C_2	7.6	0.43	$\approx \omega_x/2$
C_3	4.1	0.32	$\approx \omega_x/4$
C_4	1.0	0.24	$\approx \omega_x/16$

TABLE 12.2
Frequency Distribution of y -Component from
the Lorenz Chaotic Attractor

$y(t)$	$\langle \omega \rangle$	σ	Ratio
C_1	17.4	0.71	ω_y
C_2	7.6	0.41	$\approx \omega_y/2$
C_3	4.2	0.31	$\approx \omega_y/4$
C_4	1.9	0.26	$\approx \omega_y/8$

TABLE 12.3
Frequency Distribution of z -Component from
the Lorenz Chaotic Attractor

$z(t)$	$\langle \omega \rangle$	σ	Ratio
C_1	13.6	0.08	ω_z
C_2	5.8	0.28	$\approx \omega_z/2$
C_3	3.4	0.26	$\approx \omega_z/4$
C_4	1.7	0.85	$\approx \omega_z/8$

governing the phase dynamics of the Lorenz equations and all the auxiliary frequencies are obtained from these two.

In principle, an infinitely long chaotic signal can be decomposed into an infinite number of proper rotations. However, we find that the amplitudes and the average rotation frequencies decrease rapidly as higher-order modes are examined. Thus, a few proper rotations are sufficient to represent the phase of the chaotic signal, but we have no rigorous assurance of this.

It should be mentioned that, although the sifting procedure is necessary for chaotic flows that exhibit multiple centers of rotation in the complex plane of its analytic signal, there are systems in which the flow apparently already has a unique center of rotation. In this case the sifting procedure is not necessary, and one can define the phase associated with the flow directly from the analytic signal. Flows on the Rössler attractor appear to belong to this category (Rössler, 1976).

SUMMARY

In this chapter we described two modern techniques in *nonlinear* time series analysis: (1) the delay-coordinate embedding method and, (2) the Hilbert-transform method. The first is useful when the underlying dynamical system is deterministic and low-dimensional, whereas the second can be applied to both deterministic and stochastic systems. In addition, the Hilbert analysis has the advantage of being suitable for nonstationary time series to assess the instantaneous-frequency spectrum of the system. When combined with the empirical-mode decomposition procedure, the Hilbert transform can yield understanding of the physical mechanisms responsible for the observed time series, which cannot be revealed by the traditional Fourier or the more recent wavelet transform method. An example is the record of the time of a day. The exact time of a day on earth is not precisely 24 hours. Instead, the times have fluctuations on the order of microseconds. By analyzing the data of fluctuations using the empirical-mode decomposition and the Hilbert analysis, Huang et al. (1998) were able to identify clearly the various physical origin of the fluctuations such as the influences from the motions of the sun and the moon, as these physical motions have distinct instantaneous-frequency characterizations.

ACKNOWLEDGMENT

Part of the materials in the Hilbert analysis is based on part of the PhD thesis by T. Yalcinkaya at the University of Kansas in 1998 under YCL.

REFERENCES

- Arfken, G. B., & Weber, H. J. (1995) *Mathematical methods for physicists* (4th ed.). San Diego: Academic Press.
- Auerbach, D., Cvitanovic, P., Eckmann, J.-P., Gunaratne, G. H., & Procaccia, I. (1987). Exploring chaotic motion through periodic orbits. *Physical Review Letters*, 58, 2387–2389.
- Bedrosian, E. (1963). A product theorem for Hilbert transform. *Proceedings of the IEEE*, 51, 868–869.
- Benettin, G., Galgani, L., Giorgilli, A., & Strelcyn, J. M. (1980). Lyapunov characteristic exponents for smooth dynamical systems and for Hamiltonian systems: A method for computing all of them. Part 2: Numerical application. *Meccanica*, 15, 21–44.
- Boashash, B. (1992). Estimating and interpreting the instantaneous frequency of a signal. I. Fundamentals. *Proceedings of the IEEE*, 80, 520–538.
- Boashash, B., Powers, E. J., & Zoubir, A. M. (1995). *Higher-order statistical signal processing*. New York: Wiley.

- Brown, R., Bryant, P., & Abarbanel, H. D. I. (1991). Computing the Lyapunov spectrum of a dynamical system from an observed time series. *Physical Review A*, 43, 2787–2806.
- Chan, Y. T. (1995). *Wavelet basics*. Boston: Kluwer.
- Chen, G. (1999). *Controlling chaos and bifurcations in engineering systems* (1st ed.). Boca Raton, FL: CRC Press.
- Dawson, S., Grebogi, C., Sauer, T. D., & Yorke, J. A. (1994). Obstructions to shadowing when a Lyapunov exponent fluctuates about zero. *Physical Review Letters*, 73, 1927–1930.
- Dhamala, M., Lai, Y.-C., & Kostelich, E. J. (2000). Detecting unstable periodic orbits from transient chaotic time series. *Physical Review E*, 61, 6485–6489.
- Dhamala, M., Lai, Y.-C., & Kostelich, E. J. (2001). Analysis of transient chaotic time series. *Physical Review E*, 64, 056207(1–9).
- Ding, M., Grebogi, C., Ott, E., Sauer, T. D., & Yorke, J. A. (1993). Plateau onset for correlation dimension: When does it occur? *Physical Review Letters*, 70, 3872–3875.
- Eckmann, J. P., Kamphorst, S. O., Ruelle, D., & Ciliberto, S. (1986). Lyapunov exponents from time series. *Physical Review A*, 34, 4971–4979.
- Eckmann, J. P., & Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, 57, 617–656.
- Farmer, J. D., Ott, E., & Yorke, J. A. (1983). The dimension of chaotic attractors. *Physica D*, 7, 153–180.
- Frenking, M. E. (1994). *Digital signal processing in communication systems*. New York: Van Nostrand Reinhold.
- Grassberger, P., & Procaccia, I. (1983a). Characterization of strange attractors. *Physical Review Letters*, 50, 346–348.
- Grassberger, P., & Procaccia, I. (1983b). Measuring the strangeness of strange attractors. *Physica D*, 9, 189–208.
- Grebogi, C., Hammel, S. M., Yorke, J. A., & Sauer, T. D. (1990). Shadowing of physical trajectories in chaotic dynamics: Containment and refinement. *Physical Review Letters*, 65, 1527–1530.
- Grebogi, C., Ott, E., & Yorke, J. A. (1988). Unstable periodic orbits and the dimensions of multifractal chaotic attractors. *Physical Review A*, 37, 1711–1724.
- Hahn, S. L. (1996). *Hilbert transforms in signal processing*. Boston: Artech House.
- Hammel, S. M., Yorke, J. A., & Grebogi, C. (1987). Do numerical orbits of chaotic dynamical processes represent true orbits? *Journal of Complexity*, 3, 136–145.
- Hammel, S. M., Yorke, J. A., & Grebogi, C. (1988). Numerical orbits of chaotic processes represent true orbits. *Bulletin of the American Mathematical Society*, 19, 465–469.
- Hénon, M. (1976). A two-dimensional mapping with a strange attractor. *Communications in Mathematical Physics*, 50, 69–75.
- Huang, N. E., Shen, Z., Long, S. R., Wu, M. C., Shih, H. H., Zheng, Q., Yen, N.-C., Tung, C. C., & Liu, H. H. (1998). The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series. *Proceeding of the Royal Society of London Series A—Mathematical Physical and Engineering Sciences*, 454, 903–995.
- Janosi, I. M., & Tél, T. (1994). Time-series analysis of transient chaos. *Physical Review E*, 49, 2756–2763.
- Kantz, H., & Schreiber, T. (1997). *Nonlinear time series analysis* (1st ed.). Cambridge, UK: Cambridge University Press.
- Lai, Y.-C. (1997). Characterization of the natural measure by unstable periodic orbits in nonhyperbolic chaotic systems. *Physical Review E*, 56, 6531–6539.
- Lai, Y.-C., & Grebogi, C. (1999). Modeling of coupled chaotic oscillators. *Physical Review Letters*, 82, 4803–4806.
- Lai, Y.-C., Grebogi, C., & Kurths, J. (1999). Modeling of deterministic chaotic systems. *Physical Review E*, 59, 2907–2910.
- Lai, Y.-C., & Lerner, D. (1998). Effective scaling regime for computing the correlation dimension in chaotic time series analysis. *Physica D*, 115, 1–18.
- Lai, Y.-C., Lerner, D., & Hayden, R. (1996). An upper bound for the proper delay time in chaotic time series analysis. *Physics Letters A*, 218, 30–34.
- Lai, Y.-C., Lerner, D., Williams, K., & Grebogi, C. (1999). Unstable dimension variability in coupled chaotic oscillators. *Physical Review E*, 60, 5445–5454.
- Lai, Y.-C., Nagai, Y., & Grebogi, C. (1997). Characterization of the natural measure by unstable periodic orbits in chaotic attractors. *Physics Review Letters*, 79, 649–652.
- Lathrop, D. P., & Kostelich, E. J. (1989). Characterization of an experimental strange attractor by periodic orbits. *Physical Review A*, 40, 4028–4031.
- Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospherics and Science*, 20, 130–141.
- Okunev, Y. (1997). *Phase and phase-difference modulation in digital communications*. Boston: Artech House.
- Ott, E. (1993). *Chaos in dynamical systems* (1st ed.). Cambridge, UK: Cambridge University Press.
- Ott, E., Grebogi, C., & Yorke, J. A. (1990). Controlling chaos. *Physical Review Letters*, 64, 1196–1199.

- Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1986). *Numerical recipe* (1st ed.). Cambridge, UK: Cambridge University Press.
- Rössler, O. E. (1976). Equation for continuous chaos. *Physics Letters A*, 57, 397–398.
- Sano, M., & Sawada, Y. (1985). Measurement of the Lyapunov spectrum from a chaotic time series. *Physical Review Letters*, 55, 1082–1085.
- Sauer, T. D., Yorke, J. A., & Casdagli, M. (1991). Embedology. *Journal of Statistical Physics*, 65, 579–616.
- Shenoi, K. (1995). *Digital signal processing in telecommunications*. Englewood cliffs, NJ: Prentice Hall.
- Takens, F. (1981). Detecting strange attractors in fluid turbulence. In D. Rand & L. S. Young (Eds.), *Dynamical systems and turbulence, lecture notes in mathematics* (pp. 366–381). Berlin: Springer-Verlag.
- Theiler, J. (1986). Spurious dimension from correlation algorithms applied to limited time series data. *Physical Review A*, 34, 2427–2432.
- Wolf, A., Swift, J. B., Swinney, H. L., & Vastano, J. A. (1985). Determining Lyapunov exponents from a time series. *Physica D*, 16, 285–317.
- Yalcinkaya, T. (1998). *Phase characterization of chaos and controlling transient chaos in deterministic flows*. Unpublished PhD dissertation, University of Kansas, Kansas.

13

Distributed Data Mining

Byung-Hoon Park and Hillol Kargupta
University of Maryland

Introduction	342
Related Research	343
Data Distribution and Preprocessing	344
Homogeneous/Heterogeneous Data Scenarios	345
Data Preprocessing	345
Distributed Data Mining Algorithms	346
Distributed Classifier Learning	346
Collective Data Mining	349
Distributed Association Rule Mining	350
Distributed Clustering	351
Privacy Preserving Distributed Data Mining	352
Other DDM Algorithms	353
Distributed Data Mining Systems	353
Architectural Issues	354
Communication Models in DDM Systems	356
Components Maintenance	356
Future Directions	357
References	358

This work was supported by the National Aeronautics and Space Administration (NRA) NAS2-37143 and by the U.S. National Science Foundation CAREER award IIS-0093353.

INTRODUCTION

Advances in computing and communication over wired and wireless networks have resulted in many pervasive distributed computing environments. The Internet, intranets, local area networks, and wireless networks are some examples. Many of these environments have different distributed sources of voluminous data and multiple compute nodes. Analyzing and monitoring the distributed data require data mining—finding classifiers, associations, outliers, clusters, and other patterns from data by paying attention to computing, storage, communication, human-computer interaction, and other resources.

This chapter starts by pointing out a mismatch between the architecture of most off-the-shelf data mining systems and the needs of mining systems for distributed applications. It also claims that such mismatch may cause a fundamental bottleneck in many emerging distributed applications. Figure 13.1(a) presents a schematic diagram of the traditional data warehouse-based architecture for data mining. This model of data mining works by regularly uploading mission critical data in the warehouse for subsequent centralized data mining application. This centralized approach is inappropriate for most of the distributed and ubiquitous data mining applications. The long response time, lack of proper use of distributed resources, little attention to the cost of communication, and sometimes the need to preserve the privacy of the distributed data make the centralized data mining algorithms not suitable for distributed applications.

A scalable solution for distributed applications calls for distributed processing of data, controlled by the available resources and human factors. For example, consider an ad hoc wireless sensor network in which the different sensor nodes are monitoring some time-critical events. Central collection of data from every sensor node may create heavy traffic over the limited bandwidth wireless channels, and this may also drain a lot of power from the devices (data transmission usually requires a considerable amount of power). A distributed architecture for data mining is likely to reduce the communication load and also reduce the battery power more evenly across the different nodes in the sensor network. One can easily imagine similar needs for distributed computation of data mining primitives in ad hoc wireless networks of mobile devices such as PDAs, cell phones, and wearable computers. Potential applications include personalization, collaborative process monitoring, and intrusion detection over ad hoc wireless networks. We need data mining architectures that pay careful attention to the distributed resources of data, computing, and communication to consume them in a near optimal fashion. Distributed data mining (DDM) considers data mining in this broader context. As

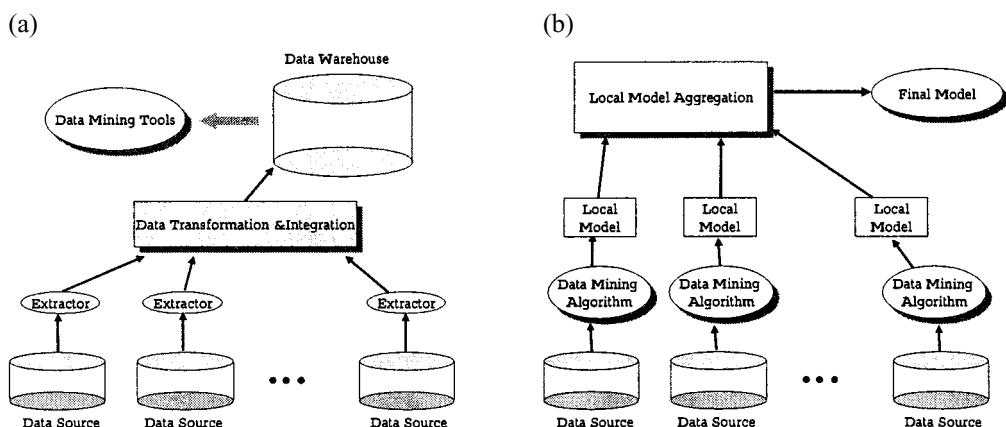


FIG. 13.1. Mining in a data warehouse-based architecture (a). The distributed data mining framework (b).

shown in Fig. 13.1(b), the objective of DDM is to perform the data mining operations based on the type and availability of the distributed resources. It may choose to download the data sets to a single site and perform the data mining operations at a central location. However, that decision in DDM should be based on the properties of the computing, storage, and communication capabilities. This contrasts with the traditional centralized data mining methodology in which collection of data at a single location prior to analysis is an invariant characteristic.

The wireless domain is not the only example. In fact, most of the applications that deal with time-critical distributed data are likely to benefit by paying careful attention to the distributed resources. The World Wide Web is a very good example. It contains distributed data and computing resources. An increasing number of databases (e.g., weather databases, oceanographic data at www.noaa.gov), and data streams (e.g., financial data at www.nasdaq.com emerging disease information at www.cdc.gov) are coming online; many of them change frequently. It is easy to think of many applications that require regular monitoring of these diverse and distributed sources of data. A distributed approach to analyze this data is likely to be more scalable and practical particularly when the application involves a large number of data sites. The distributed approach may also find applications in mining remote sensing and astronomy data. For example, the National Aeronautics and Space Administration Earth Observing System (EOS), a data collector for a number of satellites, holds 1,450 data sets that are stored, managed, and distributed by the different EOS Data and Information System (EOSDIS) sites that are geographically located all over the United States. A pair of Terra spacecraft and Landsat 7 alone produces about 350 GB of EOSDIS data per day. An online mining system for EOS data streams may not scale if we use a centralized data mining architecture. Mining the distributed EOS repositories and associating the information with other existing environmental databases may benefit from DDM. In astronomy the size of telescope image archives has already reached the terabyte range and continues to increase very fast as information is collected for new all-sky surveyors such as the GSC-II (McLean et al., 1998) and the Sloan Digital Survey (Szalay, 1998). DDM may offer a practical scalable solution for mining these large distributed astronomy data repositories.

DDM may also be useful in environments with multiple compute nodes connected over high-speed networks. Even if the data can be quickly centralized using the relatively fast network, proper balancing of computational load among a cluster of nodes may require a distributed approach. The privacy issue is playing an increasingly important role in the emerging data mining applications. If a consortium of different banks wants to collaborate for detecting frauds, then a centralized data mining system may require collection of all the data from every bank at a single location. However, this is not necessarily true if DDM is our choice of technology. DDM systems may be able to learn models from distributed data without exchanging the raw data. This may allow both detection of fraud and preserving the privacy of every bank's customer-transaction data.

This chapter presents a brief overview of the DDM algorithms, systems, applications, and the emerging research directions. The structure of the chapter is organized as follows. It first presents the related research of DDM and illustrates data distribution scenarios. Then DDM algorithms are reviewed. Subsequently, the architectural issues in DDM systems and future directions are discussed.

RELATED RESEARCH

DDM deals with data analysis algorithms, systems, and human-computer interaction issues in distributed environments. There are several other fields that deal with different subsets of these issues. This section discusses this connection between DDM and a few other related fields.

Many DDM systems adopt the multi-agent system (MAS) architecture. MAS has its root in distributed artificial intelligence (DAI), which investigates AI-based search, learning, planning, and other problem-solving techniques for distributed environments. Early research in this area includes blackboard systems (Nii, 1986); classifier systems (Holland, 1975); production systems (Newell & Simon, 1963); connectionism (Rumelhart & McClelland, 1986); Minsky's society-of-mind concept (Minsky, 1985); cooperative problem solving (Durfee, Lesser, & Corkill, 1989); actor framework (Agha, 1986); and the Contract Net protocol (Davies & Smith, 1983; Smith, 1980). The emergence of distributed environments such as the Internet and e-commerce have catalyzed many applications of DAI/MAS technology, and extensive literature on multiagent communication (Finin, Labrou, & Mayfield, 1997), negotiation (Rosenschein, 1994), search (Lander & Lesser, 1992), architectural issues (Woolridge & Jenneings, 1995), and learning (Sen, 1997) is now available. Although most of these topics are quite relevant to the DDM, DAI/MAS learning and architectural issues are probably the most relevant topics to DDM. The existing literature on multiagent learning does not typically address the issues involved with large-scale distributed data analysis. In DAI/MAS the focus is more on learning control knowledge (Byrne & Edwards, 1995; Carmel & Markovitch, 1995; Joshi, 1995; Sen & Sekaran, 1995), adaptive behavior (Mor, Goldman, & Rosenschein, 1995; Sandholm & Crites, 1995; Weiß, 1995), and other related issues. However, several efforts reported in the DAI/MAS literature do consider data intensive applications such as information discovery in the World Wide Web (Lesser et al., 1998; Menczer & Belew, 1998; Moukas, 1996).

The field of high performance parallel and distributed computing is also closely related to DDM in many ways. High-performance parallel computing environments often are used for fast mining of large data sets. There exists a large volume of parallel data mining (PDM) literature (Alsabti, Ranka, & Singh, 1997; Freitas & Lavington, 1998; Han, Karypis, & Kumar, 1997; Joshi, Han, Karypis, & Kumar, 2000; Kamath & Musick, 2000; Parthasarathy, Zaki, Ogiara, & Li, 2001; Zaki, 1996; Zaki, Parthasarathy, & Li, 1997). Most of the PDM techniques assume existence of high-speed network connection between the computing nodes. That is usually not available in many of the distributed and mobile data mining applications. Nevertheless, the development of DDM has been strongly influenced by PDM literature.

Data fusion refers to seamless integration of data from disparate sources. Among the extensive literature on data fusion, the distributed approach of multisensor data fusion is very relevant to DDM. Typically, the multisensor data fusion algorithms work based on the following general mechanism: Each sensor makes a local decision; all local decisions are then combined at a fusion center to produce a global decision. The objective of this approach is to determine the optimum local and global decision rules that maximize the probability of signal detection. This process shares striking procedural similarities with DDM. The Bayesian (Hoballah & Varshney, 1989) and the Neyman-Pearson (Viswanathan & Varshney, 1997) criteria, statistical hypothesis testing techniques, frequently are used for multisensor data fusion applications.

The following section starts the discussion on DDM by first pointing out different data distribution scenarios explored in the DDM literature.

DATA DISTRIBUTION AND PREPROCESSING

Identifying the distribution of data among different sites, collecting schema information, and understanding the metadata are necessary for developing a distributed data mining solution. Most of the DDM algorithms and systems are designed for the relational data model (tabular form); this chapter also explores this model.

Homogeneous/Heterogeneous Data Scenarios

In a relational database the schema provides the information regarding the relations stored. Information regarding different schemata from different tables is essential for posing the data mining problem. Most of the existing DDM work considers homogeneous schemata across different sites. Homogeneous schemata contain the same set of attributes across distributed data sites. Examples include local weather databases at different geographical locations and market-basket data collected at different locations of a grocery chain. There are also other DDM algorithms that consider heterogeneous schemata; this data model supports different data sites with different schemata. For example, a disease emergence detection problem may require collective information from a disease database, a demographic database, and biological surveillance databases. Most of the heterogeneous DDM literature considers cases in which every participating table shares a common key column that links the corresponding rows across the tables. The schema and the metadata information can be used for posing the data mining problem.

Preparing the data is an important step in data mining, and DDM is no exception. Data preprocessing in DDM must work in a distributed fashion. Many of the standard centralized data preprocessing techniques can be directly applied without downloading all the data sets to a single site. Some of these techniques are briefly discussed in the following section.

Data Preprocessing

Standardizing data sets across different sites is an important process in DDM. The first step is to exchange the database schema information and the metadata. Typically this involves low communication overhead. Additional information regarding the physical meaning of features, measurement units, and other domain specific information are often exchanged for better understanding of the distributed data sources.

If the data sites are heterogeneous, key-association is a necessary step. The primary purpose of the key-association step is to select a set of keys for associating the data across different sites. The schema information and the physical meaning of the features can be used for linking the distributed data sets. If a precise key is not available, we may need to use clustering or other related techniques to create approximate keys. Some features are often natural candidates for the key. For example, the coordinate location can serve as the key in a spatial database. In a temporal database the time stamp of the observation may serve the same purpose.

Data normalizations are often necessary for avoiding undesirable scaling effects. The need for data normalization depends on the application and the data analysis algorithm. For example, normalization may be critical in nearest neighbor classification for assigning uniform weight to all the features. Some of the popular normalization techniques are:

1. Decimal scaling: This technique works by moving the decimal point. It is also applicable to both homogeneous and heterogeneous DDM.
2. Standard deviation scaling: For any given feature value x this technique constructs the normalized feature x' where $x' = \frac{x - \mu_x}{\sigma_x}$; μ_x and σ_x are the mean and the standard deviation of x . In case of homogeneous DDM, the computation of the overall mean and standard deviation can be distributed among the different sites in a straightforward manner. In case of homogeneous DDM this computation is strictly local.

Missing data is a real problem in most applications. Most of the simple techniques like replacement by (1) class labels, (2) some constant value, or (3) expected value may or may

not work depending on the application domain. Usually they bias the data set and may result in poor data mining performance. However, if desired these techniques can be used directly in a DDM application. More involved techniques for handling missing data require predictive modeling of data. Typically, decision trees, Bayesian algorithms, and other inductive models are learned for predicting the missing values. The following section presents a discussion on different DDM algorithms.

DISTRIBUTED DATA MINING ALGORITHMS

Most DDM algorithms follow a general functional structure. They apply a local version of the algorithm at every participating data site and generate local models. Subsequently, all local models (sometimes along with a critical subset of the data) are sent to the global site and aggregated to produce the final global model. Local analysis captures local behavior using the local information and identifies unexplained aspects of the local behavior. The aggregation at a global site combines the local models to capture all the characteristics that can be explained using local observation and also develops models for the locally unexplained behavior using information from different sites. The later component sometimes involves transmission of some critical small subset of data to the central site, and one of the objectives of DDM algorithms is to minimize the volume of this transmitted data set. The following sections present a review of existing DDM algorithms.

Distributed Classifier Learning

Learning classifiers from distributed data sites poses an important class of problems. First we discuss the problem for homogeneous case. That will be followed the heterogeneous scenario.

Learning Classifiers From Homogeneous Sites. Most algorithms for distributed classifier learning from homogeneous data sites are related to ensemble learning techniques (Bauer & Kohavi, 1999; Dietterich, 2000; Merz & Pazzani, 1999; Opitz & Maclin, 1999). The ensemble approach has been applied in various domains to increase the classification accuracy of predictive models. It produces multiple models (base classifiers) and combines the outputs of the base modules to enhance accuracy.

The ensemble approach is directly applicable to the distributed scenario. Different models can be generated at different sites and ultimately aggregated using ensemble combining strategies. Fan, Stolfo, and Zhang (1999) discussed an AdaBoost-based ensemble approach from this perspective. Breiman (1999) considered arcing as a way to aggregate multiple blocks of data, especially in an online setting. An experimental investigation of stacking (Wolpert, 1992) for combining multiple models was reported elsewhere (Ting & Low, 1997).

The metalearning framework (Chan & Stolfo, 1993b, 1993a, 1998) offers another possible approach to learning classifiers from homogeneous distributed data. In this approach supervised learning techniques are first used to learn classifiers at local data sites; then metalevel classifiers are constructed by either learning from a data set generated using the locally learned concepts or combining local classifiers using ensemble techniques. The metalevel learning may be applied recursively, producing a hierarchy of metaclassifiers. Metalearning follows three main steps:

1. Generate base classifiers at each site using a classifier learning algorithm.
2. Collect the base classifiers at a central site. Produce metalevel data from a separate validation set and predictions generated by the base classifier on it.
3. Generate the final classifier (metaclassifier) from metalevel data.

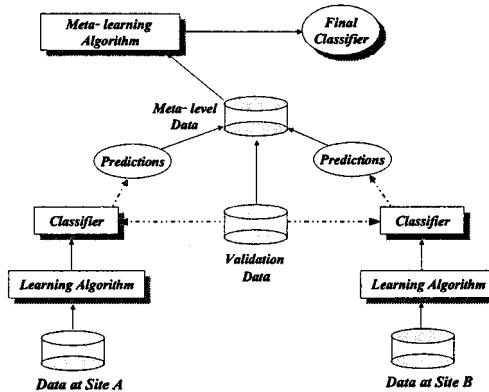


FIG. 13.2. Metalearning from distributed homogeneous data sites.

Learning at the metalevel can work in many different ways. For example, we may generate a new data set using the locally learned classifiers. We may also move some of the original training data from the local sites, blend it with the data artificially generated by the local classifiers, and then run any learning algorithm to learn the metalevel classifiers. We may also decide the output of the metaclassifier by counting votes cast by different base classifiers. Two common techniques for metalearning from the output of the base classifiers are briefly described in the following.

1. The arbiter scheme: This scheme makes use of a special classifier, called arbiter, for deciding the final class prediction for a given feature vector. The arbiter is learned using a classifier learning algorithm. Classification is performed based on the class predicted by the majority of the base classifiers and the arbiter. If there is a tie, the arbiter's prediction gets the precedence.
2. The combiner scheme: This combiner scheme offers an alternate way to perform metalearning. The combiner classifier is learned either from the correct classification and the base classifier outputs or from the data comprised of the feature vector of the training examples, the correct classifications, and the base classifier outputs.

Either technique can be used iteratively, resulting in a hierarchy of metaclassifiers. Figure 13.2 shows the overall architecture of the meta learning framework. A Java-based distributed system for metalearning is reported elsewhere (Lee, Stolfo, & Mok, 2000; Stolfo et al., 1997).

Metalearning illustrates two characteristics of DDM algorithms—parallelism and reduced communication. All base classifiers are generated in parallel and collected at the central location along with the validation set; the communication overhead is usually negligible compared to the transfer of entire raw data from every site.

Distributed learning with knowledge probing (DLKP) (Guo & Sutiwaraphun, 2000) is another metalearning based technique to produce a global model by aggregating local models. Knowledge probing initially was proposed to extract descriptive knowledge from a black box model, such as in a neural network. The key idea is to construct a descriptive model from data whose class values are assigned by a black box model. It works as follows:

1. Generate base classifiers at each site using off-the-shelf classifier learning algorithms.
2. Select a set of unlabeled data for the probing set.

3. Prepare the probing data set by combining predictions from all base classifiers.
4. Learn a final model directly from the probing set.

The probing data set (in Step 3) can be generated using various methods such as uniform voting, trained predictor, likelihood combination, and so forth. The main difference between metalearning and DLKP is the second learning phase. In metalearning a special type of classifiers (metaclassifiers) are trained to combine or arbitrate the outputs of the local models. The final classifier includes both metaclassifiers and local (base) models. In contrast, DLKP produces a final descriptive model that is learned from the probing data set as its final classifier.

Cho and Wüthrich (2002) proposed a distributed rule-based classifier that selects k appropriate rules out of $n(\gg k)$ rules learned from different sites. They empirically argued that properly chosen k rules are sufficient as a final classifier. In their approach, each rule is supposedly learned from an equal sized data called *fragment*. In case the data sizes are different among sites, the larger data should be further split so that each site has roughly one or more equal size blocks. To select the optimal set of k rules, they ordered the original n rules and picked the top k . For the ordering criteria they considered *confidence*, *support*, and *deviation*, where confidence refers to the accuracy of a rule, support denotes the significance of a rule with respect to the entire data, and deviation is the degree of misclassification.

Gorodetski, Skormin, Popyack, and Karsaev (2000) addressed distributed learning in data fusion systems. For *base classifiers* they developed a technique that learns a wide class of rules from arbitrary formulas of first-order logic. This is particularly applied as a visual technique to learn rules from databases. To overcome deficiencies of local learning (base classifiers), they adopted a randomized approach to select subsets of attributes and cases that are required to learn rules from distributed data, which results in a metalevel classifier. The following section explores the different algorithms for DDM from heterogeneous sites.

Learning Classifiers From Heterogeneous Sites. Homogeneous DDM algorithms usually do not work well for mining heterogeneous sites. In the heterogeneous case each local site observes only partial information. Therefore, a DDM algorithm must be able to learn a model using different features observed at different sites without downloading all the data to a single location. Ensemble-based approaches described in the previous section usually generate high variance local models and fail to detect interaction between features observed at different sites. This makes the problem fundamentally challenging.

In some applications, heterogeneous DDM may not require detecting interactions between features from different sites. In other words, the underlying problem may be site-wise decomposable. This scenario is relatively easy to handle. An ensemble-based approach to learn distributed classifiers is likely to work well for this case. Even if the application does not involve distributed data, vertical partitioning of data for decomposing the learning problem into smaller subproblems can speed up the process (Provost & Buchanan, 1995). However, assumption of site-wise decomposability is not necessarily correct in every application. In the general case heterogeneous DDM may require building classifiers using nonlinearly interacting features from different sites.

The WoRLD system (Aronis, Kulluri, Provost, & Buchanan, 1997) also works by making some assumptions about the class of DDM problems. It works by collecting first-order statistics from the data. It considers the problem of concept learning from heterogeneous sites by developing an “activation spreading” approach. This approach first computes the cardinal distribution of the feature values in the individual data sets. Next, this distribution information is propagated across different sites. Features with strong correlations to the concept space are

identified based on the first-order statistics of the cardinal distribution. Because the technique is based on the first-order statistical approximation of the underlying distribution, it may not be appropriate for data mining problems in which concept learning requires higher-order statistics.

There exist a few DDM algorithms that use an ensemble of classifiers for mining heterogeneous data sites. However, these techniques use special-purpose aggregation algorithms for handling some of the issues discussed earlier in this section. The aggregation technique proposed in Tumer and Ghosh (2000) uses an order statistics based approach for combining high-variance local models generated from heterogeneous sites. The technique works by ordering the predictions of different classifiers and using them in an appropriate manner. This work developed several methods, including selection of an appropriate order statistic as the classifier and taking a linear combination of some of the order statistics (“spread” and “trimmed mean” classifiers). It also analyzes the error of such a classifier in various situations. Although these techniques are more robust than other ensemble based models, they do not explicitly consider interactions across multiple sites.

Park and his colleagues (2002) developed a technique to learn decision trees from heterogeneous distributed sites. The approach can be classified as an ensemble based approach. However, they also proposed a Fourier spectrum based technique to aggregate the ensemble of decision trees. They noted that any pattern involving features from different sites cannot be captured by simple aggregation of local classifiers generated using only the local features. To detect such patterns, they first identified a subset of data that none of the local classifiers could classify with a high confidence. This identified data-subset was merged at the central site and another classifier (central classifier) was constructed from it. When a combination of local classifiers could not classify an unseen data with a high confidence, the central classifier was used instead. This approach exhibits better performance than a simple aggregation of local models. However, its performance is sensitive to the sample size (or confidence threshold). The following section discusses a framework for heterogeneous DDM that arguably can be distinguished from the ensemble based approaches.

Collective Data Mining

Kargupta and his colleagues considered the *collective* framework to address data analysis for heterogeneous environments and proposed the *collective data mining* (CDM; Kargupta, Park, Hershberger, & Johnson, 2000) framework for distributed data mining. CDM can be deployed for learning classifiers and predictive models from distributed data. Instead of combining incomplete local models, it seeks to find globally meaningful pieces of information from each local site. In other words, it obtains local building blocks that directly constitute the global model. Given a set of labeled training data, CDM learns a function that approximates it. The foundation of CDM is based on the observation that any function can be represented in a distributed fashion using an appropriate set of basis functions. When the basis functions are orthonormal, the local analysis produces correct and useful results that can be directly used as a component of the global model without any loss of accuracy. Because data modeling using canonical, nonorthogonal basis functions does not offer the problem decomposability needed in a DDM application, CDM does not directly learn data models in popular representations such as polynomial, logistic functions, decision tree, and feed-forward neural nets. Instead, it first learns the spectrum of these models in some appropriately chosen orthonormal basis space, guarantees the correctness of the generated model, and then converts the model in orthonormal representation to the desired forms. The main steps of CDM can be summarized as follows:

1. Generate approximate orthonormal basis coefficients at each local site.
2. Move an appropriately chosen sample of the data sets from each site to a single site and generate the approximate basis coefficients corresponding to nonlinear cross terms.
3. Combine the local models, transform the model into the user described canonical representation, and output the model.

Here nonlinear terms represent a set of coefficients (or patterns) that cannot be determined at a local site. In essence, the performance of a CDM model depends on the quality of estimated cross terms. Typically, CDM requires an exchange of a small sample that is often negligible compared with entire data.

The CDM approach has been explored using two important classes of problems—learning decision trees and multivariate regressors. Fourier and wavelet based representations of functions have been proposed elsewhere (Hershberger & Kargupta, 2001; Kargupta, Park, et al., 2000) for constructing decision trees and multivariate regressors, respectively. The Fourier spectrum based approach works by estimating the Fourier coefficients (FCs) from the data. It estimates the local FCs from the local data and FCs involving features from different data sites using a selected small subset of data collected at the central site. It has been shown elsewhere (Kargupta, Park, et al., 2000; Kargupta et al., 2002; Park, Ayyagari, & Kargupta, 2001) that one can easily compute the Fourier spectrum of a decision tree and vice versa. This observation can be exploited to construct decision trees from the estimated FCs. However, fast estimation of FCs from data is a nontrivial job. Estimation techniques usually work well when the data is uniformly distributed. This problem is addressed by the development of a resampling based technique (Ayyagari & Kargupta, 2002) for the estimation of the Fourier spectrum.

The collective multivariate regression (Hershberger & Kargupta, 2001) chooses wavelet basis to represent local data. For each feature in data, wavelet transformation is applied and significant coefficients are collected at the central site. Then the regression is performed directly on the wavelet coefficients. This approach has a significant advantage in communication reduction because a set of wavelet coefficients usually represents raw data in a highly compressed format.

The CDM framework also has been extended to the unsupervised DDM domain. A discussion on collective principal component analysis, collective clustering, and collective Bayesian learning from heterogeneous distributed data is presented later in this chapter.

Distributed Association Rule Mining

Two main approaches to distributed association rule mining are count distribution (CD) and data distribution (DD). CD especially considers the case when the data is partitioned homogeneously among several data sites. Each data site computes support counts for the same candidate item sets independently, which are then gathered at a central site to determine the large item sets for the next round. In contrast, DD focuses on maximizing parallelism; it distributes candidate item sets so that each site computes a disjoint subset. It requires the exchange of data partitions; therefore, it is viable only for machines with high-speed communications.

Agrawal and Shafer (1996) introduced a parallel version of Apriori. It requires $O(|C| \cdot n)$ communication overhead for each phase, where $|C|$ and n are the size of candidate item set C and the number of data sites, respectively. The fast distributed mining (FDM) algorithm (Cheung, Ng, Fu, & Fu, 1996) reduces the communication cost to $O(|C| \cdot n)$, where C_p is the potential candidate item set (or the union of all locally large item sets). The FDM notes that any globally large item set should be identified as locally large at one or more sites. However, this approach does not scale well in n , especially when the distributed data are skewed in

distribution. Schuster and Wolff (2001) proposed the distributed decision miner algorithm that reduces communication overhead to $O(Pr_{above} \cdot |C| \cdot n)$, where Pr_{above} is the probability that a candidate item set has support greater than the given threshold. DDM differs from FDM in that a locally large item set is not identified as a globally large item set until it is verified by exchange of messages.

Jensen and Sopakar (2000) proposed an association rule mining algorithm from heterogeneous relational tables. It particularly considers mining from star schema of n primary tables T_1, \dots, T_n (with one primary key) and one relationship table T_r . They assumed that T_r contains all foreign keys to each T_i , and exploit the foreign key relationships to develop a decentralized algorithm. Because each foreign key is a unique primary key in the corresponding table, explicit join operation can be avoided during the computation of the support of an item set. Another association rule mining technique from heterogeneous data also is reported (Vaidya & Clifton, 2002).

Distributed Clustering

Many clustering algorithms are designed for parallel and distributed applications. These clustering algorithms can be divided into general categories. The first approach approximates the underlying distance measure by aggregating local clusters, and the second provides the exact measure by data broadcasting. The approximation approach is sensitive to the quality of the approximation, and the exact approach imposes heavy communication load. Distributed clustering algorithms also can be categorized based on the type of data distribution that they can handle. First let us discuss the homogeneous case.

Forman and Zhang (2000) proposed a center-based distributed clustering algorithm that requires only the exchange of sufficient statistics, which is essentially an extension of their earlier parallel clustering work (Zhang, Hsu, & Forman, 2000). The recursive agglomeration of clustering hierarchies by encircling tactic (RACHET; Samatova, Ostrouchov, Geist, & Melechko, 2002) also is based on the exchange of sufficient statistics. It collects local dendograms, which are merged into a global dendrogram. Each local dendrogram contains descriptive statistics about the local cluster centroid that is sufficient for the global aggregation. However, both approaches need to iterate until the sufficient statistics converge or the desired quality is achieved.

Parthasarathy and Ogihara (2000) noted that finding a suitable distance metric is an important problem in clustering, including distributed clustering. They defined one such metric based on association rules. However, this approach is still restricted to homogeneous tables. In contrast, McClean and her colleagues (McClean, Scotney, & Greer, 2000) considered the clustering of heterogeneous distributed databases. They particularly focused on clustering heterogeneous data cubes comprised of attributes from different domains. They utilized Euclidean distance and Kullback-Leiber information divergence to measure differences between aggregates.

The PADMA system (Kargupta, Hamzaoglu, & Stafford, 1997; Kargupta, Hamzaoglu, Stafford, Hanagandi, & Buescher, 1996) is yet another distributed clustering based system for document analysis from homogeneous data sites; distributed clustering in PADMA is aided by relevance feedback based supervised learning techniques. Additional work on parallel and distributed clustering is reported elsewhere (Dhillon & Modha, 1999; Zhang et al., 2000).

Clustering heterogeneous distributed data sets constitutes an important class of problems. The CDM framework discussed in the previous section has been extended for handling representation construction and clustering from heterogeneous data.

Principal component analysis (PCA) is frequently used for constructing the representation in clustering. Clustering in DDM applications also can benefit from PCA. The collective principal component analysis (CPCA) algorithm (Kargupta, Huang, Krishnamrthy, Park, & Wang, 2000; Kargupta, Huang, S., & Johnson, 2001) offers one way to perform distributed PCA from heterogeneous sites. The main steps of the CPCA algorithm are given below:

1. Perform local PCA at each site; select dominant eigenvectors and project the data along them.
2. Send a sample of the projected data along with the eigenvectors.
3. Combine the projected data from all the sites.
4. Perform PCA on the global data set, identify the dominant eigenvectors, and transform them back to the original space.

To compute exact principal components (PCs), in principle, we need to reconstruct the original data from all projected local samples. However, because the PCA is invariant to linear transformation, the global PCs are computed directly from projected samples. The size of the samples is a lot smaller than that of the original data. In other words, we can exploit the dimensionality reduction already achieved at each of the local sites.

Kargupta et al. (2001) proposed a distributed clustering algorithm based on CPCA. This technique first applies the given off-the-shelf clustering algorithms to the local PCs. Then the global PCs are obtained from an appropriate data subset (projected) that is the union of all representative points from local clusters. Each site projects local data on the global PCs and again obtain new clusters, which are subsequently combined at the central site. A collective approach toward hierarchical clustering is proposed elsewhere (Johnson & Kargupta, 1999).

The CDM framework also is extended to other areas such as Bayesian network (BN) learning (Chen, Krishnamoorthy, & Kargupta, 2001, 2002). Within collective BN learning strategy, each site computes a BN and identifies the observations that are most likely to be evidence of coupling between local and nonlocal variables. These observations are used to compute a nonlocal BN consisting of links between variables across two or more sites. The final collective BN is obtained by combining the local models with the links discovered at the central site.

Recently an ensemble approach to combining multiple clusterings was proposed by Strehl and Ghosh (2002). Given r different clustering (possibly different numbers of clusters in each clustering), they proposed a framework that produces a combined clustering in a way to maximize the shared information between original clusterings. To quantify the shared information, they used mutual information that is borrowed from information theory. Mutual information essentially denotes how two clusterings are similar in terms of distributions of shared objects. In other words, it represents the proportion of shared objects in two clusters from different clusterings. Cluster ensembles can be applied to heterogeneous distributed data when each clustering is generated by observing different feature sets. Additional work on clustering can be found elsewhere (Sayal & Scheuermann, 2000).

Privacy Preserving Distributed Data Mining

Privacy is playing an increasingly important role in data mining. Collecting and exchanging potentially sensitive data is unlikely to be acceptable practice for many applications in future without some legal ramifications. Therefore, it is important that we explore ways to mine data from different sources without exposing the data itself to the data miner. DDM also

can enhance the privacy of the data possibly collected by different organizations because it minimizes communication of raw data. On the other hand, special privacy preserving techniques also can be used for making DDM algorithms more suitable for privacy preserving applications.

There exist two approaches to dealing with privacy preserving data mining. One aims to hide individual data instances by distorting their actual values. Value distortion is defined as $y = x + r$, where x denotes the original value and r is a random value drawn from some distribution, respectively. The key idea is that using the distribution of r , the original distribution of x can be approximated. This makes it possible for many data mining algorithms to work directly on the distorted data. Some examples include decision tree construction (Agrawal & Srikant, 2000) and association rule mining (Rizvi & Haritsa, 2002). Although it guarantees privacy, preserving data mining in the most strict sense, the data distortion approach is not well-suited for distributed data mining paradigm. It is rather beneficial to centralize data when the privacy is of importance.

The other approach, which is less strict in preserving privacy, is based on the assumption that a portion of the data is open to some parties, but no single party is allowed to access all the data. The focus of this approach is to hide any information derived from a given data set (a portion accessible to a party) from outside by using cryptographic tools. Following this, a privacy preserving data mining algorithm attempts to produce a final result without noting where each interim result comes from. In this sense the cryptographic approach is more suited for distributed data mining, because the data sites need not reveal their locally mined results. Kantarcioglu and Clifton (2002) investigated an association rule mining from homogeneous data using a commutative encryption tool.

Other DDM Algorithms

A distributed cooperative Bayesian learning algorithm was developed in Yamanishi (1997). This technique considers homogeneous data sets. In this approach different Bayesian agents estimate the parameters of the target distribution, and a population learner combines the outputs of those Bayesian models. A “fragmented approach” to mine classifiers from distributed data sources is suggested by Cho and Wüthrich (1998). In this method a single good rule is generated in each distributed data source. These rules are then ranked using some criterion, and a number of the top-ranked rules are selected to form the rule set. In Lam and Segre (1997) the authors reported a technique to automatically produce a Bayesian belief network from knowledge discovered using a distributed approach. Additional work on DDM design optimization (Turinsky & Grossman, 2000), classifier pruning (Prodromidis & Stolfo, 2000), measuring the quality of distributed data sources (Wüthrich, Cho, Pun, & Zhang, 2000), and problem decomposition and local model selection in DDM (Pokrajac, Fiez, Obradovic, Kwek, & Obradovic, 1999) also are reported. The following section discusses the systems issues in DDM.

DISTRIBUTED DATA MINING SYSTEMS

A DDM system manages a collection different components: mining algorithms, communication subsystem, resource management, task scheduling, user interfaces, and so forth. It should provide efficient access to both distributed data and computing resources, monitor the entire mining procedure, and present results to users (if needed) in appropriate formats. A successful DDM system is usually flexible enough to adapt to various situations. It should dynamically

identify the optimal mining strategy under the given resources and provide an easy way to update its components. In this section we discuss various aspects of DDM systems. In particular, architectural and communication issues are examined.

Architectural Issues

Many organizations have a cluster of high-performance workstations (e.g., SMPs) connected by a network link. Such a cluster can be a cost-effective resource for scalable data mining. However, Parthasarathy (2001) noted that performance in such an environment is largely affected by contention for processors, network link, and I/O resources. He emphasized that TCP/IP protocol is inherently designed to avoid contention, thus reducing communication rates drastically even with a small amount of resource competition. As an approach to deal with such a problem, he suggested guarding the allocation of resources and making applications adapt to resource constraints. The *three-tier client/server* architecture is one approach for efficient resource management. The Kensington system (Chatratchat et al., 1999) and Intelliminer (Parthasarathy & Subramonian, 2000) belong to this category. For example, the Kensington system is divided into client, application server, and third-tier server. The client module allows interactive creation of data mining tasks, visualization of models, and sampled data; the application server is responsible for user authentication, access control, task coordination, and data management. The third-tier server provides high-performance data mining services located in high-end computing facilities that include parallel systems. Particularly, it is placed in proximity to the databases to increase performance.

Figure 13.3 shows the Intelliminer developed by Parthasarathy and Subramonian (2000). Intelliminer is most specifically designed to support distributed *doall loop* primitive overclusters of SMP workstations. The doall loop is one in which each iteration is independent (Wolfe, 1995).

The agent-based model is another approach to address scalable mining over large distributed databases (see Fig. 13.4). Although there are many different types of software agents (Wooldridge & Jennings, 1995), they are typically distinguished from regular programs and considered to be somewhat autonomous and “intelligent.” All of the agent-based DDM systems employ one or more agents per data site. These agents are responsible for analyzing local data and *proactively* communicate with other agents during the mining stage. A globally coherent knowledge is synthesized via exchanges of locally mined knowledge. However, in a purely

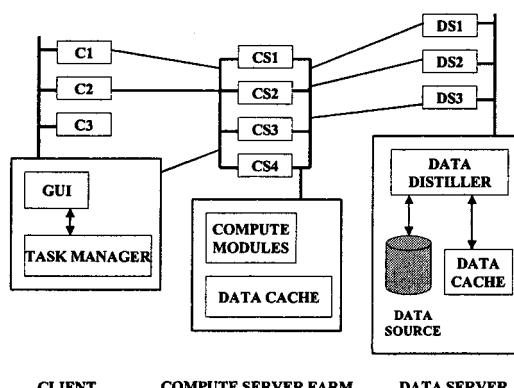


FIG. 13.3. Intelliminer DDM system. C1, C2 and C3 are clients. CS1, CS2, and CS3 are computing servers. DS1, DS2, and DS3 are data servers.

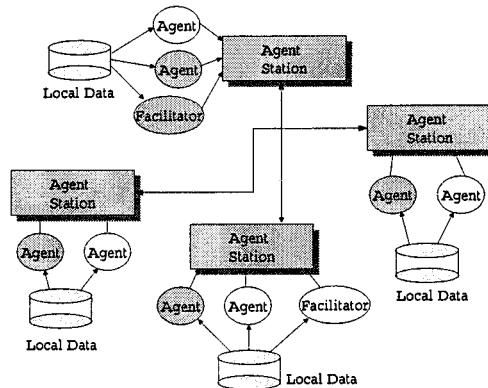


FIG. 13.4. BODHI: An agent-based DDM architecture.

autonomous agent-based model, an efficient total control over remote resources is inherently difficult. For this reason most agent based systems require a supervisory agent that facilitates the entire mining process. This agent, sometime called a *facilitator*, controls the behavior of each local agent. Java agents for metalearning (JAM) (Stolfo et al., 1997) and the BODHI system (Kargupta, Park, et al., 2000) follow this approach. In JAM, agents operating on a local database produce local classifiers. These local classifiers are then sent to a data site, where they are combined using metalearning agents. Both BODHI and JAM are implemented in Java, thus realizing a platform independent distributed data mining environment. By adopting loosely synchronized communication protocol among data sites, they seek to achieve asynchronous distributed data mining. BODHI also notes the importance of mobile agent technology. As all agents are extensions of a basic agent object, the BODHI system is easily capable of transferring an agent from one site to another, along with the agent's environment, configuration, current state, and learned knowledge.

The InfoSleuth project (Martin, Unruh, & Urban, 1999) at the Microelectronics and Computer Corporation offers an agent-based distributed infrastructure. It implements various types of agents that facilitate information gathering and analysis and event notification. The InfoSleuth is designed to provide an integrated solution to information related problems that are tailored for user-specified knowledge discovery. It especially supports continuous query subscriptions that instruct a targeted agent to continuously monitor and update its response to the query if any change is detected from an information source. The most notable aspect of the InfoSleuth system is its support of composite event detection. Events in the InfoSleuth define both the changes and observations about the data that are transferred from various heterogeneous data sources. Data-change and data-analysis events are combined to create higher-level events called composite events. InfoSleuth provides the composite event language that is adopted from active database rule language.

Distributed knowledge networks (DKN) (Honavar, Miller, & Wong, 1998) comprise another distributed data mining framework that is based on agent technology. DKN emphasizes the important role of mobile agents when the transfer of data from a private source becomes infeasible. DKN also proposes a detailed solution to extract and integrate data from heterogeneous sources that consist of different data types and semantic structures. To facilitate interoperability among heterogeneous databases, it adopts an object-oriented view that creates a uniform interface for multidatabases. An object-oriented view helps to hide the heterogeneity and distributed nature of multidatabases. Rooted in knowledge based agent software, DKN also implements an object-oriented data warehouse.

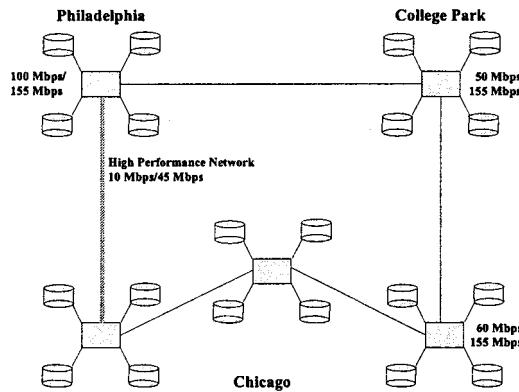


FIG. 13.5. Cluster of workstations in Papyrus.

Communication Models in DDM Systems

Architectural requirements for efficient data communication in a wide area network are explored in Grossman et al. (1998) and Turinsky and Grossman (2000). These articles note the trade-off between data transmission and local computation in DDM by defining an objective function that captures the associated costs. They formulate the overall cost function for data transfer strategies. A strategy denotes the amount of data transfer between each pair of compute nodes within the environment. The optimal strategy minimizes the cost function with respect to the given error level. The authors posed the problem as a convex linear programming problem, and the optimal strategy data and model partition strategy (OPTDMP) was proposed as a solution. However, their work made some simple assumptions about the cost and running time of the DDM algorithms. Nevertheless, this is an interesting approach that needs further attention.

Papyrus (Grossman, Bailey, Sivakumar, & Turinsky, 1999) is designed to find the optimal mining strategy over clusters of workstations connected by either a high-speed network (super-clusters) or a commodity network (metaclusters). One such environment is shown in Fig. 13.5. Papyrus supports three strategies: move results (MR), move models (MM), and move data (MD), as well as combinations of these strategies.

Costing infrastructure in the DDM architecture also is discussed by Krishnaswamy, Zaslavsky, and Loke (2000). The authors noted that DDM is evolving to embrace the e-commerce environment, especially the paradigm of application service providers (ASP). ASP technology allows small organizations or individuals to access a pool of commercial software on demand (Sarawagi & Nagaralu, 2000). The proposed architecture demonstrates how DDM can be integrated into ASP in an e-commerce environment. Krishnaswamy and colleagues (2000) emphasized that the primary issue under such highly interdomain operational environments is how to set up a standard by which to bill each user based on estimated costs and response times. The following section considers the maintenance of DDM systems.

Components Maintenance

Expandability of components is one key feature of successful DDM systems. There are far too many different approaches and algorithms for data mining to incorporate them all into a single system, and more are constantly being developed. Therefore, a DDM system must be able to incorporate new algorithms and methods as needed. For this purpose, the BODHI system

offers APIs support for creating custom-based agents. Users can easily design and insert their own distributed mining applications with the APIs. The Kensington system adopts *software component architecture*. Software components are software blocks that can be combined easily into a more complex ensemble. In particular, the application server consists of four Enterprise JavaBeans (EJB) classes. Each EJB class provides services to clients that can be accessed through remote method invocation (RMI) calls. High-performance software modules in the third-tier server are also components that are integrated via common object request broker architecture (CORBA), RMI, or java native interface (JNI).

The Parallel and Distributed Data Mining Application Suite (PaDDMAS) (Rana, Walker, Li, Lynden, & Ward, 2000) is another component based system for developing DDM applications. The overall architecture of PaDDMAS resembles the Kensington system in the sense that it identifies analysis algorithms as object components implemented as either Java or CORBA objects. It also provides a tool set that aids users in creating a DDM application by combining existing data mining components using a data flow approach. However, PaDDMAS takes one step further to allow easy insertion of custom based components. To ensure uniformity across components, each component in PaDDMAS has its interface specification written in XML. A user supplied component must also have its interfaces defined in XML. PaDDMAS allows a connection of two components only if their interfaces are compatible. A markup for data mining algorithms has emerged from the Predictive Model Markup Language (PMML) (Grossman, Bailey, Ramu, et al., 1999). PMML was designed to encode and exchange predictive data mining analysis components such as C4.5 (Quinlan, 1993). In that sense the markup used in PaDDMAS can be considered an attempt to embrace both analysis and data management components. Also, the emphasis of PaDDMAS is on encoding interfaces rather than encoding data structure of components.

The following section identifies the future directions in DDM research and concludes this chapter.

FUTURE DIRECTIONS

Current data mining products primarily are designed for off-line decision support applications. Real-time online monitoring and decision support are natural extensions with applications in many different domains. Next generation data mining products should support such applications.

This new breed of applications will require a data mining technology that pays careful attention to the distribution of computing, communication, and storage resources in the environment. DDM is a promising candidate for such applications. Several organizations are currently working toward DDM applications in different areas including financial data mining from mobile devices (Kargupta et al., 2002), sensor-network based distributed databases (Bonnet, Gehrke, & Seshadri, 2001), and care-health diagnostics analysis (Wirth, Borth, & Hipp, 2001).

However, DDM still has several major open issues that need to be addressed. First of all, many real-life applications deal with data distribution scenarios that are neither homogeneous nor heterogeneous in the traditional sense described in this chapter. We may have heterogeneous data sites that share more than one column. We may not have any well-defined key that links multiple rows across the sites. We need more algorithms for the heterogeneous scenarios. Also, distributed data preprocessing based on metadata needs further exploration.

DDM frequently requires exchange of data mining models among the participating sites. Therefore, seamless and transparent realization of DDM technology will require standardized schemes to represent and exchange models. PMML, the Cross-Industry Standard Process

Model for Data Mining (CRISP-DM), and other related efforts are likely to be very useful for the development of DDM.

Web search engines such as Yahoo! and Google are likely to start offering data mining services for analyzing the data they host (Sarawagi & Nagaraju, 2000). Combining the data mining models from such sites will be an interesting DDM application. The sites may have partially shared domains with no explicit keys linking the data; mining such semistructured data in a distributed fashion appears challenging.

Finally, human-computer interaction issues in DDM offers some unique challenges. It requires system-level support for group interaction, collaborative problem solving, development of alternate interfaces (particularly for mobile devices), and dealing with security issues.

REFERENCES

- Agha, G. (1986). *ACTORS: A model of concurrent computation in distributed systems*. Cambridge, MA: MIT Press.
- Agrawal, R., & Shafer, J. C. (1996). Parallel mining of association rules. *IEEE Transactions on Knowledge and Data Engineering*, 8, 962–969.
- Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (pp. 439–450). Dallas: ACM Press.
- Alsabti, K., Ranka, S., & Singh, V. (1997). A one-pass algorithm for accurately estimating quantiles for disk-resident data. In *Proceedings of the VLDB '97 Conference* (pp. 346–355). San Francisco: Morgan Kaufmann.
- Aronis, J., Kulluri, V., Provost, F., & Buchanan, B. (1997). The WoRLD: Knowledge discovery and multiple distributed databases. In *Proceedings of the Florida Artificial Intelligence Research Symposium (FLAIRS '97)*.
- Ayyagari, R., & Kargupta, H. (2002). A resampling technique for learning the Fourier spectrum of skewed data. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD '2002)*. Madison, WI.
- Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36, 105–139.
- Bonnet, P., Gehrke, J., & Seshadri, P. (2001). Towards sensor database systems. In *Proceedings of the Second International Conference on Mobile Data Management* (pp. 3–14). Hong Kong: Springer.
- Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36, 85–103.
- Byrne, C., & Edwards, P. (1995). Refinement in agent groups. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 22–39). New York: Springer-Verlag.
- Carmel, D., & Markovitch, S. (1995). Opponent modeling in multi-agent systems. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 40–52). New York: Springer-Verlag.
- Chan, P., & Stolfo, S. (1993a). Experiments on multistategy learning by meta-learning. In *Proceedings of the Second International Conference on Information Knowledge Management* (pp. 314–323).
- Chan, P., & Stolfo, S. (1993b). Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Workshop in Knowledge Discovery in Databases* (pp. 227–240). Menlo Park, CA: AAAI.
- Chan, P., & Stolfo, S. (1998). Toward scalable learning with non-uniform class and cost distribution: A case study in credit card fraud detection. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 164–168). Menlo Park, CA: AAAI Press.
- Chatratchat, J., Darlington, J., Guo, Y., Hedvall, S., Koler, M., & Syed, J. (1999). An architecture for distributed enterprise data mining. In *HPCN Europe* (pp. 573–582). Heidelberg, Germany: Springer-Verlag.
- Chen, R., Krishnamoorthy, S., & Kargupta, H. (2001). Distributed web mining using Bayesian networks from multiple data streams. In *IEEE International Conference on Data Mining* (pp. 281–288). Los Alamitos, CA: IEEE Computer Society Press.
- Chen, R., Krishnamoorthy, S., & Kargupta, H. (2002). Collective mining of Bayesian networks from distributed heterogeneous data. *Knowledge and Information Systems Journal*. In press.
- Cheung, D., Ng, V., Fu, Aa., & Fu, Y. (1996). Efficient mining of association rules in distributed databases. *IEEE Transaction on Knowledge and Data Engineering*, 8, 911–922.
- Chao, V., & Wüthrich, B. (2002). Distributed mining of classification rules. *Knowledge and Information Systems*, 4, 1–30.
- Cho, V., & Wüthrich, B. (1998, April). Toward real time discovery from distributed information sources. In *Second Pacific-Asia Conference, PAKDD '98* (pp. 376–377). Melbourne, Australia: Springer-Verlag.

- Davies, R., & Smith, R. (1983). Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20, 63–108.
- Dhillon, I., & Modha, D. (1999). A data-clustering algorithm on distributed memory multiprocessors. In *proceedings of the Workshop on Large-Scale Parallel KDD Systems* (pp. 47–56). New York: ACM.
- Dieterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomization. *Machine Learning*, 40, 139–158.
- Durfee, E., Lesser, V., & Corkill, D. (1989). Cooperative distributed problem solving. In A. Barr, P. Cohen, & E. Feigenbaum (Eds.), *The handbook of artificial intelligence* (Vol. 4). Reading, MA: Addison-Wesley.
- Fan, W., Stolfo, S., & Zhang, J. (1999). The application of Adaboost for distributed, scalable and on-line learning. In *Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 362–366). New York: ACM.
- Finin, T., Labrou, Y., & Mayfield, J. (1997). KQML as an agent communication language. In J. Bradshaw (Ed.), *Software agents*. Cambridge, MA: MIT Press.
- Forman, G., & Zhang, B. (2000). Distributed data clustering can be efficient and exact. In *SIGKDD Explorations* (Vol. 2).
- Freitas, A. A., & Lavington, S. H. (1998). *Mining very large databases with parallel processing*. Dordrecht, The Netherlands: Kluwer Academic Publishers.
- Gorodetski, V., Skormin, V., Popyack, L., & Karsaev, O. (2000). Distributed learning in a data fusion systems. In *Proceedings the Conference of the World Computer Congress (WCC-2000) Intelligent Information Processing (IIP 2000)*. Beijing, China.
- Grossman, R., Bailey, S., Kasif, S., Mon, D., Ramu, A., & Malhi, B. (1998). The preliminary design of Papyrus: A system for high performance, distributed data mining over clusters, meta-clusters and super-clusters. In *Fourth International Conference of Knowledge Discovery and Data Mining* (pp. 37–43). Menlo Park, CA: AAAI Press.
- Grossman, R., Bailey, S., Ramu, A., Malhi, B., Hallstrom, P., Pulley, I., & Qin, X. (1999a). The management and mining of multiple predictive models using the predictive modeling markup language. *Information and System Technology*, Vol. 41, 589–595.
- Grossman, R. L., Bailey, S. M., Sivakumar, H., & Turkinsky, A. L. (1999b). Papyrus: A system for data mining over local and wide-area clusters and super-clusters. In *Proceedings of Supercomputing IEEE*. Piscataway, NJ: IEEE.
- Guo, Y., & Sutiwaraphun, J. (2000). Distributed learning with knowledge probing: A new framework for distributed data mining. In H. Kargupta & P. Chan (Eds.), *Advances in distributed and parallel knowledge discovery* (pp. 115–132). Cambridge, MA: MIT Press.
- Han, E., Karypis, G., & Kumar, V. (1997). Scalable parallel data mining for association rules. In *Proceedings of SIGMOD'97* (pp. 277–288). New York: ACM.
- Hershberger, D., & Kargupta, H. (2001). Distributed multivariate regression using wavelet-based collective data mining. *Journal of Parallel Distributed Computing*, 61, 372–400.
- Hoballah, I., & Varshney, P. (1989). Distributed bayesian signal detection. *IEEE Transactions on Information Theory*, 35, 995–1000.
- Holland, J. H. (1975). *Adaptation in natural artificial systems*. Ann Arbor: University of Michigan Press.
- Honavar, V., Miller, L., & Wong, J. (1998). Distributed knowledge networks. In *IEEE Information Technology Conference* (pp. 87–90). Piscataway, NJ: IEEE.
- Jensen, V. C., & Soparker, N. (2000). Frequent itemset counting across multiple tables. In *Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 49–61). Springer.
- Johnson, E., & Kargupta, H. (1999). Collective, hierarchical clustering from distributed, heterogeneous data. In *Lecture notes in computer science* (Vol. 1759, pp. 221–244). New York: Springer.
- Joshi, A. (1995). To learn or not to learn. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 127–139). New York: Springer.
- Joshi, M., Han, E., Karypis, G., & Kumar, V. (2000). Parallel algorithms for data mining. In *CRPC parallel computing handbook*. San Francisco: Morgan Kaufmann.
- Kamath, K., & Musick, R. (2000). Scalable data mining through fine-grained parallelism: The present and the future. In H. Kargupta & P. Chan (Eds.), *Advances in distributed and parallel knowledge discovery* (pp. 29–77). Cambridge, MA: MIT Press.
- Kantarciooglu, M., & Clifton, C. (2002). Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *SIGMOD Workshop on DMKD*. Madison, WI.
- Kargupta, H., Hamzaoglu, I., & Stafford, B. (1997). Scalable, distributed data mining using an agent based architecture. In D. Heckerman, H. Mannila, D. Pregibon, & R. Uthurusamy (Eds.), *Proceedings of the 3rd International Conference on, Knowledge Discovery and Data Mining* (pp. 211–214). Menlo Park, CA: AAAI Press.
- Kargupta, H., Hamzaoglu, I., Stafford, B., Hanagandi, V., & Buescher, K. (1996). PADMA: Parallel data mining agent for scalable text classification. In *Proceedings of the Conference on High Performance Computing '97* (pp. 290–295). The Society for Computer Simulation International.

- Kargupta, H., Huang, W., Krishnamrthy, S., Park, B., & Wang, S. (2000). Collective principal component analysis from distributed, heterogeneous data. In *Proceedings of the Principals of Data Mining and Knowledge Discovery*.
- Kargupta, H., Huang, W., S., K., & Johnson, E. (2001). Distributed clustering using collective principal component analysis [special issue]. *Knowledge and Information Systems Journal*, 3, 422–448.
- Kargupta, H., Park, B., Hershberger, D., & Johnson, E. (2000). Collective data mining: A new perspective towards distributed data mining. In *Advances in Distributed and Parallel Knowledge Discovery* (pp. 133–184). Cambridge, MA: AAAI/MIT Press.
- Kargupta, H., Park, B., Pittie, S., Liu, L., Kushraj, D., & Sarkar, K. (2002). Mobimine: Monitoring the stock market from a PDA. *ACM SigKDD Explorations*, 3, 31–46.
- Krishnaswamy, S., Zaslavsky, A., & Loke, S. (2000). An architecture to support distributed data mining services in e-commerce environments. In *Second International Workshop on Advance Issues of e-commerce and Web-Based Information Systems (WECWIS 2000)*. Milpitas, CA.
- Lam, W., & Segre, A. M. (1997). Distributed data mining of probabilistic knowledge. In *Proceedings of the 17th International Conference on Distributed Computing Systems* (pp. 178–185). Los Alamitos, CA: IEEE Computer Society Press.
- Lander, S., & Lesser, V. (1992). Customizing distributed search among agents with heterogeneous knowledge. In *Proceedings of the First International Conference on Information and Knowledge Management* (pp. 335–344).
- Lee, W., Stolfo, S., & Mok, K. (2000). Adaptive intrusion detection: A data mining approach. *Artificial Intelligence Review*, 14, 533–567.
- Lesser, V., Horling, B., Klassner, F., Raja, A., Wagner, T., & Zhang, S. (1998). Big: A resource bound information gathering agent. In *Proceedings of the 15th National Conference on Artificial Intelligence, A-III'98*.
- Martin, G., Unruh, A., & Urban, S. (1999). *An agent infrastructure for knowledge discovery and even detection* (Tech. Rep. No. MCC-INSL-003-99). Microelectronics and Computer Technology Corp., Austin, TX.
- McClean, S., Scotney, B., & Greer, K. (2000). Clustering heterogeneous distributed databases. In *Workshop on Distributed and Parallel Knowledge Discovery*.
- McLean, B., Hawkins, C., Spagna, A., Lattanzi, M., Lasker, B., Jenkner, H., & White, R. (1998). New horizons from multi-wavelength sky surveys. *IAU Symposium*, 179.
- Menczer, F., & Belew, R. (1998). Adaptive information agents for distributed textual environments. In K. P. Sycara & M. Wooldridge (Eds.), *Proceedings of the Second International Conference on Autonomous Agents* (pp. 157–164). New York: ACM.
- Merz, C. J., & Pazzani, M. J. (1999). A principal components approach to combining regression estimates. *Machine Learning*, 36, 9–32.
- Minsky, M. (1985). *The society of mind* (1st ed.). New York: Simon and Schuster.
- Mor, Y., Goldman, C., & Rosenschein, J. (1995). Using reciprocity to adapt to others. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 164–176). New York: Springer-Verlag.
- Moukas, A. (1996). Amalthea: Information discovery and filtering using a multiagent evolving ecosystem. In *Proceedings of the Conference on Practical Applications of Intelligent Agents and Multi-Agent Technology* (pp. 421–436). London: Practical Application Company, Ltd.
- Newell, A., & Simon, H. (1963). GPS, a program that simulates human thought. In E. Feigenbaum & J. Feldman (Eds.), *Computers and Thought* (pp. 279–293). New York: McGraw-Hill.
- Nii, P. (1986). Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *AI Magazine*, 7(2), 38–53.
- Opitz, D., & Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11, 169–198.
- Park, B., Ayyagari, R., & Kargupta, H. (2001). A Fourier analysis-based approach to learn classifier from distributed heterogeneous data. In *Proceedings of the First SIAM International Conference on Data Mining*. Chicago:
- Park, B., Kargupta, H., Johnson, E., Sanseverino, E., Hershberger, D., & Silvestre, L. (2002). Distributed, collaborative data analysis from heterogeneous sites using a scalable evolutionary technique. *Applied Intelligence*, 16, 19–42.
- Parthasarathy, S. (2001). Towards network-aware data mining. In *Fourth International Workshop on Parallel and Distributed Data Mining*. San Francisco:
- Parthasarathy, S., & Ogihara, M. (2000). Clustering distributed homogeneous datasets. In *Principles and Practice of Knowledge Discovery (PKDD)* (pp. 566–574).
- Parthasarathy, S., & Subramonian, R. (2000). Facilitating data mining on a network of workstations. In K. Kargupta & P. Chan (Eds.), *Advances in distributed and parallel knowledge discovery* (pp. 233–258). Cambridge, MA: AAAI/MIT Press.
- Parthasarathy, S., Zaki, M., Ogihara, M., & Li, W. (2001). Parallel data mining for association rules on shared-memory systems. *Knowledge and Information Systems*, 3(1), 1–29.
- Pokrajac, D., Fiez, T., Obradovic, D., Kwek, S., & Obradovic, Z. (1999). Distribution comparison for site-specific regression modeling in agriculture. In *Proceedings of the International Joint Conference on Neural Networks*.

- Prodromidis, A., & Stolfo, S. (2000). Cost complexity-based pruning of ensemble classifiers. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD 2000* (pp. 30–40). Boston.
- Provost, F. J., & Buchanan, B. (1995). Inductive policy: The pragmatics of bias selection. *Machine Learning*, 20, 35–61.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Francisco: Morgan Kauffman.
- Rana, O., Walker, D., Li, M., Lynden, S., & Ward, M. (2000). PADDMAS: Parallel and distributed data mining application suite. In *14th International Parallel and Distributed Processing Symposium* (pp. 387–392). Cancun, Mexico.
- Rizvi, S., & Haritsa, J. (2002). Privacy-preserving association rule mining. In *Proceedings of 28th International Conference on Very Large DataBases, VLDB*. Hong Kong, China.
- Rosenschein, J. (1994). Designing conventions for automated negotiation. *AI Magazine*, 15, 29–46.
- Rumelhart, D., & McClelland, J. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition* (Vols. 1 & 2). Cambridge, MA: MIT Press.
- Samatova, N., Ostrochov, G., Geist, A., & Melechko, A. (2002). Rachet: An efficient cover-based merging of clustering hierarchies from distributed datasets. *An International Journal of Distributed and Parallel Databases*, 11, 157–180.
- Sandholm, T., & Crates, R. (1995). On multiagent q-learning in a semi-competitive domain. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 191–205). New York: Springer-Verlag.
- Sarawagi, S., & Nagaraju, S. (2000). Data mining models as services on the internet. *SIGKDD Explorations*, 2(1), 24–28.
- Sayal, M., & Scheuermann, P. (2000). A distributed clustering algorithm for web-based access patterns. In *Workshop on Distributed and Parallel Knowledge Discovery at KDD-2000* (pp. 41–48). Boston.
- Schuster, A., & Wolff, R. (2001). Communication efficient distributed mining of association rules. In *ACM SIGMOD International Conference on Management of Data* (pp. 473–484). New York: ACM Press.
- Sen, S. (1997). Developing an automated distributed meeting scheduler. *IEEE Expert*, 12(4), 41–45.
- Sen, S., & Sekaran, M. (1995). Multiagent coordination with learning classifier systems. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 218–233). New York: Springer-Verlag.
- Smith, R. (1980). The contract net protocol High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-12, 1104–1113.
- Stolfo, S., Prodromidis, A., Tselepis, S., Fan, D., & Chen, P. (1997). Jam: Java agents for meta-learning over distributed databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 74–81). Menlo Park, CA: AAAI Press.
- Strehl, A., & Ghosh, J. (2002). Cluster ensembles—a knowledge reuse framework for combining partitionings. In *Proceedings of AAAI 2002* (pp. 93–98). Edmonton, Canada: AAAI.
- Szalay, A. (1998). The evolving universe. *Astrophysics and Space Science Library* (231).
- Ting, K., & Low, B. (1997). Model combination in the multiple-data-base scenario. In *Ninth European Conference on Machine Learning* (pp. 250–265). New York: Springer.
- Tumer, K., & Ghosh, J. (2000). Robust order statistics based ensemble for distributed data mining. In H. Kargupta & P. Chan (Eds.), *Advances in distributed and parallel knowledge discovery* (pp. 185–210). Cambridge, MA: MIT Press.
- Turinsky, A. L., & Grossman, R. L. (2000). A framework for finding distributed data mining strategies that are intermediate between centralized strategies and in-place strategies. In *Workshop on Distributed and Parallel Knowledge Discovery*. Boston:
- Vaidya, J., & Clifton, C. (2002). Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Alberta, CA.
- Viswanathan, R., & Varshney, P. (1997). Distributed detection with multiple sensors. *Proceedings of IEEE*, 85, 54–63.
- Weiß, G. (1995). Adaption and learning in multi-agent systems: Some remarks and a bibliography. In G. Weiß & S. Sen (Eds.), *Adaption and learning in multi-agent systems* (pp. 1–21). New York: Springer-Verlag.
- Wirth, R., Borth, M., & Hipp, J. (2001). When distribution is part of the semantics: A new problem class for distributed knowledge discovery. In *Proceedings of the PKDD-2001 Workshop on Ubiquitous Data Mining for Mobile and Distributed Environments*. Freiburg, Germany.
- Wolfe, M. (1995). *High performance compilers for parallel computing*. Reading, MA: Addison Wesley.
- Wolpert, D. (1992). Stacked generalization. *Neural Networks*, 5, 241–259.
- Wooldridge, M., & Jennings, N. R. (1995). Intelligent agent: Theory and practice. *Knowledge Engineering Review*, 10, 115–152.
- Wooldridge, M., & Jenneings, N. (1995). *Intelligent agents: ECAI-94 workshop on agent theories, architectures, and languages*. New York: Springer-Verlag.
- Wüthrich, B., Cho, V., Pun, J., & Zhang, J. (2000). Data quality in distributed environments. In H. Kargupta & P. Chan (Eds.), *Advances in distributed and parallel knowledge discovery* (pp. 295–316). Cambridge, MA: MIT Press.

- Yamanishi, K. (1997). Distributed cooperative bayesian learning strategies. In *Proceedings of COLT'97* (pp. 250–262). New York: ACM.
- Zaki, M. J., Parthasarathy, S., & Li, W. (1997). A localized algorithm for parallel association mining. In *Ninth ACM Symposium on Parallel Algorithms and Architectures* (pp. 321–330).
- Zaki, M., Ogihara, M., Parthasarathy, S., & Li, W. (1996). Parallel data mining for association rules on shared memory multi-processors. In *Supercomputing '96*.
- Zhang, B., Hsu, M., & Forman, G. (2000). Accurate recasting of parameter estimation algorithms using sufficient statistics for efficient parallel speed-up: Demonstrated for center-based data clustering algorithms. In *Principles and Practice of Knowledge Discovery (PKDD)* (pp. 243–254). Heidelberg, Germany: Springer-Verlag.

II

Management of Data Mining

14

Data Collection, Preparation, Quality, and Visualization

Dorian Pyle
Data Miners

Introduction	366
How Data Relates to Data Mining	366
The “10 Commandments” of Data Mining	368
What You Need to Know about Algorithms Before Preparing Data	369
Why Data Needs to be Prepared Before Mining It	370
Data Collection	370
Choosing the Right Data	370
Assembling the Data Set	371
Assaying the Data Set	372
Assessing the Effect of Missing Values	373
Data Preparation	374
Why Data Needs Preparing: The Business Case	374
Missing Values	375
Representing Time: Absolute, Relative, and Cyclic	376
Outliers and Distribution Normalization	377
Ranges and Normalization	378
Numbers and Categories	379
Data Quality	380
What Is Quality?	382
Enforcing Quality: Advantages and Disadvantages	384
Data Quality and Model Quality	384
Data Visualization	384
Seeing Is Believing	385

Absolute Versus Relative Visualization	388
Visualizing Multiple Interactions	391
Summary	391

INTRODUCTION

Data preparation consumes 60 to 90% of the time needed to mine data—and contributes 75 to 90% to the mining project’s success. Poor or nonexistent data preparation can be 100% responsible for a project’s failure. Furthermore, preparing data properly is a skill. It can be learned intuitively—or it can be learned following a well-grounded set of principles. Although intuition alone may take years to develop, the principles can be learned in a few minutes. This chapter presents the basic principles.

Data preparation for data mining is the art of wringing the most value out of the available data, whereas data mining is the art of discovering meaningful patterns in data. But whether discovered patterns are meaningful depends on the problem to be solved—and what constitutes a problem and a solution to the problem, and thus what patterns might be meaningful, are issues wholly driven by external decisions.

For clarity, this chapter focuses on the most basic issues of data preparation, touching on mining as necessary but presenting it as much as possible as if it were separate from preparation. Many of the data preparation issues, nonetheless, are inextricably intertwined with mining issues, and none of these issues can be separated wholly from the needs that prompted the mining project in the first place. However, without data—collected, manipulated, monitored, and understood by the miner—there can be no data mining at all. Because the whole process is founded on data, it is crucial to approach it in a principled way, and understanding the principles involved dramatically raises the miner’s chance for successfully mining data.

Much of the material in this chapter is covered in much more detail in *Data Preparation for Data Mining* (Pyle, 1999).¹ Rather than making numerous references throughout this chapter to that source, for more details on most of the topics covered the reader is invited to use that as a resource.

How Data Relates to Data Mining

Essentially, data mining is no more than pattern hunting. All data mining algorithms assume that the data they will mine contains input states and output states. When the mining is “supervised” (U.S. terminology) or “directed” (European terminology), the data contains explicit examples of both input conditions and output conditions, and the data mining tool’s task requires finding some way of representing a connection between the two. When the mining is “unsupervised” or “undirected,” the output conditions are not explicitly represented in the data set: It is the mining tool’s task to discover inherent patterns in the data so that they can be compared with output conditions external to the data set to determine if the discovered patterns ultimately have any relationship to the desired outputs.

Mining tools all implement mining algorithms and the terms “tool” (in the sense of software suites) and “algorithm” are used fairly interchangeably in this chapter. Tools and algorithms are, of course, totally different entities; but for purposes of preparing data, because all tools

¹D. Pyle, *Data Preparation for Data Mining*. San Francisco, CA: Morgan Kaufmann, 1999.

apply one or more algorithms, preparation considerations need not distinguish between the two. However, the way an algorithm characterizes patterns in data is a crucial preparation concern. A principal objective of data preparation is to manipulate the data in such a way that any input and output patterns present in the data are made as obvious as possible. Thus, the different types of pattern detection employed by various algorithms call for different data preparation techniques. The appropriate preparation technique(s) in any specific case presents data patterns in ways appropriate to the method of pattern detection used by the mining tool.

Data for mining is always comprised of multiple variables. Usually the order of the records is irrelevant, and the patterns to be discovered exist within and between the values of the input and output states of each individual record. In supervised mining the patterns are the sets of values that some, or all, of the input variables assume when particular output values are present; in unsupervised mining the patterns are simply sets of values in some or all of the variables that occur more, or perhaps less, frequently than random variation in the values. In this sense, data mining attempts to match sets of input values to sets of output values. Different mining algorithms utilize different methods of set formation. Not surprisingly, each mining algorithm performs best when the data patterns are presented in a way that conforms to the data set selection method used by the algorithm.

Sometimes algorithms show marked differences in their respective data presentation requirements. Contrast neural networks with decision tree algorithms, for example. Neural networks require a data representation that is at least ordinal and numeric. Many, if not most, decision tree algorithms require categorical data. Neural networks are highly sensitive to missing values; trees and rule extractors generally are not. Neural networks create continuous estimates of the connecting relationship between input patterns and output patterns; trees and rules extractors create discontinuous categorizations of the connecting relationship. This is only a partial list of the differences between algorithms—it can easily become quite extensive.

However, sometimes the differences are more subtle. For example, neural networks inherently include interactions between the input variables, and the relationship between such interactions and any output patterns—decision trees do not. As an example of this phenomenon, create a data set with, say, five input variables of random numbers ranging in value between 0 and 1. Create a sixth variable that is the result of multiplying the other five input variables together, creating a pure multiplicative relationship, and a seventh variable that is the result of adding the five input variables together, creating a pure additive relationship. Create a decision tree and a network using the five input variables to predict each of the two output variables in turn. An example model using 1,000 records is instructive.² The tree, when learning the multiplicative relationship, produced a correlation coefficient between actual and predicted values of 0.76, and for the additive relationship a correlation coefficient between actual and predicted values of 0.87. The network learning the same relationships produced correlation coefficients of 1.00 and 1.00, respectively.

This example shows that for some algorithms it is important to represent explicitly particular relationships—interaction between variables in this case—and for other algorithms explicitly representing this particular data relationship is not important. There are several points to note here. One must be that extracting different types of relationships from data requires different algorithms—thus, *data drives tool selection*. The crucial point to note here from a data preparation perspective, however, is that different algorithms require different data representations and thus different types of preparation—thus, *tool selection drives data preparation*. A later section discusses in more detail the issues that affect tool selection and data preparation.

²The data set, example neural network, and decision tree for this example, together with an explanation, is available for download at www.modelandmine.com.

The “10 Commandments” of Data Mining

In fact, as the previous section begins to show, not only does tool selection affect data preparation and vice versa, so too is tool selection (and therefore, data preparation methods) highly affected by the problem domain. Colloquially expressed, what you want out of the process affects what you have to put into it.

So interrelated are all the parts of the data mining process that it is well worth examining the “10 Golden Rules” or “10 Commandments” of data mining to see how they help guide the miner through the data preparation process:

1. Select clearly defined problems that will yield tangible benefits.
2. Specify the required solution.
3. Define how the solution delivered is going to be used.
4. Understand as much as possible about the problem and the data set (the domain).
5. Let the problem drive the modeling (i.e., tool selection, data preparation).
6. Stipulate assumptions.
7. Refine the model iteratively.
8. Make the model as simple as possible—but no simpler.
9. Define instability in the model (critical areas where change in output is drastically different for small changes in inputs).
10. Define uncertainty in the model (critical areas and ranges in the data set where the model produces low confidence predictions/insights).

Rules 4, 5, 6, 7, and 10 have particular relevance for data preparation. Data can inform only when summarized to a specific purpose, and the purpose of data mining is to wring information out of data. Thus, to be useful, the relationships summarized by mining tools have to address some specific topic or problem. Understanding the nature of the topic (problem) and the data (rule 4) is crucial. Only by understanding the desired outcome can the data be prepared to best represent features of interest that bear on the topic (rule 5). It is also crucial to expose unspoken assumptions (rule 6) in case those, such as that described in the previous section, are not warranted. (An assumption based on the example in the previous section might be: “I assume that interactions are almost certainly not important in examining this data set and that a decision tree will work fine without explicitly including intervariable interactions in the model.”)

Modeling data is a process of incremental improvement (rule 7), or loosely: prepare the data, build a model, look at the results, reprepare the data in light of what the model shows, and so on. Sometimes, repreparation is used to eliminate problems, to explicate useful features, to reduce garbage, to simplify the model, or for other reasons. In fact, data mining often appears to be a process of working with and fixing up the data in light of what a model shows until time and resources run out. At the point that time and resources expire, the project is declared either a success or a failure, and it’s on to the next one!

Uncertainty in the model (rule 10) may reflect directly on the data. It may well indicate a problem in some part of the data domain—lack of data, the presence of noise or outliers, lack of stream synchronization, or some other data-related problem. It may or may not be possible to effect a cure for the data problem, but careful preparation can very often ameliorate the trouble.

All in all, it is no accident that 5 of the 10 golden rules of data mining bear directly on data preparation.

What You Need to Know about Algorithms Before Preparing Data

What you really need to know about algorithms is how the way they characterize relationships affects data preparation. A well-known aphorism has it that there are two kinds of people in the world—those who divide things into two categories, and those who don’t. In fact, dichotomous comparison is an easy way to explore similarities and differences, and at some danger of oversimplifying the issues, a few dichotomous descriptions will illustrate some important differences between data mining algorithms that are important to data preparation.

In this overview chapter there is space to mention only the major differences between algorithms that affect data preparation. It is very important for any miner to discover the specific data preparation needs of any data mining algorithm in addition to the “broad brush” descriptions discussed here. It is impossible to give a comprehensive list of algorithm-driven data preparation needs because mining tools are constantly changing, and most if not all tools modify the basic algorithms in ways that affect data preparation needs. This section contrasts the basic differences between decision tree and neural network data preparation only to make the discussion concrete. Also, they serve well as representative examples of algorithms requiring different types of data preparation.

A dichotomy already illustrated shows that some mining algorithms inherently include input variable interaction, and some do not. In practice, very often variable interactions are not important in building a good model, which is one reason why decision trees that do not incorporate input variable interaction do, in fact, produce excellent models in many applications. However, if there is reason to suppose that interactions may be important, and other reasons suggest the use of a mining tool that does not inherently include input variable interactions, they need to be explicitly included in the input data set as additionally prepared features.

Another significant dichotomy occurs between mining algorithms that are sensitive only to ordering of values in data, not to the magnitude of numeric values, and those that are sensitive to magnitude. Most decision tree algorithms consider one variable at a time when deciding how to split the data set—and the splitting point falls between two values in the data when the variable is numeric. As it turns out, the magnitude of the “gap” between values is of no concern to the algorithm, merely the fact that there are two discrete values. This insensitivity to the magnitude of difference in values produces a model that is sensitive only to the order of values of a variable. Neural networks, on the other hand, are affected, sometimes severely, by the magnitude of the difference in values. This difference immediately impacts data preparation. When the algorithm is sensitive to value magnitude, two preparation issues become important: (a) when the data contains univariate value clusters in a variable, it is often beneficial to normalize its distribution, or remap the values to be more evenly spread across the variable’s range; and (b) single values that are wildly different from the bulk of the values—“outliers”—can create tremendous distortion in the final model. Redistribution also alleviates genuine outlier problems. (A later section further discusses redistribution.)

The numeric/categoricalal dichotomy also enormously impacts data preparation. Some algorithms—for example, neural networks—require all values to be presented as numeric, preferably as continuous values. This requires optimal conversion of all nonnumeric values into a numerical representation. Other algorithms—for example, decision trees—require all values to be presented as categories, and thus any numeric variables have to be optimally converted into a categorical representation. Optimal value conversions are not always straightforward.

The last major dichotomy described here concerns how algorithms deal with missing values. Some, such as decision trees, are untroubled by missing values. They use the values that are

present and simply ignore the values that are not, thus making decisions on how to split the data using only values that are in the data. (The miner can usually decide whether “missing” should be considered a value or ignored.) Neural networks, in contrast, require that all inputs have a value, and one that is missing has to be interpreted somehow. In general, replacing all missing values in a variable with any single replacement value introduces distortion into the data, and using an optimal replacement strategy for missing values can have an enormous impact on model quality.

Why Data Needs to be Prepared Before Mining It

Data preparation is not an optional activity for a number of reasons. Those discussed so far include the need to represent the business problem as an object—in other words, to make the patterns in the data that relate to the object(s) of enquiry as clear as possible. However, the key reasons also include making it possible for a mining tool even to see some of the patterns at all. If, for instance, the feature “distance of customer to nearest retail store” is important, no modeling tool or mining algorithm can infer that the feature is important unless this feature is explicitly represented in the data. Such a feature may be implicitly represented if, say, the ZIP codes of the customer and store locations are present, or if latitude and longitude of both customer and store is present, but no mining tool available today will discover this feature unless the miner explicitly creates a feature to represent it. In other words, it is up to the miner to represent the real-world objects of interest in the data in a way that the mining tool can access. This is accomplished only by carefully preparing the data.

Furthermore, the requirements of the algorithm chosen to create the model drive data preparation, because all algorithms, and tools that implement them, have their idiosyncratic data preparation needs. Enhancing the patterns that do exist in the data set, or reducing noise in the data set, are equally important.

In simply assembling a data set for mining, making preparation decisions is unavoidable. The principles outlined in the following two sections are designed to help any miner produce a data set that addresses the business need, and that the mining tool of choice can characterize as easily as possible.

DATA COLLECTION

Data preparation starts with actually collecting data. It is frequently the case, and perhaps invariably, that a data miner has little or no influence on the way that data for mining is generated or captured. The best that a miner can do is to choose from data already captured. If it is possible to design the data capturing mechanism and to modify it in light of modeled results, that is, of course, ideal. However, this is so far from the normal reality of mining that this section assumes that data is already collected in some form. Starting with this assumption, this section looks at the options open to a miner in building the initial data set.

Choosing the Right Data

If there is a considerable amount of data available, choosing what to model becomes a matter of deciding on which variables and records to use. There are four fundamental principles involved in selecting the right data:

Principle. Select relevant variables.

From the selection of variables available as potential candidates in the input pool, select or construct those that common sense indicates should (or at least, could) have some relevance to the business problem. If starting with already collected data (such as in a warehouse), this might mean creating variables for mining. Select or create as many variables as possible so that each variable is as independent as possible from all variables selected so far. Stop only when no more variables can be discovered or devised that measure different relevant facets of the problem.

Principle. Choose redundant variables.

The relevant variables principle calls for variables to be as different from each other as possible. However, not all such variables will be equally stable in a data set. The selected variables may be affected differently by missing values, mismeasurement, recording errors, and other “noise.” This noise may be random in any variable or may be related in some way to the variable’s values. Adding redundant variables to a data set—that is, variables that to some extent duplicate the information carried by some other variable—attempts to compensate for noise. (Note that Golden Rule 8—making the model as simple as possible—will discard or merge many or all of the redundant variables during modeling; but that is during modeling, not during data assembly or preparation.)

Principle. Select records randomly from the source data.

The data set available from which to select a mining data set may have many kinds of problems, including all kinds of bias. This might or might not cause a problem during modeling. However, whatever bias a data set already contains, it is crucial not to inadvertently add any unknown bias to the mining data set. Random selection of records is the only way to ensure that no additional bias is added.

Principle. Ensure that the records represent the full range of within-variable and between-variable behaviors.

One potential source of noise and bias is insufficient data. If the number of records does not adequately represent some part, or all, of the data patterns or relationships well enough for a mining tool to characterize, then the mining tool’s characterization is not as good as it would have been with more data. Simple problem—simple solution: more data. (Determining how much data is needed is briefly addressed in a later section.) Of course, perhaps no amount of available data provides a perfect characterization of patterns and relationships; that is an issue to resolve during modeling. The point here is to ensure that the data set available for modeling represents the data relationships as well as allowed by the underlying data set from which the modeling data set is extracted.

Assembling the Data Set

It is still the case as of this writing that data for mining must be extracted from a data set and assembled into a flat file. This is most easily represented by the well-known spreadsheet format with rows (records) and columns (variables). Some tools recently introduced claim to perform in-place mining on existing data sets, but they appear mainly to make the extraction to a flat file invisible to the miner rather than to actually mine on normalized tables in a database. For practical purposes the row/column (record/variable) data set structure is usual.

Principle. Create at least three representative data sets.

Whether the data is extracted or mined in-place, and whatever the type of actual table layout used for mining, at least three separate data sets must be built: training, testing, and evaluation data sets. Underlying this process is the idea that a data set contains both real relationships and noise. The real relationship will be the same in both training and testing data sets, but the noise—at least inasmuch as it is nonsystematic or random noise—will be different in the two data sets. As a model improves its performance in both data sets, it is presumably learning real relationships, because the model is learning only in the training data and performing better in the test data that it has not been exposed to during training. When a model improves its performance in the training data but gets worse in the testing data, it is presumably learning noise present in the training data that is not in the testing data. (The author still creates and uses separate training and testing data sets even when the tool that wraps the algorithm itself internally, and invisibly to the miner, creates training and testing data sets as, for instance, several neural network tools do.)

However, neither the training nor testing data sets can provide an answer to the question of how well the model might perform on completely different data. This is because both training and testing data sets have in fact taken part in improving the model. The training data set provided examples from which the modeling tool learned the relationships and patterns, but that model was tuned to perform as well as possible on the testing data. Thus, any model should work well on both data sets. And because the model is specifically tuned to do as well as possible on these data sets, neither of them can be used to give any reliable indication of final performance on any other data set. The third, unseen, unused, and untouched data set—the evaluation data set—can give a better indication of expected behavior. Thus, each model needs at least three data sets for construction, and each one needs to be constructed in accord with all of the previous principles.

Assaying the Data Set

The term “assay” means to discover the fitness for purpose, or the worth of something. In this case the data is assayed to determine its fitness for creating the desired model and whether it is worth pursuing the project—that is, whether there seem to be reasonable grounds to think that the model produced with the data on hand will provide the needed functionality. The data adequacy model set (DAMS) is the tool used to perform part of the assay.

Principle. Create a DAMS.

This set of models is created in the training data set only, but is applied in training, testing, and evaluation data sets to determine data adequacy. The DAMS is created by dividing the data set into groups of 11 variables, selected randomly. (The precise number of variables used is not important, although 11 works well in practice.) In each group, one of the variables is selected at random as the output or predictor variable, and the remaining 10 are independent or input variables. Each group is used to create a predictive model, although, perhaps surprisingly, the predictive performance of each model is not important.

Principle. Ensure that DAMS models perform similarly in all three data sets.

The DAMS models created in the training data set only should perform approximately equally (well or badly) in the testing and evaluation data sets, and perhaps somewhat better in the training data set, but not remarkably so. If their performances are not similar, it is

highly probable that sufficient data is not available, or that the data sets are not all equally representative of the same patterns, or at least that some variables are not stable in the data sets—in which case, back to the drawing board!

There is insufficient space in this chapter to comprehensively cover the full use of DAMS in assaying data. However, DAMS can be used to assay data for several other characteristics, including to estimate how much more data might be needed (if that is indicated) and to assess the probable stability of the final model when applied to other execution data sets (if that is the purpose of the final model).

Assessing the Effect of Missing Values

Almost all real-world data sets, and particularly those drawn from social rather than physical data, contain records with variables that have no measured values, either numeric or categorical. These often are termed “missing” as far as mining data is concerned, and correspond, for instance, to the various kinds of “nulls” found in databases. Sometimes, depending on the needs of the algorithm and the business problem, missing values are relatively innocuous, at worst reducing the quality of the model in general but not actively damaging it. However, missing values are actively destructive remarkably often, even when the tools used are allegedly impervious to the effects of missing values.

The problem is that if missing values are not missing at random—and it is the experience of the author that values are almost never missing at random—then they are missing with some particular pattern. It is this pattern that can be dangerous if it is not detected. It may also be the case that these patterns of missing values are actually useful—but only if properly incorporated.

Principle. Assess the effect of presence/absence of values with a missing value check model (MVCm).

An MVCm is simple to construct. From the training (and, if necessary, the testing) data set, create a derived data set. In this derived data set replace all of the input variables’ values that are not missing with the value “1.” Replace all missing (or null) values with the value “0.” The variable to be predicted (the output variable) remains unaltered. This derived data set has no missing values: It is simply a flag data set with all binary variables. To construct the MVCm, create a model predicting the output variable. The author’s preferred method is to create a CHAID (chi-squared automatic interaction detection) tree and study the tree variable by variable. Remarkably often, this data set produces good predictions of output values. (CHAID tree tools are available from several vendors including Angoss and SPSS.)

Principle. Explain the effect of missing values.

Exploration of the MVCm can yield interesting insights into how data should be adjusted, rejected, and otherwise prepared for mining. Sometimes the missing/present flag variable can be usefully included in the to-be-mined data set as a useful feature, sometimes variables have to be rejected, and sometimes explaining the interaction presents an interesting challenge. Whatever is discovered, the MVCm often leads to useful insights.³

³D. Pyle, *Business Modeling and Data Mining*. San Francisco: Morgan Kaufmann, 2003.

DATA PREPARATION

Data preparation is the act of altering the contents of a data set by modifying the values of some variables and adding variables as necessary to present any patterns of interest in the data to a selected mining tool, so that the patterns are distinguished by that tool as easily as possible. As already discussed, exactly how to prepare any data set depends entirely on the exact nature of the problem, the sensitivities and capabilities of the mining algorithm chosen, and the nature of the source data from which the mining data is drawn.

Principle. Select data preparation techniques required by the business problem, the tool's (algorithm's) needs, and the state of the data set.

It is crucial to understand that there is no general purpose “best way” to prepare data; the method is entirely related to the demands of the problem, the tool, and the source data. It is without doubt a mistake to imagine that any particular transformation is always a beneficial transformation. Without reference to problem, tool, and source, there is simply no information with which to determine how to prepare data. However, there are general principles for data preparation just as there are for data set assembly. Nevertheless, all principles are relevant or applicable only when the demands of problem, tool, or source data indicate their appropriate use.

Why Data Needs Preparing: The Business Case

Although the technical need for data preparation may be clear to the miner, the business benefit is not always so clear. Because data preparation can be expected to take at least 60% of the time (and resources) of any mining project, a business case is always needed to support the time and resources required and to avoid the following scenario: Manager: “How’s that modeling project coming along? Almost three-quarters of the project has elapsed and we need an interim report about how the model is performing!” Miner: “I’m still preparing the data—haven’t started modeling yet. No results so far.”

Principle. Create and justify a business case for investing time and resources in data preparation.

Among other things, a business case helps to set expectations, especially for a large or long duration project. However, it is not just that a project plan for business purposes should include milestones that set appropriate expectations. It is equally important to note that appropriate data preparation alone—separate from tool or data choices—offers significantly improved models. The modeling improvement can be substantial.

Principle. Demonstrate and quantify the benefits of data preparation where possible.

Demonstrating the value of data preparation is not always easy. However, occasionally there are projects for which the data does not change, the modeling tool does not change, and the business objectives do not change. In these cases improved data preparation (aided, perhaps, by greater insight into the nature of the problem by the miner) is all that contributes to any performance improvement. Such improvements have amounted to substantial business benefits in several cases known to the author. A couple of examples can demonstrate the point. A credit card acquisition program with a response rate of 0.9% achieved an improvement in performance—to a response rate of 1.23%—using the same data set and modeling tool and improving only the quality of the data by repreparing it. This translates immediately into more

customers or lower acquisition cost. The acquisition cost, for instance, dropped from \$138 per acquired customer to \$101, an enormous performance improvement for the credit card issuer. In another case a commercial baker, using the same modeling source data set and the same modeling tools but with better data preparation techniques, improved model performance by about 8%, which alone directly contributed over \$1,000,000 per year in savings.

Improved data preparation practices can deliver high value directly to the bottom line, and where possible the business value of improvements possible from improved data preparation should be quantified. The business value—and thus the business case for investing in improved data preparation—can be very persuasive.

Principle. Survey data where possible to determine the potential for performance improvement of the model through improved data preparation.

It is possible to use information theory (or tools based on information theoretic principles) to survey data and measure exactly how much information a specific data set contains in the input variables about the output variables. It must be said that such a survey measures only the features in the data that are extant, not the implicit features (such as “distance of customer from nearest retail store,” mentioned earlier). Thus, information theory based data surveying indicates “at least” how much information the data set contains in its input variables about the output states. Extracting implicit features and making them explicit may result in an increase in the amount of information available due to additional understanding on the part of the miner, which, as might be expected, actually adds information content to a data set. Nonetheless, it is possible to estimate how well the best possible model could perform in any data set as it is prepared at the time of measurement. A problem, unfortunately, is that information theory gives no direction as to how to create the optimal model.

Nonetheless, all is not lost, because measuring how well any specific model has performed and comparing it against the theoretical best for extant features in a data set indicates potential improvement that might be possible through improved data preparation techniques. (Also, the selection of different modeling tools may contribute to improved performance, although appropriate selection of tools and algorithms is not the subject of this chapter.) If the model has captured only a modest fraction of the information content of a data set, a significant improvement may be possible, for instance. A survey can help actually quantify potential improvement to determine economic impact, and whether it is worth investing resources to attempt improvement. With such measures in hand, it is far easier to present a justifiable business case.

Although there is no space here to discuss the techniques of information-based data surveying, it is always worthwhile to include a strong business case for data preparation as part of the overall justification for a data mining project.

Missing Values

An earlier section of this chapter discussed the MVCM and its role in determining the impact of the pattern of missing values. However, there still remains the problem of how to find some substitute value for missing values for those algorithms that require one. Very often, a tool that incorporates an algorithm that requires a numerical representation of all values has some default method of providing a substitute. Surprisingly, such methods often do more harm than good. If possible, and where necessary, it is far better for the miner to make principled substitutions rather than simply allow the tool to use its default method.

Principle. Make principled replacements for missing values wherever possible.

Replacing missing values presents a potentially serious problem: how to do so without adding, or removing, any information from the data set when the substitute values are included. One common replacement method calls for choosing the variable's mean value as a replacement. However, any substituted value specifies, as far as the mining algorithm is concerned, that it actually occurred in concert with all of the other variable values, quite regardless of their actual values. This clearly is not the case. In fact, it might be expected that in the normal course of events the actual value in any variable has a relationship to the values of the other variables. (This had better be the case, because if the values are totally independent of each other—not even linked through the value of the output variable(s)—then the data set has no meaningful relationships in the data to find!) Single value substitutions, therefore, add erroneous information to a data set and may hide existing information. This is not conducive to mining the best model!

Principle. Replace missing values on a case-by-case (record-by-record) basis with values that preserve the within-variable and between-variable variances.

Today, some tools replace missing values with multivariate estimates from the variable values that are present in a record. The author personally prefers a method that chooses replacement values that least disturb the entropy (a measurement of information content) of a data set. However, any method, including the miner creating separate models specifically to replace missing values, is preferable to simply plugging in the mean, median, mode, or any other single value replacement. (One way to achieve this is through the use of an autoassociative neural network trained on a subset of the data with no missing variables.) If possible, it is always worth discovering exactly which missing value replacement algorithm, if any, is incorporated into those tools that need one.

Representing Time: Absolute, Relative, and Cyclic

Generally speaking, time appears in data sets for mining as variables in a record representing a specific date-time, or the status of a measurement at a specific time in the past, or the status of a measurement at a time some fixed period earlier than some other time (the “other time” usually being the “now” for the data set). Thus, data mining is not often concerned with mining time series data in which the order of the records represents the changing state of variables with the passage of time. The remarks in this section are aimed at representing time within single records so that the order of the records in the data set for mining remains irrelevant.

Principle. Convert absolute date/time information to relative temporal measurements to events or objects of business interest.

Time measurement for representation in data mining has characteristics similar to spatial representation. Absolute measurements describe fixed locations such as latitude and longitude (spatially) or specific year, month, hour, minute, and second for time. However, just as spatial relative measures are very often more useful than absolute ones (“distance of customer from local store,” again) so too are relative measures of time. For instance, “purchases in last day/week/month/quarter/year,” or “days since last purchase” are temporally relative measures.

Actually, this is a specific instance of a more general principle that is not covered here, which calls for creating relative representations of objects of business interest where appropriate and possible. For instance, how “this car” performs relative to other similar cars and/or all

cars. Or, how “this department” performs or compares, or “this insurance policy,” “this investment opportunity,” “this television viewer,” or “this consumer” relates relative to the socio- or demographic group, peer group, ZIP code, and so on. However, it is particularly important for temporal data to make a relative representation.

Principle. Represent time as a cycle of length appropriate to the objects to be represented.

One facet of representing time differs from most other representations, in that many events are cyclic. For any modeling tool to discover cyclicity, this characteristic must be made explicit in the data set.

Cyclicity can sometimes be captured with relative representation when the object of interest is known. For instance, days to next (or days from last) “xxx” (where “xxx” is, say, Christmas, Thanksgiving, August Bank Holiday, Tiradentes Day, Bastille Day, or whatever the event). However, to discover cyclicity for relevant objects that are not known to be cyclic, or perhaps are not already even known to exist, the data needs to include a representation of the temporal cycle. (Discovering these unknown cyclic relationships may be one of the results of mining.) Cycles, at least in social data such as customer relationship management or corporate data, are usually annual, although not necessarily so, but an annual cycle is assumed here to make the discussion concrete.

Cycles go round and round continuously and without sudden breaks or interruptions. A single variable can emulate this, but only incompletely. For instance, taking midsummer as “1” and midwinter as “0,” a variable could be created that continuously varies between “0” and “1.” There is no discontinuity here such as would be found in a relative variable, say “days to midsummer.” “Days to midsummer” would step from “0” to “365” on the day following midsummer’s day. However, the single-variable cyclic representation cannot distinguish between midspring (0.5, but getting larger as it increases toward 1) and midautumn (0.5, but getting smaller as it decreases toward 0). This problem holds for all values of the cyclic variable except “1” and “0.”

The answer, of course, is to use a second variable that indicates, say, direction of seasonal motion. The second variable is needed because cyclic behavior is represented as a circle, and a circle can be embedded only in two-dimensions. Representing two dimensions requires two variables.

Outliers and Distribution Normalization

In any numeric variable it is almost unknown for the values to be uniformly distributed throughout their range. Thus, some values occur more frequently than others, and when this is the case, the more frequently occurring values are said to be clustered. A normally distributed variable is clustered about its mean, and the nature of this clustering has specific well-known characteristics. Seldom are variables precisely normally distributed, and often variables have multiple clusters of values or are multimodal in statistical terms.

Any numerically sensitive data mining algorithm has some degree of sensitivity in its discrimination of values. Clearly, where the values cluster, more sensitivity is required than in the relatively less populated “spaces” between clusters. Many data mining and statistical techniques use mechanisms, sometimes highly sophisticated mechanisms, such as “kernel regression” and “adaptive polynomial kernel estimation,” to determine how best to adapt to the clusters. However, these methods of adapting the underlying algorithm to the presence of clusters are required only because there are clusters. No clusters—no adaptation needed! Life, of course, is not so easy.

Mining tools characterize and represent multivariate clusters. But univariate clusters are easy to transform through various distribution normalization techniques, and using them can lead to significant model performance improvement.

Principle. For numerically sensitive mining algorithms, always normalize the individual variable's distributions insofar as that is possible.

There are several ways to redistribute a variable's values. The most effective is without doubt continuous redistribution. The tool to do this generates a continuous function, such that the input values are translated into substitute values that result in, as nearly as can be managed, a rectangular distribution (all values equally likely.)

Another way to achieve a similar result is through equal frequency binning (EFB) with a high bin count. Although EFB results in some loss of information, it has the virtue of easy availability to a miner. The bin count should be at least 100, and 1,000 may not be too many depending on the data. EFB attempts to order the variable values and place an equal number in each bin. One binning method uses the bin mean as a substitute value for the range of values that are in the bin, but a better practice is to use uniform increments (EFBui) across the selected range. Using the EFBui method with a high bin count results in a discrete estimation of the continuous redistribution function.

Distribution normalization also handily deals with the problems presented by valid outliers. Outliers are values that are very different from the bulk of values of a variable. They lie at the extremes of the range and, by virtue of being different, may represent errors in measurement. If errors, they are invalid outliers because the real value, if it had been measured, would not have been an outlier. If the appropriate value can be discovered, that should be used instead. A cluster of outliers may be caused by a systematic bias for those measurements, which can (if identified and characterized) be corrected. However, if the true value is undiscoverable, it is probably better to treat an invalid outlier as a missing value.

An outlier is valid if it represents an accurate measurement and still falls well outside the range of the majority of values. These should not be discarded. (The data revealing the presence of the ozone hole was present for many years before it was recognized, the relevant data having been discarded as outliers.) These outliers represent genuine system behavior, but they can cause a problem.

Valid outliers are, by definition, few. Nonetheless, for many mining and statistical tools, because the effect of any value depends on the value's distance from the variable's mean, they make a very large difference to the structure of the model—a difference out of all proportion to their number. There are techniques, some of which are very sophisticated, for dealing with the outlier problem. But distribution normalization through whatever technique greatly alleviates the problem before it begins.

Distribution normalization is a simple principle but can have a significant impact on improving model performance for those techniques sensitive to differences in the magnitude of a variable's values.

Ranges and Normalization

Some algorithms require numerical inputs within specific ranges, normally 0 to 1 or -1 to $+1$. However, when this is needed, every commercial tool that implements such an algorithm performs the necessary range normalization. Unless the miner is working specifically with the raw algorithm, it is seldom necessary to normalize ranges of numeric variables before presenting them to a mining tool.

Numbers and Categories

Problems with category and number representations cut both ways because there are few raw algorithms that can handle both types of data. Numerically oriented algorithms require the conversion of categories to numbers, and categorically oriented algorithms require the conversion of numbers to categories. Whichever is made, careless transformation will lose, damage, or distort the very relationships that have to be mined.

Principle. When translating a variable's values from categorical to numerical representation with a numeric output variable: For each category, use the mean numeric value of the predictor when the category is present as the category value.

As a specific example of this principle, for a variable "gender" with at least four categories (male, female, unknown, missing), if the average value of the predictor is, say, 0.2 for the category "male" in the training data set, use 0.2 as the translated value.

Principle. When translating a variable's values from categorical to numerical representation with a categorical output variable: Use the category's reflected value from the data set.

Unfortunately, describing the algorithms for optimally finding numerical representations of input and (if necessary) output variables is beyond the scope of this chapter. Algorithms do exist for making optimal translation (reflection) of such values, and these can yield enormous model quality improvement over arbitrary assignment of numerical values to categories. Arbitrary assignment of numbers to categories makes discovering a relationship very difficult at best; at worst, not only is it impossible to discover the relationship, but discovering even those relationships not directly involving the categorical variable can be made far more difficult. Any miner needing to make these substitutions will be well advised to use the necessary substitution principles discussed elsewhere.

Placing numbers into categories seems, at first blush, a much easier proposition. Techniques such as EFBui often do produce very useable results when the bins are used as separate categories rather than as a numeric value translation mechanism. However, EFBui and many other forms of binning make use only of the information content of the variable itself (and usually only a partial use at that). These techniques provide an unsupervised (and usually arbitrary) binning. When use of a supervised binning approach is possible, such methods can produce far superior results because they make use of the information content in and between input and output variable. (For "superior" read: bin structures that produce much better models.)

Principle. When translating a variable's values from numerical to categorical representation with a numeric output variable in the data set, use supervised binning techniques.

Although this chapter cannot fully describe optimal binning techniques, it is well worth hunting for such tools and techniques. As a small example, one of the problems with many binning techniques is that they cannot create separate contiguous bins with the same class. For instance, Fig. 14.1 shows the scatterplot for a three-variable, two-class data set.⁴

Data in this data set are generated by the following rule:

- If variable 2 (V2) is less than 0.3, then class triangle.
- If V2 is greater than 0.6, then class cross.

⁴The explanation, data, and models for this demonstration are also at www.modelandmine.com. This example is after an example described in I. H. Witten & E. Frank (2000). *Data Mining*, p. 24. San Francisco: Morgan Kaufmann.

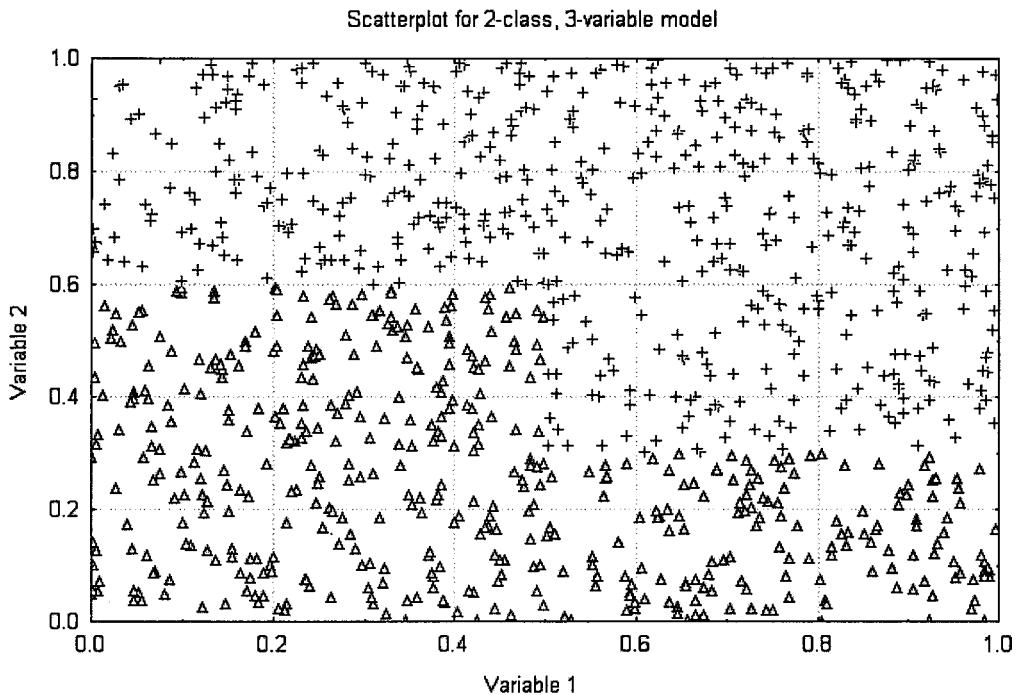


FIG. 14.1. Two-class, three-variable scatterplot.

- Otherwise, if variable 1 is greater than 0.5, then class cross.
- Otherwise, class triangle.

It is clear from looking at the figure that V2 requires three bins. Bin $V2 > 0.6$ needs to be assigned class cross. Bin $V2 < 0.3$ needs to be assigned class triangle. However, the remaining bin also must be assigned one of these two classes, and any binning mechanism that is unable to assign contiguous bins with the same class at the very least is compelled to create a suboptimal binning.

In this example data set a model created with a commercial data mining tree-based tool, and using its built-in binning mechanisms, created a model with a misclassification rate of just over 1.5%. When the same tool was given the same data optimally prebinned, the misclassification rate was 0.0% (a perfect model.) Although an optimal binning mechanism improves model classification performance in this example, a glance at Figs. 14.2 and 14.3 shows that optimal binning also contributes to tree simplicity, which enhances the ability to interpret the tree into a simple explanation.

Although this example is on a small data set created especially to demonstrate the problem clearly, optimal binning often produces significant improvements in real-world data sets, too.

DATA QUALITY

As can be deduced from the preceding sections of this chapter, data quality is of crucial importance. However, in common parlance, unqualified use of the term “quality” is taken to mean “high quality.” In this section there is no implied qualifier for the term “quality.” Why no

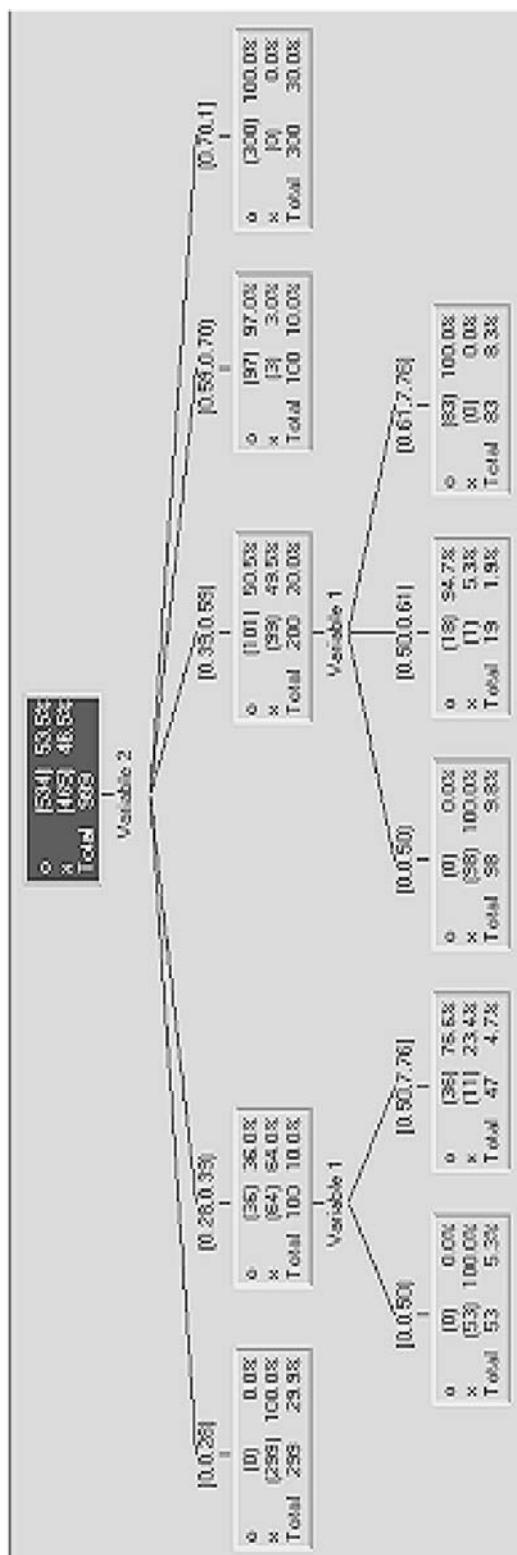


FIG. 14.2. Decision tree built on two-class, three-variable data set using default (less than optimal) binning.

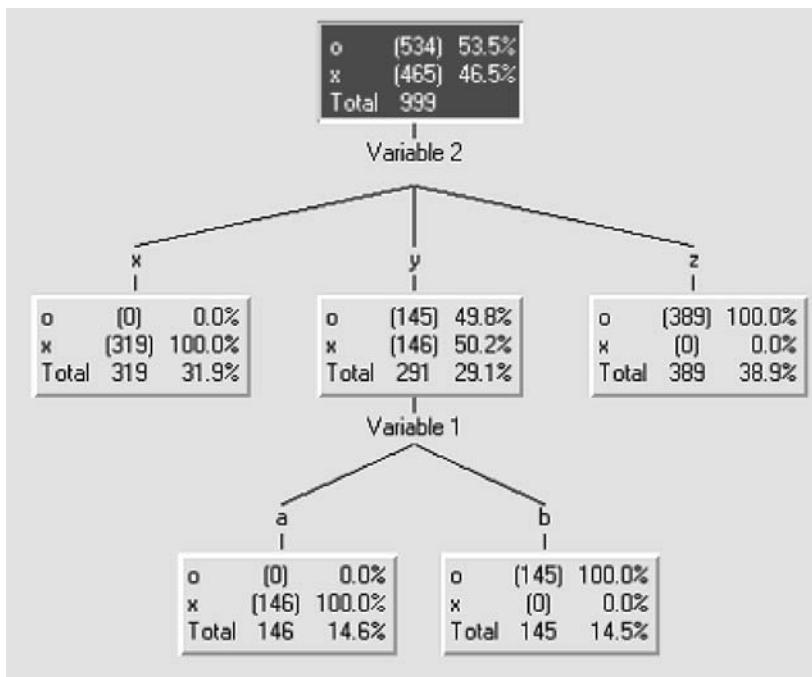


FIG. 14.3. Decision tree built on two-class, three-variable data set using externally created optimal binning.

qualifier? The most important issue for a miner when a model is to be applied to data sets other than those used for training, testing, and evaluation is that the qualities of the execution data (whatever they may be) remain constant, or at least constant enough that the model remains applicable. In explanatory models that serve only to explain a single data set, data quality determines to a great extent the quality of the explanation. A concern with quality is a concern with the old aphorism “GIGO”—garbage in, garbage out.

What Is Quality?

Data quality means different things at different stages of mining. When mining (in the training, testing, and evaluation data sets), quality concerns matters of consistency and representation.

All variables measure events imperfectly. The imperfections in measurement come from a wide variety of sources but are inherent and integral to all measurements. However, even if perfect measurement were possible, still the generating phenomena in the world are not themselves consistent and produce variations. All of these imperfections, as well as others, are often lumped together and termed “noise” in mining. Separating these sources of noise and determining the characteristics of each may be part of producing mined models or of explaining those models, but at the data preparation stage it is enough to regard them all as equally affective. However, natural variation (generating phenomenon variation) and measurement error are two principal contributors to data quality levels. Another is “bias,” which is the term used to indicate that some selectivity in choosing particular measured values is imposed on a data set so that it does not accurately represent the phenomena of interest in the world that it is intended to represent. These are the three fundamental areas to use to begin to assess data quality.

Principle. In creating training data sets, assess whether a variable measures what it is intended to measure.

It is important whether the measurement mechanism that generates a variable's values can, even in principle, capture a representation of the phenomenon it is intended to portray. Many data sets are filled with variables named in hope, and optimistically populated, that do not characterize the phenomena the miner expects. For instance, in one study a variable "driver experience" was actually populated by determining whether an individual had a vehicle registered in his or her name and finding the annual mileage of the vehicle. This was normalized and offered as a representation of the amount of driving experience an individual had. However, one anomalous part of this population was discovered to still be "experienced drivers" (by this criterion), even though these drivers were incarcerated for the duration of the study period. (Several "experienced drivers" turned out to be dead during the period of the study!) Another anomalous group turned out to be professional drivers who, by this criterion, had very little driving experience (their own registered vehicles had low annual mileage), yet they were in fact getting a very considerable amount of driving experience. This variable was mislabeled and did not, in fact could not, measure the phenomenon it was intended to measure.

Principle. In creating training data sets, assess the limits and character of measurement accuracy.

To demonstrate limits to measurement accuracy, the author (following a example quoted by Deming⁵) asked at the beginning of a 1-week course for all students to indicate their ages anonymously. At the end of the course the author again asked the students to indicate anonymously how old they had been at the beginning of the course. The result? With enough students (usually 20+), the two sets of values show a variance—that is, some students estimated at different times that they were different ages at the beginning of the course!

Principle. In creating training data sets, enumerate all known biases in a data set.

Almost all data is biased, sometimes intentionally, sometimes unintentionally. It is important to enumerate any known sources of bias so that any explanation can be suitably adjusted, and for any predictive model, so that the execution data can be verified to contain similar biases in selection. However well or poorly a model "works," when adjusted to match its output to the world as well as possible, it will only go on performing so long as it is applied to data that is in all material respects of the same quality as the training data sets—biases, noise, and all.

Principle. Ensure that run-time (execution) data sets maintain the same quality as the training data sets.

What are the key qualities? That depends largely on the data. However, for numeric variables it includes ensuring that the distributions remain similar; that the frequency and distribution of missing values and occurrence and range of outliers do not change; and that "improper" data measurements are not "corrected" during data acquisition. For categorical variables it includes determining that the relative frequencies of the various categories remain approximately similar and that previously unencountered categories (in the training data sets) do not appear.

⁵W. E. Deming (1950). *Some theory of sampling*, p. 37. New York: John Wiley.

Enforcing Quality: Advantages and Disadvantages

There are many ways of attempting to increase and maintain the quality of data, and there are many excellent books devoted entirely to the topic.⁶ The techniques fall into two broad categories. One category of techniques is concerned with improving the definition of what data is collected and the collecting mechanisms. Even automation is far from perfect—notice any supermarket checkout cashier take a single item and “swipe” it past a scanner multiple times when there appear to be several identical items for checkout, instead of passing each item separately, and you have seen this problem in action. Those seemingly identical items, although priced similarly, may well, and often do, have different stock keeping unit (SKU) codes, and multiple swipes result in data in the system no longer matching what is on the shelves or in the shopping basket. The price is right, the data is not.

Another category of techniques attempts to “fix” the data during or after it is collected, and these, too, can cause the miner problems. Some systems require fields to be populated before data entry can continue. The problem is that they do not always require accurate data to be entered, as the number of people with “birthdate” of “01/01/01” attests. Each person entering such data finds a set of values that satisfies the collection verification algorithm, but may not reflect in any way what the fields were designed to capture.

Fixing data after collection requires imposing rules of some sort on the data set, sometimes requiring or limiting particular variable values based on other variable values (no pregnant males, for instance). However, such rules create a structure in data that may hide, distort, or even remove entirely the very relationships and patterns sought by the miner.

The upshot is that data quality is a crucial issue in mining, but the process of improving and maintaining that quality requires imposing constraints and structures on the data that are not an unmixed blessing.

Data Quality and Model Quality

As in so many areas of life, GIGO reigns supreme. It is simply impossible to build high-quality models from low-quality data. Unfortunately, often the miner has no control over the data collected for mining. Available data often is collected already, although sometimes the collection is by methods unknown, for purposes uncertain, now to be used to solve problems obscure. Insofar as it is possible it is well worth the miner’s time and effort to obtain data of the highest quality possible and to improve it as much as human resource and ingenuity permit.

It may often seem, in the daily practice of data mining in the real world, that the best that the miner can accomplish, as far as improving data quality goes, amounts to no more than getting better quality garbage. Nonetheless, just as the aphorism “the proof of the pudding is in the eating” indicates, the quality of a model is in its application, and principled data preparation can mean the difference between a successful (read: “profitable for the company”) model and an unsuccessful one.

DATA VISUALIZATION

At the beginning of all data mining work comes the task of collecting and preparing the data. At the end of all data mining work comes the time to communicate the results. It matters not whether the project calls for a predictive, classificatory, explanatory, exploratory, scenario

⁶See, for instance, M. Chisom (2000). *Managing Reference Data in Enterprise Database*. San Francisco: Morgan Kaufmann; and D. Loshin (2001). *Enterprise Knowledge Management*. San Francisco: Morgan Kaufmann.

planning, strategic, tactical, or any other type or sort of model; in the end they all have to be explained, and it is always easier when using relevant and applicable graphics. This is one use for data visualization—the clear and simple graphical representation of complex relationships.

However, explaining models is by no means the only use of graphical representations of data. All data mining is a form of pattern recognition (even if it does not use traditional pattern recognition techniques). The most formidable pattern recognition apparatus known to humankind is easily accessible to any data miner—the human brain and mind. So powerful is the human mind that it will perceive patterns even where none exist. The question is how to enable this formidable apparatus to go to work most effectively in data mining. Given that vision is a predominant sense, and, not coincidentally, that computers have been created to communicate with humans in the main visually, computerized visual data representation provides an efficient connection between data and mind.

Each of these visualization topics is vast and crucially important, and each has prompted a comprehensive literature. Some of the data visualization effects possible are spectacular, but the balance of this chapter has room only for an outline sketch. It attempts, albeit briefly, to point to the benefits that visualization can offer when mining data.

Seeing Is Believing

Medical doctors looking at X-ray images can “see” things of which untrained observers are quite unaware, even when the features of interest are pointed out. Such images actually do not show the features that are “seen” (because the inside of the body is, in the main part, opaque) but the intuitive perception is that the X-ray makes hidden features visible. Trained interpreters express their interpretations in terms that clearly show that they accept what they see as fact, not interpretation.

This feeling is very convincing and usually quite justified. The art of data visualization for actually mining data is to produce “data X-rays” that can engender the same degree of confidence for the miner—at least when he or she is trained and practiced in interpreting data X-rays in particular domains. Are such data X-rays possible?

Figure 14.4 shows the results of visualizing a data set—credit card solicitation data set—using SOMiner.⁷ This figure shows the cluster map generated by this tool, together with several smaller maps to the right of the main window. These smaller maps show individual variables, whereas the larger main window is the cluster map for all of the variables combined.

The main window, shown enlarged in Fig. 14.5, shows labels on the clusters. The miner inserts these labels after inspecting and considering the individual variable maps, a few of which are shown enlarged in Fig. 14.6. Color is very important to the interpretation of these maps, and the enlarged individual variable maps shown are for binary variables.

Inspection and interpretation of such maps does indeed act very much as an X-ray of the data. Experience allows a miner to quickly “see” complex, multivariate relationships that are hard to develop with other techniques. But such maps have greater utility than for the miner’s use alone. In the author’s experience, working with such maps in group settings, particularly with technical and management teams familiar with the data, and projecting such maps for group consideration very often quickly leads to insightful comments from the team as to the meaning of particular relationships not immediately obvious to the author as significant. The learning

⁷SOMiner is a self-organizing map (SOM) tool produced by Viscovery; see www.viscovery.com. The maps shown, and the data set used, are available from www.modelandmine.com. The maps shown can be opened and explored with the evaluation version of SOMiner available for free download from the Viscovery web site.

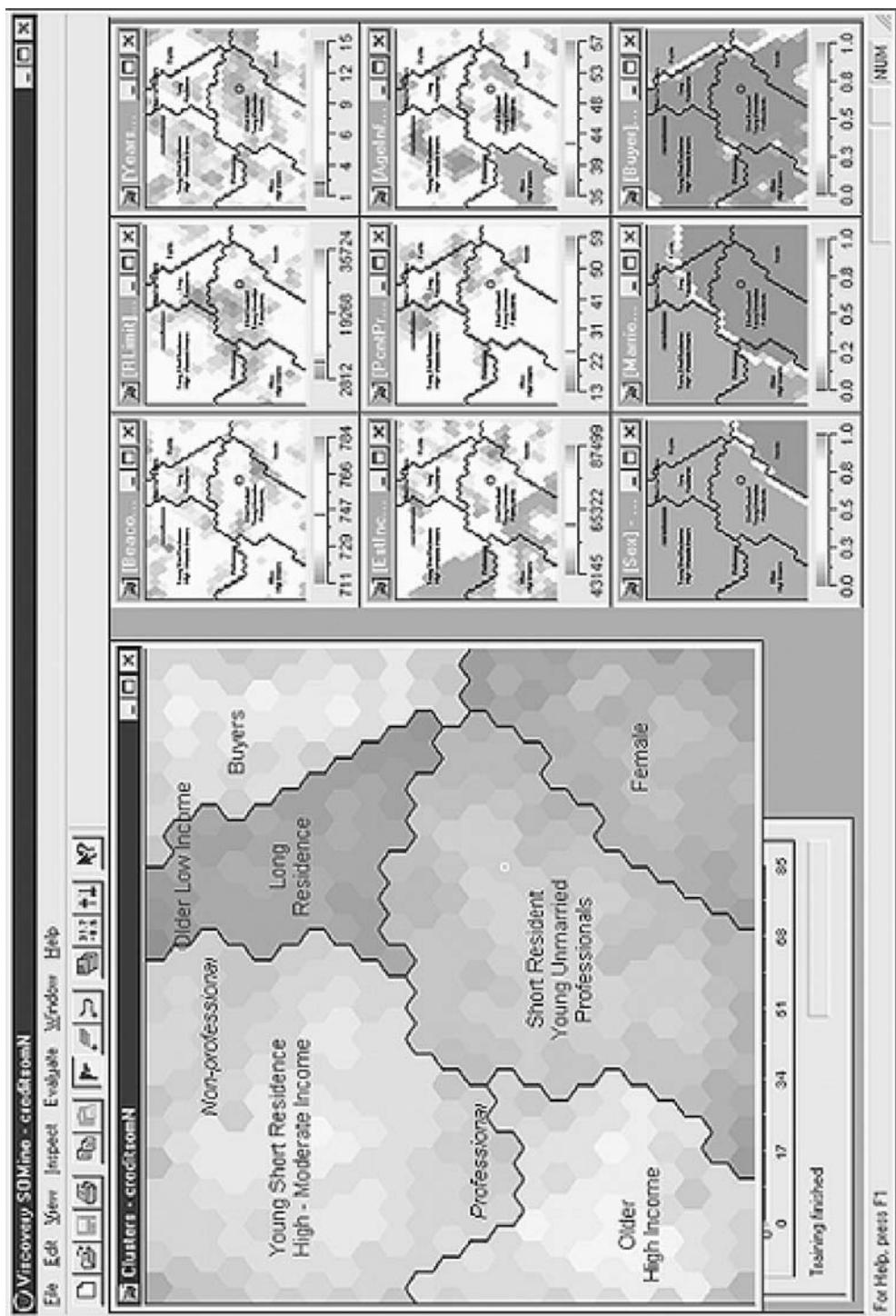


FIG. 14.4. SOM of credit data. (Reprinted with permission from D. Pyle, *Business Modeling and Data Mining*. San Francisco: Morgan Kaufmann, 2003. Morgan Kaufmann Publishers, © 2003.)

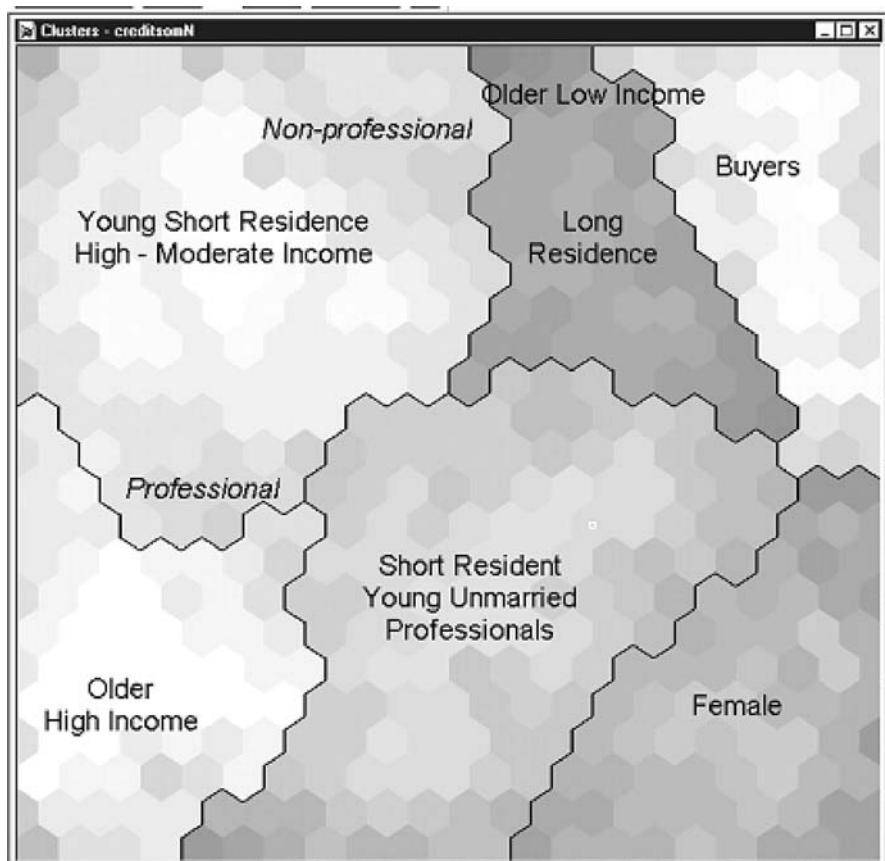


FIG. 14.5. SOM clusters identified by miner. (Reprinted with permission from D. Pyle, *Business Modeling and Data Mining*. San Francisco: Morgan Kaufmann, 2003. Morgan Kaufmann Publishers, © 2003.)

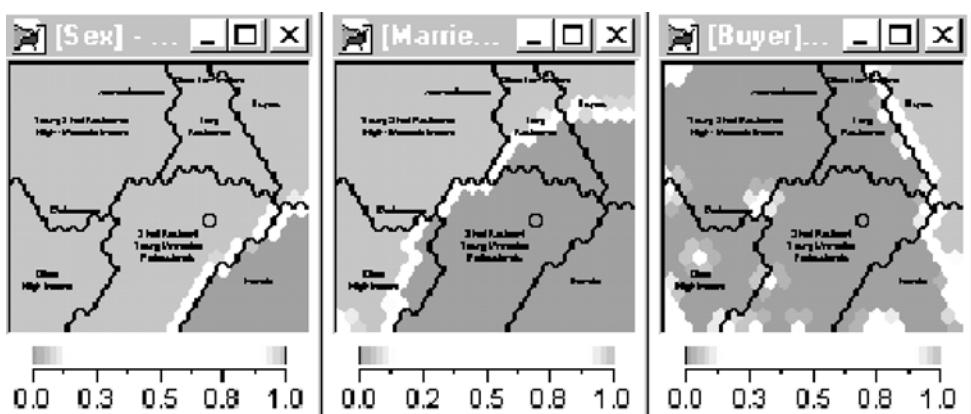


FIG. 14.6. Individual variable component maps of SOM. (Reprinted with permission from D. Pyle, *Business Modeling and Data Mining*. San Francisco: Morgan Kaufmann, 2003. Morgan Kaufmann Publishers, © 2003.)

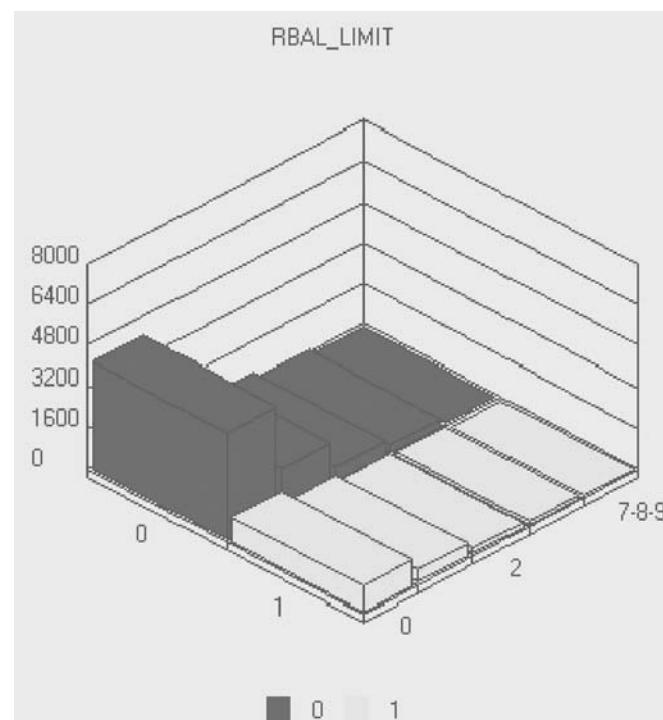


FIG. 14.7. Absolute representation of relationship.

curve for interpreting maps by such teams is surprisingly short, although careful preparation and manipulation of the data is very important in producing the maps in the first place.

The power of such maps is that, when the data is appropriately prepared for them, they reveal an enormous wealth of relationships—patterns—that few other techniques make as accessible for inspection. This is a data visualization technique that can repay the time and effort in learning the mapping techniques many times over.

Absolute Versus Relative Visualization

Figure 14.7 shows an investigation of the same credit data set as explored with the SOM discussed in the previous section. The SOM is an excellent visualization tool for initial exploration and frequently suggests complex, nonlinear, and multivariate relationships. However, such explorations are more qualitative in nature than quantitative, and the greatest benefit from them is that they suggest interesting and very often highly fruitful directions of inquiry. But whatever hints and suggestions are gleaned, the hints and suggestions need to be quantified and expressed in appropriate terms.

Figure 14.7 shows the one potentially interesting interaction suggested by the SOM investigation between a variable “RBAL_LIMIT” and the output variable. In this figure absolute numbers of interactions are shown, and it is easy to see that the left-most pair of bars (front and back) account for almost all of the data, because the height of the bar is proportional to the number of records in the segment. However, what is not apparent from this visualization of the relationship—and what is more interesting in terms of the business objectives of this data set—is the relative proportions of results in each bar. The relative effects are the comparative heights of the bars at the front and rear of this figure, and in an absolute representation are very difficult to intuit.

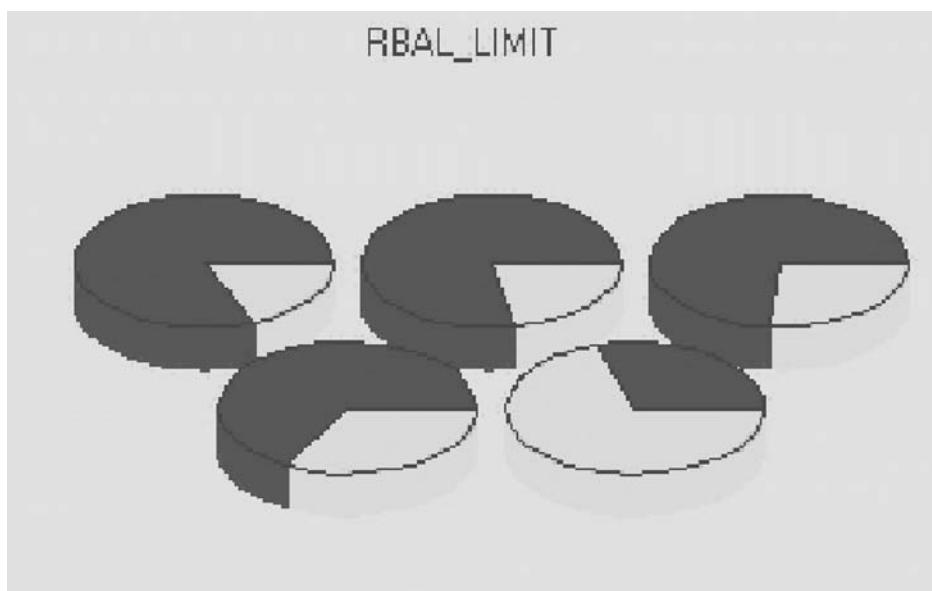


FIG. 14.8. Relative representation of relationship shown in Fig. 14.7.

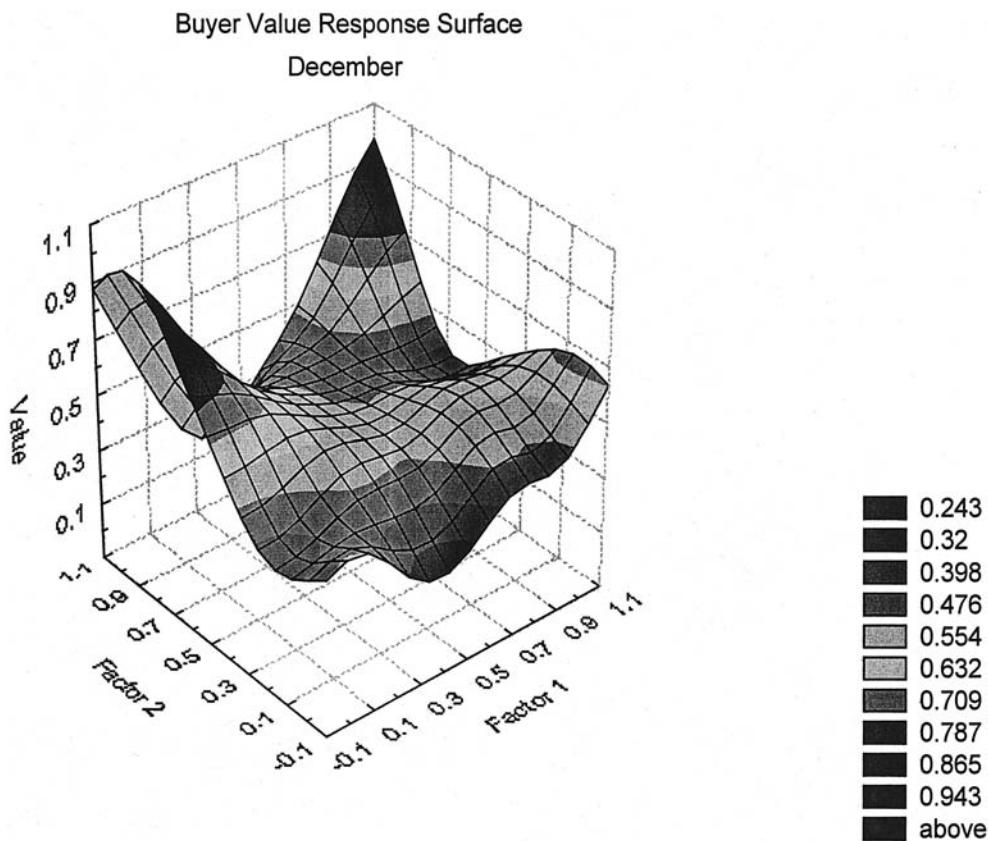


FIG. 14.9. Response surface.

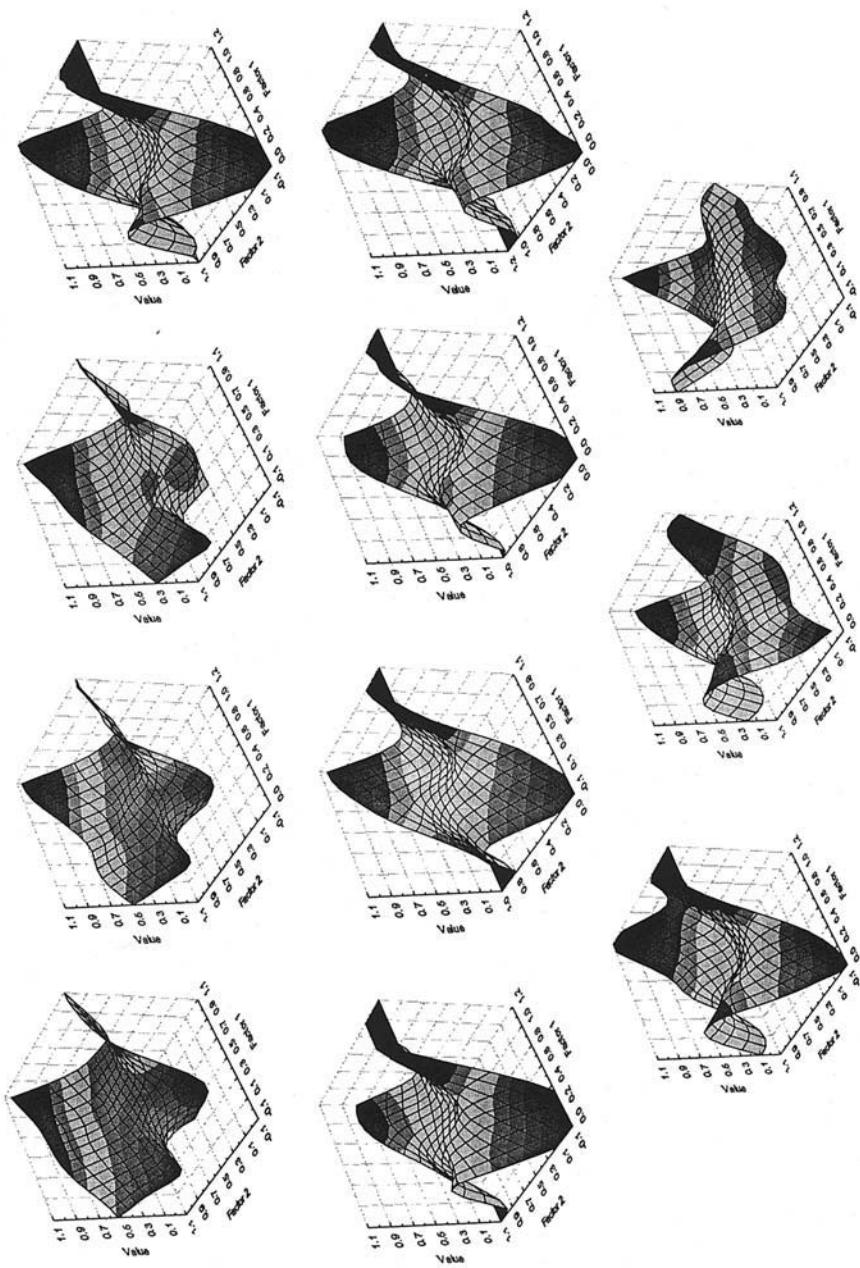


FIG. 14.10. Response surfaces showing changes in response over 11 months, February through December.

Figure 14.8 shows the same relationship. However, although any indication of the overall predominance in the data set has been lost, here the relative size of the outcome of interest is easily seen to have a definite correlation with each segment. The top three pies in Fig. 14.8 show the relative proportions for the left-hand three pairs of bars in Fig. 14.7, and the lower two pies show the relationship for the right-hand pair of bars.

Visualizing Multiple Interactions

The SOM in Fig. 14.5 shows one very useful way of visualizing multiple interactions. Another is to use a three-dimensional plot, such as that of Fig. 14.9. The image in this figure is one of a series. It shows the response surface formed by extracting two nonlinear factors from a multivariate data set using advanced data mining techniques. These two factors characterize several responses of interest over the whole of the surface. The individual image shown in the figure itself encapsulates a high density of interesting information about response during 1 month. However, it is possible to extend this approach to visualizing data to incorporate the effect of time. Figure 14.10 shows several response surfaces for the same factors, which were extracted at monthly intervals from February through December. Animating this series displays how response changes over time and gives a dynamic view of the changing interactions in the data.⁸

SUMMARY

This chapter has taken a brief look at some of the principles of effective data assembly and preparation and a quick glance at where data visualization can be of inestimable use in the process of mining data. The principles discussed here serve to set a miner on the right road to assembling and preparing data to yield the best possible model given the business problem, the nature of the available data, and the tool available. However, any single chapter can offer only a nudge in the right direction. Further tutoring in the school of experience is inevitable, but with this as a foundation much frustration and difficulty can easily be avoided.

⁸The images and an “animated” version of this response surface, together with a more detailed explanation of what it shows, are available from www.modelandmine.com.

15

Data Storage and Management

Tong (Teresa) Wu
Arizona State University

Xiangyang (Sean) Li
Rensselaer Polytechnic Institute

Introduction	393
Text Files and Spreadsheets	395
Text Files for Data	395
Spreadsheet Files	395
Database Systems	397
Historical Databases	397
Relational Database	398
Object-Oriented Database	399
Advanced Topics in Data Storage and Management	402
OLAP	402
Data Warehouse	403
Distributed Databases	404
Available Software	406
Summary	406
Acknowledgments	407
References	407

INTRODUCTION

There are many data mining algorithms, such as association rules, clustering, decision trees, discriminant analysis, artificial neural networks, genetic algorithms, and so on. These algorithms are used to process data from various fields to retrieve information and discover knowledge that

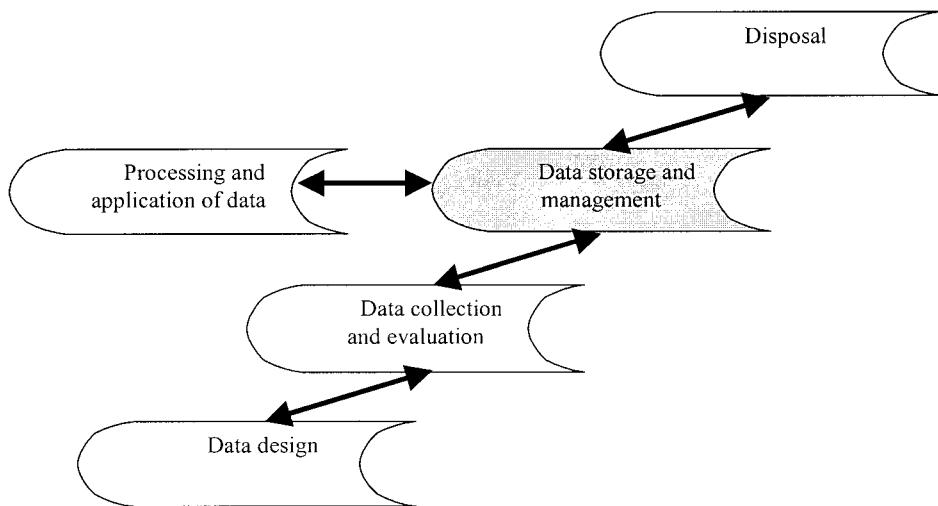


FIG. 15.1. The lifecycle of data.

can drive an executive's decisions. Information is data associated with the past and the present. Knowledge provides a basis for the prediction of future trends based on original data and the necessary information extracted from the original data. Clearly, information and knowledge are communicated through data.

The data have a lifecycle starting with initial data design, followed by collection and evaluation, storage and management, processing and application, and ending with data disposal, as shown in Fig. 15.1. In the design stage, data are carefully defined with necessary attributes based on the problem specification. After the data are collected and before they are stored, the accuracy, validity, and quality of the collected data are evaluated. The goal of data storage and management is to make data optimally available to support processing and application of the data, which are carried out to extract features and build models with data mining algorithms. When the data become obsolete, the useful pieces are kept for future use and the remaining data disposed of after backup copies are made. Note that the different phases are linked bidirectionally and interactively. The feedback from late stages to early stages helps to correct errors and problems.

Data storage and management is an important stage in the data lifecycle. It aims to give access and fusion of massive amounts of data, information, and knowledge in a timely manner. To achieve this goal, persistent storage, distributed processing, and database management must be integrated to provide users an environment in which to store and obtain relevant information, as well as update this information. There are different approaches to organizing and managing data in storage files and systems, from simple text files to databases to data warehousing. Each method has its own unique, systematic perspective for capturing and describing studied system data and designing storage structures based on this perspective. This chapter briefly introduces different data storage and management files and systems, followed by the discussion of some advanced issues such as On-Line Analytical Processing (OLAP), data warehousing, and distributed databases.

TEXT FILES AND SPREADSHEETS

Text Files for Data

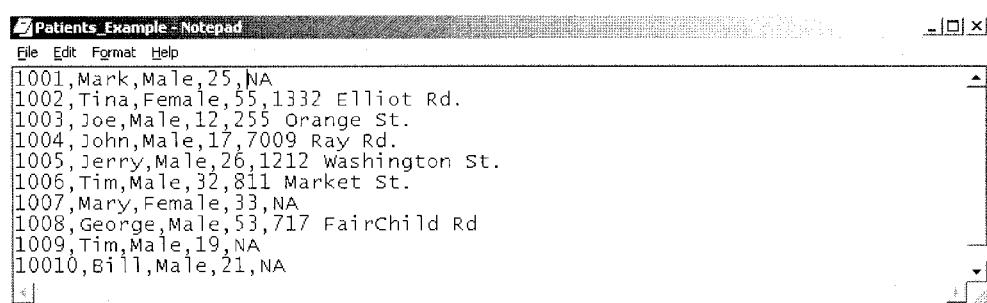
One method for storing and managing data is a text file. The most common text file format in computers is ASCII (American Standard Code for Information Interchange), which was developed by the American National Standards Institute (ANSI). In an ASCII file each alphabetic, numeric, or special character is represented with a 7-bit binary number (a string of seven 0s or 1s). In total, 128 possible characters are defined. UNIX and DOS-based operating systems use ASCII for text files. Windows NT and 2000 use a new format, Unicode, which is a model based on ASCII that extends ASCII's 7-bit character size to 16-bits. The extension of bits fulfills multilingual requirements and makes Unicode a system for “the interchange, processing, and display of the written texts of the diverse languages of the modern world” (<http://www.unicode.org>). Another example of text format is EBCDIC (Extended Binary-Coded Decimal Interchange Code), developed by IBM for its large operating systems such as the OS/390. In an EBCDIC file each character is represented with an 8-bit binary number (a string of eight 0s or 1s), and 256 possible characters are defined. Different conversion programs are available to change a file from one format to another.

Example 1. ASCII data files are simply text files, consisting of rows and columns. For instance, an ASCII data file about patients, recording identification number (ID), name, gender, age, and address might look like Fig. 15.2.

Figure 15.2 shows the first 10 records in a patient data file edited by Notepad, a text editor. Each physical record or line of data in the file represents one unit of analysis. A comma is used as a delimiter to separate the items in each record. However, without linking the columns in the file to specific defined variables, the alphanumeric characters are meaningless. With a specifically developed computer program the data can be interpreted properly. For example, the first line of Fig. 15.2 can be interpreted as follows: ID is “1001,” name is “Mark,” gender is “Male,” age is “25,” and address is “NA,” where ID, name, gender, age, and address are the defined variables for a record.

Spreadsheet Files

Another simple way to store and manage data is a spreadsheet. Professor Richard Mattessich (1961) pioneered the development of computerized spreadsheets in 1961. Robert Frankston



The screenshot shows a Microsoft Notepad window with the title "Patients_Example - Notepad". The menu bar includes File, Edit, Format, and Help. The main content area displays the following 10 records of patient data:

ID	Name	Gender	Age	Address
1001	Mark	Male	25	NA
1002	Tina	Female	55	1332 Elliot Rd.
1003	Joe	Male	12	255 Orange St.
1004	John	Male	17	7009 Ray Rd.
1005	Jerry	Male	26	1212 Washington St.
1006	Tim	Male	32	811 Market St.
1007	Mary	Female	33	NA
1008	George	Male	53	717 Fairchild Rd
1009	Tim	Male	19	NA
10010	Bill	Male	21	NA

FIG. 15.2. Patient data file in ASCII format.

and Dan Bricklin developed the first spreadsheet software, called VisiCalc, in 1978. Lotus 1-2-3 was then introduced in 1983. Lotus presented spreadsheet software as a major data presentation package as well as a complex calculation tool including charting, plotting, and database capabilities. The next milestone was Microsoft Excel in 1987, which included graphic capabilities. Spreadsheet programs in use today include Excel, Lotus 1-2-3, Appleworks, Filemaker, and Corel Quattro Pro.

In spreadsheets, data are stored in a two-dimensional table with data records represented in rows and attributes represented in columns. Spreadsheet tools such as Excel make it easy to display information and use data manipulation functions to load, modify, and save data, or transform data from another file format to a spreadsheet format. Example 2 illustrates how to load the text file from the previous example into Excel. Interested readers can refer to Web sites for more spreadsheet operations, for example, JWALK & Associates, Inc. (<http://www.j-walk.com>), and Microsoft (<http://www.microsoft.com>).

Example 2. Loading patient data file with Excel

1. Open Microsoft Excel
2. Select File -> Open Menu
3. Specify “all files” for Files of Type
4. Select the Patient Data file (example 1)
5. Select Delimiter in Text Import Wizard Step 1
6. Select Comma as Delimiter in Text Import Wizard Step 2
7. Select General as Column Data Format in Text Import Wizard Step 3

The Excel spreadsheet of Patient Data is shown in Fig. 15.3.

Besides basic data operations, spreadsheet packages such as Excel provide standard functions of mathematical and statistical calculations and data visualization and display. Because the data are loaded into a set of data cells, spreadsheet tools use mathematical and statistical calculation functions to process the cell values. Curves and charts can be plotted using the values from a set of data cells. The cells used in calculating, and their related plotted graphs, can be stored in the same data file or in a separate file. There are even application programming interface (API) libraries provided in some spreadsheet software programs such as the ones in Excel. Users can utilize scripts to automate their tasks with the aid of the API.

1001	Mark	Male	25	NA
1002	Tina	Female	55	1332 Elliot Rd.
1003	Joe	Male	12	255 Orange St.
1004	John	Male	17	7009 Ray Rd.
1005	Jerry	Male	26	1212 Washington St.
1006	Tim	Male	32	811 Market St.
1007	Mary	Female	33	NA
1008	George	Male	53	717 FairChild Rd
1009	Tim	Male	19	NA
1010	Bill	Male	21	NA

FIG. 15.3. Patient data in Excel spreadsheet.

Both text files and spreadsheets also are called “flat files,” in which the same fields are located at the same offset within each record. Flat files require less work to create data records because data is accessed by following a specific format. As a result, even minor changes in the record layout require the program to be changed accordingly. Another issue with flat files is the hidden relationships of attributes that cross records or files. The relationships are necessarily implemented within application code or logic. In flat files a common technique is to store important, repeatedly-used attribute fields redundantly across several files to facilitate the overhead of accessing the data. A problem created by this redundancy is that it increases the size and number of maintenance programs required to update the replicated copies. Flat files have been applied in some data mining applications, such as those using association rules. Nevertheless, relationships within the attributes of multiple tables are not easy to capture, which makes flat files inapplicable for many other data mining techniques. Due to these limitations, flat files are suitable only for data mining applications involving data sets with small size and volume.

DATABASE SYSTEMS

Database systems play a key role in data storage and management systems today. A database system normally consists of database files and a database management system (DBMS). Database files contain original data, indices, metadata of the database logic, and information such as log data for maintaining the database. A DBMS aims to manage data and improve operating performance. The tools in a DBMS, including mainly the database administrator, query processor, and transaction manager, accept commands from users. For example, the query processor translates and optimizes query commands and then scans and searches related database files to produce reports.

Historical Databases

Both the hierarchical model and the network model attempted to provide a foundation for database systems (Hogan, 1990). They were used in the first commercial database systems in the 1960s and 1970s. A hierarchical system (Fig. 15.4a) links records together like a tree, with the top-most record referred to as the “root.” Each record has only one owner at the

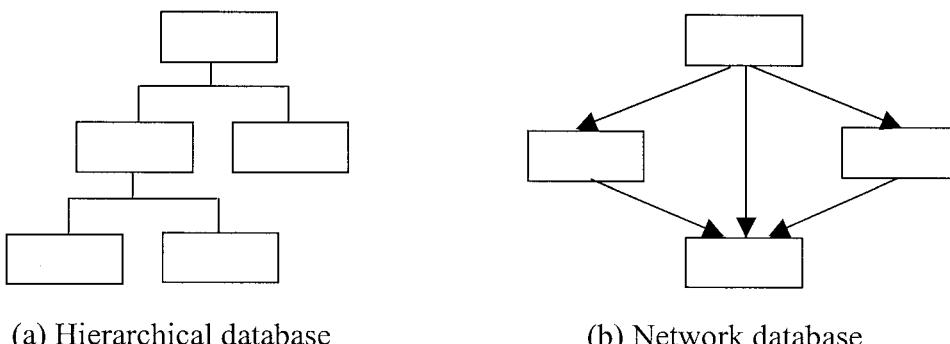


FIG. 15.4. (a) Hierarchical database and (b) network database.

next higher level. A network database system (Fig. 15.4b) provides more flexibility than a simple hierarchical database system. One subordinate record type may report to many owners at different levels. To specify the linkage between a certain type of owner record and one or more types of member records, a pointer chain is created to represent the linkage path.

Relational Database

The idea that database systems should present the user with a view of data organized as tables called “relation” was originally proposed by Ted Codd (1970). Today the most prevalent type of database is the relational database, which has been applied in many fields covering accounting systems, inventory control, banking, and so forth.

Relational Data Storage Model

Entities, attributes, and relationships are components of a relational model. An entity is a concrete object in its reality. Attributes are values describing properties of an entity. Relationships are connections among two or more sets of entities. The idea of a key on a table is central to the relational model. The purpose of a key is to identify each row uniquely. A primary key is an attribute (or combination of attributes) that uniquely identifies one row or record. A foreign key is an attribute (or combination of attributes) that appears as a primary key in another table. Foreign key relationships provide the basis for establishing relationships across tables in a relational database.

Example 3. Let us consider the previous example presented by a relation. The relation *patient* has five attributes: “ID,” “name,” “gender,” “age,” and “SSN.” Assume that there is another relation that records patient insurance information. Relation *insurance* has the attributes “name” and “SSN.” In relation *patient*, “ID” is the primary key and “SSN” is the foreign key relating *patient* to *insurance*, because “SSN” is the primary key for *insurance* as shown in Fig. 15.5.

Patient:

ID	Name	Gender	Age	SSN
1001	Mark	Male	25	480-45-3235
...

Primary Key of *Patient*

Foreign key relating *Patient* to *Insurance*

Insurance:

Name	SSN
Mark	480-45-3235
...	...

Primary Key of *Insurance*

FIG. 15.5. Relational tables.

Structural Query Language

In a relational database all the data are stored in tables and all operations on data are either done on the tables themselves or produce other tables as a result of the operation. Several languages have been created to manipulate relational tables. The most common is the ANSI Standard SQL (Structured Query Language). SQL queries are used to perform tasks such as updating data in a database or retrieving data from a database. Even though there are various versions of SQL software available from different database vendors, the ANSI standard SQL commands can still be used to accomplish most tasks.

Example 4. Suppose we want to manage the patient table from example 3. We could use SQL statements to create an empty table named “Patient”:

```
CREATE TABLE Patient (
    ID char (10),
    Name char (30),
    Gender char (10),
    Age Integer
    SSN char (20)
)
```

The “insert” statement is used to insert or add a record into the table. For example:

```
INSERT INTO Patient (ID, Name, Gender, Age, SSN)
VALUES (“1001”, “Mark”, “Male”, 25, “480-45-3235”)
```

The “update” statement is used to update or change records that match specified criteria. For example, the age of patient “Mark” is updated as follows:

```
UPDATE Patient
SET Age = 26
WHERE Name = “Mark”
```

The “select” statement is used to retrieve selected data that matches the criteria specified. For example: the name and ID of the patient with SSN (480-45-3235) are retrieved as follows:

```
SELECT Name, ID
FROM Patient
WHERE SSN = “480-45-3235”
```

Assume we want to delete the table “Patient”: The “drop” statement is used:

```
DROP TABLE Patient
```

Object-Oriented Database

With the development of the Object-Oriented concept, the flat file and relational database approaches no longer generate satisfactory results. To manage object data, the objects have

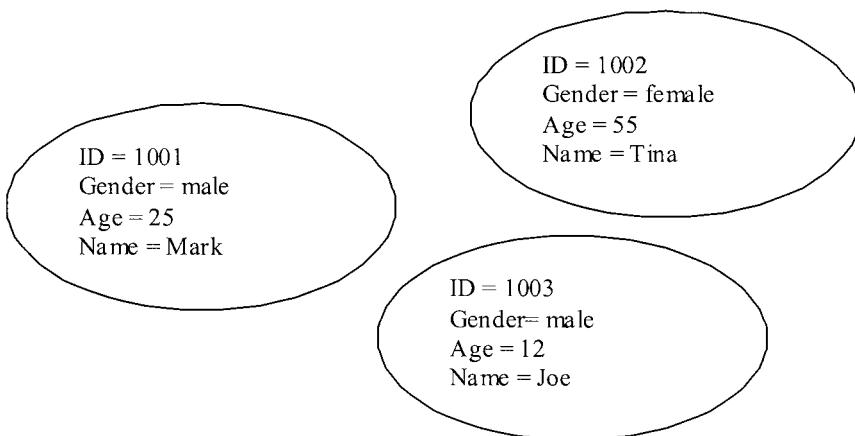


FIG. 15.6. Object-oriented data model.

to be flattened or decomposed and stored in a tabular or flat file structure. Therefore, object-oriented databases, the databases that are congruent with the data defined in object classes, have become increasingly popular.

Object-Oriented Data Storage Model

In the object-oriented data storage model, data are considered as a collection of objects, which are the instances of classes as shown in Fig. 15.6. A class is defined as a group of objects that have similar attributes and unique object IDs. Access to data records/objects follows a systematic approach using the concept of encapsulation, which provides well-defined interfaces for a class of objects. Unlike in a relational table, in which each element is an atomic value such as a string or number, an object can contain another object as its component. Object-oriented data models give a perspective that follows from the way human beings view, understand, analyze, and model the universe. Thus, object-oriented models offer great advantage in investigating complex systems and objects.

Object Definition Language

ODL (Object Definition Language) is a proposed standard language for specifying the schema or structure of object-oriented databases. The primary purpose of ODL is to allow object-oriented designs of databases to be written and then translated directly into declarations of an object-oriented database management system. In ODL the keyword “interface” is used to indicate a class; the keyword “attribute” describes a property of an object by associating a value of some simple type, for example, string or integer, with that object. The keywords “relationship” and “inverse” declare respective connections between two classes.

Example 5. The classes *patient* and *insurance* introduced in example 3 are declared in ODL as shown in Fig. 15.7.

For the class *patient*, the first attribute shown on line 2, is named ID and has string as its type. Four more attributes are declared on lines 3–6. Lines 7–8 declare the relationship between *patient* and *insurance*; that is, each patient has only one insurance, and each insurance is signed by only one patient. If each object of a class refers to a set of objects rather than to a single

```

1. interface Patient {
2.     attribute string ID;
3.     attribute string Name;
4.     attribute string Gender;
5.     attribute integer Age;
6.     attribute string SSN;
7.     relationship Insurance has ;
8.         inverse Insurance :: signed by
9. }
10. interface Insurance {
11.     attribute string Name;
12.     attribute string SSN;
13.     relationship Patient signed by
14.         inverse Patient :: has
15. }

```

FIG. 15.7. Object-oriented model in ODL.

object, for example, one patient has more than one insurance, then lines 7–8 are rewritten as:

```

relationship Set <Insurance> has ;
    inverse Insurance :: signed by

```

The keyword “set” is used. The class *insurance* is declared in lines 10–15.

Object Query Language

Like SQL, OQL (Object-Oriented Query Language) is a standard for database objects querying. OQL is an SQL-like notation for the object-oriented paradigm. It aims to provide resolution for object-oriented programming languages such as C++, Smalltalk, or Java. Objects are thus manipulated both by OQL queries and by the object-oriented programming languages.

Example 6. Suppose *p* is an instance of the class *patient*, then the syntax used to ask for the age of patient who has the name “Mark” is as follows:

```

SELECT p.Age
FROM Patient p
WHERE p.Name= "Mark"

```

Database systems provide efficient access to large quantities of data and a number of essential capabilities for data mining, which include persistent storage, data models (e.g., relational or object-oriented) that permit data to be managed using a logical rather than a physical view, and high-level query languages (SQL and OQL) that allow users to request the data they want without having to write complex programs, specifying how to access it. For complicated data mining applications further advances of database systems development need to address the following issues (Carbone, 1997):

- Large volume: Database systems are required to store and process terabytes and even petabytes of data.
- Noise and uncertainty: Noise might be introduced by faulty data collection devices leading to data inconsistency.
- Redundant information: Multiple copies of stored data cause redundant information.
- Dynamic data: Transaction databases are set up to process millions of transactions per hour. This creates difficulty for data mining tools that are oriented to look at static sets of data.
- Sparse data: The information in the database is often sparse in terms of the density of actual records over the potential instance space.
- Multimedia data: Database systems are increasingly capable of storing more than just structured data. For example, text documents, images, special data, video, and audio now can be stored as objects in databases.

Recent developments of online analytical processing tools and data warehousing have greatly increased the efficiency with which database systems can support a large number of extremely complex queries that are typical of data mining applications.

ADVANCED TOPICS IN DATA STORAGE AND MANAGEMENT

OLAP

Online analytical processing (OLAP) is a category of software technology that enables users to gain insight into data through fast and interactive access to various views of information transformed from raw data, to reflect the real dimensionality of the problem. OLAP is characterized by the function of dynamic dimensional analysis of aggregate enterprise data. A data dimension represents a special perspective of the data along with the data processed, such as time (sales for the last year, by week, month, or quarter), geography (regions or offices), and customers (target market, type, or size). The dimensions are typically hierarchical. Thus, preaggregation can be done logically according to the hierarchies. Preaggregation also allows for a logical drill-down from a large group to a small group. Another way to reduce the data cells in a multidimensional data model is to handle sparse data efficiently. These key techniques of hierarchical dimensions, preaggregation and sparse data management, can reduce the data size and the need for calculation, thus providing quick and direct answers for queries.

OLAP Architecture

There are two options for OLAP architecture: (1) Build OLAP using custom programs with relational databases or (2) choose integrated programs and a specialized data storage mechanism together. The first option relies on relational database tools for system performance and provides limited functionalities.

The second option is becoming a challenger to traditional database programmed solutions. One approach is multidimensional OLAP (MOLAP). In MOLAP, multidimensional databases called MDDBs are designed to optimize OLAP retrieval with the consideration of relational database systems. Another approach is relational OLAP in which relational databases are integrated within an application interface that supports OLAP functions.

TABLE 15.1
OLAP Tools

Users	System Class and Business Need	Time Span	Data Profile
Executive	Executive information system, strategic decision making	Long term	Highly summarized, trend data, critical success measures
Middle manager	Executive information system, decision support system, tactical decision making	Short term	Midlevel summarized, tactical decision support, business performance monitoring
Business analyst	Data mining, forecasting, what-if scenarios	Long term, short term	Disaggregated data, new business insights, scenarios building with local data inputs
End-User	End-user query tools	Long term, short term	Aggregated data, trend view, information kiosk

Note: From OLAP: A Market Overview by D. Henderson and H. Chervinsky, 1998, *Handbook of Data Management* (pp. 599–608). Copyright 1998 by CRC Press. Reprinted with permission.

Choosing OLAP Tools

A number of OLAP tools have been developed to target different customers. Henderson and Chervinsky (1998) classify the customers into four categories: end-users, business analysts, middle managers, and executives. Some basic OLAP categories, with their respective targeted users and the functions they support, are exhibited in Table 15.1.

Although OLAP offers an enhanced storage mechanism for data retrieval, some limitations of using OLAP for data mining exist. One limitation is that OLAP does not provide low-level granular data, whereas the data used for data mining is required to be of the same granular level. In the case that data mining applications aim to predict individual line items such as sales, OLAP totally fails. For instance, if a marketing department wants to predict the sales potential of a certain make of automobile without considering the individual features and condition of the vehicle, OLAP is used to aggregate data and generate sales summaries. However, if the prediction is about individual customer behavior, OLAP is incapable of allowing individual cases as input.

Data Warehouse

A data warehouse is a decision support system; a structured environment designed to store and analyze all or significant parts of a set of data. The data are logically and physically transformed from multiple source applications into business structure and are updated and maintained over a long time period. The data warehouse is organized around the major subjects in an enterprise such as customer, vendor, product, and activity. The alignment around subject areas affects the design and implementation of data in the data warehouse. Data that will not be used for decision support system processing is excluded from the data warehouse.

The architecture of a typical data warehouse and its environment is shown in Fig. 15.8. The main components of a data warehouse include a database; data loading and extracting tools; administration and management platforms; and application tools such as data mining, query, and reporting, visualization and display, and so on (Berson & Smith, 1997). The major functions of these components are as follows (Zagelow, 1997):

- Access, transform, clean, and summarize data into data warehouse database.
- Store and manage data.

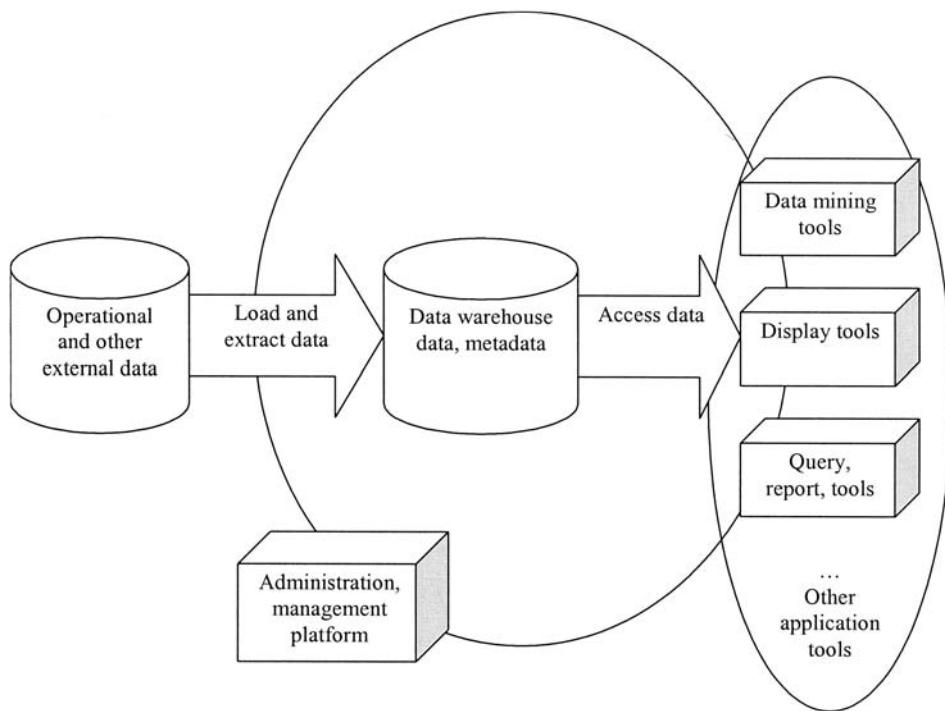


FIG. 15.8. Data warehouse architecture.

- Manage metadata.
- Display and analyze data, discover useful patterns, and produce reports.

To develop such a data warehouse, three steps—mapping from source documentation to enterprise model to tabular model to dimensional model—are taken as shown in Fig. 15.9. Source documentation captures the physical data structure of an operational system. The enterprise model describes all important data in the enterprise at a high level. The outcomes are usually consistent naming conventions and a list of attribute names and definitions, which are useful in mapping source data to the warehouse. The tabular model supports queries and reports for a specific business function such as finance or customer support. The dimensional model supports numerical analysis and represents data as an array. The values in the array are numeric facts that measure business performance, and the dimensions of the array are parameters of the fact, such as date of sale, region, and product.

A well-designed data warehouse makes the right information available to the right people at the right time, and therefore helps a company compete aggressively and sustain leadership.

Distributed Databases

Ideally, all the data related to a data mining application is stored and managed in a centralized structure. This occurs with most small data mining applications. However, with the rapid increase of data volume and system scale, many data storage and processing systems are geographically dispersed, and the need for the development of a distributed database system

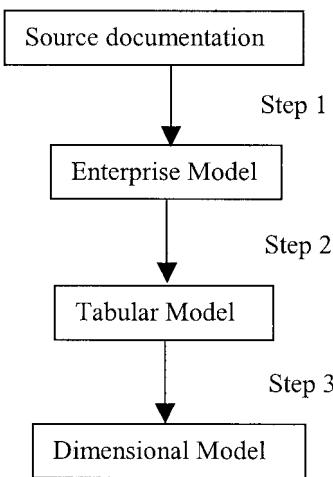


FIG. 15.9. Data warehouse design procedure.

becomes evident. Typical examples include nationally or globally distributed branches of retailers, airlines, or financial service institutions. Providing added urgency to the need for distributed database systems are the recent developments in computer networking technology, primarily the advancements in communication and network architecture and distributed computing technologies such as CORBA and Java standards. Advantages of distributed databases include tremendous expansion in storage capacity and processing speed and improvements on robustness.

A typical distributed data storage system is shown in Fig. 15.10. The separated databases are operated by the distributed DBMS communicating through the network framework.

Several important issues arising in distributed databases are distributed data processing, consistency control, distributed metadata management, interoperability between heterogeneous databases, security, and so on. The interested reader can refer to Thuraisingham (1998) and Elmagarmid, Rusinkiewicz, and Sheth (1999) for details.

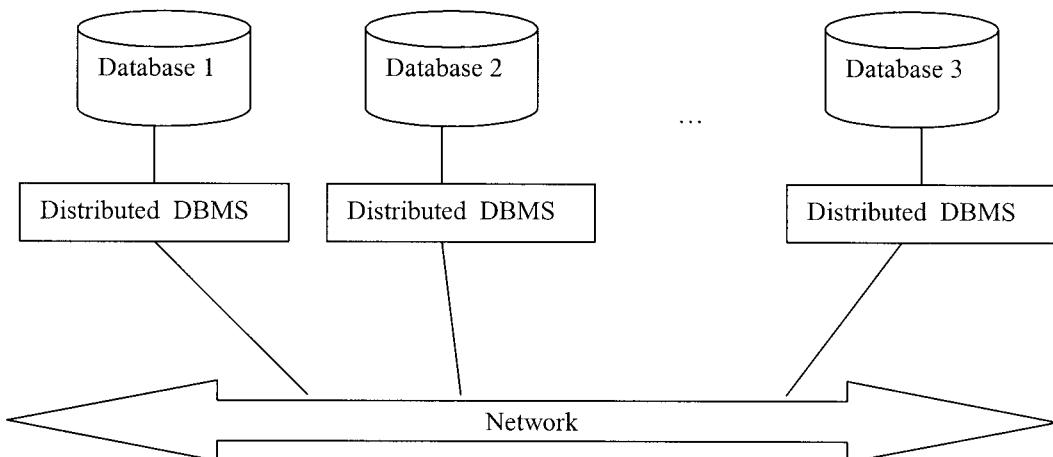


FIG. 15.10. Distributed database structure.

TABLE 15.2
List of Software

Category	Software	Description
Text editors	Notepad	A basic text editor that can be used to create simple documents. The most common use for Notepad is to view or edit text (.txt) files. Notepad files can be saved as Unicode, ANSI, or UTF-8, which brings the user the flexibility to work with documents that use different character sets.
	UltraEdit-32	A text and hexadecimal editor. It includes column-mode editing, and conversion from DOS to Unix file types, etc. UltraEdit-32 also supports HTML.
Spread sheets	Microsoft Excel	Dominates the spreadsheet market with approximately 90% of the market share.
	Lotus 1-2-3	Spreadsheet software compatible with Excel, Lotus Notes, and leading databases such as IBM DB2 and Oracle.
	Quattro Pro	Originally developed by Borland International and sold to Novell and later sold to Corel. Quattro Pro was modeled after Lotus 1-2-3. Current versions are modeled after Microsoft Excel.
Databases	Microsoft SQL	Microsoft SQL Server provides extensive database programming capabilities built on Web standards. Rich XML and Internet standard support give the user the ability to store and retrieve data in XML format easily with built-in stored procedures.
	Oracle 9i	A client/server DBMS. It features full XML database functionality. In addition, the built-in OLAP functionality has been expanded, and significant enhancements have been made for Windows and Linux operating systems.
	IBM DB2	As the foundation for e-business, IBM DB2 is the industry's first multiplatform multimedia, Web-ready relational database management system.
OLAP	Microsoft OLAP	A service provided by Microsoft SQL server.
	Oracle Discoverer	The OLAP solution from Oracle. It provides functions including query, report, search, and Web publication.
Data warehouses	SAP	Includes many predefined analysis-models, uses Microsoft Excel and Web browsers as reporting tools, allows easy drag and drop construction of new queries, and includes geographical information system.
	SAS	SAS data warehouses enhance quality extraction and transformation and loading processes with value-added data quality technologies.

AVAILABLE SOFTWARE

In this section, we list some available software tools, categorized into Text Editors, Spread Sheet Programs, Databases, OLAP tools and Data Warehouses. They are summarized in Table 15.2. For detailed information regarding the software tools, readers are encouraged to visit the associated commercial WWW site.

SUMMARY

Data mining is a rapidly emerging field capable of analyzing vast amounts of data gathered from applications such as manufacturing, bioinformatics, and geological information systems. In data mining applications it is hard to overestimate the value of data storage and management.

A well-designed data storage and management architecture is the foundation for efficiently extracting useful information from stored data. This chapter summarizes several data storage and management files/systems. Recent advancements such as OLAP, data warehousing, and distributed databases are then discussed. Finally, some available software tools are listed and summarized.

ACKNOWLEDGMENTS

Special thanks to Toni Farley for her detailed reviews and assistance in editing this chapter.

REFERENCES

- Berson, A., & Smith, S. J. (1997). *Data warehousing, data mining, and OLAP*. New York: McGraw-Hill.
- Carbone, P. L. (1997). Data mining: Knowledge discovery in data bases. In B. Thuraisingham (Ed.), *Handbook of data management* (pp. 611–624). Boca Raton, FL: CRC Press.
- Codd, E. F. (1970). A relational model for large shared data banks. *Communication Association for Computing Machinery*, 13, 377–387.
- Elmagarmid, A., Rusinkiewicz, M., & Sheth, A. (1999). *Management of heterogeneous and autonomous database systems*. San Francisco: Morgan Kaufmann Publishers.
- Henderson, D., & Chervinsky, H. (1998). OLAP tools: A market overview. In B. Thuraisingham (Ed.), *Handbook of data management* (pp. 599–608). Boca Raton, FL: CRC Press.
- Hogan, R. (1990). *A practical guide to data base design*. Englewood Cliffs, NJ: Prentice Hall.
- Mattessich, R. (1961). Budgeting models and system simulation. *Accounting Review*, July, 384–397.
- Thuraisingham, B. M. (1998). *Data management systems: Evolution and interoperation*. Boca Raton, FL: CRC Press.
- Zagelow, G. (1997). Data warehousing—client/server for the rest of the decade. In R. C. Barquin & H. A. Edelstain (Eds.), *Building, using, and managing the data warehouse* (pp. 3–23). Upper Saddle River, NJ: Prentice Hall.

16

Feature Extraction, Selection, and Construction

Huan Liu and Lei Yu
Arizona State University

Hiroshi Motoda
Osaka University

Introduction	410
Feature Extraction	411
Concepts	411
Algorithms	412
An Example	413
Summary	413
Feature Selection	414
Concepts	414
Algorithm	415
An Example	416
Summary	417
Feature Construction	417
Concepts	417
Algorithms and Examples	418
Summary	419
Some Applications	420
Summary	421
References	422

This chapter is based in part on work supported by the National Science Foundation under grant no. IIS-0127815 for H. Liu.

INTRODUCTION

Researchers and practitioners realize that to effectively use data mining tools introduced in other chapters of this handbook, data should be preprocessed before it is presented to any learning, discovering, or visualizing algorithms (Liu & Motoda, 1998a; Han & Kamber, 2001). In many discovery applications, such as marketing data analysis and customer relationship management (Ortega, 2000), a key operation is to find subsets of the population that behave sufficiently similarly to be worthy of focused analysis. This kind of task necessarily has become important and pertinent in many real-world data mining applications as data are seldom collected for the purpose of data mining. It is often the case that one faces a plethora of data and wishes to take advantage of the data for some application. Therefore, data reduction is frequently a necessary step. Feature extraction, selection, and construction are effective approaches to data reduction among others such as instance selection as discussed in Liu and Motoda (2001) and data selection as covered in Liu, Lu, and Yao (2001). Feature extraction is a process that extracts a set of new features from the original features through some functional mapping. Feature selection is different from feature extraction in that no new features will be generated. It is a process that chooses a subset of features from the original set of features. Feature construction is a process that discovers missing information about the relationships between features and augments the space of features by inferring or creating additional features. The goal of feature extraction, selection, and construction is threefold:

- reducing the amount of data;
- focusing on the relevant data; and
- improving the quality of data and hence the performance of data mining algorithms, such as learning time and predictive accuracy.

There could be two main approaches. One is to rely on data mining algorithms, and the other is to conduct preprocessing before data mining. It seems natural to let data mining algorithms deal with data directly, as the ultimate goal of data mining is to find hidden patterns from data. Indeed, many data mining methods attempt to select, extract, or construct features; however, both theoretical analyses and experimental studies indicate that many algorithms scale poorly in domains with large numbers of irrelevant and/or redundant features. In other words, when data is massive, the very first problem data mining algorithms encounter is how to process the huge amount of data to mine nuggets from data. That is, it is about the scaling up of algorithms, and it is an extremely important research issue itself. All the evidence suggests the need for alternatives to overcoming the difficulties caused by large amounts of data.

The other approach is to preprocess the data so that it is made suitable for data mining. Feature extraction, selection, and construction are tasks of preprocessing and are independent of data mining. There are several reasons for this approach. First, it can be done once and used for all subsequent data mining tasks. Second, it usually employs a less expensive evaluation measure than a data mining algorithm. Hence, it can handle larger amounts of data than data mining can. Third, it often works offline. Therefore, if necessary, many different algorithms can be tried.

Feature extraction, selection, and construction can be considered three independent issues. Each is extremely difficult on its own. However, in addition to their being mainly preprocessing tasks, there are other commonalities among them: (a) they all try to achieve the same goal as stated earlier for data reduction, (b) they require some criteria to ensure that the resulting data allow the mining algorithm to accomplish as much or more, and (c) their effectiveness has to be measured in multiple aspects such as reduced amounts of data; relevance of the reduced data; and, if possible, their direct impact on data mining algorithms.

Feature extraction, selection, and construction can be used in combination. In many cases feature construction expands the number of features with newly constructed ones that are more expressive but that might include redundant features. Feature selection can help automatically reduce those excessive features. The following are possible combinations: feature selection followed by feature extraction, feature construction followed by feature selection. Exactly how they should be used depends on many factors in an application, such as expected results from data mining, data mining algorithm used and so forth.

We present feature extraction, selection, and construction independently in the next three sections. For each we first introduce basic concepts; second, discuss some representative algorithms; third, illustrate one algorithm using a simple data set to demonstrate the input and output of each method, followed by a summary. In the fifth section we present some applications of using feature extraction, selection, and construction. We conclude this chapter with some possible further research.

FEATURE EXTRACTION

Concepts

Feature extraction is a process that extracts a set of new features from the original features through some functional mapping (Wyse, Dubes, & Jain, 1980). Assuming there are n features (or attributes) A_1, A_2, \dots, A_n , after feature extraction, we have a new set of features B_1, B_2, \dots, B_m ($m < n$), $B_i = F_i(A_1, A_2, \dots, A_n)$, and F_i is a mapping function. For instance, one mapping can be $B_1 = c_1A_1 + c_2A_2$, where c_1 and c_2 are coefficients. Intensive search generally is required in finding good transformations. The goal of feature extraction is to search for a minimum set of new features via some *transformation* according to some *performance measure*. The major research issues, therefore, can be summarized as follows.

The performance measure investigates what is most suitable in evaluating extracted features. The crux of performance evaluation is to make sure that extracted features preserve certain aspects of the original data after the transformation. Hence, which measure should be chosen is to some extent determined by the application that requires feature extraction. Common data mining tasks such as clustering or classification can impose vastly different constraints on determining the performance measure. For classification the data has class labels, and predictive accuracy on the training set may be used as a performance measure. For clustering, the data does not have class labels, and one has to resort to other measures such as intercluster/intracluster similarity, variance among data, and so forth.

Transformation studies ways of mapping original features to new features. The aim of transformation is to find a way to represent the original data in a more concise manner. In the context of data mining, transformation can be defined as fewer new features than original features or as some new features that can be easily visualized and manipulated. Different mappings can be employed to extract features. In general the mappings can be categorized into linear or nonlinear transformations. Some transformations can only be applied to certain types of data. One often encountered case is whether data is labeled. Thus, one could categorize transformations along two dimensions: linear and labeled, linear and nonlabeled, nonlinear and labeled, nonlinear and nonlabeled. Many data mining techniques can be used in transformation. For example, EM, k -means, and k -medoids can be used for nonlabeled data, and multilayer perceptrons can be used for labeled nonlinear data.

Number of new features surveys methods that determine the minimum number of new features. This is seemingly an easy problem to be solved. As in data visualization, the most

intuitive solution is to plot a three-dimensional model for the data. However, when the original dimensionality of data is large (greater than 20), it is seldom possible to have a comprehensive view of data in a three-dimensional model. With our objective to create a minimum set of new features, the real question here is how many new features can ensure that “the true nature” of the data remains after transformation. Some common approaches to this problem are (a) thresholding, in which one subjectively determines a threshold on some variable used in the performance measure based on past experience, which in turn decides the number of new features; or (b) automatically determining the number of features based on some objective measures such as predictive accuracy.

One can take advantage of data characteristics as a critical constraint in selecting performance measure, number of new features, and transformation. In addition to with/without class labels, data features can be of various types: continuous, nominal, binary, mixed.

Feature extraction may have many uses: dimensionality reduction for further processing (Liu & Motoda, 1998a), visualization (Fayyad, Grinstein, & Wierse, 2001), compound features used to booster some data mining algorithms (Liu & Setiono, 1998). However, feature extraction does come with some costs. The most salient are the following:

1. It is time-consuming in searching for new features that satisfy performance criteria. This shortcoming can often be tolerated because feature extraction is performed only periodically to obtain the mapping. The most frequent part in data mining is to apply the mapping to the data and then use the new features to discover valuable patterns.
2. Original features have to be kept. In other words, feature extraction does not reduce the number of features in the data source. This would be an unwarranted characteristic if data collection is costly. In the next section we will discuss feature selection that can reduce dimensionality in the original data.

Algorithms

Functional mapping can be realized in several ways. We present here two exemplar algorithms to illustrate how they treat different aspects of feature extraction. A simple example with some data is given in the next subsection.

Feedforward neural networks (presented in chap. 3) can be used to extract new features (Setiono & Liu, 1998). In particular, a multilayer perceptron with one single hidden layer is adopted for feature extraction. The basic idea is to use the hidden units as newly extracted features. Let us examine how the three major issues of feature extraction are handled here. The first issue is how to evaluate the performance of new features. Predictive accuracy is estimated and used as the performance measure. This entails that data should be labeled with classes. We choose the extracted features that result in the best predictive accuracy. The second issue is the mapping from original features to new ones. In this context it is the nonlinear transformation from input units to hidden units. The third issue is how to determine the number of new features. Clearly, the last two issues are closely associated with the topology of the neural networks. Two algorithms are designed to construct a network with the minimum number of hidden units (i.e., minimum number of new features) and the minimum of connections between the input and hidden layers: the network construction algorithm parsimoniously adds one more hidden unit to improve predictive accuracy; the network pruning algorithm generously removes redundant connections between the input and hidden layers if predictive accuracy does not deteriorate.

Principal component analysis (PCA) is a classic technique in which the original n features are replaced by another set of m new features that are formed from linear combinations

TABLE 16.1
Iris Data: Covariance Matrix, Ordered Eigenvalues, and Their Proportions

<i>Covariance Matrix</i>			<i>Eigenvalue</i>	<i>Proportion</i>
1.0000	-.1094	0.8718	0.8180	2.9108
-.1094	1.0000	-.4205	-.3565	0.9212
0.8718	-.4205	1.0000	0.9628	0.1474
0.8180	-.3565	0.9628	1.0000	0.0206
				0.0052

of the original features. Let us look at how PCA is used to determine performance measure, transformation, and a number of new features. The basic idea is straightforward: to form an m -dimensional projection ($1 \leq m \leq n - 1$) by those linear combinations that maximize the sample variance subject to being uncorrelated with all these already selected linear combinations. The aim here is to capture the intrinsic variability in the data through linear mapping from original features to newly extracted ones. Specifically, performance measure is sample variance; the number of new features, m , is determined by the m principal components that capture the amount of variance subject to a predetermined threshold; the transformation is linear combination. PCA does not require that data be labeled with classes. The search for m principal components can be rephrased to finding m eigenvectors associated with the m largest eigenvalues of the covariance matrix of a data set (Hand, Mannila, Smyth, & Uthurusamy, 2001). We now illustrate the working of PCA using the frequently cited data set, Iris.

An Example

The Iris data is a 150×4 data matrix in which there are 150 rows or instances and 4 continuous features. Its covariance matrix is 4×4 (shown in Table 16.1) after each feature's value is normalized into a range of $[0, 1]$.

The four eigenvalues are ordered in a descending order shown in Table 16.1. When we calculate the proportion of the m largest eigenvalues over all n eigenvalues, $r = \sum_{i=1}^m \lambda_i / \sum_{i=1}^n \lambda_i$, the sum of the first two is 0.95801; that is, over 95% of the variance is captured by the first two principal components. The corresponding two eigenvectors can then be used in transforming the original data of four dimensions into the new data of two features: Let M be a 4×2 matrix that consists of the two eigenvectors; D is the original Iris data; the new data $D' = DM$, is two-dimensional data. More details and comparisons of this approach with other approaches can be found in (Liu and Motoda 1998b).

Summary

Many feature extraction algorithms can be categorized in terms of linear or nonlinear transformation as well as data labeled with or without classes. We present two algorithms that can be used in feature extraction. One is the feedforward multilayer neural network approach that employs class information of the data and generates a nonlinear mapping between original data and new data. The other is the classic PCA that does not require class information and attempts to capture the predetermined amount of sample variance in data with the minimum number of principal components. An example of this approach is given. The two approaches apply different performance measures (predictive accuracy versus sample variance).

FEATURE SELECTION

Concepts

Feature selection is different from feature extraction in that no new features are generated. It is a process that chooses a subset of M features from the original set of N features ($M \leq N$), so that the feature space is optimally reduced according to a certain criterion (Blum & Langley, 1997). The role of feature selection in machine learning is (a) to reduce the dimensionality of the feature space, (b) to speed up a learning algorithm, (c) to improve the predictive accuracy of a classification algorithm, and (d) to improve the comprehensibility of the learning results. Recent studies of feature selection in an unsupervised learning context shows that feature selection can also help to improve the performance of clustering algorithms with reduced feature space (Dash & Liu, 2000; Talavera, 1999). In general, feature selection is a search problem according to some evaluation criterion. A typical feature selection method consists of four basic steps (shown in Fig. 16.1): subset generation, subset evaluation, stopping criterion, and result validation (Dash & Liu, 1997).

Subset generation is a search procedure. Basically, it generates subsets of features for evaluation. Let N denote the number of features in the original data set, then the total number of candidate subsets is 2^N , which makes an exhaustive search through the feature space infeasible with even moderate N . We discuss some more practical subset generation procedures in the following subsection. Each subset generated by the generation procedure needs to be evaluated by a certain evaluation criterion and compared with the previous best one with respect to this criterion. If it is found to be better, then it replaces the previous best subset. We discuss different evaluation criteria in a later subsection. Without a suitable stopping criterion the feature selection process may run exhaustively before it stops. A feature selection process may stop under one of the following reasonable criteria: (a) a predefined number of features are selected, (b) a predefined number of iterations are reached, (c) when addition (or deletion) of any feature fails to produce a better subset, (d) an optimal subset according to the evaluation criterion is obtained. The selected best feature subset needs to be validated by carrying out different tests on both the selected subset and the original set and comparing the results using artificial data sets and/or real-world data sets. We see a simple example in a later section.

Subset Generation. First, one must decide the starting point in the search space. One direction is to grow a feature subset from an empty set (i.e., forward generation). As the search starts, features are added one at a time. At each iteration the best feature among unselected ones is chosen based on the evaluation criterion. The subset grows until it reaches a full set of original features or the stopping criterion is reached. An opposite direction is to start from the full set (i.e., backward generation). At each iteration the least important feature is removed based on the evaluation criterion. The subset shrinks until there is only one feature in

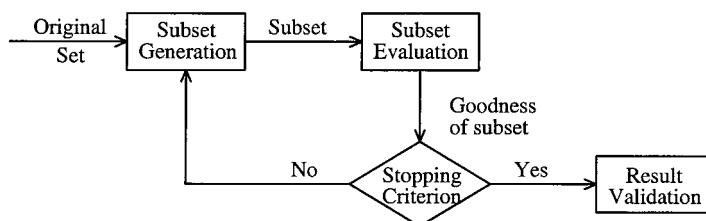


FIG. 16.1. The typical four steps of feature selection.

the set or the stopping criterion is reached. Another way is to start the search from a randomly selected subset (i.e., random generation) and add or delete a feature at random. There is no particular direction to follow, which tries to avoid being trapped into local optima.

Second, one must decide the search strategy. As we mentioned earlier, an exhaustive search of the space to find the optimal subset is impractical. There are three different strategies to solve this problem: complete, heuristic, and nondeterministic. Complete search arises from the answer to the question “Do we have to resort to an exhaustive search to guarantee the optimality of the resulted subset?” In some cases being complete (i.e., no optimal subset is missed) does not necessarily mean that the search must be exhaustive (i.e., every subset has to be evaluated). If the evaluation criterion possesses some certain property (e.g., monotonicity), we can find an optimal subset without evaluating all of the 2^N subsets. However, this improvement is not enough, because the search space is still in the order of $O(2^N)$. Heuristic search, as the name suggests, employs heuristics in conducting the search. It avoids being complete, but at the same time risks losing optimal subsets. It is sort of a depth-first search guided by heuristics. The space complexity could be $O(N^2)$ or less. Nondeterministic search, unlike the first two types of strategies, searches for the next set at random (i.e., a current set does not directly grow or shrink from any previous set following a deterministic rule). The search generates current-best subsets constantly and keep improving the quality of selected subsets as the algorithm runs.

Evaluation Criteria. An optimal subset is always relative to a certain evaluation criterion (i.e., an optimal subset chosen using one evaluation criterion may not be the same as that using another evaluation criterion). Evaluation criteria can be categorized broadly into two groups based on their dependence on the learning algorithm applied on the selected feature subset. Typically, an independent criterion (i.e., filter) tries to evaluate the goodness of a feature or feature subset without the involvement of a learning algorithm in this process. Some of the independent criteria are distance measure, information measure, dependency measure, and consistency measure. A dependent criterion (i.e., wrapper) tries to evaluate the goodness of a feature or feature subset by evaluating the performance of the learning algorithm applied on the selected subset. In other words, it is the same measure on the performance of the applied learning algorithm. For supervised learning the primary goal of classification is to maximize predictive accuracy; therefore, predictive accuracy is generally accepted and widely used as the primary measure by researchers and practitioners. For unsupervised learning there exist a number of heuristic criteria for estimating the quality of clustering results, such as cluster compactness, scatter separability, and maximum likelihood. Recent reviews on developing dependent evaluation criteria for unsupervised feature selection based on these criteria can be found in Dy and Brodley 2000 and Kim, Street, and Menczer (2000).

Algorithm

In the previous section we introduced a general picture of feature selection with a relatively detailed study of various subset generation strategies and evaluation criteria. Various combinations of generation strategies and evaluation criteria result in different feature selection algorithms. In this section we illustrate one variation by introducing a popular feature selection algorithm, LVF, a Las Vegas algorithm for filter feature selection.

LVF starts with a randomly selected subset and employs a nondeterministic search strategy to randomly search the space of subsets. It uses a consistency measure, which attempts to find a minimum number of features that separate classes as consistently as the full set of features can. The inconsistency rate is calculated as follows.

1. Two instances are considered inconsistent if they are the same except for their class labels (we call these “matching instances”). For example, for two instances (0 1 1) and (0 1 0), their values of the first two features are the same (0 1), but their class labels are different (1 and 0).

2. The inconsistency count is the number of all the matching instances minus the largest number of instances of different class labels. For example, in n matching instances, c_1 instances belong to $label_1$, c_2 to $label_2$, and c_3 to $label_3$ where $c_1 + c_2 + c_3 = n$. If c_3 is the largest among the three, the inconsistency count is $(n - c_3)$.

3. The inconsistency rate is the sum of all the inconsistency counts divided by the total number of instances (N). An inconsistency threshold γ is fixed in the beginning of the algorithm, and any candidate subset, having an inconsistency rate greater than γ , is rejected. The algorithm uses another predefined parameter, $MAX\text{-TRIES}$, which is the maximum number of subsets randomly generated, as a stopping criterion. Let D denote a data set with N features. LVF can be illustrated as follows.

$LVF(D, N, \gamma, MAX\text{-TRIES})$

```

initialize: list  $L$       /* $L$  stores equally good sets*/
 $C_{best} = N$ 
for  $MAX\text{-TRIES}$  loops
begin
     $S = randomSet(seed)$ 
     $C = numberOfFeatures(S)$ 
    if ( $C < C_{best} \wedge CallIncon(S, D) \leq \gamma$ )
         $S_{best} = S$ 
         $C_{best} = C$ 
         $L = S$           /* $L$  is reinitialized*/
    else if ( $C = C_{best} \wedge CallIncon(S, D) \leq \gamma$ )
         $L = append(S, L)$ 
    end
    return  $L$       /* all equally good sets found by LVF*/

```

The LVF algorithm generates a random subset S from N features in every round. If the number of features (C) is less than the current best, that is, $C < C_{best}$, the data D with the features prescribed in S is checked against the consistency criterion described earlier. If its inconsistency rate is below the predefined threshold γ , C_{best} and S_{best} are replaced by C and S , respectively; the list L is replaced by the new current best set S . If $C = C_{best}$ and the consistency criterion is satisfied, an equally good current best is appended to the list L . When LVF loops $MAX\text{-TRIES}$ times, it stops and returns the final list L , which stores all equally good sets found by LVF.

An Example

We now show the effectiveness of feature selection by validating the result of LVF on a synthetic data set, CorrAL, designed in John, Kohavi, and Pfleger (1994). This data set has 64 instances in total with six Boolean features (A_0, A_1, B_0, B_1, I, C), where feature I is irrelevant, feature C is correlated to the class label 75% of the time, and the other four features in combination define the target concept: $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$. Running classification algorithm C4.5 (see chap. 1) directly on the original data set chooses feature C as the root, with 54 (84.38%) of all the instances correctly classified. Running LVF on this data set generates

a best subset (A_0, A_1, B_0, B_1). After feature I and C are removed, a more accurate tree will result from C4.5, with A_0 chosen as the root and all the 64 instances correctly classified.

One thing worthy of being pointed out is that the CorrAL data set described above contains features with only discrete values. LVF cannot handle data sets containing features with continuous values properly because the evaluation criterion assumes the values of each feature to be discrete in calculating the inconsistency rate. One way of solving this problem is to apply feature discretization methods (Liu & Setiono, 1997) to discretize the values of continuous features into a small number of intervals, where each interval is mapped to a discrete symbol, before applying LVF.

Summary

There exist many different feature selection algorithms, each of which has its own strengths and limits; therefore, how to correctly choose a feature selection algorithm for a specific problem remains an important and challenging issue in this research area.

FEATURE CONSTRUCTION

Concepts

Feature construction is a process that discovers missing information about the relationships between features and augments the space of features by inferring or creating additional features (Liu & Motoda, 1998). Assuming there are n original features A_1, A_2, \dots, A_n , after feature construction, we may have the additional m features $A_{n+1}, A_{n+2}, \dots, A_{n+m}$. For example, a new feature A_k ($n < k \leq n + m$) could be constructed by performing a logical operation on A_i and A_j from the original set. Another example: a two-dimensional problem (say, $A_1 = \text{width}$ and $A_2 = \text{length}$) may be transformed into a one-dimensional problem ($B_1 = \text{area}$) after B_1 is constructed.

All newly constructed features are defined in terms of original features; as such, no inherently new information is added through feature construction. Feature construction attempts to increase the expressive power of the original features. Usually, the dimensionality of the new feature set is expanded and is bigger than that of the original feature set. As its immediate result, it may not directly reduce the number of features. However, many features are rendered redundant after new features are constructed. Intuitively, there could be exponentially many combinations of original features in search for new constructed features, and not all combinations are necessary and useful. Constructing features manually has been shown to be difficult. Feature construction aims to automatically transform the original representation space to a new one that can better achieve data mining objectives: improved accuracy, easy comprehensibility, truthful clusters, revealing hidden patterns, and so forth.

The major research issues of feature construction are as follows.

1. How to construct new features. There are various approaches to feature construction. They can be categorized into four groups: data driven, hypothesis driven, knowledge based, and hybrid (Wnek & Michalski, 1994). The data driven approach is used to construct new features based on analysis of the available data by applying various operators. The hypothesis driven approach is used to construct new features based on the hypotheses generated previously. Useful concepts in the induced hypotheses (e.g., rules) can be extracted and used to define new

features. The knowledge based approach is used to construct new features applying existing knowledge, and domain knowledge, which is particularly helpful in determining the type of new compound features and choosing suitable operators.

2. How to choose and design operators for feature construction. There are exponentially many operators for combining features to form compound features. Data type plays an important role in determining each operator's suitability. Conjunction, disjunction, and negation are commonly used constructive operators for nominal features. Other common operators are Cartesian product (Pazzani, 1998) of two or more nominal features, *M-of-N* and *X-of-N* (Zheng, 1998), where *M*, *N* are constants and *M-of-N* is true iff at least *M* out of *N* conditions are true, and *X-of-N* is true iff *X* of *N* conditions are true where *X* is a variable and *M* is a given constant. For numerical features, simple algebraic operators such as equivalence, inequality, addition, subtraction, multiplication, division, maximum, minimum, and average often are used to construct compound features.

3. How to use operators to construct new features efficiently. It is clear that too many operators can be used in feature construction. It is impossible to explore every possible operator. The problem is further complicated by the fact that we could combine any number of features in search of compound features, which leads to combinatorial explosion. It is imperative to find intelligent methods that can avoid exhaustive search of all operators and heuristically try potentially useful operators. This line of research investigates the connections between data mining tasks, data characteristics, and operators that could be effective. Examples in Liu and Motoda 1998 show that one can employ greedy search of different constructed features for decision tree learning or exploit the association of fragmentary domain knowledge to feature construction.

4. How to measure and select useful new features. Not all constructed features are good ones. The danger of including all features (both original and constructed) is to artificially increase the dimensionality—the curse of dimensionality. We have to be selective. One option is to handle the selection part by applying feature selection as discussed in the previous section. Basically, we can construct the new features first, unite them with the original features, and then remove redundant and irrelevant features. When the number of features is very large, it is sensible to keep only those features that are potentially useful, making decisions while a new compound feature is generated to avoid too many features. This would require an effective measure that can evaluate a new feature and provide an indicator. The choice of this measure can hardly be independent of data mining tasks. Doing so, however, slows down the feature construction process because it is time-consuming to involve data mining in this process that requires evaluation of every single new feature. Therefore, researchers are investigating various measures that are not computationally expensive. Some examples are measures of consistency and distance as discussed in the previous section on feature selection. In the following, we show some successful examples of feature construction.

Algorithms and Examples

Genetic algorithms are adaptive search techniques. They commonly maintain a constant-sized population of individuals that represent samples of the space to be searched. Each individual is evaluated based on its overall fitness with respect to the given application domain. New individuals are constructed by selecting individuals to produce the next generation that preserve many of the characteristics of their parents. The process results in an evolving population that has improved fitness (Vafaie & De Jong, 1998). The two main genetic operators often used to create the next generation are crossover and mutation. In the context of feature

construction one can use these operators to create new generations of individuals, that is, new features. The most natural way to construct useful new features is to form combinations of existing features via a well-chosen set of operators. Specially designed representation is needed to allow for more traditional constructive operators. For continuous features we can use a set of arithmetic operators such as $+$, $-$, $*$, $/$. To evolve good features an individual is a variable-length structure representing a set of original and compound features. If there are four features (f_1, f_2, f_3, f_4) , a possible feature set can be $(f_1 * f_2, f_3 - f_4, f_1 + f_3, f_2)$, in which we have still four features, but three of them are compound. Genetic algorithms avoid exhaustive search by relying on random mechanisms to generate new generations of features, the fittest individuals (both new and original features) will be selected for the next round of generation and selection. Fitness evaluation is performed frequently, so an efficient measure is required. For classification, for example, predictive accuracy can be part of a fitness measure.

Using domain knowledge to construct new features is often recommended because one can thus quickly rule out many impossible combinations and determine an effective measure to evaluate new compound features. The caveat is that any bias may lead to an impasse. For the example of Iris data we used earlier, one can try to construct two new compound features $(i_1 * i_2, i_3 * i_4)$ without using the single features (i_1, i_2, i_3, i_4) . That is possible because they form two approximate area measures for sepal and petal. Obviously, other combinations would likely generate unusable compound features. Another useful constructive operate is $\text{Count}(S, C)$ as suggested in Bloedorn and Michalski (1998). It counts how many features in set S satisfy condition C . For the second Monk's problem an instance is a monk (the concept to be learned) if exactly two of six features take their first value ($\text{Count}(A_1, \dots, A_6, \text{FirstValue}) = 2$). This compound feature basically reduces the data mining problem to a search of compound features. Use of domain knowledge does not necessarily mean abandoning the search. Search is still required, but just becomes more focused. Going back to the example of $(\text{Count}(A_1, \dots, A_6, \text{FirstValue}) = 2)$, the search could still be extensive: At a minimum, condition C can vary from FirstValue to LastValue , and set S can have as many variations as $\sum_{i=1}^{n=6} \binom{n}{i}$.

Combining feature construction and selection often is needed. For example, the Iris data can be described by one compound feature of petal for a classification task; it is not necessary to keep any individual features and the other compound feature of sepal. In this case the data becomes one-column and can be modeled by a piecewise linear model. Compound features can help mitigate the problems of replication and fragmentation suffered by selective induction algorithms that adopt the divide-and-conquer strategy (Setiono & Liu, 1998). Whereas feature construction generates compound features to increase the representational power of the data, feature selection streamlines the representation of the data and simplifies its description.

Summary

Feature construction is itself an inductive learning task. If performed properly, it can significantly simplify the description of the data. In this section we define the problem of feature construction, outline its research issues, and present some examples to illustrate the aspects of feature construction. Some more elaborate examples can be found in Liu and Motoda (1998a). In essence, feature construction requires a set of relevant operators, an effective way to combine features in search of useful compound features, and a fast goodness measure to evaluate each newly constructed feature. Domain knowledge is very important in determining the set of operators and search strategies.

SOME APPLICATIONS

In this section we briefly discuss some applications of using feature extraction, selection, and construction. The essence of these applications lies at the recognition of a necessity of data preprocessing: Data mining can be effectively accomplished with the aid of feature transformation. This is because on one hand, data is usually collected for many reasons other than data mining (required by law, easy to collect, or simply for bookkeeping); on the other hand, data mining algorithms have constraints to be satisfied so that they work as well as they are designed to. In real-world applications one often encounters problems such as having too many features to focus, individual features that are unable to capture significant characteristics of data independently, high dependency among the individual features, and emergent behaviors of combined features.

Feature selection has had many successes in real-world applications because it can often notably reduce dimensionality to enable many proven data mining algorithms to work on data with large dimensionality. In Ng and Liu (2000) a case of feature selection is presented for customer relationship management. In the context that each customer means a big revenue and the loss of one will likely trigger a significant segment to defect, it is imperative to have a team of highly experienced experts monitor each customer's intention and movement based on massively collected data. A set of key indicators are used by the team and proven useful in predicting potential defectors. The problem is that it is difficult to find new indicators describing the dynamically changing business environment because the machine recorded data is simply too enormous for any human expert to browse and obtain any insight from it. Feature selection is employed to search for possible new indicators that are presented to experts for scrutiny. This approach considerably improves the team's efficiency in finding changing indicators.

Feature extraction also has been used extensively in reducing the dimensionality and enhancing the interpretability of the data. In Mallet, de Vel, and Coomans (1998) discrete wavelets transformation is used in extracting the important local features in spectra data sampled by a spectrometer. Spectroscopy provides an efficient and nondestructive procedure for analyzing substances, which is of great benefit to research and quality control procedures in industry. Spectrometers measure the change in reflected or absorbed radiation that has been directed at some substance for hundreds of wavelengths, usually at regular intervals, where each wavelength can be considered equivalent to a variable or attribute. Such data are complex and extremely multivariate, consisting of many highly correlated variables or features. Wavelets transformation reduces the number of variables while at the same time retaining as much information as possible and facilitating the automated analysis and interpretation of spectra. The work reported in Mallet, de Vel, and Coomans (1998) adopts a wrapper approach in which an optimal subset of wavelet coefficients is selected such that it minimizes the prediction error of a regression model. The spectral data analyzed in the work are of importance to the agricultural, pharmaceutical, and mining industries as well as the environmental sciences.

Feature construction is not as well-established a technique as the other two, and much of the work is still in exploratory research. In Donoho and Rendell (1998) fragmentary knowledge is used to guide feature construction by providing prior evidence for or against groups of constructed features in bankruptcy prediction problems. The fragmentary knowledge used is dimensional analysis (the unit of a new feature must make sense), correlation knowledge (features are positively or negatively correlated to the target class), and normalization knowledge (values of some features make sense relative to the values of some other features). The accuracy obtained using the automatically constructed features is much higher than using the original features using any induction techniques and is comparable to that of the set of popular financial ratios. The genetic algorithm described earlier has been used in combination with

feature selection for image understanding, specifically eye detection (Vafaie & De Jong, 1998). Extracting a good set of features from raw images is extremely difficult. Most commonly used raw features are pixel intensities, their variances and entropies within overlapping windows. In their application this gives 105 features. Use of these original features gives a very poor performance. Using a set of arithmetic operators as constructing operators, the genetic algorithm constructs many new features, some of which are not informative. Feature selection using again the same genetic algorithm prunes these useless features. Repeated use of feature construction and selection several times results in 10 informative features with a great improvement of eye detection accuracy.

SUMMARY

This chapter presented the concepts and the representative algorithms of feature selection, extraction, and construction and gave illustrative examples and some applications. Feature extraction and construction are the variants of feature transformation through which a new set of features is created. Feature construction often expands the feature space, whereas feature extraction usually reduces the feature space. Feature transformation and feature selection are not two totally independent issues. They can be viewed as two sides of the representation problem. We can consider features as a representation language. Sometimes when this language contains more features than necessary, feature selection helps simplify the language; when this language is not sufficient to describe the problem, feature construction helps enrich the language by constructing compound features. It is common that some constructed features are not useful at all. Feature selection can then remove these useless features. The use of feature selection, extraction, and construction depends on the purpose—for simpler concept description or for better data mining task performance.

Despite recent advances in feature selection, extraction, and construction, much work is needed to unify this currently still diversified field. Many types of data exist in practice. Boolean, nominal, and numerical types are popular, but others like structural, relational, and temporal should also receive our equal attention in data mining applications of real-world problems. We need to study the links between various techniques and data types (Dash, Liu, & Motoda, 2000) in face of an ever increasing number of choices of transformation and selection methods. A contingency theory would allow a naive user to make best use of the techniques available. Most efforts have been directed toward improving performance, such as estimated classification accuracy. In the case of data mining, however, we also need to pay attention to issues such as comprehensibility of newly extracted or constructed features. People want to know not only whether the data contains valuable information, but also what the information is through discovered rules and features. Different kinds of knowledge, from complete to fragmentary knowledge, can all contribute to feature selection, extraction, and construction. To make the best use of knowledge, we need to handle conflicts arising from using knowledge of different sources and to balance domain-specific knowledge and domain-independent bias. Many unsolved problems still await intelligent solutions. Little work has been devoted to studying how human experts come up with good representations. We know little about how human experts relate primitive features to higher-level abstract intermediate concepts in real-world domains. It would be very useful if we could know more about involving human efforts in the process of developing good representations.

Feature selection, extraction, and construction are key techniques in answering the pressing needs for data mining. These techniques can help reduce data for mining or learning tasks and enable those mining algorithms that were unable to mine.

REFERENCES

- Bloedorn, E., & Michalski, R. (1998). Data-driven constructive induction: A methodology and its Applications. In H. Liu & Motoda (Eds.), *Feature extraction, construction, and selection: A data mining perspective* (pp. 51–68). Boston: Kluwer Academic Publishers.
- Blum, A., & Langley, P. (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97, 245–271.
- Dash, M., & Liu, H. (1997). Feature selection methods for classifications. *Intelligent Data Analysis: An International Journal*, 1, (3). Retrieved from <http://citeseer.nj.nec.com/dash97feature.html>
- Dash, M., & Liu, H. (2000). Feature selection for clustering. In *Proceedings of Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining*. Kyoto, Japan: Springer-Verlag.
- Dash, M., Liu, H., & Motoda, H. (2000). Consistency based feature selection. In *Proceedings of Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining*. Kyoto, Japan: Springer-Verlag.
- Donoho, S., & Rendell, L. (1998). Feature, construction using fragmentary knowledge. In H. Liu & H. Motoda (Eds.), *Feature extraction, construction, and selection: A data mining perspective* (pp. 273–288). Boston: Kluwer Academic Publishers.
- Dy, J. G., & Brodley, C. E. (2000). Feature subset selection and order identification for unsupervised learning. In *Proceedings of the Seventeenth International Conference on Machine Learning* (pp. 247–254). San Francisco: Morgan Kaufmann.
- Fayyad, U., Grinstein, G., & Wierse, A. (2001). *Information visualization in data mining and knowledge discovery*. San Francisco: Morgan Kaufmann.
- Han, J., & Kamber, M. (2001). *Data mining: Concepts and techniques*. San Francisco: Morgan Kaufmann.
- Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of Data Mining*. Cambridge, MA: MIT Press.
- John, G., Kohavi, R., & Pfleger, K. (1994). Irrelevant feature and the subset selection problem. In W. a. H. H. Cohen (Ed.), *Machine learning: Proceedings of the Eleventh International Conference* (pp. 121–129). New Brunswick, NJ: Rutgers University.
- Kim, Y., Street, W., & Menczer, F. (2000). Feature selection for unsupervised learning via evolutionary search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 365–369). New York: Association for Computing Machinery.
- Liu, H., Lu, H., & Yao, J. (July/August 2001). Toward multidatabase mining: Identifying relevant databases. *IEEE Transactions on Knowledge and Data Engineering*, 13, 541–553.
- Liu, H., & Motoda, H. (Eds.). (1998a). *Feature extraction, construction and selection: A data mining perspective*. Boston: Kluwer Academic Publishers.
- Liu, H., & Motoda, H. (1998b). *Feature selection for knowledge discovery data mining*. Boston: Kluwer Academic Publishers.
- Liu, H., & Motoda, H. (Eds.). (2001). *Instance selection and construction for data mining*. Boston: Kluwer Academic Publishers.
- Liu, H., & Setiono, R. (1997). Feature selection via discretization. *IEEE Transactions on Knowledge and Data Engineering*, 9, 642–645.
- Liu, H., & Setiono, R. (1998). Feature transformation and multivariate decision tree induction. In S. Arikawa & H. Motoda (Eds.), *First International Conference on Discovery Science* (pp. 279–290). Kyoto, Japan: Springer.
- Mallet, Y., de Vel, O., & Coomans, D. (1998). Integrated feature extraction using adaptive wavelets. In H. Liu & H. Motoda (Eds.), *Feature extraction, construction, and selection: A data mining perspective* (pp. 175–189). Boston: Kluwer Academic Publishers.
- Ng, K., & Liu, H. (2000). Customer retention via data mining. *Artificial Intelligence Review—An International Science and Engineering Journal*, 14, 569–590.
- Ortega, J. (Eds.). (2000). Issues on the application of data mining [Special issue]. *Artificial Intelligence Review—An International Science and Engineering Journal*, 14.
- Pazzani, M. (1998). Constructive induction of Cartesian product attributes. In H. Liu & H. Motoda (Eds.), *Feature extraction, construction, and selection: A data mining perspective* (pp. 341–354). Boston: Kluwer Academic Publishers.
- Setiono, R., & Liu, H. (1998). Feature extraction via neural networks. In H. Liu & H. Motoda (Eds.), *Feature extraction, construction, and selection: A data mining perspective* (pp. 191–204). Boston: Kluwer Academic Publishers.
- Talavera, L. (1999). Feature selection as a preprocessing step for hierarchical clustering. In *Proceedings of the Sixteenth International Conference on Machine Learning* (pp. 389–397). San Francisco: Morgan Kaufmann.
- Vafaie, H., & De Jong, K. (1998). Evolutionary feature space transformation. In H. Liu & H. Motoda (Eds.), *Feature extraction, construction, and selection: A data mining perspective* (pp. 307–323). Boston: Kluwer Academic Publishers.

- Witten, I., & Frank, E. (2000). *Data mining—practical machine learning tools and techniques with JAVA implementations*. San Francisco: Morgan Kaufmann.
- Wnek, J., & Michalski, R. (1994). Hypothesis-driven constructive induction in AQ17-HCI: A method and experiments. *Machine Learning*, 14, 139–168.
- Wyse, N., Dubes, R., & Jain, A. (1980). A critical evaluation of intrinsic dimensionality algorithms. In E. Gelsema & L. Kanal (Eds.), *Pattern recognition in practice* (pp. 415–425). San Francisco: Morgan Kaufmann.
- Zheng, Z. (1998). A comparison of constructing different types of new features for decision tree learning. In H. Liu & H. Motoda (Eds.), *Feature extraction, construction and selection: A data mining perspective* (pp. 239–255). Boston: Kluwer Academic Publishers.

17

Performance Analysis and Evaluation

Sholom M. Weiss and Tong Zhang
IBM T. J. Watson Research Center

Overview of Evaluation	426
Training versus Testing	426
Measuring Error	427
Error Measurement	427
Error from Regression	428
Error from Classification	429
Error from Conditional Density Estimation	429
Accuracy	430
False Positives and Negatives	430
Precision, Recall, and the F Measure	430
Sensitivity and Specificity	431
Confusion Tables	431
ROC Curves	432
Lift Curves	432
Clustering Performance: Unlabeled Data	432
Estimating Error	433
Independent Test Cases	433
Significance Testing	433
Resampling and Cross-Validation	435
Bootstrap	436
Time Series	437
Estimating Cost and Risk	437
Other Attributes of Performance	438
Training Time	438
Application Time	438
	425

Interpretability	438
Expert Evaluation	439
Field Testing	439
Cost of Obtaining Labeled Data	439
References	439

OVERVIEW OF EVALUATION

After examining the data and applying automated methods for data mining, we must carefully consider the quality of the end-product of our efforts. We might describe this next step as an evaluation of the performance of a proposed solution to the data mining task. For example, one of the primary tasks of data mining is prediction, sometimes called classification or regression. We apply automated learning methods, and we reach a proposed solution. Of obvious importance is the accuracy of that solution. Are its predictions sufficiently accurate to exceed chance? Does it achieve a level of predictive performance that makes its future application worthwhile?

Predictive performance measurement often is considered the key evaluator of the success of an application. If we are a marketing organization, and we send out a mass mailing, we are clearly interested in having the highest response rate. In that application most people do not respond, so even tiny changes in probability may be extremely valuable. In a medical application we are typically at the other end of the spectrum, where a very high accuracy is expected. The possibility of error is not the only concern. Mistakes have costs. The risk of missing the diagnosis may be higher than the cost of further testing. Each mailing has an associated cost, and these costs must be considered. For example, a credit card company may try to detect fraud by investigating anomalies in purchasing habits of customers. Too many investigations cost too much, but when the probability of fraud exceeds a certain threshold, the customer may be contacted.

Additional factors besides accuracy and risk may also be considered. Some methods may give better accuracy, yet may consume more resources. They may be computationally expensive or difficult to revise for new data.

In this chapter we consider issues related to the evaluation of data mining results. It is convenient to view data mining as a process of examining data, learning a solution, and then evaluating the proposed solution. In reality, evaluation is not necessarily distinct from proposing a model for the data. We can expect to learn from data, evaluating the result, and then directing the learning system until the “best” result has been achieved.

Training versus Testing

To perform data mining, we obtain a sample of data. The sample contains examples of past performance. If these examples contain the complete universe of possibilities, then we need only compose a table and look up the correct answers. The objective of data mining is almost always the generalization of prior experience to new examples that will arrive in the future. Thus, classification and regression are sometimes described as prediction.

Data mining can be distinguished from other statistical problems by the sheer volume of data. We can often expect that instead of working with smaller samples with little hope of obtaining additional data, we will have the luxury of getting new independent data that makes evaluation much easier. The classical technique for evaluation is to assemble a sample of cases

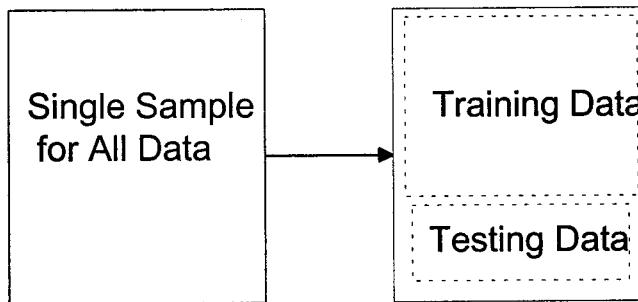


FIG. 17.1. Data organised for empirical evaluation.

that is devoted exclusively to evaluation. Thus, in Fig. 17.1 we see two views of collecting data. The first one has one sample that contains all available data. The second view has two samples, the larger one for training and the smaller one for evaluation.

Why do we need to do anything special for evaluation? Why not train on one set of data? If we rely solely on training data, it is very easy to get exceptional results. Again, we just do direct table lookup and get perfect results. However, these results would likely not generalize to new examples, which would be missing from the stored table. Researchers have developed training techniques that reduce the likelihood of “overfitting” to the training data. Still, they are subject to potential problems, whether they specialize too much and overfit the training data or attempt to ensure generality by “underfitting” and not using the training data to its full potential. A strong and effective way to evaluate results is to hide some data and then do a fair comparison of training results to unseen test results. Of course, one could wait until new data arrives during application of the solution, but it is wise to test performance prior to actual application. It prevents unexpected poor results and gives the developers time to extract the best performance from the application system.

MEASURING ERROR

For most data mining applications we are given a sample of past experience, and the objective is to project to new data. In the typical prediction application the sample is collected with known correct answers: labeled data. We try to find some general solution that uses the same type of data to give answers when the labels are not known. To estimate future performance, we need an objective measure of current performance. Unless a perfect model is found, our model will make errors when asked to make predictions. Measuring predictive performance for labeled data is well understood, and we will consider the classical frameworks for measuring current performance and projecting to future performance. It should be noted that although it is easy to measure current performance, projections to future performance often assume that the population is stable. If the original data can evolve so that the sample changes, then the task of evaluation becomes more complex and less reliable.

Consider the base situation, in which the task is to measure current performance.

Error Measurement

The goal of a prediction problem is to approximate an unknown output value y based on observed input x . If the output y corresponding to an input x is observed, we say that the

data point is labeled. For labeled data the label is a numerical value. The data for each case or example are stored, and the label represents the correct answer. During training some general solution was found that accepts a set of measurements as input, and then a prediction is made on the output. Depending on the nature of the labels we can compute a measure of predictive performance.

Formally the output of the training procedure is a functional relationship $y \approx p(x)$ that is estimated based on labeled examples. The quality of this trained predictor can often be measured by a cost function $L(p(x), y)$, and usually we want to minimize the average cost over the future data that are not labeled.

Error from Regression

In regression the label is an ordered numerical value. For example, in a basketball game we might try to predict the difference between the scores of the winning and losing teams. To measure performance in terms of error, we can measure the difference between the predicted and the true values. The mean error is then computed over all examples. Two measures naturally come to mind. The MSE (mean squared error) and the MAD (mean absolute deviation). The MSE is defined in Equation 1, and the MAD is defined in Equation 2, where x_i is the i th input, $p(x_i)$ is the prediction of example i , y_i is the true output value, and n is the number of examples. The squared error amplifies larger errors. The MAD simply takes the absolute value of the error.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (p(x_i) - y_i)^2 \quad (1)$$

$$\text{MAD} = \frac{1}{n} \sum_{i=1}^n |p(x_i) - y_i| \quad (2)$$

The definition of error is the test-set error for a particular training set. The error can also be averaged over training set X . It is often useful to decompose the overall expected MSE error as bias term and variance term. Let $p_X(x)$ be the predictor estimated from random training data X , then the error

$$\overline{\text{MSE}} = E_x E_{x,y} (p_X(x) - y)^2 = E_{x,y} (E_x p_X(x) - y)^2 + E_x E_x (p_X(x) - E_x p_X(x))^2. \quad (3)$$

$\overline{\text{MSE}}$ is the average MSE over all training data. E_X denotes mean over training data, and $E_{x,y}$ denotes mean over future data. The first term in Equation 3 is the average of the squared bias terms that describe the system error:

$$b_x(p_X) = E_x p_X(x) - y(x). \quad (4)$$

The second term is a variance term that describes the random portion of the error:

$$\text{VAR}_x(p_X) = E_x (p_X(x) - E_x p_X(x))^2. \quad (5)$$

We can thus write

$$\overline{\text{MSE}} = E_{x,y} b_x(p_X)^2 + E_x \text{VAR}_x(p_X). \quad (6)$$

To have a small average MSE, the algorithm should have a small bias and a small variance. This decomposition allows one to understand where the overall error of the mining algorithm comes from.

Error from Classification

For classification, the labels are unordered variables that are often called categorical outputs; for example, value 1 for the first class and value 2 for the second class, and so on (James, 1985). For the basketball game example, we might predict which team is a winner or loser but not the actual score differential, the spread. There is no special order for the assigned codes or label values. In this situation it is natural to compute an error rate. The predicted answer is compared with the actual label. If the two match, there is no error. If they do not match, then an error has occurred. The overall performance is measured by the number of errors divided by the number of examples. The overall error rate is given in Equation 7.

$$\text{Error rate} = \frac{\text{number of errors}}{\text{number of examples}}. \quad (7)$$

Error from Conditional Density Estimation

Some classification algorithms learn class labels, others learn probability distributions over classes. In many practical classification applications such probability information is very desirable because it can be used to indicate the confidence of classification outputs.

Assume we have m possible categorical outputs $y = 1, 2, \dots, m$. A classifier that learns class probability takes an input x and returns a probability distribution $p_1(x), \dots, p_m(x)$ over the m -classes: $p_j \geq 0$ and $\sum_{j=1}^m p_j = 1$. Many statistical classification methods, such as decision trees and logistic regression, can provide such information.

A frequently used method to check how well the predicted probability output fit the data is to compute the overall probability of output values based on the input over independent test data. That is, $\prod_{i=1}^n p_{y_i}(x_i)$. The larger the probability, the better the predictor. Equivalently, one can use the average negative log-likelihood cost defined as:

$$\frac{1}{n} \sum_{i=1}^n -\ln(p_{y_i}(x_i)). \quad (8)$$

A better model has a smaller cost.

If true probability distribution is $q(x)$, then the expected negative log-likelihood over future data is

$$E_x \sum_{j=1}^m q_j(x) \ln \frac{1}{p_j(x)}. \quad (9)$$

The choice of p_j that minimizes this cost function is $p_j(x) = q_j(x)$. Subtracting the cost at the optimal value, a smaller average negative log-likelihood cost corresponds to a smaller average Kullback-Leibler (KL) distance between the true model $q(x)$ and the computed model $p(x)$:

$$E_x \text{KL}(q(x) || p(x)) = E_x \sum_{j=1}^m q_j(x) \ln \frac{q_j(x)}{p_j(x)}. \quad (10)$$

Knowing the true conditional density $q(x)$, the optimal classification performance can be achieved using the Bayes decision rule that simply selects the class label j with the largest class probability $q_j(x)$ for each input x . The resulting optimal classification error is called optimal Bayes error.

Many statistical data mining methods fit probability models over the data, hence they approximately minimize the KL distance (Equation 10). The corresponding classification rule for such a method is just the Bayes rule based on the estimated conditional probability $p(x)$. If the estimated $p(x)$ is close to $q(x)$ in their KL distance (Equation 10), then the classification rule derived from $p(x)$ is close to the optimal Bayes error. For these statistical data mining methods, it is often useful to report both the negative log-likelihood (Equation 8) and the classification error rate (Equation 7).

Accuracy

For classification problems we have described the measurement of performance as the measurement of error. If we confine our goals to measuring an overall rate of error, then we can readily reverse the computation and speak in terms of accuracy. A solution that predicts with an error rate of 10% is the same as one that is 90% accurate. Many times our interest is not just in overall performance. Instead, we know that a predictor yields different types of errors, and our attention may focus on the specific breakdown of error, where not all errors are treated equally. We will soon consider different types of errors and how they influence our interpretation of performance.

False Positives and Negatives

The most common classification application is binary classification. For the two-class problem we can view a simple 2-by-2 table describing predictive performance.

Let A+ be actual class label is true, and A– actual class label is false. P+ is predicted class label is true, and P– is predicted class label is false. Thus, we have four combinations, two for correct combinations on one diagonal, and two incorrect combinations on the other diagonal. The numbers of true positives (TP) and true negatives (TN) correspond to the correct answers, and the number of errors is broken into two directions, false positives (FP) and false negatives (FN).

It does not necessarily follow that these categories can be collapsed into the simpler categories of correct and incorrect classifications. In medical diagnostic testing the consequences of completely missing a diagnosis can be great, and therefore one may accept many more false positives than negatives for a screening test. Only with knowledge of the application can we gauge the importance of the different errors. This simple table is widely used in different fields to understand the types of errors that are made in classification and prediction.

Precision, Recall, and the *F* Measure

Binary classification is a prototypical type of problem that occurs in many fields. Different variations of the four measures of error and correctness have achieved prominence in various specialties.

For information retrieval applications there is usually a large amount of negative data (Spark-Jones & Willet, 1997). A classifier can achieve a very high accuracy by simply saying that all data are negative. It is thus useful to measure the classification performance by ignoring correctly predicted negative data. Three ratios have achieved particular prominence: precision, recall, and *F* measure.

Assume that documents are stored in a database and each document has a label, for example, a document about sports. Then when an automated system assigns labels to new documents, it

TABLE 17.1
Error for Binary Classification

	P+	P-
A+	TP	FN
A-	FP	TN

may make mistakes. The percentage of all sports documents that are retrieved correctly is the recall. The percentage of documents correctly labeled as sports is the precision. F measure is defined as the harmonic mean of precision and recall. It is often used to measure the performance of a system when a single number is preferred. In terms of the four measures of Table 17.1, here are the formal definitions.

$$\text{precision} = \text{TP}/(\text{TP} + \text{FP}),$$

$$\text{recall} = \text{TP}/(\text{TP} + \text{FN}),$$

$$F \text{ measure} = \frac{2}{1/\text{precision} + 1/\text{recall}}.$$

Sensitivity and Specificity

These same four basic units of error and correctness are used in medicine, where the measures are slightly different. Sensitivity is a measure identical to recall, but specificity is the reverse of sensitivity. For example, a diagnostic test might be given to patients with sensitivity of 90%, when 90% of people having the disease are detected. A specificity of 90% means that 90% of the people not having the disease are correctly identified (Galen & Gambino, 1975).

$$\text{sensitivity} = \text{TP}/(\text{TP} + \text{FN}),$$

$$\text{specificity} = \text{TN}/(\text{TN} + \text{FN}).$$

Confusion Tables

The 2-by-2 error table for binary classification covers the most prominent situation, and often multiclass problems are defined in terms of a series of binary classifications. Sometimes many hypotheses are competing for a single classification. Although there may be many classes, they can be mutually exclusive, and for any example, only one may be applicable. Instead of a 2-by-2 error table, an m -by- m table may be constructed, in which m is the number of classes. This type of table is called a confusion matrix. It generalizes the simple 2-by-2 matrix. Table 17.2 illustrates a 3-by-3 confusion matrix in which the column labels are the actual class labels and the row labels are the predicted labels. The diagonal is all the correct classifications, and the number of specific errors is observable. For example, in Table 17.2 each class has

TABLE 17.2
Confusion Matrix

	1	2	3
1	80	15	5
2	10	70	20
3	0	0	100

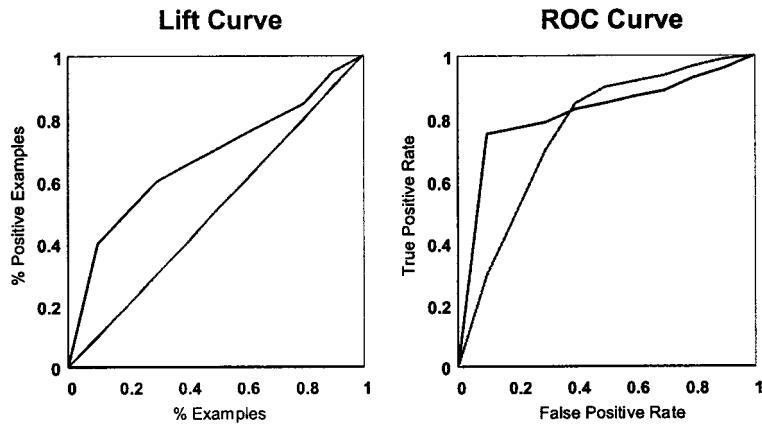


FIG. 17.2. Lift and ROC curves.

100 examples, and 250 examples are classified correctly and 50 incorrectly. Class1 has been classified 15 times as Class2.

ROC Curves

Two alternative ways of evaluating a model's performance, *receiver operating characteristic (ROC) curves* and *lift curves*, are of interest. These are not new, but they can offer insight into how different models will perform for many application situations. Many classification models can be modified so that the output is a probability of the given class, and hence depending on the threshold or some other decision making parameter value, one can get a family of models from one—for example, a tree or a rule set.

The ROC curve originated from signal detection theory. It plots the true positive rate (y-axis) against the false positive rate (x-axis). If there are two models in this space one can obtain any performance on the connecting line of the two just by randomly using the models with some probability in proportion to the desired position on the line. This curve allows one to select the optimal model depending on the assumed class distribution at the time of prediction. Figure 17.2 shows an example of a ROC curve and a lift curve.

Lift Curves

For many applications the aim of prediction is to identify some desired class members (e.g., customers) for whom some action (e.g., mailing advertisement circulars) is to be performed. It is more flexible than classification if the prediction is in the form of ranking based on the predicted class probability. The lift curve then plots cumulative true positive coverage (y-axis) against the rank-ordered examples (x-axis). A random ranking results in a straight diagonal line on this plot. A lift curve of a model is usually above this line, the higher the better.

Clustering Performance: Unlabeled Data

For prediction applications the concept of evaluation is straightforward. Compare the predicted answer to the true answer. For some applications we do not have the true answer. Clustering is an attempt to create labels that characterize group membership. Because there is no exact

answer, evaluation is less definitive, and a multiplicity of different criteria, called the *objective functions*, have been proposed to measure clustering performance. These are measures of how well the clusters achieve various desirable properties: (a) the members of a cluster are similar and (b) the members of one cluster are dissimilar to the members of other clusters.

No single objective function is right or wrong. Their purpose is to fulfill goals relative to similarity of membership in a cluster. If we have two different clustering methods and a specific objective function, then we have a basis for comparison. Here are some ideas for evaluation that have been used in evaluating clustering performance:

- Compute the mean vector for each cluster. That becomes the simulated true answer. Compute the MSE or MAD of each vector from the mean vector, averaged over all examples. Compare it with the average distances among the mean vectors.
- Compute the distance between each vector and the most dissimilar vector within the same cluster. Find the average over all clusters. Alternatively, just find the average distance between all vectors within a cluster. The same concept can be extended to measure distance between vectors in different clusters or combinations of intracluster and intercluster similarity.

These functions have advantages and disadvantages that have been studied extensively, especially for information retrieval. It is up to the analyst to select the evaluation criteria that best match the application.

If a human constructed category scheme exists, then it may be desirable to have the clusters close to the existing category scheme. In this case we can consider the clustering scheme as a classifier by assigning each cluster the label of a category that has the most significant overlap with the cluster. The clustering performance can then be measured using the error rate of the resulting classifier with respect to the existing category scheme.

ESTIMATING ERROR

Independent Test Cases

The assumption of many methods for learning is that the sample used for training is representative, and that future data are independent and identically distributed. The sample is randomly drawn from a general population, as future examples will be. Although this principle may not fully hold in the real world, it is important to separate training from testing and to draw new data independently for testing.

Sometimes we need data to evaluate the many choices that we have for tuning our learning method. If we do too much tuning, the test data are no longer independent. Measures have to be introduced to check the statistical significance of comparing different methods on the test data. In practice, we may need more than one test set: one for tuning and one for evaluation. Doing everything with one data set is dangerous, although various simulation techniques that we will mention can help in maximizing the use of sample data.

Significance Testing

Significance testing is a standard way to determine whether two results are different or if the difference is possibly due to chance. For example, we may try two different learning methods, such as a decision tree and linear discriminant, each yielding different results. Significance

testing helps determine whether the results are significantly different. Data mining implies processing large volumes of data. As the number of examples increases in a sample, the value of formal significance testing decreases. That is because significance is dependent on sample size. With a very large sample of independent test cases almost all differences will be considered significant. When we toss a fair coin 10 times, it is not unusual to see six or seven heads. When we toss that same coin 10,000 times, it is extremely unusual to see 7,000 heads.

If we are measuring the overall error rate for classification, then we can expect the standard error to be that of Equation 11, where $erate$ is the true error rate, which can be substituted by the error rate found on the test cases; and n is the number of test cases. It is highly unlikely that the true error of a predictor estimate error varies more than 2 standard errors of the estimate found for the test cases, except for near-perfect classification.

$$SE = \sqrt{\frac{erate \cdot (1 - erate)}{n}} \quad (11)$$

To compare many results it is also useful to measure the significance through the p value calculation, which is the probability of test errors differing by at least a certain amount from the true error. The probability can be estimated from a tail probability bound such as the Chernoff bound (Hoeffding, 1963):

$$\text{Prob}(\text{test error} \geq \text{true error} + \epsilon) \leq \exp(-2n\epsilon^2), \quad (12)$$

$$\text{Prob}(\text{test error} \leq \text{true error} - \epsilon) \leq \exp(-2n\epsilon^2), \quad (13)$$

where $\epsilon > 0$. This bound can be very useful for comparing the performance of multiple classifiers on the test data. Assume we want to compare K classifiers that produce K test errors “test error₁, …, test error _{K} ” on the same test data; then the probability that true error _{i} \leq test error _{i} + ϵ for all classifiers is at least

$$1 - K \exp(-2n\epsilon^2). \quad (14)$$

In other words, for any confidence value $\eta \in (0, 1)$, with probability at least $1 - \eta$,

$$\text{true error}_i \leq \text{test error}_i + \sqrt{\frac{\ln K + \ln \frac{1}{\eta}}{2n}} \quad (15)$$

for all K classifiers to be compared. This indicates that using n examples, we can compare exponentially in n number of classifiers reliably.

The Chernoff bound also can be applied to regression problems. In general, assume the test error can be expressed as the average of a cost function L :

$$\frac{1}{n} \sum_{i=1}^n L(p(x_i), y_i), \quad (16)$$

and assume that $L \in [0, M]$. Then for all $\epsilon > 0$ the difference of the true error and the test-set error can be bounded by

$$\text{Prob}(\text{test error} \geq \text{true error} + \epsilon) \leq \exp(-2n\epsilon^2/M^2), \quad (17)$$

$$\text{Prob}(\text{test error} \leq \text{true error} - \epsilon) \leq \exp(-2n\epsilon^2/M^2). \quad (18)$$

Resampling and Cross-Validation

When independent data for testing are not readily available, it is possible to obtain accurate evaluations by simulation (Weiss & Kulikowski, 1991). The simplest approach is to randomly divide the sample into a training set and test set. For smaller sets of data, a two thirds to one third partition is reasonable. For much larger samples, in the tens of thousands, a smaller percentage for testing might be considered.

The smaller the sample, the higher the error of the estimate. Instead of creating a single division of data into a training and test set, we might repeat the process more than once. We could randomly select two thirds of cases for training and one third for testing and then repeat the process many times. This process is called resampling. This reduces the error of the estimates of future performance.

A nice procedure for resampling called *k*-fold cross-validation takes advantage of all the data for both training and testing. Although it may appear to violate the principle of never looking at the test data, cross-validation relies on the honesty of the computer to train on data, test on different data, store the results, but then forget about any temporary solutions induced from the data.

In k -fold cross-validation the data are randomly divided into k parts. The most common division is 10 random partitions of data. The learning method is applied to $k - 1$ of the partitions, and the remaining partition is used for testing. This process is repeated k times, and the results are summed for the estimate. For example, in 10-fold cross-validation the data are partitioned into 10 groups of 10% each. Ten simulations are run, with 90% training and 10% testing. Cycling through all 10 partitions, we see that every group eventually is used for testing, and a group appears in a training group every time except when it is used for testing. Figure 17.3 illustrates the technique, where nine groups are used for training and one group for testing.

These methods are usually quite accurate estimators for samples having more than 1,000 examples. They were originally developed for smaller samples in which data are scarce for both training and testing. The leave-one-out method is the extreme variation of cross-validation, in which k in k -fold cross-validation is the number of examples, and each test partition is just one example.

The expected k -fold cross-validation error is the expected true error of the algorithm trained on a random sample size of $n(k - 1)/k$ training data. One reason to use cross-validation rather than an independent hold-out test set is to reduce the variance of the estimate. This is important if the available data size is small. In many cases the cross-validation error estimate can have a much smaller variance than the hold-out test set approach. Moreover, the variance of the

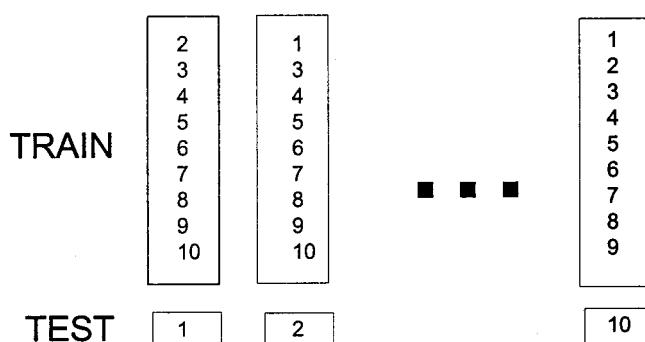


FIG. 17.3. Ten-fold cross-validation

cross-validation error estimate is never larger than the variance of the error estimate with an independent hold-out set. To see this, we write the cross-validation error e_c as the average of error e_i for the i th data partition:

$$e_c = \frac{1}{k} \sum_{i=1}^k e_i. \quad (19)$$

Now assume e is the expected error of the algorithm trained on a random sample size of $n(k-1)/k$ training data. Then $e = E e_i$ for each i , and thus, $e = E e_c$. We use E to denote the average over all training and test data. The variance of e_c is

$$\text{VAR}(e_c) = E(e_c - e)^2 = E \left(\frac{1}{k} \sum_{i=1}^k e_i - e \right)^2 \leq \frac{1}{k} \sum_{i=1}^k E(e_i - e)^2 = \text{VAR}(e_1). \quad (20)$$

Clearly, if the random variable e_i is completely correlated, then $\text{VAR}(e_c) = \text{VAR}(e_1)$. However, this is not likely to happen. Therefore, in practice, the cross-validation error estimate can be a much more reliable way of comparing the performance of different algorithms.

Note that if we use k -fold cross-validation, we effectively estimate the performance of an algorithm with n/k less training data than what is available. To estimate the performance for the full training data, one can use a special case of cross-validation with $k=n$, which is often called leave-one-out error estimate. Because the effective independent test-set size is one for each partition, the leave-one-out error estimate may not always be reliable. It has a small variance only when error e_i (obtained by leaving the i th datum out) are not correlated. This will be the case when the algorithm is relatively stable. That is, the computed predictor approximately remains the same when we remove one data point from the training set.

Bootstrap

Another frequently useful method for error estimation is bootstrap (Davison & Hinkley, 1997; Efron, 1982; Efron & Tibshirani, 1993). Given a data set of size n , a bootstrap training set is generated by randomly selecting n samples uniformly from the data with replacement. After the bootstrap sampling the probability of a data point not chosen is $(1-1/n)^n \approx e^{-1} \approx 0.368$. The expected number of data points that are not in the training set is thus approximately $0.368n$. These data are used as test data in the bootstrap error estimate.

The algorithm for which we would like to obtain the error estimate is then trained on the bootstrap training data and tested on the bootstrap test data. Although the bootstrap error estimate can be obtained based on the test data alone, in practice other formulas also have been used. For example, the .632 bootstrap error estimate is defined as

$$\frac{1}{k} \sum_{i=1}^k [0.632 \times \text{test error}_i + 0.368 \times \text{repl error}_i], \quad (21)$$

where k is the number of random bootstrap sampling we use to obtain the estimate, test error_i is the test error for the i th bootstrap split, and repl error_i is the error on all of the available data for the i th bootstrap predictor.

Similar to leave-one-out cross-validation, bootstrap tries to capture the expected error of the algorithm with a training set size of about n . Computationally it can be more efficient than leave-one-out. Bootstrap methods also are used often on small data sets for which leave-one-out cross-validation may have a larger variance. Similar to the cross-validation approach, the

TABLE 17.3
Time Series Evaluation

<i>Original Data</i>									
1	2	3	4	5	6	7	8	9	10
<i>Train</i>						<i>Test</i>			
1	2	3	4	5	6	7	8	9	5
2	3	4	5	6	7	8	9	10	6
3	4	5	6	7	8	9	10		7
4	5	6	7	8	9	10			8
5	6	7	8	9	10				9
6	7	8	9	10					

potential success of bootstrap also relies on the assumption that the underlying data mining algorithm is stable.

Time Series

Time series is a form of data ordered by time of occurrence. If we are tracking a stock price, the values are ordered from oldest to most recent. If we are collecting news stories, we might also order these chronologically. The objective of data mining is usually to project results to the future. The data may not belong to a stationary population, and therefore we cannot randomly sample from the entire group. Instead, we divide the data by time of occurrence, taking the earliest data for training and the later data for testing. This partition of data directly simulates the evaluation of predictive performance on new cases that may change over time. We train on the past and evaluate on the future.

For time series there is also an analog to cross-validation, but not in its random form of k equal groups of cases. The data can be surrounded by a moving window on both the past and future. We can do an extensive simulation in which at any give time, t , we train on the k most recent examples and test on the immediate future j items. For example, we might examine stock price data. Starting at time t , we train on the previous 10 prices and evaluate the resulting model on the single next price. If we repeat this process for all times, we have performed an extensive evaluation of the effectiveness of our trained solutions.

Table 17.3 illustrates the evaluation of a time series by simulation. We are given 10 initial data points, and a window of size 4 is examined and projected to the following data point. A sample of 6 examples is created from the original 10 data points.

ESTIMATING COST AND RISK

Error is not the sole criterion for judging the effectiveness of a proposed solution. Sometimes we want to consider costs for obtaining solutions and making errors. The simplest situation to examine is binary classification, in which we have two kinds of errors, false positives and negatives. We may not consider these two types of errors equally. Therefore, a cost is assigned to each error. The objective is not to minimize error, but to minimize average or overall cost as in Equation 22

$$\text{Average Cost} = \frac{(FP \cdot \text{Cost}(FP)) + (FN \cdot \text{Cost}(FN))}{n} \quad (22)$$

The concept of assigning cost of errors can be generalized beyond binary classification. For a confusion matrix with more than two classes we can assign costs for every pairwise combination of error that can occur. For example, true class4 could be falsely classified as class10, and we would have a specific cost for that type of error.

Cost of error is not the only type of cost. The cost of taking some action or obtaining a result may also be a key consideration. For example, we wish to detect fraud in usage of a credit card. It costs money to investigate a transaction and to contact the credit card owner. Thus, we might evaluate performance based on actual costs, not just errors and their associated costs. Similarly, we might develop an investment strategy that makes few errors, but we must still consider transaction costs.

OTHER ATTRIBUTES OF PERFORMANCE

Although the tendency for researchers is to emphasize the sterling predictive performance of their methods, real-world practitioners must consider other factors for proper evaluation. Here, we list a few of these that are noteworthy for practitioners of data mining.

Training Time

Given a single sample, it is relatively easy to evaluate its performance on additional test cases. That type of evaluation emphasizes measuring predictive performance, such as accuracy. To measure predictive performance, we may obtain new cases for test proposes. If training is less stationary, when new training cases are arriving in relatively short time frames, we may have to take action to retrain. For example, we may retrain on the new data combined with prior data, or we may train on the new sample obtained from the most recent time period.

Retraining on a newly composed sample will take time. Not all methods are amenable to quick training, and they can consume additional resources such as memory. Thus, the expense of retraining many times should be factored into any evaluation of candidate solutions.

Application Time

Although training time may be an important factor in selecting a learning method, to a lesser extent application time also will be useful to consider. A method such as nearest neighbor may have no training time, but its application in high dimensional situations can consume exorbitant amounts of time and memory, sometimes turning aside its practical use. Any practical application must take a reasonable amount of time, for both development and application. For large volumes of data these issues become more critical, and predictive performance must be balanced by reasonable application times.

Interpretability

In many data mining applications the interpretability of the models used is an important characteristic to be considered in addition to the accuracy achieved and the computational requirements. Interpretability of the model allows people to gain insights into the data and possibly to correct flaws in the data mining system.

The requirement of interpretability can be satisfied by using a rule based system, such as rules obtained from a decision tree. Rule based systems are particularly appealing because

a person can examine the rules and modify them. Initial rules also can be written manually without using any training data.

Expert Evaluation

Some applications have purely an empirical basis. If we could develop an accurate stock price predictor, no one would debate its merits if it made money. Its value could be measured in purely financial terms. For other applications, such as medical applications, it may be essential for experts to evaluate the proposed model's performance. These experts may not be willing to rely on a "black box" and may feel that some clear explanatory support will be needed by the software. The main theme is to evaluate performance relative to accumulated knowledge. The proposed model should be consistent with current expertise, or it should be explained when it diverges.

Although it is easy to justify empirically some strange results for an induced model, these may be due to biases in the sample. We like to think that the sample is completely representative, but some misunderstood problems may be uncovered by the solution, for example, a feature that unintentionally provides the correct label. An expert familiar with the area may readily determine the flaw in a proposed solution, whereas the naive developer may see only good results in unfamiliar territory.

Field Testing

Data are obtained, data mining methods are applied, proposed solutions are tested, and eventually results are given, and projections may be made for future performance. Ultimately, the strongest form of evaluating is testing in the field. It is quite easy to introduce artifacts into data that may lead to unexpected results in the field. Moreover, the sample may contain elements that do not correspond to true real-world conditions. These will likely be uncovered when actual real-world testing occurs. In scientific experimental research, investigators should wait for confirmation by other researchers who can replicate those results. Similarly, once we develop a data mining solution, such as a predictor, we must await confirmation from real-world experience.

Cost of Obtaining Labeled Data

Many data mining methods require labeled data, for example, classification and regression methods. They examine examples with known correct answers, that is, their labels. Some methods can actually improve their results by (a) examining unlabeled data or (b) only asking for additional labeled data when they make errors. In terms of evaluation, those methods that need fewer examples to train effectively can be advantageous. This is especially true when additional training will be needed in the future. The amount of data and frequency of data collection are important factors in comparing different results and methods of data mining.

REFERENCES

- Davison, A. C., & Hinkley, D. V. (1997). *Bootstrap methods and their application*. Cambridge, U.K.: Cambridge University Press.
- Efron, B. (1982). *The jackknife, the bootstrap and other resampling plans*. Philadelphia: Society for Industrial and Applied Mathematics.

- Efron, B., & Tibshirani, R. J. (1993). *An introduction to the bootstrap*. New York: Chapman & Hall.
- Galen, R., and Gambino, S. (1975). *Beyond normality: The predictive value and efficiency of medical diagnoses*. New York: Wiley.
- Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58, 13–30.
- James, M. (1985). *Classification algorithms*. New York: Wiley.
- Sparck-Jones, K., & Willet, P. (Eds.). (1997). *Readings in information retrieval*. San Francisco: Morgan Kaufmann.
- Weiss, S., & Kulikowski, C. (1991). *Computer systems that learn*. San Francisco: Morgan Kaufmann.

18

Security and Privacy

Chris Clifton
Purdue University

Introduction: Why There Are Security and Privacy Issues with Data Mining	441
Detailed Problem Analysis, Solutions, and Ongoing Research	442
Privacy of Individual Data	442
Fear of What Others May Find in Otherwise Releasable Data	448
Summary	451
References	451

INTRODUCTION: WHY THERE ARE SECURITY AND PRIVACY ISSUES WITH DATA MINING

Concern with information security and individual privacy has become a major factor in the design and use of information systems. Data mining opens new challenges in this area. This chapter looks at two issues: protecting the privacy of information sources and exercising control over potential data mining results.

The first issue, privacy of individual data, has the potential to stop beneficial data mining projects. A prime example is data from the U.S. Census Bureau. The base data contains detailed descriptions of many U.S. residents. Such multifeatured data in a large data set could be useful in many data mining projects. However, concerns about misuse of individual data prevent the raw data from being made available. The same is true of many other sources: insurance company records, credit report information, and so forth.

Inability to release data is not the only deterrent to data mining projects caused by privacy issues. Even when the data are available, data mining may give the ability to predict individual

data values. This poses a potential public relations nightmare for the data miner. In other cases there may be regulatory or contractual limitations on the use of the data. It is difficult to ensure that data mining projects do not overstep boundaries.

A second security issue posed by data mining is the capabilities it gives to adversaries. Suppose we wish to make data available—how can we ensure that others do not misuse it? The U.S. Census Bureau is again a good example. Detailed summaries of census data are made available. Considerable research has gone into ensuring that the summaries do not reveal private information. Smaller concerns cannot afford such detailed analysis. Can they ensure that released data is not susceptible to misuse? An example would be hidden “secrets” in the data. The individual data items may be benign, but through use of data mining techniques an adversary may learn something the information provider does not want disclosed.

This chapter looks at these issues and presents solutions if research has addressed them. We first look at concerns with privacy: How can we carry on a data mining effort when concerns about privacy limit access to the needed data? We then go into issues about safely releasing data.

DETAILED PROBLEM ANALYSIS, SOLUTIONS, AND ONGOING RESEARCH

Privacy of Individual Data

Personal privacy—the right of individuals to control data about themselves—is a hot topic. Information technology has lead to a fundamental change in the discussions of privacy rights. Computers have given us an unprecedented ability to store and process information. An example is public records; for years, the difficulty of accessing information that was legally public made the information private in practice. The use of public record information has long been limited to specific private and public investigations—journalist investigations of public figures, criminal investigations, civil trials—events that affected only a small minority of the population. Information technology has enabled the use of this information on a wide scale. Targeted marketing is a prime example. Data about us that has long sat unused now affects our everyday lives.

Privately gathered information, such as credit reports and customer relationship management data, are an even bigger issue. The ability to capture and share such data has grown drastically. Applications such as Amazon.com’s trial of offering different prices to different customers (“Amazon,” 2000) would have been unthinkable until recently; with manually processed information such applications were only possible for the largest customers. Data mining is one of the key enabling technologies behind such applications. However, as Amazon.com discovered, people are not always receptive to these new applications. Public outcry quickly ended their differential pricing experiment.

Privacy issues are being discussed on several fronts. Legal systems are trying to adapt to the capabilities of new technology. The laws and regulations that govern the use of individual data are changing. The private sector is also trying to adapt. Companies now provide statements on how they will use the information they gather—this is a contractual relationship with their customers.

Privacy issues can affect a data mining project in many ways. We now go into some of the ways privacy can affect data mining and explore techniques to deal with these issues.

Reconstructing Data from Results. People are very concerned about the release of information about themselves. Release of individually identifiable information is often

controlled by both legal and contractual mechanisms. The holder of individually identifiable data may be allowed to use it, but must ensure that the data is not released to others.

A data mining project may utilize such individual data but release only the results. Initially, this is fine. The results are aggregates or summaries and do not include the protected individually identifiable data.

Release of aggregates works as long as we are not able to reconstruct the data. This demands some care. For example, suppose we are a bookseller, and we want to sell publishers information that helps them market their products. An example rule might be “Age > 40 and Income > \$50,000 → purchases financial planning books.” But what if we came up with a rule like “Address = ‘10 INDUSTRIAL AVENUE’ and Title = ‘President’ → purchases books on explosives.” This association rule applies to a single person only; we have released the protected data of who purchases what.

Although the preceding example is trivial, it does allow us to make some general statements. If an association rule applies to a single entity only, it has the potential to release individually identifiable data. High-confidence rules also pose a danger; a rule that holds in all cases releases data for every entity to which the rule applies.

What makes the problem difficult is that combinations of rules may reveal protected data even when the individual rules do not. There has been substantial work on this problem with respect to data from the U.S. Census Bureau (Subcommittee on Disclosure Limitation Methodology, 1994). The Census Bureau makes data available that summarizes census blocks. To ensure that individually identifiable data is not released, the summaries must be based on a population of a minimum size (300 people in the 2000 census). In addition, the results are grouped into ranges; instead of providing a single value (average income for the census block), the result is a range. These protect the data values for individuals.

These issues also have been addressed for statistical queries. With statistical queries a user is allowed to query protected data, but only by using statistical operators that summarize groups of data. This protects the individually identifiable data. For example, a query that returns total income for a group of people does not disclose the income of any individual in that group. However, multiple summary queries can disclose individual data. Take two queries: one that requests total salary for all employees of a company and one that requests the total salary for all employees but the president. Neither query by itself reveals individually identifiable information, but the difference between the two is the salary of the president.

There have been efforts to address these issues. Denning (1980) developed methods to identify when a set of queries was “safe,” that is, when they do not reveal individual values. A survey of these and other approaches to privacy protection through statistical queries is given by Adam and Wortmann (1989). One method is through query restriction, for example, tracking queries and ensuring that the combination of query results does not reveal information, or restricting queries to portions of the database. In the “president’s salary” example, a query restriction method could use *query set overlap control* to prevent the above combination of queries. A history of queries would be kept, and all results would be checked to ensure that (a) the result is generated from at least k items and (b) the items used to generate the result have at most r items in common with those used to produce any previous query result. Dobkins, Jones, and Lipton (1979) showed that at least $1 + (k - 1)/r$ queries are needed to compromise data with such an approach.

A second method is data perturbation: introducing noise into the original data. A third is output perturbation: leaving the original data intact but introducing noise into the results. The challenge of perturbation techniques is ensuring that the query results are far enough from real values that results from multiple queries cannot be used to reconstruct the original data values, but are still close enough to the real results to be meaningful. Palley and Simonoff (1987)

showed that regression (read “data mining”) techniques can be used to estimate individual data even when the statistical query techniques protect exact data values. This is done by constructing a *synthetic database* to generate the same results as the statistical queries on the real database. First, histograms are created for independent variables that are not protected (e.g., job title in the previous example), based on queries counting the number of instances of each value. Next, statistical queries on the protected value (e.g., average salary) are run for each of the variables. These results are used to create a synthetic database that captures the relationships between the unprotected variables and protected values found in the statistical queries. Data mining techniques can now be applied to the synthetic database directly.

Part of what makes the statistical query problem difficult is that the adversary gets to create the queries. The likelihood that a set of data mining results compromise individual data is remote, provided we eliminate straightforward channels such as rules that apply to only a single entity. Proving that individual data is not released may be difficult, but reasonable care should be adequate for many projects.

Inability to Release Data. A bigger roadblock for many data mining projects is obtaining access to the needed data. Privacy concerns may prevent the owner/custodian of the data from making it available to others. A prime example is medical records. Insurance companies have volumes of data on patient diseases and treatments. Mining this data could be beneficial to public health agencies, drug companies, or other researchers. However, releasing the raw data exposes individually identifiable information to the researchers—this would violate the relationship between the insurance companies and patients.

This points to a tradeoff between individual privacy and the greater good. Data mining may provide benefits, but the fear that individual data may be misused by the data miner places a roadblock in front of a project. Fortunately, there are solutions to this. One approach is to modify the data. The U.S. Census Bureau has used this approach (Moore, 1996). They make some data available showing complete data for a sample of individuals in a region. Identifying information is removed. In addition, various techniques (such as swapping data values among similar individuals) further prevent individual identification. Thus, the data does not really represent real people; the data values have been modified so that they are representative of real data but do not actually reflect specific individuals. This preserves the privacy of the individuals, as no entity in the data contains actual values for any real individual. The question is, can we reliably mine this perturbed data?

Agrawal and Srikant (2000) presented an approach for developing decision trees on numeric data that has been perturbed by randomly modifying the data. The basic idea is that the modifications should average out over many instances, so the resulting tree should be correct. The difficulty is how to transform the input from continuous data to categorical data. As discussed in Chapter 1 in this volume, appropriate splits are made at natural break points. However, the modifications to the data may obscure these natural breaks, resulting in poor split points. The key idea of Agrawal and Srikant (2000) is that determining good split points can be done if we know the *distribution* of the data—we do not need to know the actual values. For example, if we know that the distribution of the data values is concentrated in the ranges (0, 1) and (2, 3), we can determine that 1.5 would be a good split point. However, the data perturbation can cause many of the values that were originally in the (0, 1) and (2, 3) ranges to migrate into the (1, 2) range, obscuring this split point. Their method reconstructs the distribution from the perturbed data values and knowledge of the distribution of the perturbing values. They present evidence that the resulting decision trees are nearly as good as decision trees built from the original data and substantially better than trees built directly from the perturbed data.

The next question concerns the privacy of the original data. Does the ability to reconstruct the distribution and construct decision trees enable us to reconstruct the original data values? Agrawal and Aggarwal (2001) provided a thorough treatment of this issue. In some cases we can use knowledge of the distribution to tighten bounds on the data. For example, suppose the released data contains a value 1.5. If we know that data has been perturbed by adding a value in the range $[-1, 1]$, we know the real value corresponding to the released 1.5 is in the range $[0.5, 2.5]$. However, if we reconstruct the distribution and determine that all values are in either the range $[0, 1]$ or $[4, 5]$, we know that the original value corresponding to the released 1.5 is really in the range $[0.5, 1]$. Although we do not know the real original value, we have obtained much tighter bounds than expected. Agrawal and Aggarwal (2001) gave metrics to evaluate privacy achieved by different ways of perturbing the data. Rizvi and Haritsa (2002) applied a similar approach to the problem of learning association rules from perturbed data.

A second approach to mining private data is to have the data owner/custodian do the mining and release only the results. At first glance, this poses few technical challenges. We have previously discussed the problem of ensuring that the released results do not compromise individual data values. Business issues must still be resolved: What is the benefit to the data custodian, how do we convince them to expend the effort, and how do we ensure that they have the expertise to produce the needed results? There is still one problem that does pose significant technical challenges. What do we do when the data is split among several owners?

Let us assume a public health agency wants to mine data on disease outbreaks, and the source data is owned by insurance companies. The traditional approach would be to combine the data into a data warehouse, as shown in Fig. 18.1. However, this requires a site trusted by all parties—what do we do when no such trusted site exists?

Distributed data mining, discussed in Chapter 13 in this volume, can provide an answer. The idea is to perform local operations on each site that produces intermediate data that can be used to obtain the results, without revealing private information. Although not all distributed data mining algorithms preserve privacy, they can serve as a starting point to solve some privacy problems. There are many variants of these problems depending on how the data is

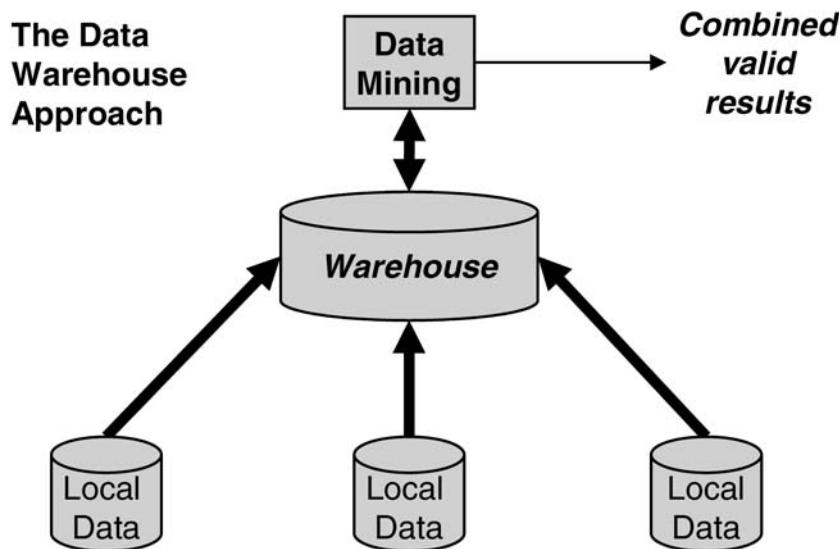


FIG. 18.1. Data warehouse approach to mining data with multiple owners.

distributed, what type of data mining we wish to do, and what restrictions are placed on sharing of information. Some problems are quite tractable, others are more difficult. For example, if we are trying to learn association rules with support and confidence thresholds, there is a simple distributed solution that provides a degree of privacy to the individual sites. Suppose we want to find association rules from insurance company data such as:

Received Flu shot and age > 50 \Rightarrow hospital admission,
 where at least 5% of insured meet all the
 criteria (support), and at least 30% of those meeting
 the *flu shot* and *age* criteria actually require
 hospitalization (confidence).

Chapter 2 in this volume presents methods to find all association rules with a minimum level of support. We can easily extend this to the distributed case using the following lemma: If a rule has *support* > $k\%$ globally, it must have *support* > $k\%$ on at least one of the individual sites. For proof of this, assume all the data is together, and divide data items into those that support the rule and those that don't. Next partition data such that no site has *support* > $k\%$. For each supporting item sent to a site, at least $1/k$ non-supporting items must be sent to that site. It can be seen that we will run out of non-supporting items before assigning all the supporting items to sites.

We can compute global association rules without sharing individual data as follows: Request that each site send all rules with support at least k . For each rule returned, request that all sites send the count of items they have that support the rule, and the total count of all items at the site. From this, we can compute the global support of each rule. From the lemma, we are certain that all rules with support at least k have been found. An example of how this works is shown in Fig. 18.2.

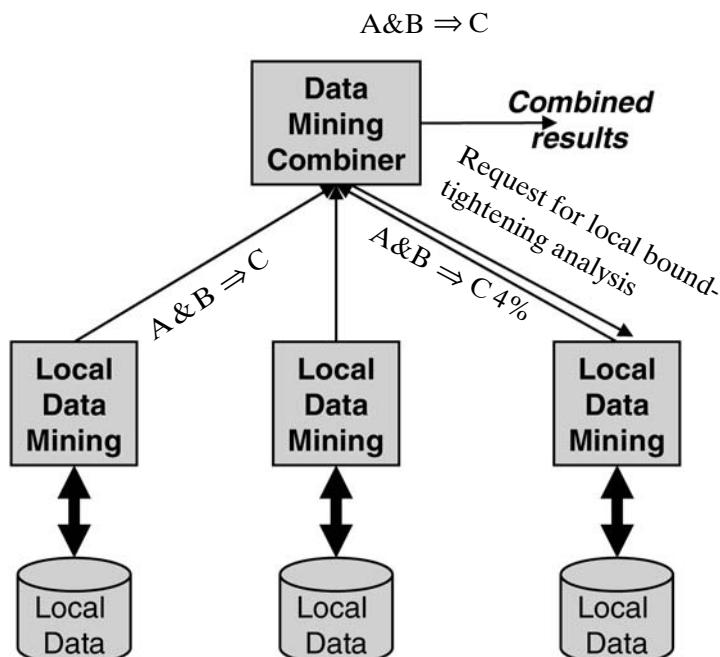


FIG. 18.2. Example of computing association rules from individual site results.

This is straightforward, but as we vary the problem the challenge becomes more difficult. What if we want to protect not only the individual items at each site, but also how much each site supports a given rule? The method reveals this information, although Kantarcıoglu and Clifton (2002) presented a solution that does not. Their solution follows this approach, but each site encrypts its rules before sharing them. The encrypted rules are then reencrypted by the other sites using commutative encryption techniques. The rules are then decrypted by all sites. Because the encryption is commutative, decryption can be done in any order, and the knowledge of which site supported which rule is lost. This gives the global candidate set. Computing which candidates are supported globally is done in a round-robin fashion. The first site computes its *excess support*: the number of items it has beyond (or below) the support threshold. It adds a random number to this value and passes the total to the second site, which adds its excess support and passes it on. At the end the first site subtracts the random number. If the remaining value is positive, the rule is supported globally.

Another variant of the distributed association rule problem is when the data is partitioned vertically: A single item may have part of its information at one site and part at another. A solution to this problem for two sites was given by Vaidya and Clifton (2002). The idea is that the first site sends its data to the second, but only the key is “in the clear.” The data values are obscured by random numbers. The second site computes the product of its values for each row with the corresponding (obscured) value from the first site and returns the sum of these products. By remembering the process of obscuring its values and the random numbers used, the first site is able to extract the real sum, which is the support for the given item set in the combined database.

There also has been work addressing problems other than association rules. Lindell and Pinkas (2000) presented a method to build decision trees from data split between two parties. This is an ongoing area of research; solutions to additional privacy-preserving distributed data mining problems are likely.

Limitations on Use of Results. Another problem facing data mining projects is restrictions on how data may be used. For example, in the United States the regulated portions of the telecommunication industry are allowed to use calling, billing, and personal information records for fraud detection. The data cannot be used for marketing purposes, however. This extends beyond individually identifiable information; companies are enjoined from using summaries or statistics on the data for purposes other than fraud detection.

Another example is mining of customer relationship management data. The gathering of this data is often controlled by contractual relationships—most of us have seen privacy policies when we register with a Web site. These policies generally limit use of individually identifiable data but could also be interpreted to place restrictions on how the results of mining the data are used.

What does this mean for data mining projects? In many companies the same data mining experts serve different parts of the organization. Crossover—carrying results from one project to another—can easily happen in such an environment. Although such crossover is often beneficial, it can result in inadvertent misuse of restricted data. The “Chinese wall” approach—erecting a barrier between parts of an organization—is not an efficient use of expertise. Data mining experts must educate themselves on the allowable uses of data and ensure that each project’s results can be obtained from data it is allowed to use.

Legal systems are gradually coming to grips with technology’s ability to utilize individual data. The European Union has strong privacy laws governing the use of personal data. As these laws evolve, the impact on data mining will change. Even today the differences in laws between nations can have a big impact on multinational corporations. The privacy preserving distributed data mining approaches discussed previously may help. For example, a corporation

may not be able to share data between subsidiaries in different countries, but using distributed data mining techniques they may be able to mine the entire corporate data set. Although privacy issues are likely to remain a stumbling block to many data mining projects, they should not be viewed as insurmountable.

Fear of What Others May Find in Otherwise Releasable Data

Even when there are no privacy concerns with individual data items, data mining projects can still face security issues. Privacy is generally concerned with preserving individual entities from disclosure. But what if it is not the individual entities, but the summaries and rules learned from data mining, that must be kept secure?

Perhaps the most obvious instance is when data mining can be used to expose a hidden secret in otherwise benign data. We will look at several techniques that have been proposed to address this issue. But first, we look at a wider problem: the potential for misuse of data mining results.

Misuse. Although data mining can be beneficial, it can also be used to accomplish unacceptable or even illegal goals. One example comes from the U.S. mortgage industry: a practice known as “redlining.” At one point banks used race as a factor in deciding whether a mortgage should be approved. When this became illegal, a practice known as redlining took its place. Redlining was highlighting neighborhoods with a high default rate; mortgages in those neighborhoods were not approved. Due to historical segregation in housing and historically lower incomes among minorities, minority neighborhoods often had high default rates. The practical result of redlining was to deny mortgages based on race, independent of individual creditworthiness—Thus, redlining was outlawed.

Data mining gives us the ability to evaluate a much greater variety of factors. The banking industry now generates credit scores based on credit history, allowing faster and more accurate decisions on creditworthiness. The users of this technology must be careful to ensure that the factors used do not serve as de facto surrogates for prohibited factors (such as race).

This issue affects not only the users of data mining, but also providers of data. In 1990 Lotus Development Corporation proposed to offer *Household Marketplace*, a CD-ROM database of information on individual U.S. households. Lotus by itself did nothing to harm anyone—much of the data was already a matter of public record. However, by making the data more easily accessible, the potential for misuse increased. The public outcry against this project caused Lotus to withdraw the offering rather than face the public relations damage from going through with the project (“Lotus,” 1990).

What lesson should data mining projects learn from this? First, be careful when the goals of the project come close to socially or legally dangerous ground. The Lotus case shows that it is not just those who misuse data that face problems. The providers of the misused data can find themselves with both legal and public relations costs. This leads to a question: How do we ensure that data we make available is not misused?

The first key to ensuring proper use of data is to know what we mean by proper use. Lotus may have intended the *Household Marketplace* database to be used for direct marketing. It is unlikely this alone would have caused an uproar—credit reporting agencies constantly make use of even more confidential data for direct marketing (witness the flood of credit card offers creditworthy people in the United States receive). The difference is in how the data is made available. Credit reporting agencies do not distribute data for marketing purposes; they take in parameters from the marketers and provide lists of people who meet those parameters. This

protects against misuse. The Lotus database had no such protection. A data owner can take a similar approach with data mining projects. Instead of providing raw data to the project, the data owner does the mining and provides the results. This may increase the cost of the mining project—the data owner may not have the expertise needed, and the feedback loop between results and revising the mining parameters grows—but the data owner gains considerable control over how data is used.

Sometimes we want to make data public but are prevented by concerns over potential misuse. A simple example is a corporate telephone book. A single entry—name, department, office, and telephone number of an employee—may not be sensitive. However, given the entire telephone directory, we may be able to learn “corporate secrets.” An unusually large department containing a few people whose area of expertise is known may allow us to guess areas for new products.

Knowing the intended use of data allows us to address the problem of unintended release of secrets. If the goal of the telephone book is to allow someone to find out how to locate an employee, it is not necessary to release the entire book. A simple single-entry query mechanism is sufficient.

This still leaves a question. How do we prevent someone from getting access to the entire data set through repeated queries? One way is to alter the data—a telephone book that contains all employees but also includes a large number of fictitious entries meets the “given a name, find information” goal. Data mining applied to the augmented telephone book will give incorrect results. Another approach is to track the history of a user’s access. Marks (1996) gave a method for efficiently tracking and limiting accesses to prevent learning specific inductive rules.

What if we do not know what we want to protect? The approach of Marks (1996) protects specific secrets by ensuring data that can be combined to disclose those secrets are not released. An alternative, when we do not know what secrets the data might contain, is to limit the total size of the released data to ensure that data mining results are not statistically valid.

Clifton (2000) presented an analysis of classification that shows for small enough samples, data mining results cannot be relied on. Because many types of data mining can be used to classify (e.g., we can build a decision tree from rules), if we cannot build a reliable classifier, other types of data mining results would also be suspect.

The basic idea behind this work is that the best an adversary can do is to choose a classifier that works well for a sample of data. This may be substantially different than the best classifier on the overall data (see Fig. 18.3). We want to ensure that the combined estimation and approximation error is sufficiently large. The adversary controls the type of data mining and

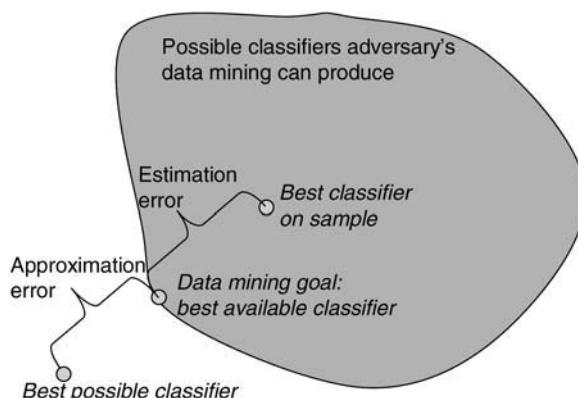


FIG. 18.3. Error based on sample size.

thus the space of possible classifiers and approximation error. A small sample increases the likelihood of a high estimation error.

The problem is that as the space of possible classifiers shrinks, it becomes easier to get a good classifier: The potential for high estimation error is lost. However, this increases the expected approximation error. As one example, if we assume the approximation error is 0, we can determine a sample size from the complexity of the data mining technique used in terms of the Vapnik-Chervonkis dimension V (Vapnik and Chervonenkis, 1974). The following sample size gives us probability $\geq 1/10$ that the estimation error will be greater than ϵ :

$$\text{number of samples} \leq \frac{V - 1}{12\epsilon}$$

Clifton (2000) presented sample size formulas for cases in which the approximation error is not 0.

The problem with this approach is that allowable samples tend to be small—for reasonable choices of V and ϵ , 100 to 1,000 rows of data. For certain purposes this is reasonable. For example, a few samples of actual data may be useful for development of new systems. This approach gives the ability to state “this is a safe amount of data to release,” without worrying about the specific inferences that may be drawn.

Hidden Secrets. When we do know the secrets we want to protect from potential data miners, we have more options. In addition to limiting data we can specifically falsify data to cover up the secrets. The problem is how to alter data to ensure that the secrets are protected without making the data unusable for its intended purpose.

An approach proposed by Atallah, Bertino, Elmagarmid, Ibrahim, and Verykios (1999) addresses this for the generation of association rules. The goal is to hide a set of sensitive rules by altering data values. A rule is considered hidden when the support or confidence for that rule falls below a threshold. To keep the data useful, the alterations are made so as to minimize the number of nonsensitive rules hidden. They show that an optimal solution to the problem is NP-hard. However, they do present algorithms that are guaranteed to hide the sensitive rules, while using heuristics to limit the effect on nonsensitive rules. Their approach is to hide the item sets from which the sensitive rules are constructed. The algorithm first searches through all of the rules that need to be hidden for the single item that is contained in the most rules. The data is modified by removing this item from one transaction that supports the rule. To minimize the effect on nonsensitive rules, the transaction chosen is the one that supports the fewest high support two-item sets. Modifying this transaction reduces the support for the chosen item set by one. The process is repeated until all sensitive rules have support less than the threshold.

One problem with this approach is that the data now contains incorrect values. An alternative from Saygin, Verykios, and Clifton (2001) is to place unknowns in selected entries. The unknowns hide sensitive rules by reducing either the *guaranteed* support or the confidence of the rules. Guaranteed support means the actual support if all of the unknowns “go against you”—every unknown that could support a rule is actually 0. Confidence has a similar definition. Thus, a sensitive rule is still possible given the correct assignment of values to unknowns, but many “false” rules are also possible with incorrect (but plausible) assignments of values to unknowns. The values to convert to unknowns are chosen so as to limit the number of nonsensitive rules hidden.

A problem with this approach is that an adversary already has a starting point to “guess” the sensitive rules: unknowns in the database. Unless the database already contains unknowns, thus

hiding the values modified to hide sensitive rules, care must be taken to ensure an adversary cannot reconstruct the data.

SUMMARY

This chapter presented some of the security and privacy issues that face data mining projects. Although privacy concerns can be an impediment to data mining, there are many solutions to the problem. Some key things to remember:

- Data mining results do not automatically protect the confidentiality of the underlying data. If data privacy is an issue, ensure that the results do not compromise this privacy.
- If data privacy stands in the way of obtaining data for a project, there are ways to address privacy concerns. Do not give up—look for alternative ways to obtain results without compromising data confidentiality.
- Data mining, if misused, opens new doors to compromising privacy. Be aware of the possibilities.
- Be careful how data mining results are applied. Apparently innocuous data and results can be misused, leading to potential problems for all involved.

Data mining gives us great potential to constructively utilize the data we gather. Fear of data mining is easily overblown. Paying attention to privacy and security concerns raised by a data mining project and being open about the purpose and results can help alleviate these fears.

REFERENCES

- Adam, N. R., & Wortmann, J. C. (1989). Security-control methods for statistical databases: A comparative study. *ACM Computing Surveys*, 21, 515–556.
- Agrawal, D., & Aggarwal, C. C. (2001). On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems* (pp. 247–255). New York: ACM.
- Agrawal, R., & Srikant, R. (2000). Privacy-preserving data mining. In *Proceedings of the 2000 ACM SIGMOD Conference on Management of Data* (pp. 439–450). New York: ACM.
- Amazon may spell end for “dynamic” pricing. (2000, September 29). *USA Today*, <http://www.usatoday.com/life/cyber/tech/ci595.htm>
- Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., & Verykios, V. (1999). Disclosure limitation of sensitive rules. In *Knowledge and Data Engineering Exchange Workshop (KDEX '99)* (pp. 25–32). Piscataway, NJ: IEEE.
- Clifton, C. (2000). Using sample size to limit exposure to data mining. *Journal of Computer Security*, 8, 281–307.
- Denning, D. E. (1980). Secure statistical databases with random sample queries. *ACM Transactions on Database Systems*, 5, 291–315.
- Dobkins, D., Jones, A. K., & Lipton, R. J. (1979). Secure databases: Protection against user influence. *ACM Transactions on Database Systems*, 4, 97–106.
- Kantarciooglu, M., & Clifton, C. (2002). Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD '2002)* (pp. 24–31).
- Lindell, Y., & Pinkas, B. (2000). Privacy preserving data mining. In *Advances in cryptology—CRYPTO 2000* (pp. 36–54). New York: Springer-Verlag.
- Lotus—new program spurs fears privacy could be undermined. (1990, November 13). *The Wall Street Journal* (p. B1).
- Marks, D. G. (1996). Inference in MLS database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8, 46–55.

- Moore, R. A. (1996). *Controlled data-swapping techniques for masking public use microdata sets* (Statistical Research Division Report Series RR 96-04). Washington, DC: U.S. Bureau of the Census.
- Palley, M. A., & Simonoff, J. S. (1987). The use of regression methodology for the compromise of confidential information in statistical databases. *ACM Transactions on Database Systems*, 12, 593–608.
- Rizvi, S. J., & Haritsa, J. R. (2002). Maintaining data privacy in association rule mining. In *Proceedings of the 28th International Conference on Very Large Data Bases* (pp. 682–693). San Francisco: Morgan Kaufmann.
- Saygin, Y., Verykios, V. S., & Clifton, C. (2001). Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30, 45–54.
- Subcommittee on Disclosure Limitation Methodology, Federal Committee on Statistical Methodology. (1994). *Report on statistical disclosure limitation methodology* (Statistical Policy Working Paper 22). Washington, DC: Office of Management and Budget. (NTIS PB94-16530)
- Vaidya, J. S., & Clifton, C. (2002). Privacy preserving association rule mining in vertically partitioned data. In *The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 639–644). New York: ACM.
- Vapnik, V., & Chervonenkis, A. (1974). Theory of pattern recognition. (in Russian). Moscow: Nauka.

19

Emerging Standards and Interfaces

Robert Grossman

University of Illinois at Chicago and Two Cultures Group

Mark Hornick

Oracle Corporation

Gregor Meyer

IBM

Introduction	453
XML Standards	454
XML for Data Mining Models	454
XML for Data Mining Metadata	456
APIs	456
SQL APIs	456
Java APIs	457
OLE DB APIs	457
Web Standards	457
Semantic Web	457
Data Web	458
Other Web Services	458
Process Standards	458
Relationships	458
Summary	459
References	459

INTRODUCTION

Data mining standards are concerned with one or more of the following issues:

1. A standard representation for data mining and statistical models. This is the more straightforward to standardize and can be done easily, for example, with Extensible

Markup Language (XML). This includes, for example, the parameters defining a classification tree.

2. A standard representation for cleaning, transforming, and aggregating attributes to provide the inputs for data mining models. This includes, for example, the parameters defining how zip codes are mapped to three digit codes prior to their use as a categorical variable in a classification tree.
3. A standard representation for specifying the settings required to build models and to use the outputs of models in other systems. This includes, for example, the name of the training set used to build a classification tree.
4. Interfaces and application programming interfaces (APIs) to other languages and systems. There are standard data mining APIs for Java and Structural Query Language (SQL). This includes, for example, a description of the API so that a classification tree can be built on data in an SQL database.
5. The overall process by which data mining models are produced, used, and deployed. This includes, for example, a description of the business interpretation of the output of a classification tree.
6. Standards for viewing, analyzing, and mining remote and distributed data. This includes, for example, standards for the format of the data and metadata so that a classification tree can be built on distributed Web based data.

In the following sections we discuss XML standards, data mining APIs, process standards, and emerging standards for integrating data mining into data webs and other emerging Web based applications.

This chapter is based in part on a previous summary of data mining standards (Grossman, Hornick, & Meyer, 2002).

XML STANDARDS

There are XML standards for the statistical and data mining models themselves that arise in data mining, as well as for the metadata associated with them, such as the metadata that specifies the settings for building and applying the models.

XML for Data Mining Models

The Predictive Model Markup Language (PMML) is being developed by the Data Mining Group (2002), a vendor led consortium that currently includes more than a dozen vendors such as Angoss; IBM; Magnify; MINEit; Microsoft; National Center for Data Mining at the University of Illinois (Chicago); Oracle; NCR; Salford Systems; SPSS; SAS; and Xchange (Data Mining Group, 2002a). PMML is used to specify the models themselves and consists of the following components:

1. Data dictionary. The data dictionary defines the attributes input to models and specifies the type and value range for each attribute.
2. Mining schema. Each model contains one mining schema that lists the fields used in the model. These fields are a subset of the fields in the data dictionary. The mining schema contains information that is specific to a certain model, whereas the data dictionary contains data definitions that do not vary with the model. For example, the mining schema specifies the

usage type of an attribute, which may be active (an input of the model), predicted (an output of the model), or supplementary (holding descriptive information and ignored by the model).

3. Transformation dictionary. The transformation dictionary defines derived fields. Derived fields may be defined by normalization, which maps continuous or discrete values to numbers; by discretization, which maps continuous values to discrete values; by value mapping, which maps discrete values to discrete values; or by aggregation, which summarizes or collects groups of values, for example, by computing averages.

4. Model statistics. The model statistics component contains basic univariate statistics about the model, such as the minimum, maximum, mean, standard deviation, median, and so forth of numerical attributes.

5. Model parameters. PMML also specifies the actual parameters defining the statistical and data mining models per se. Models in PMML Version 2.0 include regression models, clusters models, trees, neural networks, Bayesian models, association rules, and sequence models. Additional models are being planned for PMML Version 2.1. There is no mechanism in PMML to define statistical or data mining models that are not one of the supported types.

As an example, here is a data dictionary for Fisher's Iris data set:

```
<DataDictionary numberOfFields="5">
<DataField name="Petal_length" optype="continuous"/>
<DataField name="Petal_width" optype="continuous"/>
<DataField name="Sepal_length" optype="continuous"/>
<DataField name="Sepal_width" optype="continuous"/>
<DataField name="Species_name" optype="categorical">
<Value value="Setosa"/>
<Value value="Virginica"/>
<Value value="Versicolor"/>
</DataField>
</DataDictionary>
```

and here is the corresponding mining schema:

```
<MiningSchema>
<MiningField name="Petal_length" usageType="active"/>
<MiningField name="Petal_width" usageType="active"/>
<MiningField name="Sepal_length" usageType="supplementary"/>
<MiningField name="Sepal_width" usageType="supplementary"/>
<MiningField name="Species_name" usageType="predicted"/>
</MiningSchema>
```

Finally, here is a fragment describing a node of a decision tree built from the data:

```
<Node score="Setosa" recordCount="50">
<SimplePredicate field="Petal_length" operator="lessThan" value="24.5"/>
<ScoreDistribution value="Setosa" recordCount="50"/>
<ScoreDistribution value="Virginica" recordCount="0"/>
<ScoreDistribution value="Versicolor" recordCount="0"/>
</Node>
```

PMML Version 1.0 basically concerned itself with defining standards for various common data mining models assuming that the inputs to the models had already been defined. The inputs are called data fields. PMML Version 2.0 introduced the transformation dictionary, which contains derived fields. The inputs to models in PMML Version 2.0 may be data fields or derived fields. In principle, this approach is powerful enough to capture the process of preparing data for statistical and data mining models and for deploying these models in operational systems.

XML for Data Mining Metadata

Through the Object Management Group, a new specification for data mining metadata recently has been defined using the Common Warehouse Metadata (CWM) specification (Object Management Group, 2002). CWM supports interoperability among data warehouse vendors by defining document type definitions (DTDs) that standardize the XML metadata interchanged between data warehouses. The CWM standard generates the DTDs using the following three steps: First, a model using the Unified Modeling Language (Object Management Group, UML, 2002) is created. Second the UML model is used to generate a CWM interchange format called the metaobject facility (MOF)/XML metadata interchange (XMI) (Object Management Group, UML, 2002). Third, the MOF/XML is converted automatically to DTDs. CWM for data mining (CWM DM) was specified by members of the java data mining (JDM) expert group, and as such, has many common elements with JDM. For example, CWM DM defines function and algorithm settings that specify the type of model to build with parameters to the algorithm. CWM DM also defines tasks that associate the inputs to mining operations, such as build, test, and apply (score).

APIs

Data mining must coexist with other systems. In addition to XML standards such as PMML there are efforts defining data mining APIs for Java, SQL, and Microsoft's Object Linking and Embedding (OLE DB).

SQL APIs

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) are in the process of adopting a data mining extension to SQL. It consists of a collection of SQL user-defined types and routines to compute and apply data mining models.

ISO and IEC form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of international standards through technical committees. The data mining extensions in SQL are part of the SQL Multimedia and Applications Packages standard or SQL/MM. The particular specification, called SQL/MM Part 6: Data Mining, specifies an SQL interface to data mining packages.

Database users are interested in a standard for data mining so they can arbitrarily combine databases and the data mining based applications. From this perspective, data mining is just a sophisticated tool to extract information or to aggregate the original data. This is pretty similar to the functionalities that are provided by SQL today. Hence, standardizing data mining through SQL is a natural extension of the more basic storage and retrieval mechanisms supported by SQL today.

Java APIs

Turning to Java, the Java Specification Request 73 (JSR-73), known as JDM, defines a pure Java API to support data mining operations. These operations include model building, scoring data using models, and the creation, storage, access, and maintenance of data and metadata supporting data mining results (“Java Specification Request 73,” 2002). It also includes selected data transformations.

JDM not only defines a representative set of functions and algorithms, but also provides a framework so that new mining algorithms can be introduced. A key goal of the JDM team was to make it relatively easy to add additional models. Because many vendors specialize and support only certain data mining models, JDM is defined in a number of packages to support à la carte package compliance. For example, a vendor specializing in neural networks is likely interested in the classification and approximation functions, along with specific neural network algorithms and representations. JDM strives to make data mining accessible to both experts and nonexperts by separating high level *function* specifications from lower level *algorithm* specifications. JDM leverages the J2EE platform to enable scalability, integration with existing information systems, extensibility, and reusability, as well as choices of servers, tools, and components for vendor implementations.

JDM influenced and was influenced by several existing standards, such as SQL/MM, CWM DM, and PMML. JDM makes explicit provision for the import and export of data mining objects. The most common representation will be XML, in which PMML can be leveraged for model representations and CWM DM for other mining objects such as settings, tasks, and scoring output.

OLE DB APIs

Turning to Microsoft’s SQL environment, OLE DB for data mining (OLE DB for DM) defines a data mining API to Microsoft’s OLE DB environment (“Microsoft OLE DB,” 2002). OLE DB for DM does not introduce any new OLE DB interfaces, but rather uses an SQL-like query language and a specialized data structure called a rowset so that data mining consumers can communicate with data mining producers using OLE DB.

Recently, OLE DB for DM has been subsumed by Microsoft’s Analysis Services for SQL Server 2000 (Microsoft Corp., 2001). Microsoft’s Analysis Services provide APIs to Microsoft’s SQL Server 2000 services, which support data transformations, data mining, and online analytic processing (OLAP).

WEB STANDARDS

Semantic Web

The World Wide Web Consortium (W3C) standards for the semantic web define a general structure for knowledge using XML, RDF, and ontologies (World Wide Web Consortium, 2002). This infrastructure in principle can be used to store the knowledge extracted from data using data mining systems, although at present, one could argue that this is more of a goal than an achievement. As an example of the type of knowledge that can be stored in the semantic web, RDF can be used to code assertions such as “credit transactions with a dollar amount of \$1 at merchants with an MCC code of 542 have a 30% likelihood of being fraudulent.”

Data Web

Less general than semantic webs are data webs. Data webs are web based infrastructures for working with data (instead of knowledge). For example, in the same way that a web client can retrieve a web document from a remote web server using hypertext transfer protocol, with a data web, a data web client can retrieve data and metadata from a remote data web server (Bailey, 2000; National Center for Data Mining, 2002).

Data web client applications have been developed that allow users to browse remote data in data webs and set the PMML parameters defining the mining dictionary and transformation dictionary, enabling data webs to serve as the foundation for remote and distributed data mining (Grossman, 2001).

Other Web Services

More recently, Microsoft and Hyperion have introduced XML for Analysis, which is a simple object access protocol (SOAP) based XML API designed for standardizing data access between a web client application and an analytic data provider, such as an OLAP or data mining application (Microsoft Corp., 2001; Microsoft SQL, 2002). No details are available about XML for Analysis at this time (April 2002).

PROCESS STANDARDS

The CRoss Industry Standard Process for Data Mining (CRISP-DM) is the most mature standard for the process of building data mining models (Chapman et al., 2002). Process standards define not only data mining models and how to prepare data for them, but also the process by which the models are built and the underlying business problems the models are addressing. The CRISP 1.0 Process Model has six components: business understanding, data understanding, data preparation, modeling, evaluation, and deployment.

CRISP was an ESPRIT program that included Teradata, ISL (which was acquired by SPSS), DaimlerChrysler, and OHRA.

RELATIONSHIPS

Data mining is used in many different ways and in combination with many different systems and services. Each of these emerging standards and interfaces was developed to address a particular need, which is why there are several different standards.

XML standards such as PMML and CWM are a natural choice to support vendor neutral data interchange. PMML and CWM DM are complementary in that PMML specifies the detailed content of models, whereas CWM DM specifies data mining metadata such as settings and scoring output.

Application programmer interfaces such as SQL/MM and JDM use existing XML standards (PMML) for exchanging model data and metadata. Good coordination and good working relationships exist between the standards committees developing PMML, CWM, JSR-73, and SQL MM. For this reason we expect that terminology, concepts, and object structures will continue to be shared as these standards evolve.

XML standards such as PMML can serve as a common ground for the several different standards efforts in progress. CWM, JSR-73, and SQL MM use PMML concepts and infrastructure

when appropriate. In addition, for each large community interfacing to data mining systems there will be a continuing need for an API. This includes SQL, Java, Microsoft's COM and .Net, and the W3C's Semantic Web.

SUMMARY

As data mining has matured, so have the standards supporting it. PMML is a relatively mature XML standard for data mining models per se. Version 2.0 of PMML has begun to add support for the transformations, normalizations, and aggregations required for preparing data for modeling and for scoring and deploying models.

There are now well-defined APIs for interfacing data mining with SQL, Java, and Microsoft's COM and .Net environments. These APIs incorporate PMML when appropriate.

There are also emerging Web standards and services, such as WSDL, SOAP, and DSTP, for working with remote and distributed Web data. These standards hold the promise of opening up data mining into entirely new application domains.

REFERENCES

- Bailey, S., Creel, E., Grossman, R., Gutti, S., & Sivakumar, H. (2000). A high performance implementation of the Data Space Transfer Protocol (DSTP). In M. J. Zaki & C.-T. Ho (Eds.), *Large-scale parallel data mining* (pp. 55–64). Berlin: Springer-Verlag.
- Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinhertz, T., Shearer, C., & Wirth, R. (2002). CRoss Industry Standard Process for Data Mining Process (CRISP), CRISP 1.0 Process and User Guide. Retrieved March 10, 2002, from <http://www.crisp-dm.org>
- Data Mining Group. (2002a). Predictive model markup language (PMML). Retrieved March 10, 2002, from <http://www.dmg.org>
- Data Mining Group. (2002b). Retrieved March 10, 2002, from www.dmg.org
- Grossman, R., Creel, E., Mazzucco, M., & Williams, R. (2001). A dataspace infrastructure for astronomical data. In R. L. Grossman et al. (Eds.), *Data mining for scientific and engineering applications* (pp. 115–123). Dordrecht, The Netherlands: Kluwer Academic Publishers.
- Grossman, R., Hornick, M., & Meyer, G. (2002). Data Mining Standards Initiatives. *Communications of the ACM*, 45, 59–61.
- Java specification request 73. (2002). Retrieved March 8, 2002, from <http://jcp.org/jsr/detail/073.jsp>
- Microsoft Corp. (2002). Microsoft OLE DB for data mining specification 1.0. Retrieved March 8, 2002, from <http://www.microsoft.com/data/oledb/default.htm>
- Microsoft Corp. (2002). Microsoft SQL server 2000 analysis services. Retrieved March 8, 2002, from <http://www.microsoft.com/SQL/techinfo/bi/analysis.asp>
- Microsoft Corp. (2001). Microsoft analysis. Introduction to XML for analysis, Microsoft, April, 2001. Retrieved April 12, 2002, from <http://www.microsoft.com/data/xml/XMLAnalysis.htm>
- National Center for Data Mining. (2002). Data space transfer protocol. Retrieved March 8, 2002, from www.ncdm.uic.edu
- Object Management Group. (2002). Common warehouse metamodel—Data Mining. Retrieved March 8, 2002, from <http://cgi.omg.org/cgi-bin/doclist.pl>
- Object Management Group. (2002). Unified modeling language (UML). Retrieved May 10, 2002, from <http://www.omg.org/uml/>
- World Wide Web Consortium. (2002). Semantic web. Retrieved March 8, 2002, from <http://www.w3c.org/2001/sw>

III

Applications of Data Mining

20

Mining Human Performance Data

David A. Nembhard

University of Wisconsin-Madison

Introduction and Overview	463
Mining for Organizational Learning	464
Methods	464
Individual Learning	467
Data on Individual Learning	468
Methods	468
Individual Forgetting	474
Distributions and Patterns of Individual Performance	474
Other Areas	476
Privacy Issues for Human Performance Data	477
References	477

INTRODUCTION AND OVERVIEW

The mining of human performance data has potential for wide-reaching benefits in manufacturing and service organizations, for researchers and human behavior specialists. Until recently the vast majority of research in these areas was based on relatively small controlled experiments and resultant data. With the rapid growth of data acquisition systems in the workplace, there are now massive stores of data that have potential for use in addressing many existing research questions and in informing decision-making processes in these organizations themselves. The primary purpose of many of these data acquisition systems is often mundane, such as barcode tracking of raw materials and products, or payroll accounting systems that keep track of employees' working hours. However, the sheer volume of data collected often creates an

opportunity to mine these records as secondary data. Much as the data from point-of-sale systems are transforming marketing research, the records from bar-code scanners and networked systems offer vast new repositories of information on individual and group performance. As will be illustrated later in this chapter, a common key component of human performance data mining is the availability of temporal databases related to human work or activities.

The sections that follow describe several useful approaches for mining data related to human productivity. These approaches include measurements of organizational improvement and learning, individual learning, forgetting, human computer interaction, and cognitive science data.

MINING FOR ORGANIZATIONAL LEARNING

Scholars have studied learning behavior in organizations for more than a century. Important contributions have been made by behavioral psychologists studying the individual learning process, industrial engineers studying manufacturing costs, systems theorists studying organizational dynamics, and artificial intelligence scholars modeling the human thought process. More recently behaviorists have added important insights into corporate culture and mechanisms for promoting and managing change.

The use of learning curves for measuring and improving efficiency and cost has been demonstrated in numerous manufacturing and service organizations. Effective organizations use their experience to “tune” operations to do the same work at a lower unit cost. Observers of aircraft manufacture in the 1930s noted that “as the quantity of units manufactured doubles, the number of direct labor hours it takes to produce an individual unit decreases at a uniform rate” (Yelle, 1979). Unfortunately, the observed cost reductions result from a complex mix of individual learning, technology changes, economies of scale, and changes in management systems (Dutton & Thomas, 1985). Thus, the internal structure and causal mechanisms that affect organization-level learning curves are still not well understood (Argote, 1993).

Various researchers and practitioners have used systems analysis, case studies, and observation to find ways for companies to increase the rate of organizational learning. This “action research” agenda tries new approaches and observes their effects on organizational processes and outcomes (Argyris, 1989). These efforts have created a strong interest in helping employees to “learn how to learn” (Argyris, 1982; Senge, 1992). An increase in learning rates may provide benefits in terms of cost reduction, productive output, and efficiency. The measurement of organizational learning is also useful for cost estimation and production planning.

A second stream of work has studied the economic impact of midlevel organizational learning. They show how diversity in learning behavior can exert a complex influence on overall productivity (Adler, 1990; Kantor & Zangwill, 1991; Meredith & Camm, 1989). Both types of intermediate level research have focused more on cognition and problem solving than on motor learning and skill acquisition. They reflect an implicit assumption that learning in professional and managerial ranks will have a greater impact on organizational and economic performance. In particular, scholars want to help organizations foster “double loop learning,” in which an organization’s decision-makers learn how to learn more effectively (Argyris, 1982).

Methods

Organizational learning (productivity improvement) has been studied from a variety of perspectives. Most commonly, the overall productivity of the organization has been examined over a time scale to gauge the rate and overall effect of increased productivity with increased

organizational experience. Methods for achieving this have often centered on linear and non-linear regression. Example 1 illustrates a straightforward approach using linear regression.

Example 1: Regression Based Organizational Learning Model.

This example describes a linear regression based model of organizational learning adapted from Epple, Argote, and Devadas (1991), wherein the conventional learning curve is written in the form given in Equation 1, where y denotes output, x is cumulative output (as a surrogate for acquired experiential knowledge), T is the time worked, with C and r parameters related to the organization's characteristics. The larger the parameter r , the more rapidly productivity increases due to learning. Parameter C in this formulation is a time constant.

$$\frac{y}{T} = Cx^r \quad (1)$$

Because organizational data generally is stored based on a discrete time increment (e.g., hours, days, weeks), Equation 1 may be rewritten in a form suitable for estimating an organization's parameters given such discrete data, where t is a discrete time increment. Thus, Equation 2 allows for the prediction of an organization's output at time t , y_t , given cumulative experience up to the previous time increment, x_{t-1} .

$$\frac{y_t}{T_t} = Cx_{t-1}^r \quad (2)$$

To linearize this expression so that standard linear regression methods may be used, logarithms are taken of Equation 2, letting $c = \ln(C)$ and yielding a form more convenient for estimation as shown in Equation 3.

$$\ln(y_t/T_t) = c + r \ln(x_{t-1}) \quad (3)$$

To illustrate the application of Equation 3, consider the data for t , x_t , and y_t given in Table 20.1, where for simplicity each period is of equal length (i.e., $T_t = 1$ for all t). By taking logarithms as indicated in Equation 3 and using the values in Table 20.1, the dependent and independent variable terms may be written in matrix format as

$$Y = \begin{bmatrix} 2.48 \\ 2.71 \\ 3.00 \\ 3.04 \\ 3.22 \\ 3.26 \end{bmatrix} \quad \text{and} \quad X = \begin{bmatrix} 1 & 2.30 \\ 1 & 3.09 \\ 1 & 3.66 \\ 1 & 4.08 \\ 1 & 4.38 \\ 1 & 4.65 \end{bmatrix},$$

respectively. Using least squares estimation, the estimates of the model parameters, c , and r in Equation 3 may be determined using Equation 4 in which X' is the transpose of matrix X

TABLE 20.1
Fitting a Learning Curve Using Linear Regression: A Numerical Example

Time, t	0	1	2	3	4	5	6
Output, y_t	10	12	17	20	21	25	26
Cumulative output, x_{t-1}	—	10	22	39	59	80	105
In (x_{t-1})	2.30	3.09	3.66	4.08	4.38	4.65	
In (y_t/T_t)	2.48	2.71	3.00	3.04	3.22	3.26	

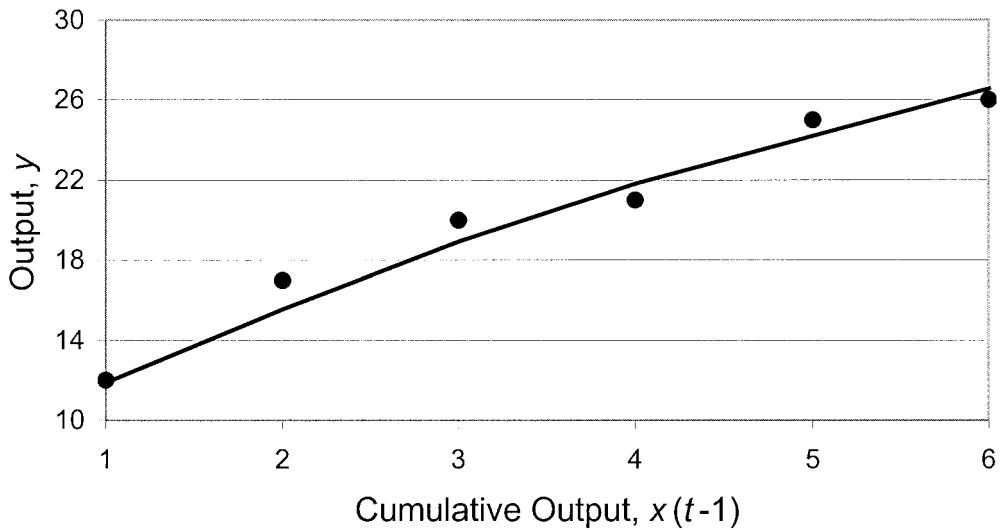


FIG. 20.1. Fitted linear model for Equation 3.

(see Neter, Kutner, Nachtsheim, & Wasserman, 1996 for a detailed discussion of the derivation and use of Equation 4).

$$\begin{pmatrix} c \\ r \end{pmatrix} = (X'X)^{-1}(X'Y) \quad (4)$$

The least squares parameter estimates for the data given in Table 20.1 are then $\begin{pmatrix} c \\ r \end{pmatrix} = \begin{pmatrix} 1.688 \\ 0.342 \end{pmatrix}$ with the corresponding plot of the fitted line and original data as shown in Fig. 20.1.

Equation 3 may be extended as shown in Equation 5, with an additional parameter, α , to represent potential diminishing returns on labor. Equation 3 is a special case of Equation 5 in which $\alpha = 1$.

$$\ln(y_t) = c + \alpha \ln(T_t) + r \ln(x_{t-1}) \quad (5)$$

The form in Equation 5 is suitable for estimating and predicting organizational learning characteristics given for each time period t , the productivity during t , the cumulative organizational experience from all previous periods, x_{t-1} , and the present time T_t . Additional information may be mined from an organization's production history to answer a broader range of questions such as whether the number of hours in a work shift affects the diminishing returns on increased labor hours. Adding an additional term, $\beta \ln(n_t)$, to the right hand side of Equation 5 yields such a model, where n_t represents the number of shifts per week, and h_t , the number of hours per shift with organizational parameters α and β , respectively. Note that by definition $T_t = h_t n_t$, making Equation 5 a special case of Equation 6, where $\alpha = \beta$.

$$\ln(y_t) = c + \alpha \ln(h_t) + \beta \ln(n_t) + r \ln(x_{t-1}) \quad (6)$$

The implicit assumption in forming the models shown in Equations 5 and 6 is that knowledge acquired through learning by doing does not depreciate (i.e., there is no forgetting effect). Models of organization knowledge depreciation vary widely. However, in the context of a linear model this may be represented by replacing the cumulative experience term x with a

cumulative knowledge term that accounts for the depreciation of knowledge over time with depreciation parameter λ , as given in Equation 7.

$$\ln(y_t) = c + \alpha \ln(h_t) + \beta \ln(n_t) + r \ln \left(\sum_{s=1}^{t-1} \lambda^{t-s-1} x_s \right) \quad (7)$$

Depending on the information content in the data and the decision support questions being raised, a greater or lesser number of terms may be included in such models and fitted in an analogous manner to that shown for Equation 3. In general it will be difficult to calibrate the choice of model *a priori*, because the models have not yet been fitted to the data. However, a common approach to addressing this issue is to fit the data to each of the candidate models to help gauge both the appropriateness of the models and information content in the data.

A complete review of the organizational learning literature is broader than space permits. However, the interested reader can find recent comprehensive overviews of the organizational learning literature by Argote (1999), Cohen and Sproull (1996), and Dar-El (2000). Argote (1999) discusses data driven models such as that illustrated in example 1. Another segment of the literature is conceptual in nature such as the compilation by Cohen and Sproull (1996). A third related segment of the literature is prescriptive in nature and is based somewhat on previous results, many of which were data driven. Additional prescriptive outlines on improving organizational performance can be found in Druckman, Singer, and VanCott (1997) and Druckman and Bjork (1991, 1994).

INDIVIDUAL LEARNING

There are gaps in the understanding of human learning that go beyond the interaction of groups within organizations and extend to the individual level. An important limitation to research and decision-making advancements in this area has long been the difficulty in going beyond the laboratory to obtaining access to adequate field data and the subsequent knowledge embedded therein. Recently data warehousing efforts and information systems technologies have helped by providing sources of human performance data from the field. To analyze these data, effective tools are required to allow us to measure and preferably visualize a multiplicity of human behaviors.

At the individual level, human performance and learning is affected by many factors including individual ability; individual variability; financial incentives; organizational norms and constraints; task complexity; training; the nature of the social environment; technology; and other factors. Learning and improvement also occur in teams, work groups, departments, and other collections of people. In these groups interactions, both obvious and subtle, occur among the technologies, workers, and management systems that often lead to measurable performance improvement (Argote, 1993).

Knowledge of individual learning characteristics and distributions of these characteristics may be related to a number of factors at the individual level, such as workforce training, education, and the local industrial base. Building an understanding of how these factors affect learning rates may be central to improving them. As with organizational level learning, the literature on individual learning is extensive. Belkaoui (1986), Dar-El (2000), and Yelle (1979), provide comprehensive overviews of the numerous models and functional forms for measuring individual human learning behavior. Comparative studies of many of these and other models are given by Badiru (1992) and Nembhard and Uzumeri (2000a). Several of these models also are shown in a later section.

Data on Individual Learning

Data on individual learning generally takes the form of an individual-related spacio-temporal database. At a minimum an individual record contains information such as an individual (human) identifier, a time stamp, and a location. Additionally, information on quality, complexity, teamwork, and other characteristics of the work may be beneficial but are not always available. The unit of analysis may vary, but it may be convenient to use a unit of production (batch), a part, or in the case of a service related task, a customer. The spatial dimension tends not to constrain the ability to mine such data.

Methods

The mining of human productivity data often has been based on individual curve estimation, or regression analyses. However, there may be opportunities to combine some of these with other techniques in the future, including classification, clustering, and online analytical processing (OLAP; see, e.g., Hasan & Hyland; 2001; Li & Wang, 1996). Regression analyses by Argote (1993), Argote Beckman and Epple (1990), and Epple et al. (1991) allow for the examination of learning characteristics among various groups. Curve estimation models by Nembhard (2000), Shafer, Nembhard, and Uzumeri (2001), and Uzumeri and Nembhard (1998) have helped develop knowledge about individual performance in industrial settings.

Models of organizational learning are effective and valuable when the type of activity and the granularity of the data warrant this view. However, greater detail may allow for the use of individual based models of organizational learning (e.g., Nembhard & Uzumeri, 2000a). The distinction between an individual based model of organizational learning and a model of individual learning is somewhat one of name and desired purpose. For example, Fig. 20.2a

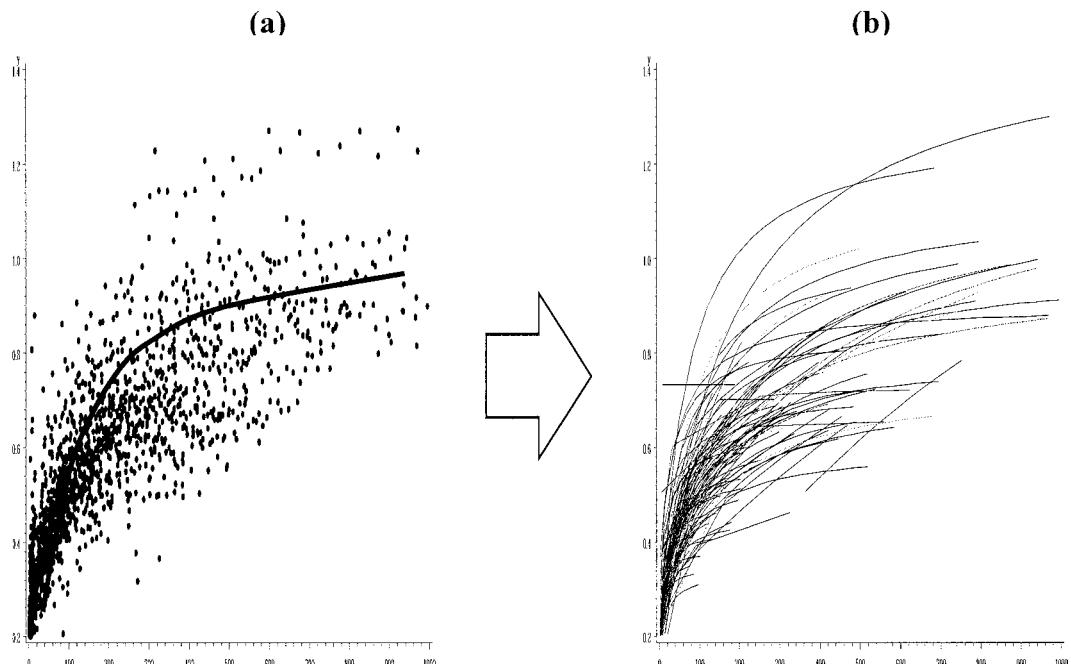


FIG. 20.2. Example of organizational and individual based models.

illustrates the organizational learning curve model given in Equation 3 fitted to organizational data. This view can adequately represent an organization's progress over time. However, in Fig. 20.2b, the same data may be fitted to a set of individual learning models to achieve a similar goal. It is clearly necessary for the data to contain identifiers of individual workers, teams, worker classifications, or other categories for this alternate view to be used. An individual based model then allows for further examination of individual characteristics, capabilities, variations, and potential for improvement. That is, the parameterizations of the individual characteristics can be related to the tasks, technologies, individual experience, age, gender, and education, and so forth. This type of secondary analysis has provided useful insights into human behavioral characteristics (see, e.g., Lance et al., 1998; Nembhard, 2000).

By taking an individual based perspective to organizational learning and forgetting behavior, a mathematical function may be used to describe the observed performance history in much the same manner as curves may be fitted to an organization's history. Standard curve-fitting methods can be applied to produce curves that capture the essential shape of the underlying "signal," as illustrated in Fig. 20.1. Some useful examples of nonlinear curve fitting methods may be found in Dennis and Schnabel (1983). Many of these and similar methods are available in off-the-shelf software such as SAS, SPSS, Minitab, and so forth. The fitted curves provide a filtered summary of the process that is more compact and offers several important benefits including the following.

1. *Reduced data dimensionality.* By fitting the data to an appropriate continuous function that describes the underlying signal in the data, the volume of information required to describe the process is reduced. For example, Fig. 20.3 illustrates 18 discrete data points representing

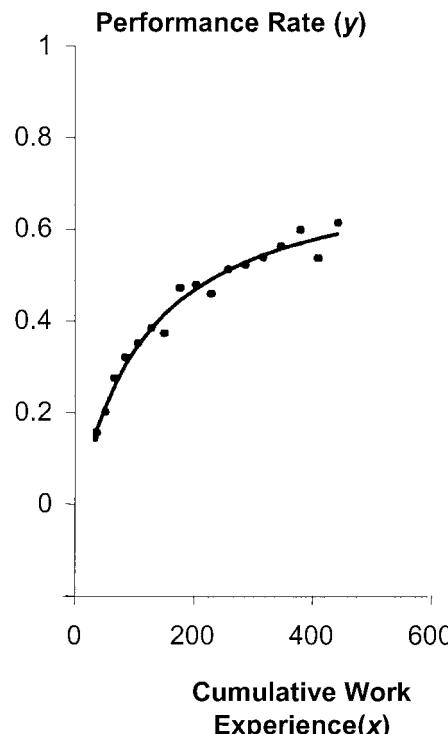


FIG. 20.3. Fitted individual learning curve (using Equation 11).

productivity rates, each with a pair of (x, y) coordinates that are described by a continuous function of only three parameters. Reduced dimensionality may facilitate visualization for decision makers while maintaining the embedded knowledge that aids comprehension and insight.

2. *Noise reduction.* By fitting a continuous function, noise contained in the discrete data is inherently reduced, assuming that the model chosen is appropriate for the human characteristic being modeled. When relevant, the distribution of the noise can be measured and recorded.

3. *Implicit mathematical analysis.* Because the signal pattern is described by the variables in the best-fit equation, an analyst can use the fitted equation to directly perform a number of otherwise complex calculations. For example, the derivative of the fitted function provides an estimate of the signal slope at a given point, which represents the instantaneous rate of improvement. Such information might then be aggregated across individuals and related to specific workplace interventions. For example, we may be able to address questions such as how a change in procedure affects the learning rates across a population of workers.

A general approach for mining human performance data for individual learning and forgetting characteristics and skill measurement is described as follows. Individual productive performance, y_i , can be modeled based on a nonlinear model, f_i , and model parameters, $\underline{\theta}_i$, that best fit the data set for an individual unit i , such that the fitted curve is given by,

$$y_i = f_i(\underline{x}_i, \underline{\theta}_i) + \varepsilon_i, \quad (8)$$

where \underline{x}_i represents the data for curve i , with error term ε_i . That is, in fitting a curve for each individual i , we first select the most preferred function, f_i , for an individual, and then estimate the parameters $\underline{\theta}_i$ from the data using an appropriate nonlinear estimation technique. The underscores on x_i and θ_i indicate vectors the length of which depends on the data and model used, respectively.

When considering multiple individuals, there are two predominant methodological approaches. The first is to select both the best model and the best model parameters for each individual based on a model selection approach such as maximum likelihood estimation (MLE) or least squares estimation (LSE) (see, e.g., Gujarati, 1978). This tends to minimize model selection error because a variety of functional forms, f_i , may be considered, with the best candidates chosen for each individual. However, if comparisons among a large number of individual units, i , is desirable, this approach may be somewhat cumbersome because different models may be most preferred for the various individuals. This realization motivates the second approach, which is to select a single consistent function, g , to be used for all of the individuals in the data being considered. That is, for each individual i fit the model given by

$$y_i = g(\underline{x}_i, \underline{\theta}_i) + \varepsilon_i. \quad (9)$$

By requiring a single model, g , the consistency and relative ability to compare individual units that a single model affords introduces an additional model selection error, δ_i , as given in Equation 10.

$$\delta_i = [f_i(\underline{x}_i, \underline{\theta}_i) - g(\underline{x}_i, \underline{\theta}_i)]. \quad (10)$$

The choice of model, g , can be based on a model scoring approach that maps model-data combinations to numbers the numerical ordering of which corresponds to a preference ordering over the space of models, given the data (Glymour, Madigan, Pregibon, & Smyth, 1996).

Popular rules include the Akaike information criterion (Akaike, 1974), minimum description length (Rissanen, 1978), and Bayes information criterion (Raftery, 1995). A specific scoring system for selecting an individual based learning model is described by Nembhard and Uzumeri (2000a), in which the criteria include model efficiency, stability, and parsimony. Criteria may additionally include other considerations including estimation balance, the interpretability of resultant parameters, and the ability to estimate the model parameters. Several of these potential criteria are described in turn later.

Efficiency. It is important that a model be flexible with respect to the potential shapes of the responses. That is, it should describe any of the regularly observed patterns within the population of individuals. In the case of measuring learning and forgetting, for example, it is beneficial to have the ability to fit both positive and negative improvements and any prior learning based on previous task experience. An efficient model is flexible enough to capture the various shapes of learning among the individuals in a population. For the efficiency measure the average of the mean squared error (MSE) statistics, given by $\hat{\mu}_{MSE}$, across individuals is useful wherein lower $\hat{\mu}_{MSE}$ values indicate a better average fit among all individuals.

Stability. In addition to a good average fit, it is important that every curve in the population of curves fits the model reasonably well. An efficient model may fit a given curve extremely well but fit other individual curves relatively poorly. A stable model is not overly dependent on specific individuals to obtain good model fits (Borowiak, 1989). Stability allows a model to be robust with respect to variations between individuals within the population. Stability may be measured using the variability in the goodness-of-fit, given by $\hat{\sigma}_{MSE}$, for the population of individual curves. Lower $\hat{\sigma}_{MSE}$ values indicate that the model fits individuals in the population more consistently. For instance, given a relatively efficient model, high values of $\hat{\sigma}_{MSE}$ would indicate low stability and, hence, the presence of poor model fits for some individual curves.

Balance. A third scoring criterion is that of estimation balance, in which a balanced model tends to have an equal probability of providing overestimates and underestimates, an overestimate relates to the model *predicting* a higher output than the actual output, on average, and underestimates, a lower output than actual. This may be particularly important if the model is to be used for forecasting, in which imbalance creates a general bias toward overestimation or underestimation. It would generally be desirable to have little or no bias introduced as a result of the model choice. The skewness of the goodness-of-fit may be used as the measure for balance, or alternately, a raw percentage score as used, for example, by Nembhard and Osofsky (2001).

Parsimony. In general, it is desirable to limit the number of degrees of freedom used by a model, particularly when the data corresponding to the least data-rich individuals is relatively sparse. This parsimony is additionally useful if a model can be limited to two or three parameters, because they are more readily summarized in two-dimensional or pseudo-three-dimensional graphical representations. We might expect, *a priori*, that a four-parameter model would tend to fit an individual's data somewhat better than a model with fewer parameters, simply in terms of overall flexibility. However, there is a clear tradeoff between reducing the dimension of the parameter space at the expense of increasing the residual error. The number of model parameters is useful as a measure of such parsimony.

Estimableness. The relative ease or difficulty in fitting a model to each individual's performance history largely depends on the curve fitting algorithm, the starting conditions,

step sizes, and termination criteria. In general, it may not be feasible for a single set of starting conditions to result in model convergence for all of the individuals in a data set. For large data sets and/or automated analyses the estimableness of model parameters given a standardized estimation procedure is an important consideration. A potential measure of estimableness is the percentage of individuals for which model parameter convergence can be obtained based on a consistent method and starting conditions. For example, other things being equal one would prefer parameter convergence in 99% of individuals for one model versus only 95% for another model.

Interpretability. Another potentially valuable model selection criterion, interpretability, measures the degree to which a model's parameters have clear contextual interpretations. It is preferable for each parameter to have clear and useful meanings for decision makers. This is somewhat more likely when the models were developed from theoretical bases as opposed to empirically, because theoreticians often start with a set of rational and interpretable parameters and then build models around them based on current knowledge in the discipline. For example, this is true for the model used later, in example 2 (Equation 11).

The literature reveals numerous functional forms, g , for measuring human learning and forgetting. Psychologists have studied and modeled individual learning processes (Mazur & Hastie, 1978), whereas engineers have modeled learning as it relates to manufacturing costs (Yelle, 1979), process times (Adler & Nanda, 1974a, 1974b; Axsater & Elmaghraby, 1981; Pratsini, Camm, & Raturi, 1993; Smunt, 1987; Sule, 1981), setup time learning (Karwan, Mazzola, & Morey, 1988; Pratsini, Camm, & Raturi, 1994), and line balancing (Dar-El & Rubinovitz, 1991). Although the majority of work has centered on the log-linear model (e.g., Badiru, 1992; Buck & Cheng, 1993; Glover, 1966; Hancock, 1967), there have been continuing efforts to identify alternative formulations. Some alternative forms were found to represent observed behavior better than the log-linear model in their respective industrial settings (e.g., Argote et al., 1990; Asher, 1956; Carlson & Rowe, 1976; Carr, 1946; Ebbinghaus, 1885; Elmaghraby, 1990; Globerson, Levin, & Schub, 1989; Knecht, 1974; Jaber & Bonney, 1996; Levy, 1965; Nembhard & Uzumeri, 2000a; Pegels, 1969; Ramos & Chen, 1994).

Example 2. This section summarizes a numerical example of applying the above approach using real data from 3,874 individuals. The example is adapted from Nembhard and Uzumeri (2000a). Mazur and Hastie (1978) discuss a model of individual learning developed from psychological principles and later tested using experimental data (Mazur & Hastie, 1978) and field data (Uzumeri & Nembhard, 1998). The model is intended to describe learning based on three fitted parameters, k , p , and r , where p represents the prior expertise attributable to a task based on a fit of the model to the data and may be viewed as an estimate of a workers' expertise acquired from past and similar experience. It is in effect shifting the learning curve backward in cumulative work to estimate prior expertise. The fitted parameter k estimates the asymptotic *steady-state* productivity rate per unit time relative to the work standard. This is the rate that can be expected once all learning has been completed. The fitted parameter r is the cumulative production and prior expertise reach $k/2$, starting from the production rate corresponding to zero cumulative work and prior expertise. Thus, r represents the rate of learning *relative* to the individual's steady state productivity rate, k . Note that this definition holds for both positive and negative learning, and that smaller values of r correspond to a more rapid approach to steady state. The response variable, y_x , is a measure of the productivity rate corresponding to x units of cumulative work (i.e., the number of units completed thus far).

$$y_x = k \left(\frac{x + p}{x + p + r} \right) + \varepsilon_x \quad (11)$$

TABLE 20.2
Performance of Learning Curve Models

Model Name	Form	Efficiency, $\hat{\mu}_{MSE}$ (change in error)	Stability, $\hat{\sigma}_{MSE}$ (change in error)	Parsimony, Number of Parameters
Hyperbolic-3	$y = k[(x + p)/(x + p + r)]$.00577 (0%)	.01183 (0%)	3
Exponential-3	$y = k(1 - e^{(x+p)/r})$.00598 (+3.6%)	.01382 (+16.8%) ^a	3
DeJong	$y = C_1[M + (1 - M)x^{-b}]$.00665 (+15.3%)	.01331 (+12.5%)	3
Stanford-B	$y = C_1(x + B)^b$.00671 (+16.3%)	.01340 (+13.3%)	3
Log-linear	$y = C_1x^b$.00711 (+23.2%)	.01434 (+21.2%)	2
S-curve	$y = C_1[M + (1 - M)(x + B)^b]$.00736 (+27.6%)	.01506 (+27.3%)	4
Hyperbolic-2	$y = k[x/(x + r)]$.00873 (+51.3%)	.01410 (+19.2%)	2
Pegels	$y = A x^{a-1} + b$.00984 (+70.5%)	.01327 (+12.2%)	3
Exponential-2	$y = k(1 - e^{-x/r})$.01122 (+94.5%)	.01563 (+32.1%) ^a	2
Levy	$y = [1/\beta - (1/\beta - x^b/C_1)k^{-kx}]^{-1}$.08096 (+1300%)	.12561 (+962%) ^a	4
Knecht	$y = C_1 x^b e^{cx}$.09817 (+1600%)	.16354 (+1280%)	3

Note: Table is adapted from Nembhard and Uzumeri (2000a).

^aDenotes: convergence criterion not met for some of the individual curves.

Table 20.2 illustrates the use of several model selection criteria (efficiency, stability, parsimony, estimableness) to select a single learning model for multiple individuals. Noting that lower values are preferred for the first three criteria, the hyperbolic-3 model shown in Fig. 20.4 may be preferred, because it performed best on the first two criteria. However, in general such criteria may produce a set of nondominated alternatives, making the eventual choice among models somewhat less clear.

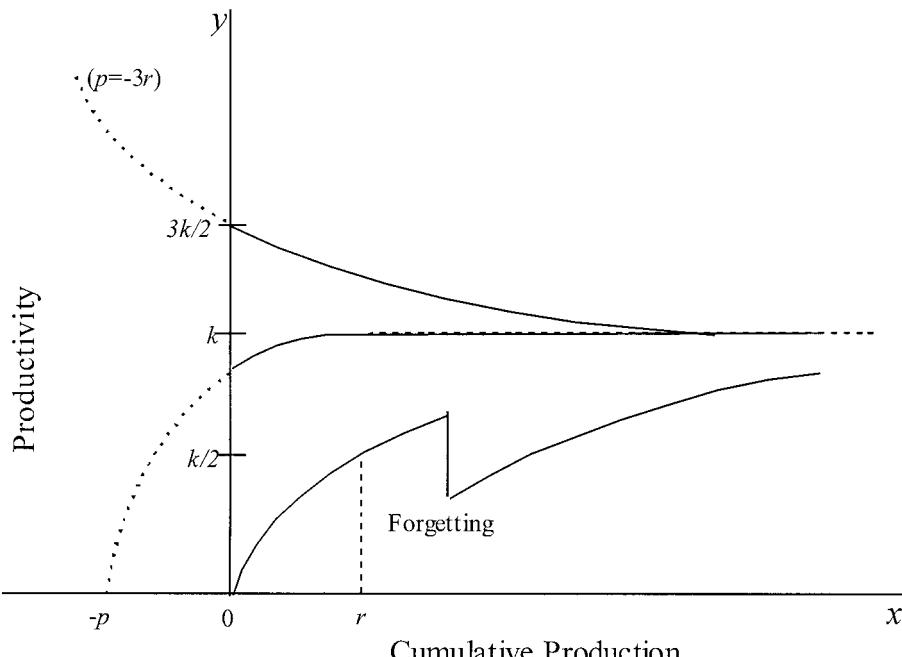


FIG. 20.4. Hyperbolic-3 learning model (in Equation 11).

Individual Forgetting

Research addressing the forgetting or the retention of learned skills suggests that the longer a person studies, the longer the retention (Ebbinghaus, 1885). The forgetting process has been shown to be describable by the traditional log-linear performance function (Anderson, 1985). Based on individuals' ability to recall nonsense syllables, Farr (1987) found that as the meaningfulness of the task increased, retention also increased. The retention of learning at an organizational level also has been found to decline rapidly (Argote et al., 1990). Several research efforts attempted to explain the impact that forgetting has on production scheduling (Fisk & Ballou, 1982; Khoshnevis & Wolfe, 1983; Smunt, 1987; Sule, 1983). Globerson et al. (1989) developed a model of learning and forgetting for a data correction task. The model indicates that forgetting behaves much like the mirror image of the learning process. Bailey (1989) found that forgetting of a procedural task was a linear function of the product of the amount learned and the log of the elapsed time. Nembhard and Osothsilp (2001) presented a survey and comparative study of many of the forgetting models in the literature employing an approach similar to that illustrated in example 2 for the learning process. The study compares forgetting models from Carlson and Rowe (1976), Elmaghriby (1990), Globerson and Levin (1987), Globerson, Levin, and Ellis (1998), Globerson et al. (1989), Jaber and Bonney (1996), and Nembhard and Uzumeri (2000b).

DISTRIBUTIONS AND PATTERNS OF INDIVIDUAL PERFORMANCE

Although modeling individual human performance has many potential direct uses for decision makers and researchers, additional insights may be obtained by studying the *distributions* of improvement behavior across populations of individuals. For example, Fig. 20.5 depicts parameter estimates for k , p , and r , from Equation 11 in three orthogonal views (a): $r \times k$; (b): $r \times p$; (c): $k \times p$. These distributions in general show that workers tend to fall along a continuum. At one extreme, workers learn quickly and then plateau at a relatively low level of performance (bottom left of Fig. 20.5a). At the other extreme, some workers improve slowly and steadily, but climb to high ultimate performance levels (upper right of Fig. 20.5a). Regardless of the mechanisms behind such behavior, these *maps* of individual performance, such as that in Fig. 20.5, can help an organization to build a quantitative description of human performance within its workforce. However, applications extend beyond managerial curiosity. Organizational learning is a group characteristic, and many managerial interventions (e.g., education, training, and technology) are designed to change both individual and group performance. Managers may be more concerned about the mix of learning curve shapes in the workforce than in changing the curves of specific individuals. To deal with individual curves, one would abandon the power of statistics and understand the many subtle forces acting on that specific individual. That is, organizational improvements are more likely to be due to aggregate changes in the mean or the shape of the learning distributions than from changes in a few specific individuals.

Focusing on the group level distributions of learning curve shapes may provide clues to systemic influences that would otherwise be difficult to discern. One straightforward approach for forming such a distribution is to consider the parameter estimates themselves as drawn for a multivariate distribution. Thus, one would estimate multivariate distribution parameters given the population of individual parameter estimates. Figure 20.5 illustrates such a multivariate distribution of three parameters. Using these parameter estimates, one may fit a given

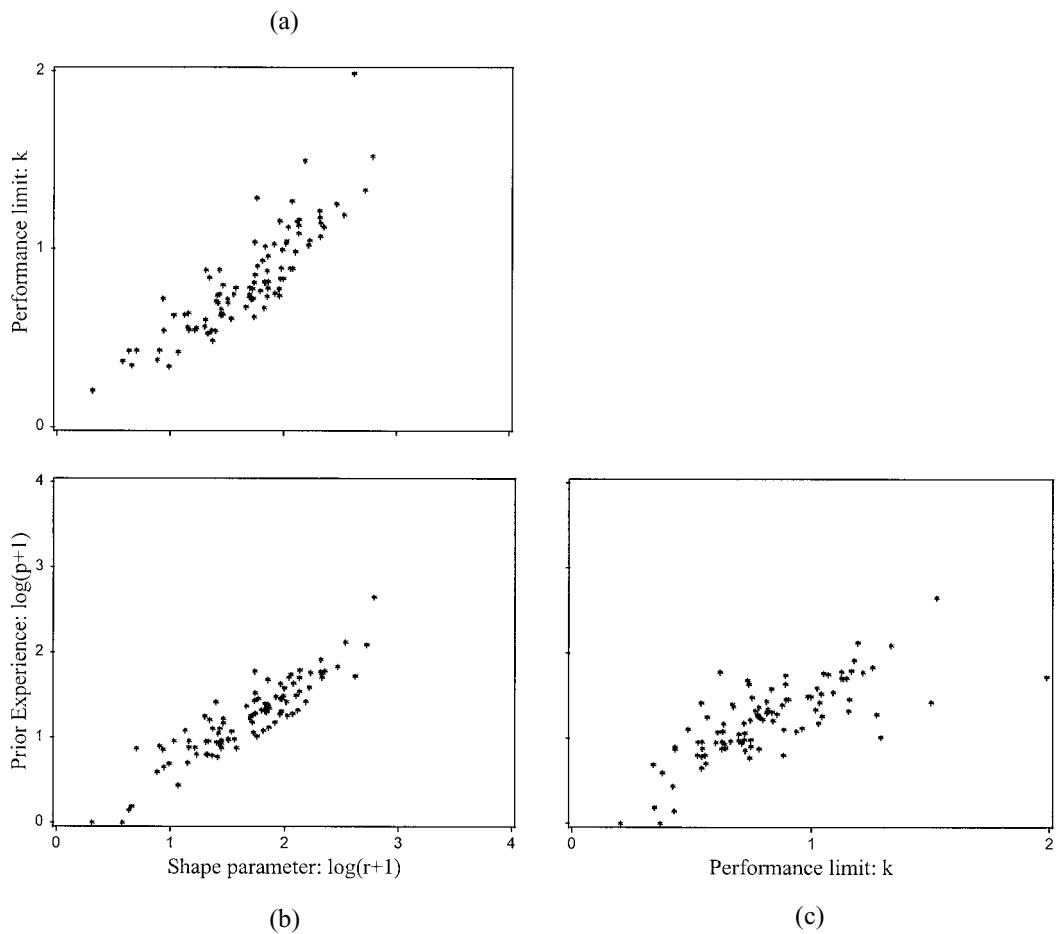


FIG. 20.5. Orthogonal views of the distributions of parameters k , p , r .

distribution such as a multivariate normal (MVN) to further simplify the representation of the original data. For example, the original data for Fig. 20.5 consists of hundreds of observations for each of 110 individuals. The MVN distribution consists of a 3×1 vector of means, and a 3×3 variance–covariance matrix.

The regularity of the learning map in Fig. 20.5 suggests that although parameter estimates are highly variable across individuals, they may be relatively predictable in the aggregate. For instance, we might view a newly hired individual as a random trial selected from the workplace distribution. Managers and researchers may employ statistical modeling and analysis tools to describe and apply the distributions involved to simulations, worker-task assignment, scheduling, line balancing, and other disciplines.

Empirical distributions may be particularly useful when rapid organizational change is taking place. For example, a company opening a plant in a previously unindustrialized area could estimate maps based on early performance data at the new plant or on educational demographics in the region. By comparing these maps to those of mature plants, one might identify systematic differences in the workforces before they are visible directly, thereby providing better production and cost estimates. Maps may also help management to better understand the nature of heterogeneity in their workforces, which has potential for creating

new opportunities for taking patterns of worker variability into account in making key decisions. As an example, consider a company that is introducing a new product line. Should the company make the product in a new plant with new workers? Should it make the product in an old plant with experienced workers and shift other products to the new plant? Or should it simply expand its existing plant and use a mix of experienced and inexperienced workers?

Finally, it is important to note that the fitted curves represent a *descriptive* model of learning behavior that does not need to assume a specific causal mechanism. When used in this manner, descriptive curves can be fit to any variable. Instead of production rates, maps could be drawn for reductions in defect rates, for changes in cycle times, or for improvements in worker safety.

To this point, the discussion has assumed modeling of all available data. It should be noted that sampling individual performance records is also a viable approach, particularly when extremely large data sets are available and the data miner intends to estimate a distribution. For sampling, a variety of statistical sampling methods are available. Descriptions of these approaches may be found in Kowalewski and Tye (1990).

At the opposite end of the spectrum there may be circumstances in which data are relatively sparse and aggregation is necessary to obtain useful quantities of information. Although there is no standard established for aggregating human performance data, we may consider various attributes of the individual and the task conditions as candidate variables on which to aggregate. For example, one might aggregate all subjects performing a specific task, or a category of tasks. Alternatively, one might aggregate by years of experience, human skill level, or seniority level, among other variables. The choice of where to aggregate may ultimately be best driven by the intended purpose of the data mining efforts.

OTHER AREAS

The area of human-computer interaction has elements in common with human performance in that it deals with human characteristics and behavior given specific inputs or requirements. Researchers have made use of neural networks, pattern recognition, and visualization approaches for understanding how humans respond and perform computer related tasks (see, e.g., Beale & Finlay, 1992; Brecht & Jones, 1988; Rasmussen, 1983). A large portion of past work relied on closely controlled experiments to expand scientific understanding of underlying human mechanisms and behaviors. However, there are increasing opportunities for mining larger data sets in this area in much the same manner as that for human productivity. For example, given the wealth of data generated daily from Web based activities, information helpful to the understanding of human-computer interaction has significant data mining potential.

There is a relatively long history of research in the cognitive sciences that makes use of human performance data. Much of this work has been in the area of experimental psychology, which seeks to understand the underlying causality and mechanisms by which humans' performance is affected by various internal and external factors. For example, Ebbinghaus (1885) conducted numerous experiments to uncover the underlying mechanisms of human memory. Although these early efforts dealt with modest quantities of data by today's standards, some of these approaches are scaleable to the volumes of data generated by modern organizations.

The cognitive and physiological mechanisms by which individuals acquire knowledge have been carefully examined. Centered in behavioral psychology, artificial intelligence and industrial engineering, this research stream has generated sophisticated models of the processes by which humans acquire and retain knowledge (Anderson, 1982; Hancock, 1967; Mazur & Hastie, 1978).

PRIVACY ISSUES FOR HUMAN PERFORMANCE DATA

Maintaining individual privacy and organizational confidentiality is important and often necessary when dealing with human performance information. Research related studies are generally required to have human subject protocols in place, thereby protecting subjects' privacy. However, as organizations themselves begin to employ the approaches developed by researchers, concern arises from workers and labor organizations seeking to protect workers from potential or perceived abuses of the information.

Employee performance is evaluated in a variety of ways across industries and organizations to make decisions including promotions, layoffs, transfers, and raises. Given that it is not unusual for such evaluations to be made purely based on the judgment of supervisors and peers, one might expect that a more objective measure of performance based on automated data acquisition systems would be openly welcomed by workers. However, although workers in some organizations embrace these new tools, they are not universally accepted. One means of alleviating such fear is to openly communicate what is to be measured and how it will be used, including the protection of privacy when appropriate. For example, if data are collected for the purposes of pay-for-performance, employees are likely to be relatively receptive. Whereas, if their individual learning characteristics are determined, they may not want this information to become public knowledge. A "double blind" is often effective, in which the data miner obtains data with individual identifiers pre-coded to "anonymize" the individuals. Prior to any analysis, the miner recodes the individual identifier so that those with later access to any analysis will be unable to reassociate the information with specific individuals. Because human performance data mining often yields distributions, the organizations can focus on how the group behaves, which may be more acceptable, and potentially no less valuable, to those being studied, their representatives, and their organizations.

REFERENCES

- Adler, P. S. (1990). Shared learning. *Management Science*, 36, 938–957.
- Adler, G. L., & Nanda, R. (1974b). The effects of learning on optimal lot size determination: Multiple product case. *AIEE Transactions*, 6, 20–27.
- Adler, G. L., & Nanda, R. (1974a). The effects of learning on optimal lot size determination: Single product case. *AIEE Transactions*, 6, 14–20.
- Akaike, H. (1974). A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19, 716–723.
- Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 89, 369–406.
- Anderson, J. R. (1985). *Cognitive psychology and its implications* (2nd ed.). New York: W. H. Freeman.
- Argote, L. (1993). Group and organizational learning curves: Individual, system and environmental components. *British Journal of Social Psychology*, 32, 31–51.
- Argote, L. (1999). *Organizational learning: Creating and transferring knowledge*. Boston: Kluwer Academic Publishers.
- Argote, L., Beckman, S. L., & Epple, E. (1990). The persistence and transfer of learning in industrial settings. *Management Science*, 36, 140–154.
- Argyris, C. (1989). Participatory action research and action science compared. *American Behavioral Scientist*, 32, 612–623.
- Argyris, C. (1982). How learning and reasoning processes affect organizational change, In P. S. Goodman (Ed.), *Change in organizations: New perspectives on theory, research and practice*. San Francisco: Jossey-Bass.
- Asher, H. (1956). *Cost and quantity relationships in the airframe industry* (Report No. R291). Santa Monica, CA: Rand Corp.
- Axsater, S., & Elmaghriby, S. E. (1981). A note on EMQ under learning and forgetting. *AIEE Transactions*, 13, 86–90.

- Badiru, A. B. (1992). Computational survey of univariate and multivariate learning curve models. *IEEE Transactions on Engineering Management*, 39, 176–188.
- Bailey, C. D. (1989). Forgetting and the learning curve: A laboratory study. *Management Science*, 35, 340–352.
- Beale, R., & Finlay, J. (1992). *Neural networks and pattern recognition in human-computer interaction*. New York: Ellis Horwood.
- Belkaoui, A. (1989). *The learning curve*. London: Quorum Books.
- Borowiak, D. S. (1989). *Model discrimination for nonlinear regression models*. New York: Marcel Dekker.
- Brecht, B., & Jones, M. (1988). Student models: The genetic graph approach. *International Journal of Man Machine Studies*, 28, 483–504.
- Buck, J. R., & Cheng, S. W. J. (1993). Instructions and feedback effects on speed and accuracy with different learning curve models. *IIE Transactions*, 25, 34–47.
- Carlson, J. G., & Rowe, R. G. (1976). How much does forgetting cost? *Industrial Engineering*, 8, 40–47.
- Carr, G. W. (1946). Peacetime cost estimating requires new learning curves. *Aviation*, 45, 76–77.
- Cohen, M. D., & Sproul, L. S. (Eds.). (1996). *Organizational learning*. Thousand Oaks, CA: Sage.
- Dar-El, E. M. (2000). *Human learning: From learning curves to learning organizations*. Boston: Kluwer Academic Publishers.
- Dar-El, E. M., & Rubinovitz, J. (1991). Using learning theory in assembly lines for new products. *International Journal of Production Economics*, 25, 103–109.
- Dennis, J. E. Jr., & Schnabel, R. B. (1983). *Numerical methods for unconstrained optimization and nonlinear equations*. Englewood Cliffs, NJ: Prentice Hall.
- Druckman, D., Singer, J. E., & VanCott, H. (Eds.). (1997). *Enhancing organizational performance*. Washington, DC: National Academy Press.
- Druckman, D., & Bjork, R. A. (Eds.). (1994). *Learning, remembering, believing*. Washington, DC: National Academy Press.
- Druckman, D., & Bjork, R. A. (Eds.). (1991). In the mind's eye: Enhancing human performance. Washington, DC: National Academy Press.
- Dutton, J. M., & Thomas, A. (1985). Relating technological change and learning by doing, research on technological innovation. *Management and Policy*, 2, 187–224.
- Ebbinghaus, H. (1885/1964). *Memory: A contribution to experimental psychology*, trans. H. A. Ruger & C. E. Busse-nius. New York: Dover.
- Elmaghraby, S. E. (1990). Economic manufacturing quantities under conditions of learning and forgetting (EMQ/LaF). *Production Planning and Control*, 1, 196–208.
- Epple, D., Argote, L., & Devadas, R. (1991). Organizational learning curves: A method for investigation of intra-plant transfer of knowledge acquired through learning by doing. *Organization Science*, 2, 58–70.
- Farr, M. J. (1987). *The long-term retention of knowledge and skills: A cognitive and instructional perspective*. New York: Springer-Verlag.
- Fisk, J. C., & Ballou, D. P. (1982). Production lot sizing under a learning effect. *IIE Transactions*, 14, 257.
- Globerson, S., & Levin, N. (1987). Incorporating forgetting into learning curves. *International Journal of Production Management*, 7, 80–94.
- Globerson, S., Levin, N., & Ellis, S. (1998). Rate of forgetting for motor and cognitive tasks. *International Journal of Cognitive Ergonomics*, 2, 181–191.
- Globerson, S., Levin, N., & Shtub, A. (1989). The impact of breaks on forgetting when performing a repetitive task. *IIE Transactions*, 21, 376–381.
- Glover, J. H. (1966). Manufacturing progress functions: An alternative model and its comparison with existing functions. *International Journal of Production Research*, 4, 279–300.
- Glymour, C., Madigan, D., Pregibon, D., & Smyth, P. (1996). Statistical themes and lessons for data mining. *Data Mining and Knowledge Discovery*, 1, 25–42.
- Gujarati, D. (1978). *Basic econometrics*. New York: McGraw-Hill.
- Hancock, W. M. (1967). The prediction of learning rates for manual operations. *Journal of Industrial Engineering*, 18, 42–47.
- Hasan, H., & Hyland, P. N. (2001). Using OLAP and multidimensional data for decision making. *IT Professional*, 3, 44–50.
- Jaber, M. Y., & Bonney, M. (1996). Production breaks and the learning curve: The forgetting phenomenon. *Applied Mathematical Modeling*, 20, 162–169.
- Kantor, P. B., & Zangwill, W. I. (1991). Theoretical foundation for a learning rate budget. *Management Science*, 37, 315–330.
- Karwan, K. R., Mazzola, J. B., & Morey, R. C. (1988). Production lot sizing under set-up and worker learning. *Naval Research Logistics*, 35, 159–179.

- Knecht, G. R. (1974). Costing, technological growth and generalized learning curves. *Operations Research Quarterly*, 25, 487–491.
- Knoshnevis, B., & Wolfe, P. M. (1983). An aggregate production planning model incorporating dynamic productivity. Part II: Solution methodology and analysis. *IIE Transactions*, 15, 283–291.
- Kowalewski, M. J., & Tye, J. B. (1990). *Statistical sampling: Past, present, and future theoretical and practical*. Philadelphia: American Society for Testing and Materials.
- Lance, C. E., Parisi, A. G., Bennett, W. R., Teachout, M. S., Harville, D. L., & Welles, M. L. (1998). Moderators of skill retention interval/performance decrement relationships in eight U.S. Air Force enlisted specialties. *Human Performance*, 11, 103–123.
- Levy, F. K. (1965). Adaptation in the production process. *Management Science*, 11, 136–154.
- Li, C., & Wang, X. S. (1996). A data model for supporting on-line analytical processing. *Proceedings of the Conference on Information and Knowledge Management* (pp. 81–88). New York: Association for Computing Machinery.
- Mazur, J. E., & Hastie, R. (1978). Learning and accumulation: A reexamination of the learning curve. *Psychological Bulletin*, 85, 1256–1274.
- Meredith, J., & Camm, J. (1989). Modeling synergy and learning under multiple advanced manufacturing strategies. *Decision Sciences*, 20, 258–271.
- Nembhard, D. A. (2000). The effects of task complexity and experience on learning and forgetting: A field study. *Human Factors*, 42, 272–286.
- Nembhard, D. A., & Uzumeri, M. V. (2000a). An individual-based description of learning within an organization. *IEEE Transactions on Engineering Management*, 47, 370–378.
- Nembhard, D. A., & Uzumeri, M. V. (2000b). Experiential learning and forgetting for manual and cognitive tasks. *International Journal of Industrial Ergonomics*, 25, 315–326.
- Nembhard, D. A., & Osothsilp, N. (2001). An empirical comparison of forgetting models. *IEEE Transactions on Engineering Management*, 48, 283–291.
- Neter, J., Kutner, M. H., Nachtsheim, C. J., & Wasserman, W. (1996). *Applied linear statistical models* (4th ed.). Chicago: Irwin.
- Pegels, C. C. (1969). On startup or learning curves: An expanded view. *AIEE Transactions*, 1, 216–222.
- Pratsini, E., Camm, J. D., & Raturi, A. S. (1993). Effects of process learning on manufacturing schedules. *Computers and Operations Research*, 20, 15–24.
- Pratsini, E., Camm, J. D., & Raturi, A. S. (1994). Capacitated lot sizing under set-up Learning. *European Journal of Operational Research*, 72, 545–557.
- Raftery, A. E. (1995). Bayesian model selection in social research (with discussion). In P. V. Marsden (Ed.), *Sociological methodology* (pp. 111–196). Oxford, UK: Blackwells.
- Ramos, M. A. G., & Chen, J. J. G. (1994). On the integration of learning and forgetting curves for the control of knowledge and skill acquisition for non-repetitive task training and retraining. *International Journal of Industrial Engineering*, 1, 233–242.
- Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, 13, 257–266.
- Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14, 465–471.
- Shafer, S. M., Nembhard, D. A., & Uzumeri, M. V. (2001). The effects of worker learning, forgetting, and heterogeneity on assembly line productivity. *Management Science*, 47, 1639–1653.
- Senge, P. M. (1992). Systems thinking and organizational learning: Acting locally and thinking globally in the organization of the future. *European Journal of Operational Research*, 59, 137–151.
- Smunt, T. L. (1987). The impact of worker forgetting on production scheduling. *International Journal of Production Research*, 25, 689–701.
- Sule, D. R. (1981). Effect of learning and forgetting on economic lot size scheduling. *International Journal of Production Research*, 21, 771–786.
- Sule, D. R. (1983). Effect of learning and forgetting on economic lot size scheduling problems. *International Journal of Production Research*, 21, 771–786.
- Uzumeri, M. V., & Nembhard, D. A. (1998). A population of learners: A new way to measure organizational learning. *Journal of Operations Management*, 16, 515–528.
- Yelle, L. E. (1979). The learning curve: Historical review and comprehensive survey. *Decision Sciences*, 10, 302–328.

21

Mining Text Data

Ronen Feldman

ClearForest Corp. & Bar-Ilan University, Israel

Introduction	482
Architecture of Text Mining Systems	483
Statistical Tagging	485
Text Categorization	485
Term Extraction	489
Semantic Tagging	489
DIAL	491
Development of IE Rules	493
Auditing Environment	499
Structural Tagging	500
Given	500
Find	500
Taxonomy Construction	501
Implementation Issues of Text Mining	505
Soft Matching	505
Temporal Resolution	506
Anaphora Resolution	506
To Parse or Not to Parse?	507
Database Connectivity	507
Visualizations and Analytics for Text Mining	508
Definitions and Notations	508
Category Connection Maps	509
Relationship Maps	510
Trend Graphs	516

Summary	516
References	517

INTRODUCTION

Data mining and knowledge discovery seek to turn the vast amounts of data available in digital format into useful knowledge. Classic data mining concentrates on structured data stored in relational databases or in flat files. However, it is now clear that only a small portion of the available information is in a structured format. It is estimated that up to 80% of the data available in digital format is nonstructured data. Most notably, much of information is available in textual form with little or no formatting. Hence, the growing interest in text mining, which is the area within data mining that focuses on data mining from textual sources.

The first issue to address when performing text mining is to determine the underlying information on which the data mining operations are applied. Our approach to text mining is based on extracting meaningful concepts that annotate the documents. A typical text mining system begins with collections of raw documents without any labels or tags. Initially, documents are automatically labeled by categories and terms or entities extracted directly from the documents. Next, the concepts and additional higher-level entities (that are organized in a hierarchical taxonomy) are used to support a range of Knowledge Discovery in Databases (KDD) operations on the documents. The frequency of co-occurrence of concepts can provide the foundation for a wide range of KDD operations on collections of textual documents, such as finding sets of documents with concept distributions that differ significantly from those of the full collection, other related collections, or collections from other points in time.

A straightforward approach is to use the entire set of words in the documents as inputs to the data mining algorithms. However, as was shown by Rajman and Besançon (1997), the results of the mining process in this approach are often rediscoveries of compound nouns (such as that “Wall” and “Street” or that “Ronald” and “Reagan” often co-occur), or of patterns that are at too low a level to be significant (such as that “shares” and “securities” co-occur).

A second approach (Feldman, Aumann, Amir, Klösgen, & Zilberstein, 1997; Feldman & Dagan, 1995; Feldman & Hirsh, 1997; Feldman, Rosenfeld, Stoppi, Liberzon, & Schler, 2000) is to use tags associated with the documents and to perform the data mining operations on the tags. However, to be effective this requires:

- Manual tagging, which is unfeasible for large collections; or
- Automated tagging using any one of the many automated categorization algorithms.

This approach suffers from two drawbacks. First, the number of distinct categories that such algorithms can effectively handle is relatively small, thus limiting the broadness of the mining process. More important, the process of automated categorization requires defining the categories *a priori*, thus defeating the purpose of discovery within the actual text.

A third approach is text mining via information extraction (Appelt et al., 1993a, 1993b; Cowie & Lehnert, 1996; Lin, 1995; Riloff & Lehnert, 1994; Tipster, 1993), whereby we first perform information extraction on each document to find events, facts, and entities that are likely to have meaning in the given domain, and then to perform the data mining operations on the extracted information. A possible “event” may be that a company has entered a joint venture or has executed a management change. The extracted information provides much more concise and precise data for the mining process than in a word-based approach and tends to represent

more meaningful concepts and relationships in the document's domain. On the other hand, in contrast to the tagging approach, the information-extraction method allows for mining of the actual information present within the text rather than the limited set of tags associated to the documents. Using the information extraction process, the number of different relevant entities, events, and facts on which the data mining is performed is unbounded—typically thousands or even millions, far beyond the number of tags that any automated tagging system could handle.

Although on a basic level one can rely on generic proper name recognition that is mostly domain-independent, the power of this text mining approach is most apparent when coupled with extraction specific to the domain of interest. Thus, for example, when mining a collection of financial news articles, we would want to extract pertinent information on companies, industries, and technologies—information such as mergers, acquisitions, management positions, and so forth.

We start by describing the typical architecture of a text mining system. We then describe the various techniques for document preprocessing. Next we discuss document categorization and term extraction and move to information extraction methodologies. We then describe how to generate and utilize a taxonomy based on the output of the preprocessing phase. The automatically created taxonomy is the basis for all the analytic text mining operations. We continue by describing the analytic level of a typical text mining system including various types of visualizations. We then outline the primary hurdles when implementing a text mining system. A summary concludes this chapter.

ARCHITECTURE OF TEXT MINING SYSTEMS

A text mining system is composed of three major components:

1. Information feeders: A component to enable the connection between various textual collections and the tagging modules. This component connects to any Web site, streamed source (such a news feed), internal document collections, and any other types of textual collections.
2. Intelligent tagging: A component responsible for reading the text and distilling (tagging) the relevant information. This component can perform any type of tagging on the documents such as statistical tagging (categorization and term extraction), semantic tagging (information extraction), and structural tagging (extraction from the visual layout of documents).
3. Business intelligence suite: A component responsible for consolidating the information from disparate sources, allowing for simultaneous analysis of the entire information landscape.

In addition, the output of the tagging component can be fed into external systems such as corporate databases, workflow systems, or file systems. The tagging system is connected to the business intelligence suite or to the external systems either by producing rich XML documents or by using an API (application programming interface) that can connect directly to the internal data structures of the tagging process.

An example of such an XML document is shown in Fig. 21.1. We can see several examples of annotation of entities (like Dynegy Inc as an instance of a company, or Roger Hamilton as an instance of a person) and relationships such as the CreditRating relationship, which correlates a rating, a trend, the rating company, and the rated company; and the PersonPositionCompany relationship, which correlates a person a position and a company.

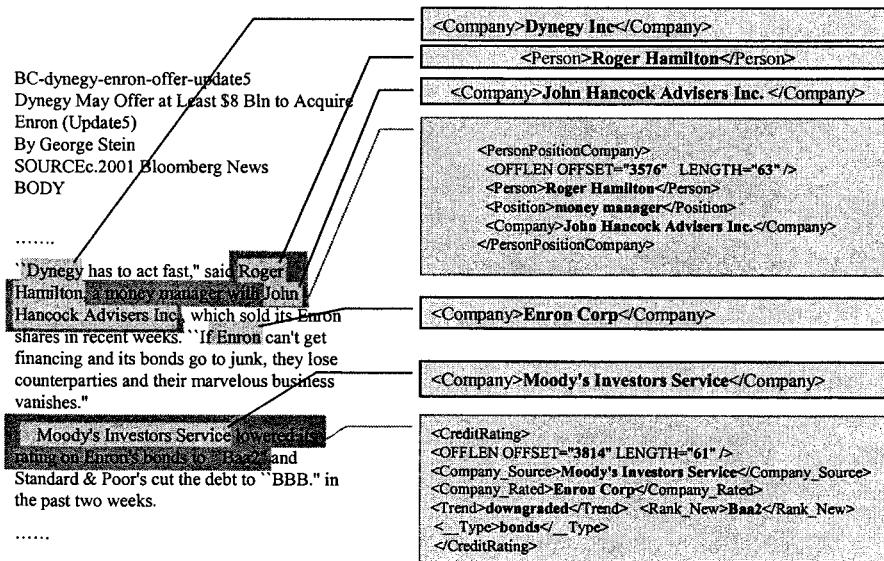


FIG. 21.1. XML annotation of an input document.

In Fig. 21.2 we can see the schematic diagram of a typical text mining system as was described. A more detailed description of the intelligent tagging component is shown in Fig. 21.3. Each of the taggers is using a separate training module that is based on annotated examples. A more detailed discussion of the training modules will be presented in the following sections. The training module for the structural tagging is producing document signatures that are then saved and mapped against new documents. The training module for the

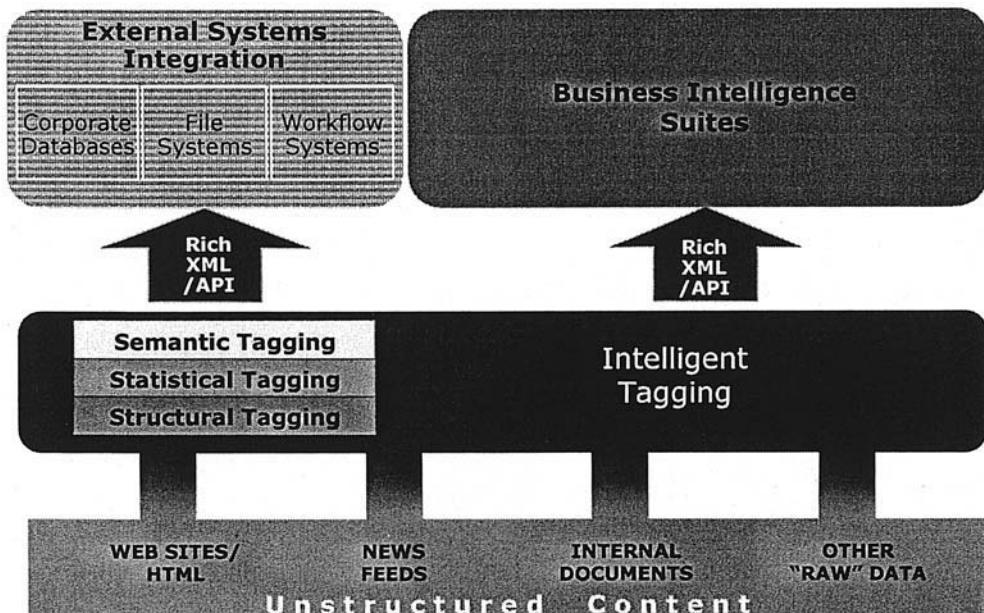


FIG. 21.2. Architecture of text mining systems.

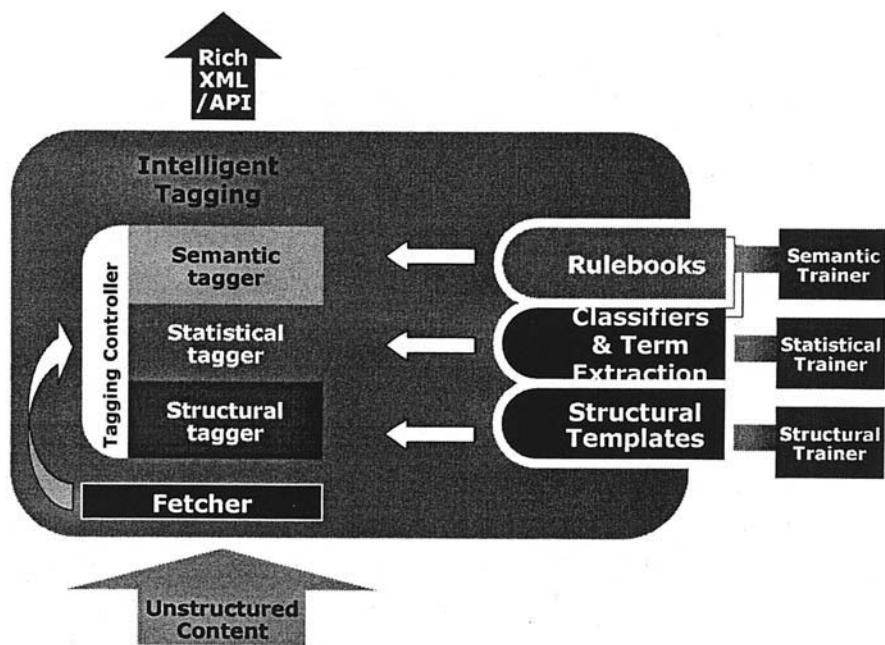


FIG. 21.3. Detailed description of the intelligent tagging component.

statistical tagging is producing classifiers for each of the categories, and the training module for the semantic training is producing information extraction rules based on annotated documents.

STATISTICAL TAGGING

Statistical tagging is based on the existence of a large collection of documents and usually relies on the presence of a training collection that is pretagged. The two main families of techniques within this approach are text categorization algorithms and term extraction algorithms. The next two subsections provide an overview of these families of algorithms.

Text Categorization

Text categorization (Cohen, 1992; Cohen & Singer, 1996; Dumais, Platt, Heckermann, & Sahami, 1998; Lewis, 1995; Lewis & Hayes, 1994; Lewis & Ringuette, 1994; Lewis, Schapire, Callan, and Papka, 1996; Sebastiani, 2002) is the activity of labeling natural language texts with thematic categories from a predefined set of categories. There are two main approaches to the categorization problem. The first is the knowledge engineering approach, in which the user is defining manually a set of rules encoding expert knowledge how to classify documents under given categories. The other approach is the machine learning approach (Sebastiani, 2002), in which a general inductive process automatically builds an automatic text classifier by learning from a set of preclassified documents.

An example of knowledge engineering approach is the CONSTRUE (Hayes, 1992; Hayes & Weinstein, 1990) system built by the Carnegie group for Reuters. A typical rule in the CONSTRUE system:

if DNF (disjunction of conjunctive clauses) formula then category
Example:

```
If ((wheat & farm)      or
     (wheat & commodity) or
     (bushels & export)   or
     (wheat & tonnes)     or
     (wheat & winter & ¬ soft))
then Wheat
else ¬ Wheat
```

The main drawback of this approach is the knowledge acquisition bottleneck. The rules must be manually defined by a knowledge engineer interviewing a domain expert. If the set of categories is modified, then these two professionals must intervene again. Hayes and Weinstein (1990) reported a 90% breakeven between precision and recall on a small subset of the Reuters test collection (723 documents). However, it took a tremendous effort to develop the system (several person-years), and the test set was not significant to validate the results. It is not clear that these superb results will scale up when a bigger system needs to be developed.

The machine-learning based approach is based on the existence of a training set of document that are pretagged using the predefined set of categories. A diagram of a typical ML based categorization system is shown in Fig. 21.4. There are two main methods for performing machine-learning based categorization. One method is to perform “hard” (fully automated) classification, in which for each pair of category and document we assign a truth value (either TRUE if the document belongs to the category or FALSE otherwise). The other approach is to perform a ranking (semiautomated) based classification. In this approach rather than returning a truth value the classifier return a *categorization status value (CSV)*, that is, a number between 0 and 1 that represents the evidence for the fact that the document belongs to the category. Documents are then ranked according to their CSV value. Specific text categorization algorithms are discussed later.

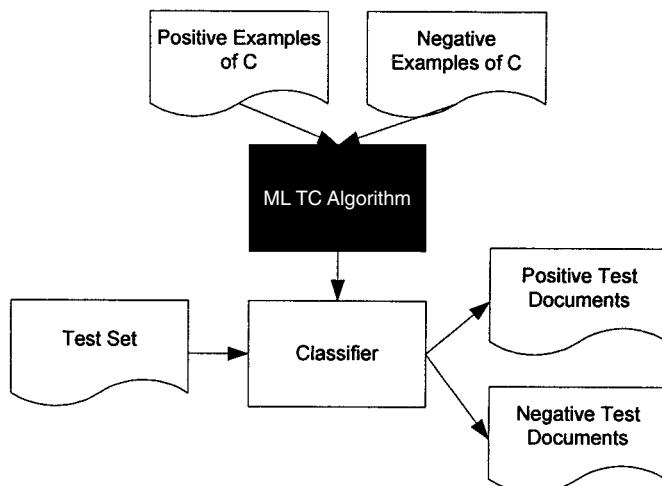


FIG. 21.4. Diagram of a typical machine-language based categorization system.

Definitions:

- $D = \{d_1, d_2, \dots, d_n\}$: the training document collection
- $C = \{c_1, c_2, \dots, c_k\}$: the set of possible categories to be assigned to the documents
- $T = \{t_1, t_2, \dots, t_m\}$: the set of terms appearing in the documents
- w_{ij} : the weight of the j th term of the i th document
- $\text{CSV}_i(d_j)$: a number between 0 and 1 that represents the certainty that a category c_i should be assigned to document d_j
- $\text{Dis}(D_i, D_j)$: the distance between document D_i and D_j ; this number represents the similarity between the documents

Probabilistic Classifiers. Probabilistic classifiers view $\text{CSV}_i(d_j)$ in terms of $P(c_i | d_j)$, that is, the probability that a document represented by a vector $\vec{d}_j = \langle w_{1j}, \dots, w_{mj} \rangle$ of (binary or weighted) terms belongs to c_i , and compute this probability by an application of Bayes' theorem

$$P(c_i | \vec{d}_j) = \frac{P(c_i)P(\vec{d}_j | c_i)}{P(\vec{d}_j)}.$$

To compute $P(d_j)$ and $P(d_j | C_i)$, we need to make the assumption that any two coordinates of the document vector, when viewed as random variables, are statistically independent of each other; this independence assumption is encoded by the equation

$$P(\vec{d}_j | c_i) = \prod_{k=1}^{|T|} p(w_{kj} | c_i).$$

Example-Based Classifiers. Example-based classifiers do not build an explicit, declarative representation of the category c_i , but rely on the category labels attached to the training documents similar to the test document. These methods have thus been called lazy learners, because they defer the decision on how to generalize beyond the training data until each new query instance is encountered. The most prominent example of example-based classifier is *KNN* (*K* nearest neighbor).

For deciding whether $d_j \in c_i$, *KNN* looks at whether the k training documents most similar to d_j also are in c_i ; if the answer is positive for a large enough proportion of them, a positive decision is made, and a negative decision is taken otherwise. A distance-weighted version of *KNN* is a variation of *KNN* such that we weight the contribution of each neighbor by its similarity with the test document. Classifying d_j by means of *KNN* thus comes down to computing

$$\text{CSV}_i(d_j) = \sum_{d_z \in Tr_k(d_j)} \text{Dis}(d_j, d_z) \cdot C_i(d_z).$$

One interesting problem is how to pick the best value for k . Larkey and Croft (1996) use $k = 20$, whereas Yang and Chute (1994) and Yang and Liu (1999) found $30 \leq k \leq 45$ to yield the best effectiveness. Various experiments have shown that increasing the value of k does not significantly degrade the performance.

Propositional Rules Learners. There is a family of algorithms that try to learn the propositional definition of the category. One of the prominent examples of this family

of algorithms is Ripper (Cohen, 1992; Cohen & Singer, 1996). Ripper learns rules that are disjunctions of conjunctions. Here is an example of two rules that define the category “Ireland”.

Ireland \leftarrow ira \in document, killed \in document.
 Ireland \leftarrow ira \in document, belfast \in document.

Ripper builds a rule set by adding new rules till all positive exemplars are covered. Conditions are added to the rule until no negative exemplars are covered. Initially, examples were represented as feature vectors. Because the matrix was so sparse, each document was represented as a set. Ripper can also use negative features (i.e., $w \notin S$).

One of the attractive features of Ripper is its ability to bias the performance toward higher precision or higher recall. This is done via the use of a special parameters call the Loss ratio. This is the ratio of the cost of a false negative to the cost of false positive. High loss ratio will increase recall and decrease precision.

Support Vector Machines. The support vector machine (SVM) algorithm was proven to be very fast and effective for text classification problems (Dumais et al., 1998; Joachims, 1998). SVMs were introduced by Vapnik in his work on structural risk minimization (Vapnik, 1979, 1995). A linear SVM is a hyperplane that separates with the maximum margin a set of positive examples from a set of negative examples. The margin is the distance from the hyperplane to the nearest example from the positive and negative sets. The diagram shown in Fig. 21.5 is an example of a two-dimensional problem that is linearly separable.

Clustering as a Preprocess Step for Categorization. Document clustering algorithms can be used in a preprocessing phase and enable finding the main themes in the documents without any need for corpus annotation. The fact that clustering is an instance of unsupervised learning makes it very suitable for the exploration phase of a text mining project. In addition, clustering algorithms were used in the link analysis phase to cluster together related entities. This approach proved to be very useful in identifying groups of related objects and in identifying their internal organization.

Comparison Between Text Categorization and Information Extraction. In contrast to the information extraction approach, in which the entities that tag the document are based on actual terms extracted from the document, text categorization tags

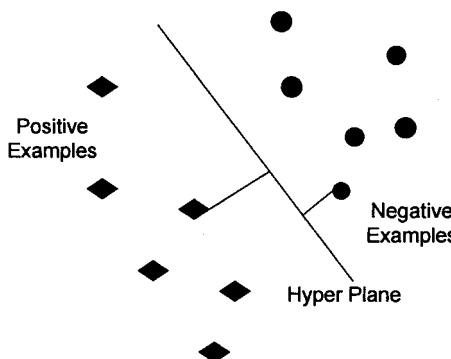


FIG. 21.5. Diagram of a two-dimensional linear SVM.

the document with concepts that do not need to be mentioned in the document itself. The main advantages of using a categorization approach are that it is less time consuming to prepare the training corpus, and there is no need to manually craft rules. On the other hand, one to five tags would be assigned to any given document. That would usually capture just some of the main topics of the document and certainly miss most of the important entities mentioned inside the document. In contrast, a document tagged by an information extraction system will have around 20 to 50 tags (for a two-to-three-page document). In a nutshell, information extraction was found to provide a much better infrastructure for text mining than was text categorization.

A Visual Interface to Document Classification. In Fig. 21.6 is a visual front end that enables the user to create classifiers for various categories and then test them on other document collections. In Fig. 21.6 we created 10 classifiers for categories such as “British banks,” “CRM software,” or “baseball.” We then tested all classifiers on a test collection of 476, and focused on the performance of the baseball classifier on the test set. The system ranks all documents in a decreasing order certainty that baseball should be assigned to the document. The user can then set the threshold for the category to any of the scores of the ranked documents. This threshold will be used in the operation mode of the system for attaching the baseball tag to newly arriving documents. In this particular example the best threshold was 0.829, which provides a precision of 97% and recall of 98%.

Term Extraction

The term extraction module is responsible for labeling each document with a set of terms extracted from the document. An example of the output of the term extraction module is given in Fig. 21.7. The excerpt is taken from an article published by Reuters Financial on May 12, 1996. Terms in this excerpt that were identified and designated as interesting by the term extraction module are underlined.

The overall architecture of the term extraction module is illustrated in Fig. 21.8. There are three main stages in this module: linguistic preprocessing, term generation, and term filtering.

The documents are loaded into the system through a special reader. The reader uses a configuration file that informs it of the meaning of the different tags annotating the documents. In such a way, we are able to handle a large variety of formats. The Text Processing Language (TPL) reader packages the information into a Standardized General Markup Language file.

The next step is the linguistic preprocessing that includes tokenization, part-of-speech tagging and lemmatizations (i.e., a linguistically more founded version of stemming; see Hull, 1996). The objective of the part-of-speech tagging is to automatically associate morpho-syntactic categories such as *noun*, *verb*, *adjective*, and so forth, to the words in the document. Some systems use a rule based approach similar to the one presented in Brill (1995), which is known to yield satisfying results (96% accuracy) provided that a large lexicon (containing tags and lemmas) and some manually hand-tagged data is available for training.

SEMANTIC TAGGING

Rule based information extraction techniques rely on the fact that the information to be extracted from documents can be specified in such a way that a relatively small number of extraction rules are needed. The knowledge-intensive information extraction approach requires trained developers and is very laborious; however, some attractive features of this approach are the

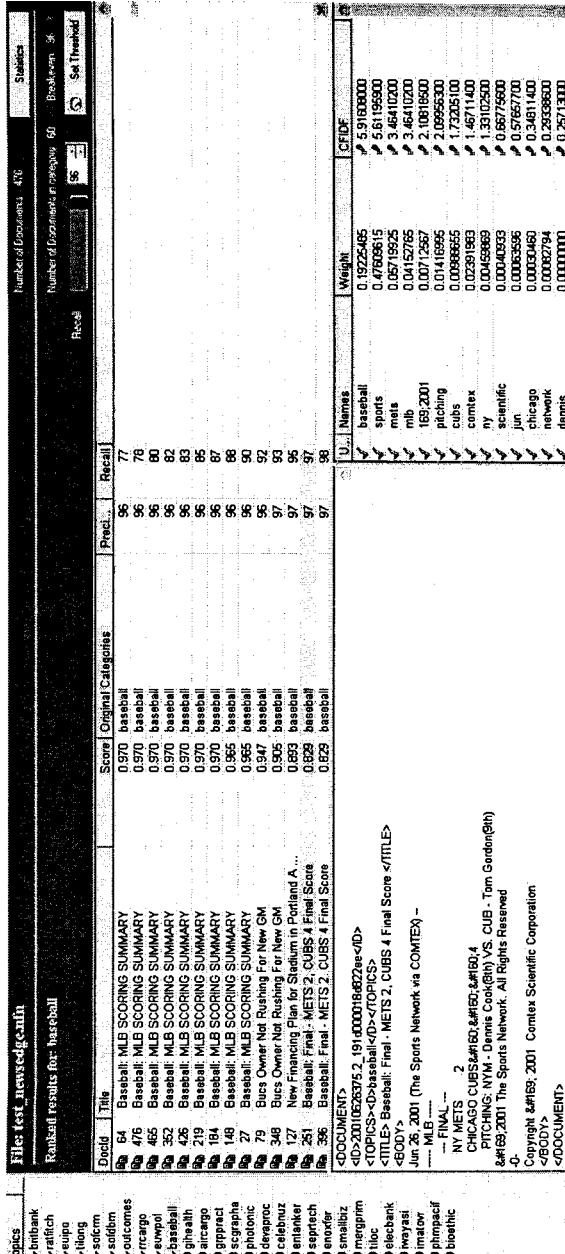


FIG. 21.6. A visual interface to classifying documents.

Profits at Canada's six big banks topped C\$6 billion (\$4.4 billion) in 1996, smashing last year's C\$5.2 billion (\$3.8 billion) record as Canadian Imperial Bank of Commerce and National Bank of Canada wrapped up the earnings season Thursday. The six banks each reported a double-digit jump in net income for a combined profit of C\$6.26 billion (\$4.6 billion) in fiscal 1996 ended Oct. 31.

But a third straight year of record profits came amid growing public anger over perceived high service charges and credit card rates, and tight lending policies.

Bank officials defended the group's performance, saying that millions of Canadians owned bank shares through mutual funds and pension plans.

FIG. 21.7. An example of the output of the term extraction process.

specificity of the extracted information and the precision and recall with which information can be extracted. The architecture of a typical information extraction system is shown in Fig. 21.9. We start our discussion of information extraction with a description of the Declarative Information Analysis Language (DIAL) information extraction language (Feldman et al., 2001, 2002).

DIAL

In this subsection we describe DIAL. DIAL is designed specifically for writing IE (information extraction) rules. The complete syntax of DIAL is beyond the scope of this chapter. Here we describe the basic elements of the language.

Basic Elements. The basic elements of the language are syntactic and semantic elements of the text and sequences and patterns thereof. The language can identify the following elements:

- Predefined strings such as “merger”
- Word class element: a phrase from a predefined set of phrases that share a common semantic meaning—for example, WC-Countries, a list of countries.

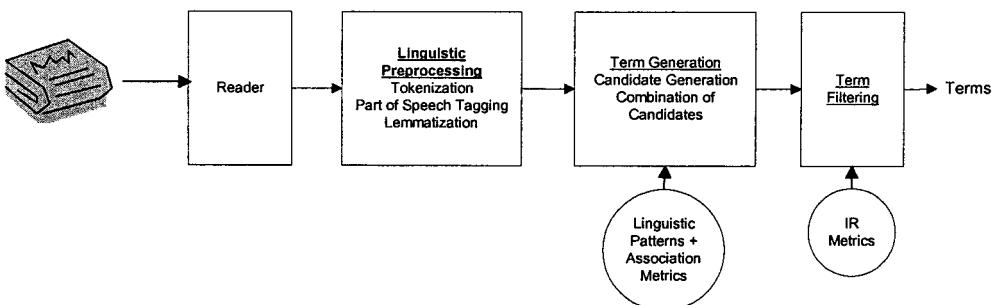


FIG. 21.8. Architecture of the term extraction module.

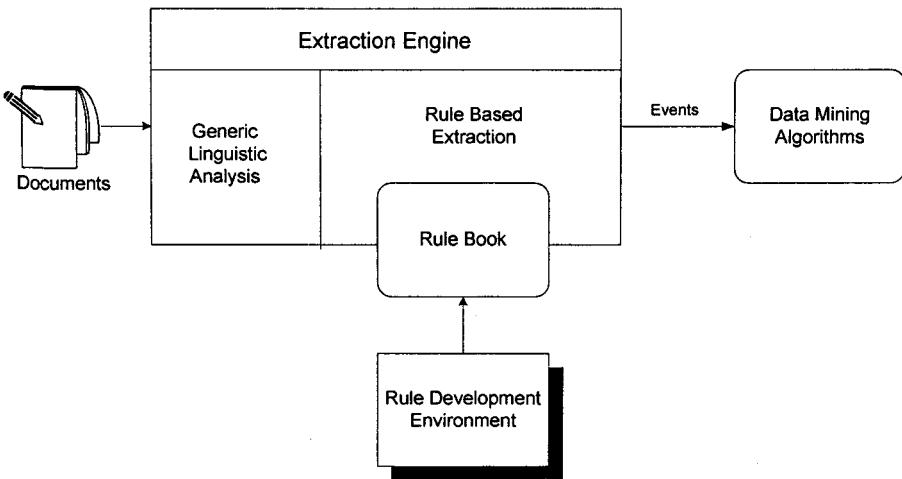


FIG. 21.9. Architecture of an information extraction system.

- Scanner feature (basic characteristic of a token), for example, @Capital or @HtmlTag
- Compound feature: a phrase comprising several basic features. Thus, Match (@Capital & WCCountries), for example, will match a phrase that both belongs to the word class WCCountries and starts with a capital letter.
- Part-of-speech tag—for example, noun or adjective
- Recursive predicate call—for example, Company (C)

Constraints. Constraints carry out on-the-fly Boolean checks for specific attributes. These can be applied to fragments of the original text or to results obtained during processing extraction process.

The marker for a constraint is the word *verify*, followed by parentheses containing a specific function, which governs what it is checking for. For example:

```
verify ( StartNotInPredicate ( c , @PersonName ) )
```

ensures that no prefix of the string assigned to variable *c* is a match for the predicate PersonName.

IE Rule Bases. The rule base can be viewed as a logic program. Thus, a *rule base*, Γ , is a conjunction of definite clauses $C_i : H_i \leftarrow B_i$ where C_i is a clause tag, H_i (called the *head*) is a literal, and $B_i = \{B_{i1} B_{i2} \dots\} = P_i \cup N_i$ (called the *body*) is a set of literals, where $P_i = \{p_{ij}\}$ is a set of pattern matching elements and $N_i = \{n_{ij}\}$ is a set of constraints operating on P_i . The clause $C_i : H_i \leftarrow B_i$ represents the assertion that H_i is implied by the conjunction of the literals in P_i while satisfying all the constraints in N_i .

An example of a DIAL rule is the following, which is 1 of 10 rules to identify a merger between two companies:

```
FMergerCCM(C1, C2) :-
    Company(Comp1) OptCompanyDetails "and" skip(Company(x)),
    SkipFail, 10) Company(Comp2) OptCompanyDetails
```

```

skip(WCMergerVerbs, SkipFailComp, 20) WCMergerVerbs
skip(WCMerger, SkipFail, 20) WCMerger
verify(WholeNotInPredicate(Comp1, @PersonName))
verify(WholeNotInPredicate(Comp2, @PersonName))
@% @!
{ C1 = Comp1; C2 = Comp2 };

```

The rule looks for a company name (carried out by the predicate `Company`, which returns the parameter `Comp1`) followed by an optional phrase describing the company, and then the word *and*. The system then skips up to 10 tokens (within the same sentence, and while not encountering any phrase prescribed by the predicate `SkipFail`) until it finds another company, followed by an optional company description clause. The system then skips up to 20 tokens until it finds a phrase of the word class `WCMergerVerbs`. (This may be something like “approved,” “announced,” etc.). Finally, the system skips up to 10 tokens scanning for a phrase of the word class `WCMerger`. In addition, the rule contains two constraints ensuring that the company names are not names of people.

Each rulebook can contain any number of rules that are used to extract knowledge from documents in a certain domain. Here are a few examples of rulebooks that were developed in DIAL:

- Financial rulebook: containing 11,500 rules, can identify more than 50 different entity types including company names; people names; organizations; universities; products; positions; locations (cities, countries, states, and addresses); dates, and amounts. In addition, it can identify more than 120 different event types such as: mergers (including a fine-grained distinction between known merger, new merger, rumored merger, planned merger, and cancelled merger); acquisitions (with a similar distinction between acquisition types); joint ventures; takeovers; business relationships; investment relationships; customer–supplier relationships; new product introductions; analyst recommendations for stocks and bonds, associations between companies and people; associations between companies and technologies; associations between companies and products; and many others.
- Business intelligence rulebook: contains 7,000 rules, can identify thirty different entity types, including company names, people, positions, prices, and products. In addition, it can identify more than eighty events, including joint ventures; business relationships; mergers and acquisitions; customer–supplier relationships; rival relationship; layoffs; strategic reorganizations; investments; new management; IPO plans; senior appointments; product-related events (product features); awards; and so forth.
- Intellectual property rulebook: contains 100 rules and can identify 30 different types of entities in patent files, including inventor, assignee, examiner, and a set of noun-phrase classes based on the context in which they appear.
- A protein relationship rulebook: enabling the extraction of relationships between protein pathways from articles featured in Medline. This rulebook contains 500 rules and can identify 30 different types of entities, including proteins and 10 different relationships including phosphorylation.

Development of IE Rules

Developing an IE module for a new domain can be very tedious and time-consuming. To speed up the process, the developer needs an environment with a range of productivity tools. The focus of this section is on these editing and debugging tools. In addition, the environment should

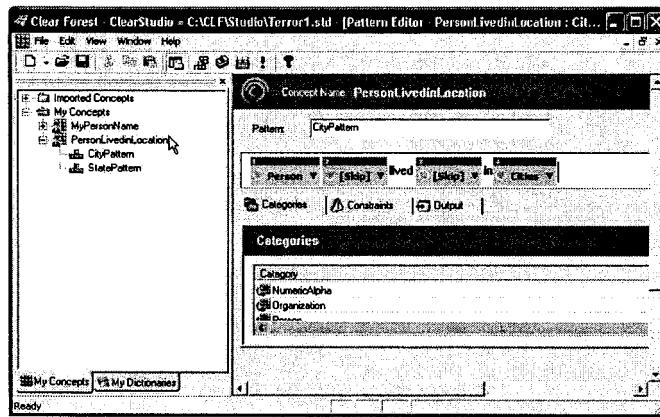


FIG. 21.10. Defining patterns by using visual pattern editors.

provide tools for checking the quality of the rulebook by examining its output on documents streams.

Visual Rule Authoring. To increase the productivity of the application developers, it is prudent to consider using rule editors and visual tools. The template editor enables the user to select a text fragment from any document and build a new pattern based on that. The user can generalize any part of the text fragments by selecting the part and replacing it with one of the predefined building blocks. The building blocks can be either primitives such as company name, person name, product, location and so forth, or a reference to predefined lexicons. Most of the pattern matching elements discussed previously can be defined using the visual pattern editor.

As an example, in Fig. 21.10, the pattern that was extracted from the sentence “Kamfar, also known as Amer Taiybkamfar, was reported to have lived in Florida for the last 18 months” is shown. The pattern generalizes the sentence to extract a relationship that correlates a person with a city or state. It was found that the users are much more comfortable working at this abstraction level than using a formal language to define the grammar of the patterns.

Debugging Tools. The DIAL environment includes a variety of tools for monitoring the integrity and performance of the rule base during its development. The tools available reflect the many types of problems that may arise, ranging from simple syntax errors that prevent the code’s compilation (e.g., omitting a vital punctuation mark or misspelling the name of a predicate or word class) to inefficiencies in the rules themselves that lead to inaccuracies in the results (e.g., *Bank of England* as a company), or events that are missed altogether. The user can then make modifications to the rule and rerun the code to ensure that the problem has been fixed or the accuracy improved.

Central to all these operations is the interpreter, which can act on the code line-by-line without precompilation. This is used to check the code for syntactical integrity before it is used for any information extraction. The offending line is highlighted, usually with an accompanying comment in the output pane, allowing the user to zoom in on the problem (see Fig. 21.11).

Once the code has passed the compilation test, it is tried out on a number of sample texts. The following debugging tools are available:

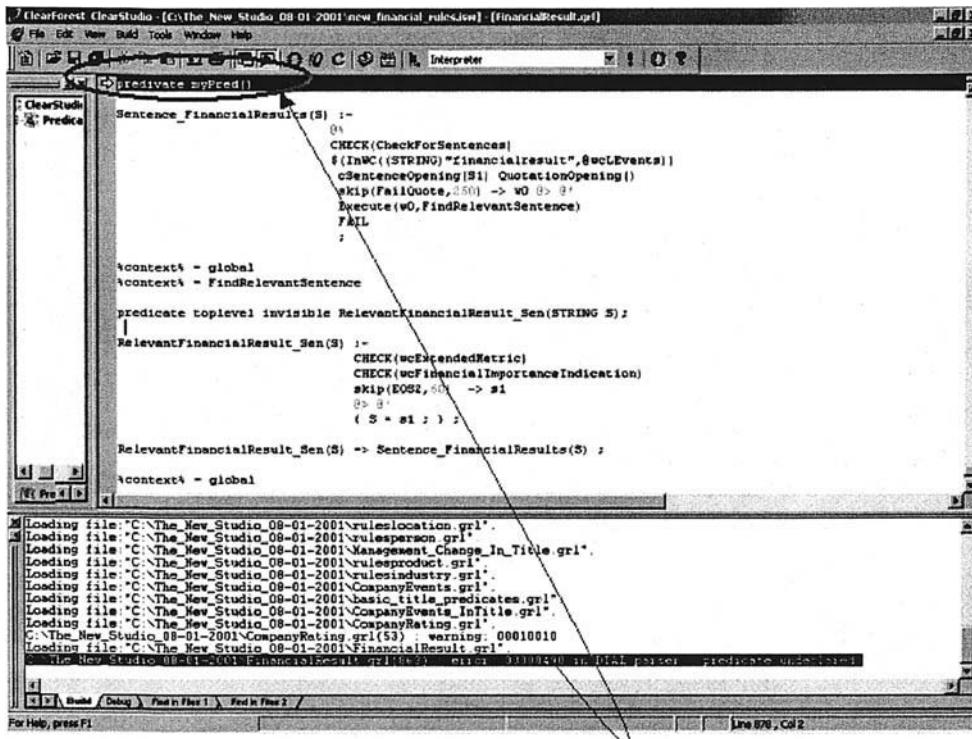


FIG. 21.11. ClearLab's built-in interpreter pinpoints syntax errors that prevent compilation.

- Reviewing event tables for rapid spotting of erroneous events. These can then be double-clicked to highlight the text fragment in the source text that caused the problem. This is usually enough to alert the user to the nature of the special case that caused the erroneous output and to make adjustments to the rule to prevent such occurrences in future.
- Right-clicking events. This allows the user to go directly to the relevant predicate behind the event—and specifically to the culprit definition in the code—and amend it as necessary.
- Tracer. This is used to monitor the incidence of *recall errors*—events that should have been caught but were not. A relevant rule is applied to the specific text in question. The success or failure of each component of the rule is then clearly shown in a report, featuring green check marks (success), red crosses (failure), and blue question marks (unchecked section), as in Fig. 21.12. Appropriate action can then be taken as necessary to improve the relevant rule(s).
- Event diff. This utility that allows you to assess the comparative effectiveness of incremental changes to the rules by comparing the list of events extracted using the new and the old versions. Typically, this is done soon or immediately after changing or adding any number of predicates.
- Profiler. This tool analyzes the rule file's performance, how long each predicate took to process a given document collection, and which in particular need tweaking, revising, or even complete removal to streamline the IE process. Its report is created as part of the compilation process, and thus the tool typically is applied at the end of the development process.
- Low-level debugging tool. It is similar to the tracer in that it tests the code against a sample text of the user's choosing (Fig. 21.13B). But it is more comprehensive because it examines the processing by the entire rule file up to a breakpoint of their choosing (A). All aspects of

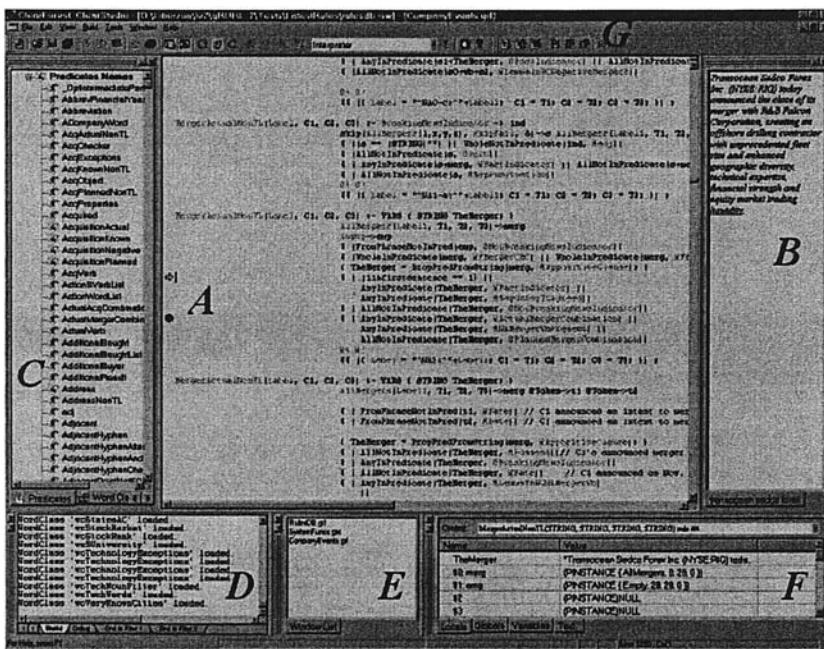


FIG. 21.12. The low-level debugger tool built into the environment allows the whole or part of the rulebook (up to a user-defined breakpoint, A) to run on any sample text (B), and all aspects of the results to be examined (D, E, F).

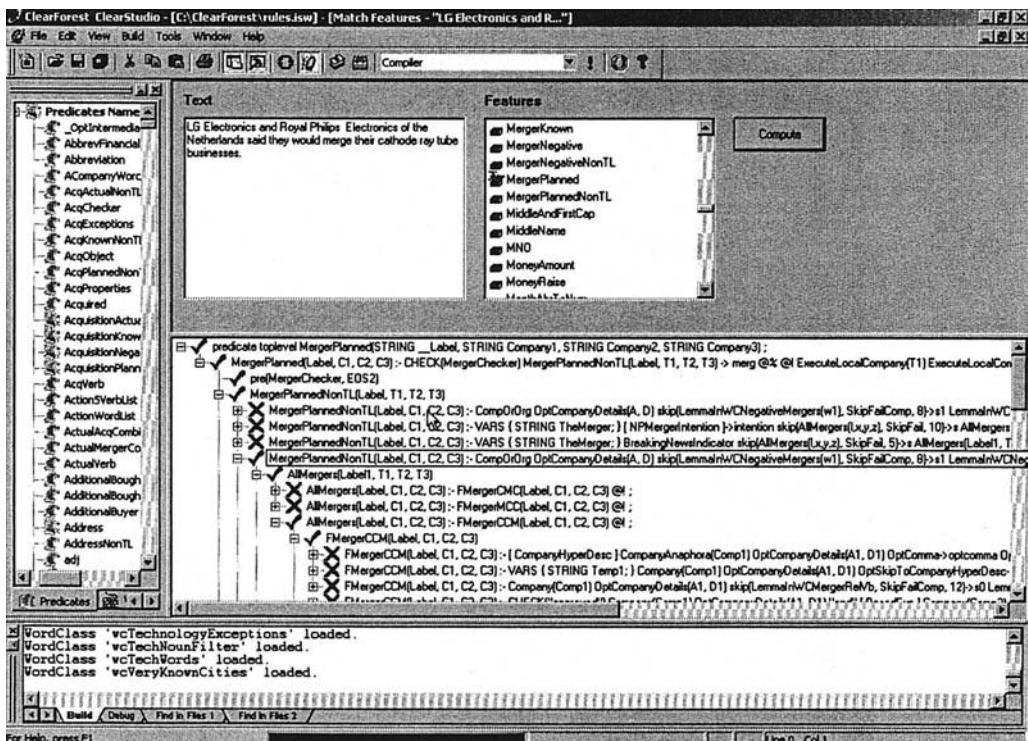


FIG. 21.13. The tracer tool allows specific predicates to be tested for effectiveness on sample texts that are pasted or typed into the relevant section.

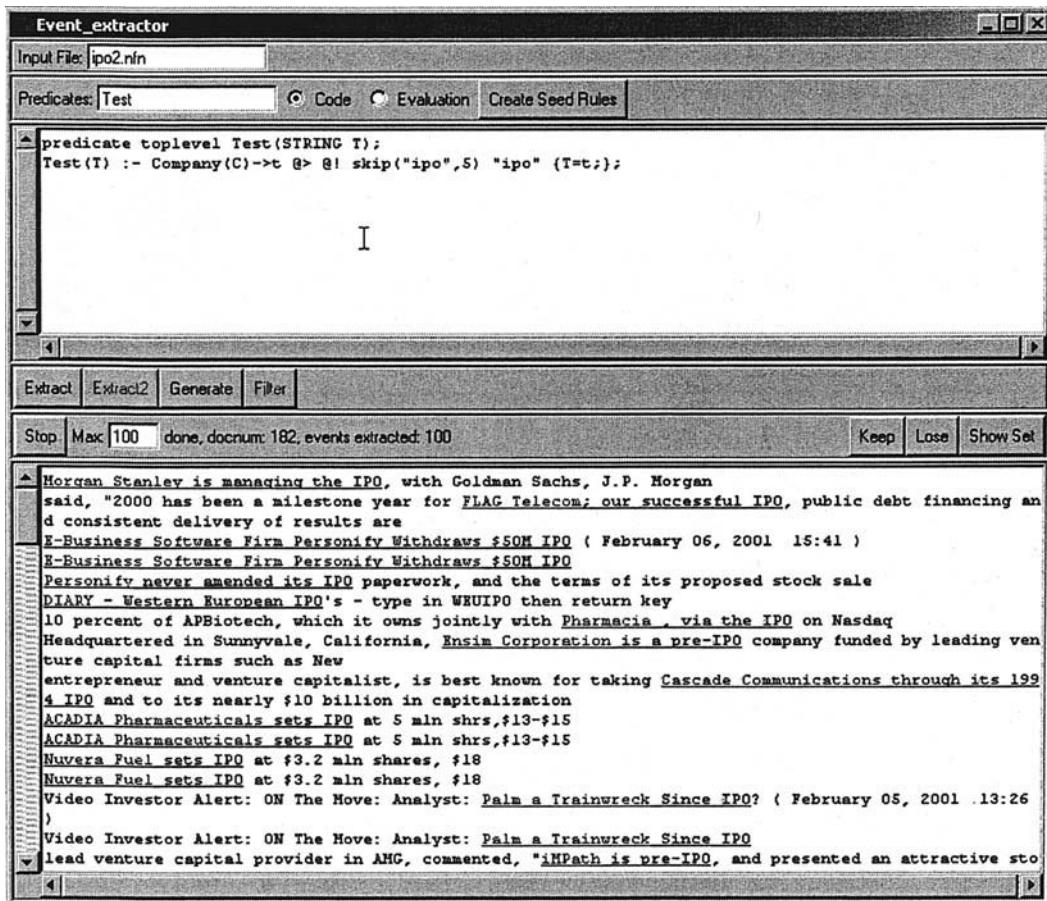


FIG. 21.14. The pattern locator tool.

the process may be examined subsequently, from the full list of predicates and functions in the rule file (C), to the word classes and other resources actually loaded, the active windows, and variables (E, F). *Stepinto*, *Stepover*, and *Stepout* tools (G) allow one to examine each rule call-by-call individually within the same stack frame or outside it.

- Pattern locator tool. This enables the user to enter a DIAL pattern and locates all instances of the pattern in a document collection. It then enables generalizing a selected subset of those instances. Portions assigned to variables in the pattern are shown in red and blue. The full instance of the patterns is underlined (see Fig. 21.14).

Contrasting Rule Engineering with Automatic Rule Learning. In contrast to the rule based approach, in which most of the effort is focused on writing rules, other systems use a machine learning approach in which IE rules are automatically learned based on an annotated corpus (Lehnert et al., 1991; Riloff & Lehnert, 1994; Soderland, Fisher, Aseltine, & Lehnert, 1995). Most of the effort in this case would then be devoted to annotating the corpus with entities and relationships. Although experience has shown that the accuracy of entity extraction rules was quite on par with the handcrafted rules, this was not the case for automatically learned relations extraction rules. From our experience, we managed to get by

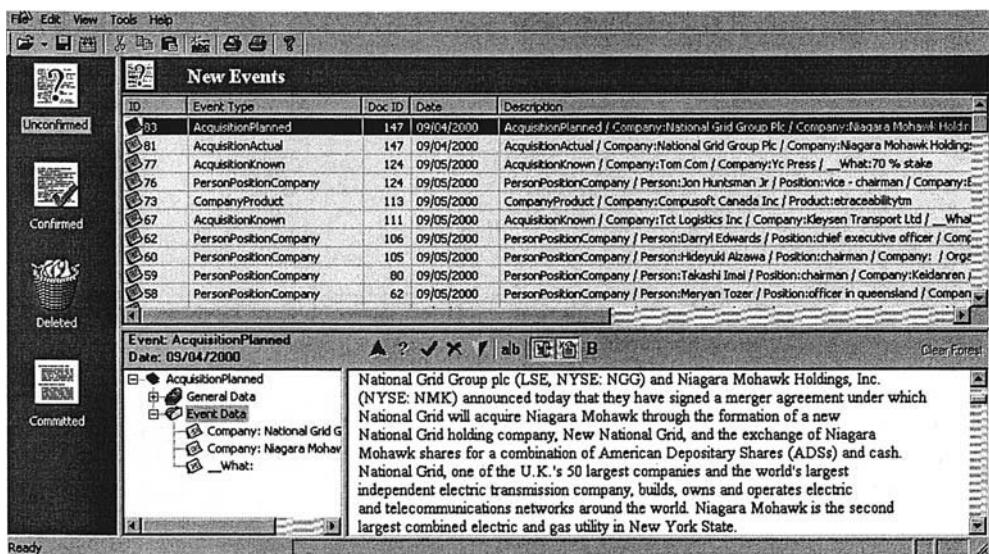


FIG. 21.15. The main auditing display.

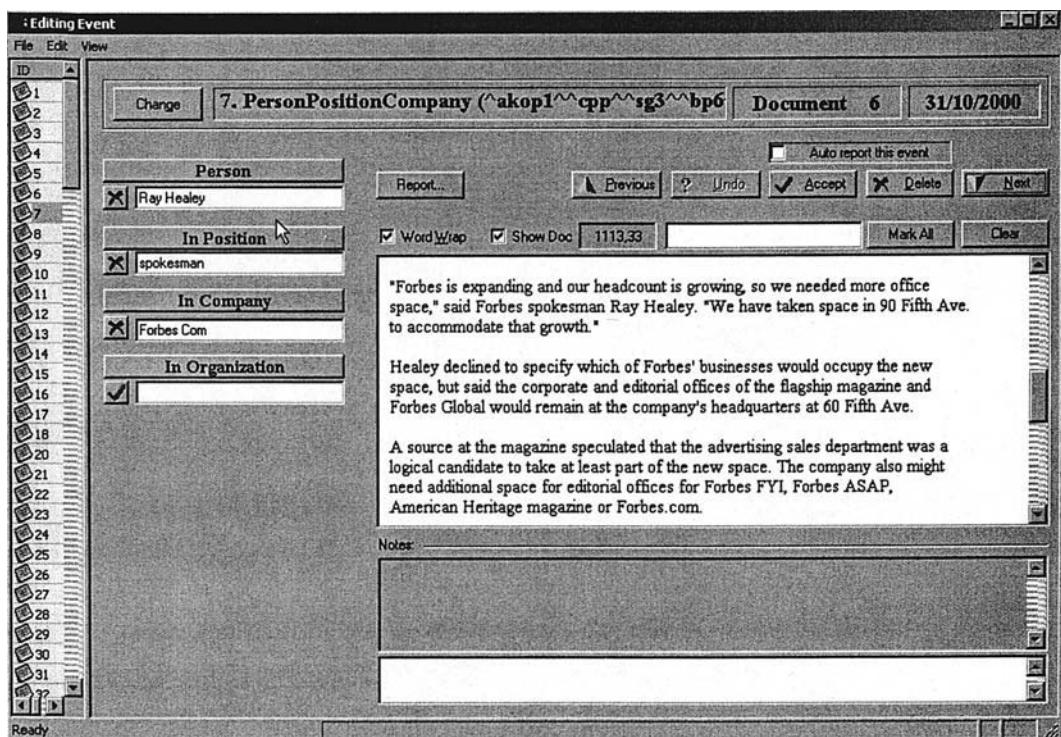


FIG. 21.16. Auditing a Person Position Company event.

using handcrafted rules producing a breakeven score of around 92% (for event and fact extraction). Machine learning based approaches were able to achieve only around 70% breakeven.

Auditing Environment

The auditing environment enables the user to view all events extracted from the document collection and easily fix erroneous events. The errors may be of several types: The event is completely wrong, and in that case we will delete it altogether; the event is correct, but some of the fields are incorrect, in which case we fix the erroneous fields; or the event is accurate (and all fields are accurate as well). In this case we may update the taxonomy or the thesaurus with new entities that may have been discovered in this event.

In Fig. 21.15 we can see the main screen of the auditing environment. The top pane shows all the events that were found in the collection. The lower pane shows the text fragment from which the event was extracted along with the fields that comprise the event.

In Fig. 21.16 we can see how to audit a specific PersonPositionCompany event. The tool allows the user to see the text fragment from which the event was extracted in context. In addition, it enables the user to change any of the individual fields that comprise the event. If the field value is already in the taxonomy (under the right category) then a check mark appears next to the field, otherwise, an X will appear. When the user clicks on any of the fields, he or she gets a screen that makes it possible to add the entry to the taxonomy and thesaurus or to change the value for the given field. Such a screen is shown in Fig. 21.17. The user gets the value of “president and CEO” for the position field for the event PersonPositionCompany. The system shows similar values from the same category (president, in this case), and the user has the option to add this as a new value of position in the taxonomy or to select another value.

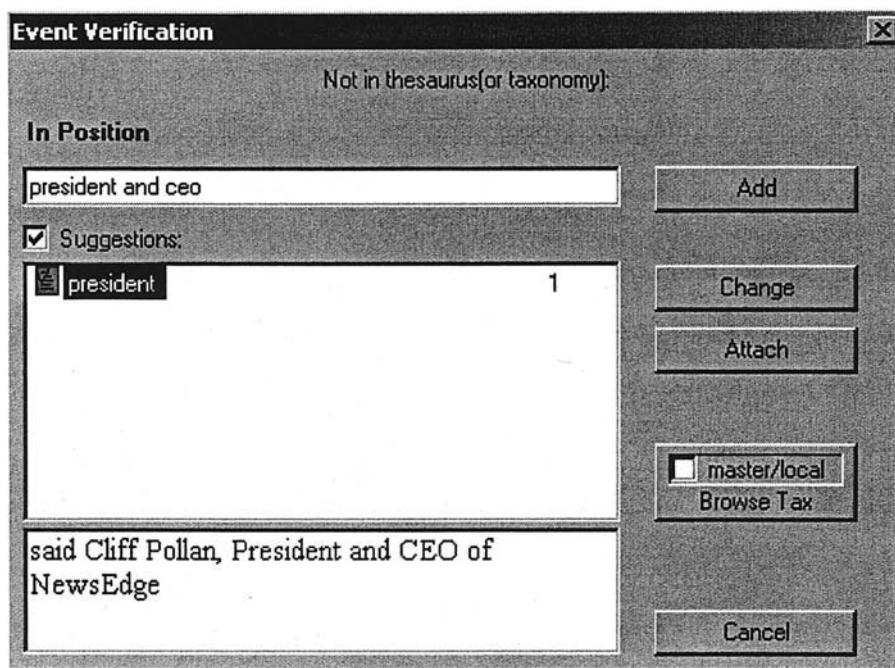


FIG. 21.17. Updating the taxonomy and the thesaurus with new entities.

STRUCTURAL TAGGING

Most text-processing systems simplify the structure of the documents they process. The visual form and layout of the documents is ignored and only the text itself is processed, usually as linear sequences or even bags of words. This allows the algorithms to be simpler and cleaner at the cost of possible loss of valuable information. In this section we show an approach that ignores the content of words while focusing on their superficial features, such as size and position on the page. Such an approach is not a rival but a complement to the conventional text extraction systems and can also function as a preprocessor or a converter.

We implemented this approach in a system called PES (PDF extraction system). PES accepts its input in the form of Acrobat PDF documents. A document page in PDF format is represented by a collection of primitive objects, which can be characters, simple graphic shapes, or embedded objects. Each primitive has properties, such as font size and type for characters, and position on the page, given as coordinates of the object's bounding rectangle. We are interested in an automatic process that accepts a formatted document as input and returns a predefined set of elements of the document, each assigned to a corresponding field, for example, "AUTHOR = ..., TITLE = ..." The set of field names and which parts get assigned to them is problem-dependent and may be different for different types of documents. Thus, we seek a system that learns how to extract the proper document elements based on examples provided by a domain expert. In PES, described in this chapter, a domain expert annotates a set of documents, marking the fields to be extracted. Each annotated document functions as a template, against which new documents can be matched.

At the heart of the extraction system we have the following problem:

Given:

1. Document A (a template)
2. Set of primitives in A (annotated fields), denoted P_A
3. Document B (a query document)

Find:

1. The degree of similarity between documents A and B
2. The set of primitives in B that corresponds to P_A

The first step in the process of finding the primitives of B that correspond to P_A , is to find similarities between the original document A and the new document B . The simplest way to match two documents is coordinate-wise: Return as answer all objects of the query document that fit into the bounding rectangle of the marked subset in the template. The disadvantages of such an approach are obvious. If the same field has different visual sizes, for example, a document title containing different number of words and text lines, or if the field is shifted a bit, the system will not identify the correct match. Nevertheless, the coordinates form a good basis for more refined heuristics, as the same fields tend to reside in more or less the same place across documents. However, the correspondence must be established between objects, not coordinates. In addition, the correspondence must be between higher-level groups and not only between primitive objects.

The PDF document representation does not contain any information about text lines, paragraphs, columns, tables, and other meaningful groups of primitives. The format is designed

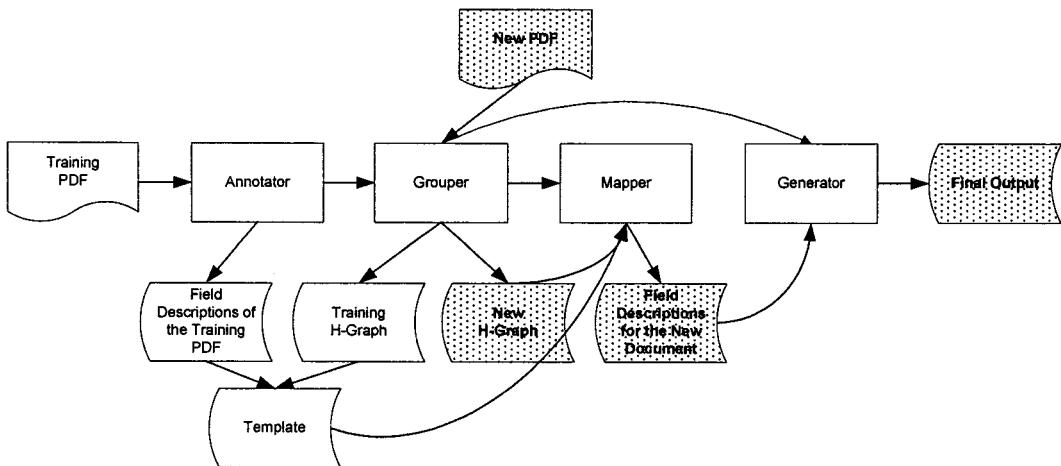


FIG. 21.18. Architecture of the PES.

for human reading, where the human mind does the necessary grouping unconsciously. For an information extraction system to take advantage of the visual clues available in the PDF format, the system must perform perceptual grouping as the first stage of processing a document. The approach we take is to take the physical/visual representation of the document and transform it into a complex abstract representation consisting of nested objects and relationships between them. We call this step *perceptual grouping*. Once perceptual grouping has been performed, the resulting structure is independent of the specific document type. This approach allows us to provide a general procedure applicable to diverse formats and rapidly adaptable to new formats.

Once the document structures are generated, these structures can be used to extract information. A representative set of documents is annotated by a domain expert, with parts of the documents' structures being assigned to certain fields. These documents serve as templates to be matched against the new documents. In the process of structural mapping, a correspondence is created between two document structures, mapping the objects in a template document to the objects in the nonannotated query document.

PES contains several components: annotator, grouper, mapper, and extractor. Annotator is a GUI tool that allows the user to mark fields in a PDF document and store their names and positions in a separate file. Grouper takes a PDF document as input, does the grouping, and saves the document structure. Mapper's input is a template (document structure + fields data) and a document structure for a query document. The template is mapped onto the query document, and the elements assigned to the various fields are produced as output, together with the overall quality of the mapping. Extractor takes a document structure and the selected elements and outputs the elements' text. The architecture of PES is shown in Fig. 21.18.

TAXONOMY CONSTRUCTION

One of the crucial issues in performing text mining is the need for term taxonomy. A term taxonomy also enables the production of high-level association rules, which are similar to general association rules (Srikant & Agrawal, 1995). These rules capture relationships between

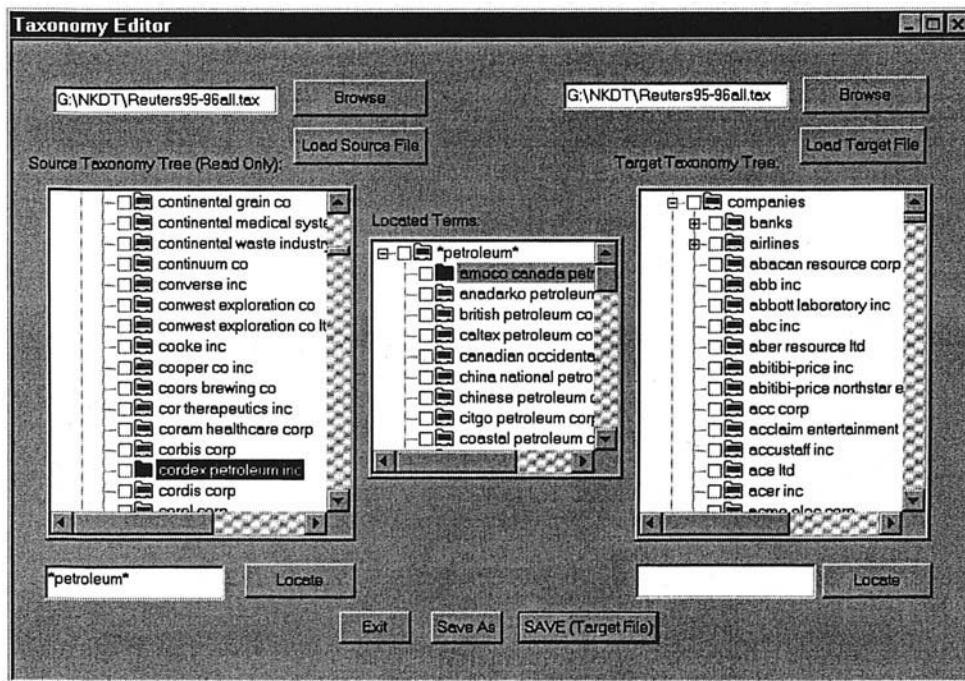


FIG. 21.19. Taxonomy editor.

groups of terms rather than between individual terms. A taxonomy is also important in other text mining algorithms such as maximal association rules and frequent maximal sets (Feldman et al., 1997).

A taxonomy also enables the user to specify mining tasks in a concise way. For instance, when trying to generate association rules, rather than looking for all possible rules the user can specify interest only in the relationships of companies in the context of business alliances. To do so, we need two nodes in the term taxonomy marked “business alliances” and “companies.” The first node contains all terms related to alliance such as “joint venture,” “strategic alliance,” “combined initiative” and so forth, whereas the second node is the parent of all company names in our system.

Building term taxonomy is a time-consuming task. Hence, there is a need to provide a set of tools for semiautomatic construction of such taxonomy. One such tool is the taxonomy editor shown in Fig. 21.19. This tool enables the user to read a set of terms or an external taxonomy and use them to update the system’s term taxonomy. The user can drag entire subtrees in the taxonomies or specify a set of terms via regular expressions. In Fig. 21.19 we can see the terms found when specifying the pattern **petroleum**. The initial set of terms is the set of all terms extracted from a collection of 64,000 Reuters documents from 1995–1996 (shown in the left tree), the terms matching the query are shown in the middle tree, and the right tree is the target taxonomy. The user can utilize the entries in the middle tree to create a new node in the target taxonomy (such as “petroleum companies”).

The taxonomy editor also includes a semiautomatic tool for taxonomy editing called the taxonomy editor refiner (TER). TER compares generated frequent sets against the term taxonomy. When most of the terms of a frequent set are determined to be siblings in the taxonomy

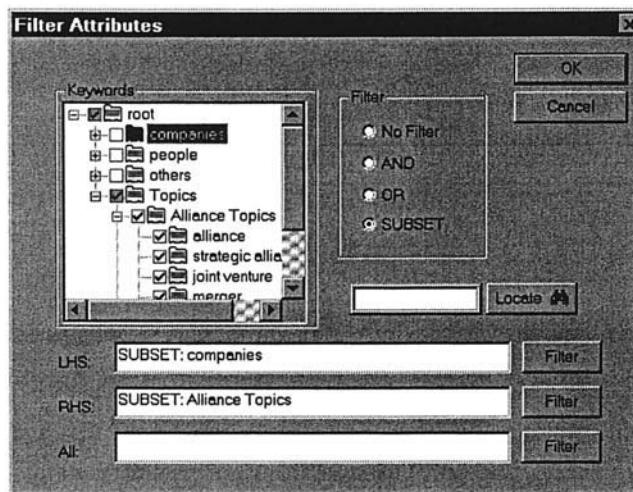


FIG. 21.20. Specifying a filter to display only association rules with “Companies” to the left of the rule and “Alliance Topics” to the right.

hierarchy, the tool suggests adding the remaining terms as siblings as well. For example, if our taxonomy currently contains 15 companies under “tobacco companies” and the system generates a frequent set containing many tobacco companies, one of which does not appear in the taxonomy, the TER will suggest adding this additional company to the taxonomy as a tobacco company. The TER also has a term clustering module, again suggesting that terms clustered together be placed as siblings in the taxonomy.

In the example presented in Fig. 21.20 the user is interested in business alliances between companies. The user therefore specifies a filter for the association rules generation algorithm, requesting only association rules with companies on the left hand side of the rule and business alliance topics on the right hand side. Fig. 21.20 shows the filter definition window.

Using the Reuters document corpus described previously, the system generated 12,000 frequent sets that comply with the restriction specified by the filter (with a support threshold of five documents and confidence threshold of 0.1). These frequent sets generated 575 associations. A further analysis removed rules that were subsumed by other rules, resulting in a total of 569 rules. A sample of these rules is presented in Fig. 21.21. The numbers presented at the end of each rule are the rule’s support and confidence.

The example in Fig. 21.21 illustrates the advantages of performing text mining at the term level. Terms such as “joint venture” would be totally lost at the word level. Company names, such as “santa fe pacific corp” and “bank of boston corp,” would not have been identified either. Another important issue is the construction of a useful taxonomy such as the one used in Fig. 21.21. Such a taxonomy cannot be defined at the word level because many logical objects and concepts are, in fact, multiword terms.

In addition to analysis tasks, we have in the system a set of tools for exploring the document collections based on the created taxonomy. In Fig. 21.22 we can see such an interactive distribution-browsing tool. We started by computing the distribution of all alliance-related topics. The most frequent topic was “join-venture,” for which we then computed the company distribution. IBM was the company that cooccurred the most with “join-venture.” We then chose to compute the company distribution of MCI (in the context of “joint venture”); Sprint

america online inc, bertelsmann ag \Rightarrow joint venture 13/0.72

apple computer inc, sun microsystems inc \Rightarrow merger talk 22/0.27

apple computer inc, taligent inc \Rightarrow joint venture 6/0.75

sprint corp, tele-communications inc \Rightarrow alliance 8/0.25

burlington northern inc, santa fe pacific corp \Rightarrow merger 9/0.23

lockheed corp, martin marietta corp \Rightarrow merger 14/0.4

chevron corp, mobil corp \Rightarrow joint venture 11/0.26

intuit inc, novell inc \Rightarrow merger 8/0.47

bank of boston corp, corestates financial corp \Rightarrow merger talk 7/0.69

FIG. 21.21. A sample of the association rules that comply with the constraints specified in the rule filter shown in Fig. 21.20.

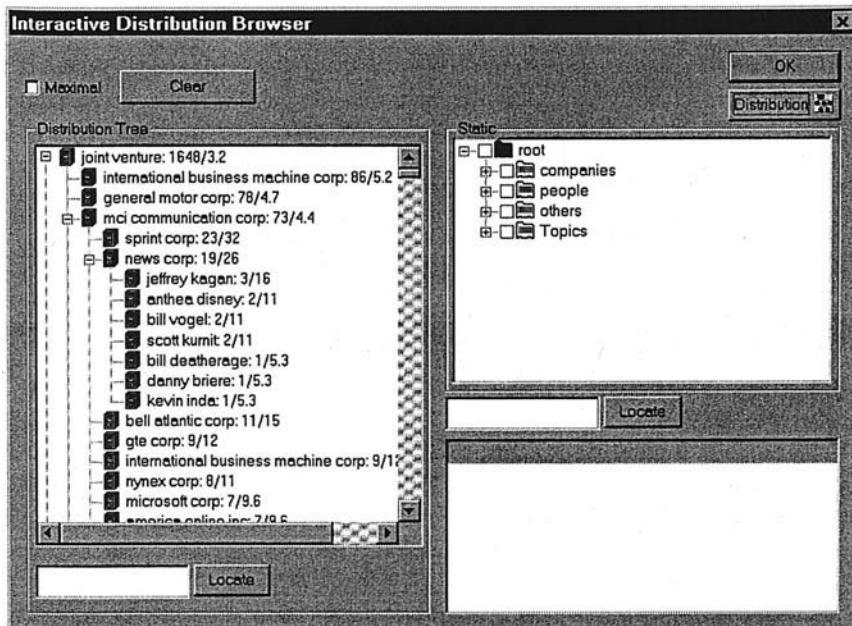


FIG. 21.22. Interactive exploration of term distributions.

was the company with the highest frequency. We then chose to compute the people distribution of News Corp. in the context of “joint venture” and “MCI Communication Corp.”

IMPLEMENTATION ISSUES OF TEXT MINING

In this section we outline several of the practical problems that one would typically face when trying to develop a text mining system. We describe the problems and review some of the common solutions to these problems.

Soft Matching

Soft matching is the problem of matching synonyms that refer to the same entity. As is often the case, different authors may use different phrases when referring to the same entity. Sometimes, even the same author may refer to the entity in different ways. In addition, in some cases the multiple names of the same entity are due to errors and variation in spelling. In this section we refer just to proper name reference to entities; the problem of pronouns will be discussed later in the anaphora resolution section.

Examples. We list here several common scenarios that cause the text mining system to have multiple names for the same entity.

1. Punctuation variations: WalMart versus Wal-mart or Wal Mart, Microsoft Corp. versus Microsoft Corp
2. Spelling mistakes: Microsoft versus Micorsoft
3. Use of abbreviations: GM for General Motors, IBM for International Business Machines
4. Formal name versus Informal name: Microsoft versus Microsoft Corp. or Microsoft Corporation
5. “Nicknames”: Big Blue for IBM

This problem of soft matching (Tejada, Knoblock, & Minton, 2001) is applicable to many types of proper names such as those of companies, organizations, countries, cities (Big Apple for NYC), people, product names, and so forth.

Solutions. The most widely used technique for correcting spelling mistakes is using a soundex algorithm. This algorithm can match words that have a similar phonetic pronunciation. The problems of abbreviations and nicknames can be solved by using a lookup table that will contain for each entity all known abbreviations and nicknames. The problems of punctuation variations and variation in the formality of the entity names can be solved by coding name conversion rules. An example of such a conversion rule is X Corporation or X Corp. are mapped to X. Note that the application of such broad rules across documents can be dangerous at times and link together names that do not refer to the same entity (for example, there might be Highland Partners, Highland Corporation, and Highland Inc., each referring to a different entity). In many systems a more conservative approach is used and only names that appear within the same documents are candidates for becoming synonyms.

Temporal Resolution

When extracting events and facts from the documents we want to associate them with the date and time in which they occurred. This time-stamp will be used for temporal analysis of the document collection (see the later section about Trend Graphs).

There are two main families of date and time formats: absolute format and relative format. Examples of absolute date formats are 10/5/2002, or January 5, 2001. Examples of relative date formats are 3 days ago, yesterday, last month, a year ago, and so forth.

To “time-stamp” any event mentioned inside a document we need to perform the following two-step procedure:

1. Determine the absolute data of the document: This can be done either by analyzing the document and extracting the date of the document, or if no date can be extracted directly from within the document, use the date the document was created.
2. Identify the relative phrase that describes when the event took place and, based on the absolute date of the document, compute the absolute date of the event.

Some of the challenges in identifying the absolute date of the document are related to the large variety of possible date formats (American date, European date, short notation, long notation, abbreviations, etc.). Most of those problems can be solved by coding date conversion rules that convert all dates to one canonic form. Some temporal phrases that are fuzzy in nature pose an even more difficult problem because it is hard to resolve them into a specific date. Examples are “at a later date,” “in the very near future,” and others. The solution in this case is to utilize fuzzy logic, and rather than providing a sharp date, to provide a fuzzy set that represents a fuzzy date.

Anaphora Resolution

One of the main challenges in developing comprehensive text mining systems is anaphora resolution, or the ability to resolve coreferences (Frantzi, 1997; Hobbs 1986; Hobbs, Stickel, Appelt, & Martin, 1993; Ingria & Stallard, 1989; Lappin & McCord, 1990). Consider, for example, the following text fragment from the Chicago Tribune:

Mohamed Atta, a suspected leader of the hijackers, had spent time in Belle Glade, Fla., where a crop-dusting business is located. Atta and other Middle Eastern men came to South Florida Crop Care nearly every weekend for two months.

Will Lee, the firm’s general manager, said the men asked repeated questions about the crop-dusting business. He said the questions seemed “odd,” but he didn’t find the men suspicious until after the Sept. 11 attack.

It is fairly easy to conclude that “Atta” refers to Mohammad Atta; it is a little more difficult to conclude that “he” refers to Will Lee. However, it is much more difficult to infer that “men” refers to Mohammad Atta and his friends, because this coreference appears in a different paragraph and does not include any direct reference to Atta. In general, it was found that resolving proper names and aliases (such as GM for General Motors) was fairly easy; resolving pronominals such as “he,” “she,” and “we” was harder; and resolving definite noun phrases such as “the ruthless man” was the most difficult and the most error prone. The approach taken is a knowledge based approach in which for each referring phrase all accessible antecedents are collected. The accessible antecedents are computed based on the type of the referring phrase.

For proper names all previous entities serve as candidates. For pronouns, entities that appear within the previous sentences of the current paragraph are used. For definite noun phrases all entities that appear within the current paragraph and the preceding paragraph are used. One exception to this heuristic is that for entities of the form “the X” (where X was one of company, organization, corporation, etc.) the scope was extended to the whole preceding text. To select the right antecedent from the set of all possible candidates, the candidates that are incompatible with the referring phrase are eliminated (either due to gender, type, or plurality). From the filtered set the final candidate is selected according to the following heuristics (in order of importance):

1. Prefer the candidate that appears earlier in the current sentence
2. Prefer the candidate that appears earlier in the previous sentence
3. Prefer the candidate that appears later within other sentences (prioritized in descending order of their position in the document)

To compute the effectiveness of these anaphora resolution heuristics for the example from the Chicago Tribune, the number of pairs (of referring expression and antecedent) that correctly matched was computed. It was found that in 82% of cases the right match for the referring expression was performed.

TO Parse or Not to Parse?

Based on actual empirical evaluation, it was found that it is enough to focus just on the core constituents and use shallow parsing augmented by “smart skips.” These skips enable the information extraction engine to skip irrelevant parts and focus just on the important phrases of each sentence. Other researchers have attempted to use full parsing as a component in their information systems and have concluded that it was not worthwhile to invest the extra effort. Specifically, full parsing was included in the SRI TACITUS system (implemented for Message Understanding Conference [MUC]-3) (Hobbs et al., 1992, 1993) and the NYU PROTEUS system (implemented for MUC-6) (Grishman, 1996). Neither of these systems gained any improvement in accuracy due to the full parsing employed. The main problem with using full parsing is that, due to the combinatorial explosion of possible parses, it is very slow and very error prone.

Database Connectivity

Without database connectivity a text mining application is isolated. The application developer must consider database interfaces both for bringing data into the text mining application and for outputting and storing the extracted information.

Input. During the development of many text mining applications, it was noticed that often the analyst would like to use background information about entities, which is not available in the documents. This background information enables the analyst to perform filtering and clustering and to automatically color entities in various colors. For instance, when dealing with relationships between companies, an analyst would often like to use the Standard Industrial Code (SIC) assigned to companies. The SIC enables the analyst to focus, for instance, on relationships between software companies and hardware companies. The relationships are extracted from the documents, but often the actual SIC is not mentioned in the documents. To enable the usage of such background information, a direct connection to an external database

is needed. A database gateway that can connect to external relational databases was developed. This gateway can create virtual nodes in the taxonomy based on properties stored in the database. As an example, all the information from Hoovers was stored in a database and a utility enabled defining groups of companies on the fly based on their various properties. Such a group might be all companies with headquarters in New York City, the industry code “Financial Services—Investment Firms,” and more than 1,000 employees. This utility allows one to create flexible taxonomies that are based on live data residing in relational databases and apply them to entities extracted from any stream of documents.

Output. The ability to output the extracted features from mining text into a database is important for practical text mining applications. Text mining applications typically need to perform various kinds of postmining analysis on the features extracted. Database output of features significantly aids this process.

API. The text mining application typically would mine volumes of text (for example, tens of thousands of news articles per day), and it is essential to store the text mining results in a relational database in a seamless manner. The text mining application must provide an API to perform this function.

VISUALIZATIONS AND ANALYTICS FOR TEXT MINING

When developing a text mining system, one of the crucial needs is the ability to browse through the document collection and be able to “visualize” the various elements within the collection. This type of interactive exploration enables one to identify new types of entities and relationships that can be extracted and better explore the results of the information extraction phase.

We provide an example by using a visualization tool called ClearResearch (Aumann et al., 1999; Feldman et al., 2001, 2002). This visualization tool enables the user to visualize relationships between entities that were extracted from the documents. The system enables the user to view collocations between entities or a semantic map that will show entities that are related by any of a user-definable set of relationships.

We first give some basic definitions and notations.

Definitions and Notations

Let T be a taxonomy. T is represented as a DAG (directed acyclic graph), with the terms at the leaves. For a given node $v \in T$, we denote by $\text{Terms}(v)$ the terms that are decedents of v .

Let D be a collection of documents. For terms e_1 and e_2 we denote $\sup_D(e_1, e)$ the number of documents in D that indicate a relationship between e_1 and e_2 . The nature of the indication can be defined individually according to the context. In the current implementation we say that a document indicates a relationship if both terms appear in the document in the same sentence. This has proved to be a strong indicator. Similarly, for a collection D and terms e_1, e_2 and c , we denote by $\sup_D(e_1, e_2, c)$ the number of documents that indicate a relationship between e_1 and e_2 in the context of c (e.g., relationship between the UK and Ireland in the context of peace talks). Again, the nature of indication may be determined in many ways. In the current implementation we require that they all appear in the same sentence.

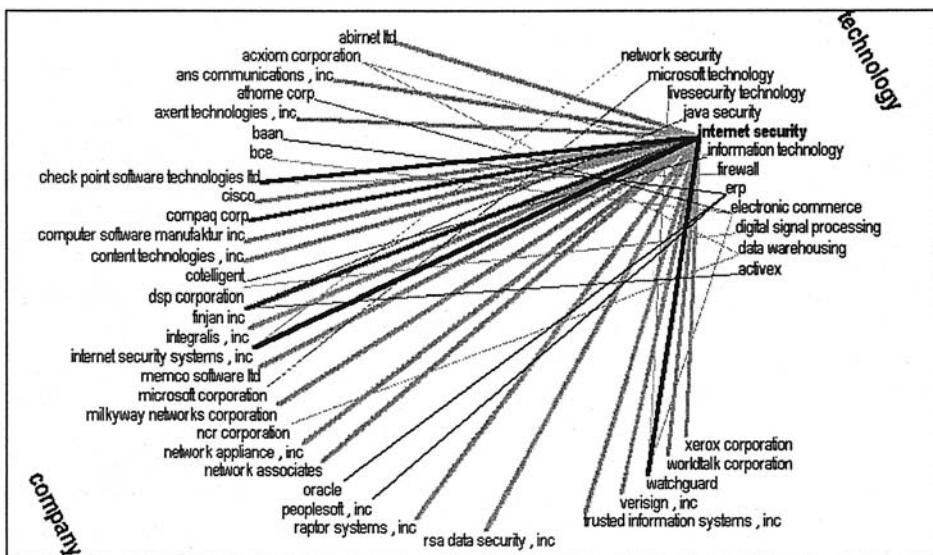


FIG. 21.23. Category connection map. The graph presents the connection between “companies” and “technologies.” The information is based on 5,413 news articles obtained from Marketwatch.com. The user chose to highlight the “internet security” connections.

Category Connection Maps

Category connection maps (Aumann et al., 1999) provide a means for concise visual representation of connections between different categories, for example, between companies and technologies, countries and people, or drugs and diseases. To define a category connection map, the user chooses any number of categories from the taxonomy. The system finds all the connections between the terms in the different categories. To visualize the output, all the terms in the chosen categories are depicted on a circle, with each category placed on a separate part on the circle. A line is depicted between terms of different categories that are related. A color coding scheme represents stronger links with darker colors. An example of a category connection map is presented in Fig. 21.23. Here, the map is for the categories “companies” and “technologies.” The underlying data set consists of 5,413 news articles downloaded from Marketwatch.com.

Formally, given a set $C = \{v_1, v_2, \dots, v_k\}$ of taxonomy nodes and a document collection D , the category connection map is the weighted G defined as follows. The nodes of the graph are the set $V = \text{terms}(v_1) \cup \text{terms}(v_2) \cup \dots \cup \text{terms}(v_k)$. Nodes $u, w \in V$ are connected by an edge if u and w are from different categories and $\text{sup}_D(u, w) > 0$. The weight of the edge (u, w) is $\text{sup}_D(u, w)$.

An important point to notice regarding category connection maps is that the map presents in a single picture information from the entire collection of documents. In the specific example of Fig. 21.23, there is no single document that has the relationship between all the companies and the technologies. Rather, the graph depicts aggregate knowledge from hundreds of documents. Thus, the user is provided with a bird’s-eye summary view of data from across the collection.

Category connection maps are dynamic in several ways. First, the user can choose any node in the graph, and the links from this node are highlighted. In the example in Fig. 21.23, the user chose “Internet Security,” and all the edges emerging from this node are highlighted. In

addition, a double-click on any of the edges brings the list of documents that support the given relationship, together with the most relevant sentence in each document. Thus, in a way the system is the opposite of search engines. Search engines point to documents, in the hope that the user will be able to find the necessary information. Category connection maps present the user with the information itself, which can then be backed by a list of documents.

Relationship Maps

Relationship maps provide a visual means for concise representation of the relationship between many terms in a given context. To define a relationship map the user defines:

- A taxonomy category (e.g., “companies”), which determines the nodes of the circle graph (e.g., companies)
- An optional context node (e.g., “joint venture”), which will determine the type of connection to be found among the graph nodes

Formally, for a set of taxonomy nodes vs , and a context node C , the relationship map is a weighted graph on the node set $V = \text{terms}(vs)$. For each pair $u, w \in V$ there is an edge between u and w , if there exists a context term $c \in C$, such that $\text{sup}_D(u, w, c) > 0$. In this case the weight of the edge is $\sum_{c \in C} \text{sup}_D(u, w, c)$. If no context node is defined, then the connection can be in any context. Formally, in this case the root of the taxonomy is considered as the context.

A relationship map for “companies” in the context of “joint venture” is depicted in Fig. 21.24. In this case the graph is clustered, as described below. The graph is based on 5,413 news documents downloaded from Marketwatch.com. The graph gives the user a summary of the

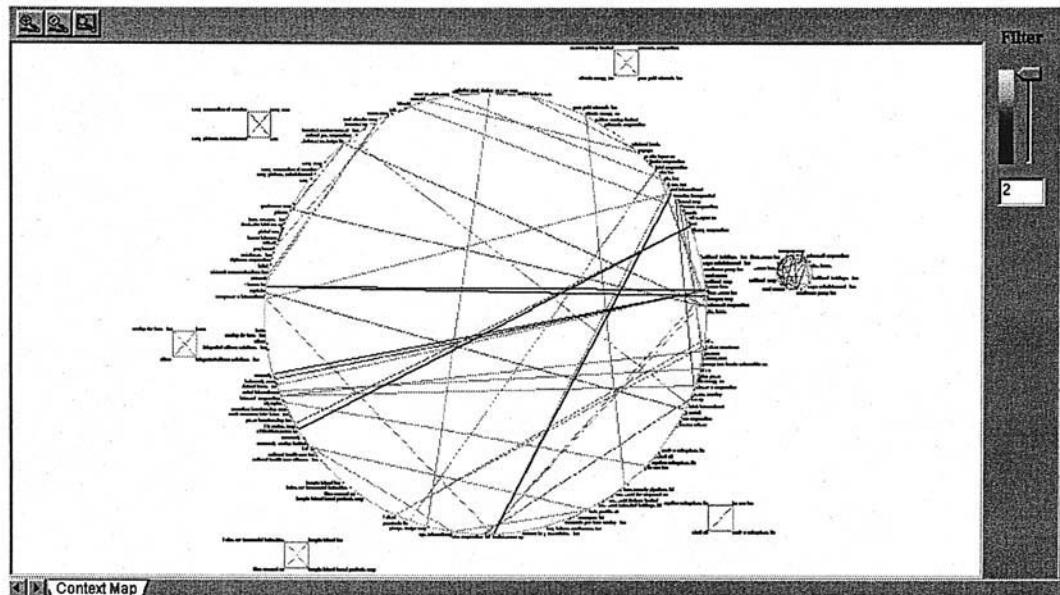


FIG. 21.24. Relationship map. The graph presents the connections between companies in the context of “joint venture.” Clusters are depicted separately. The information is based on 5,413 news articles.

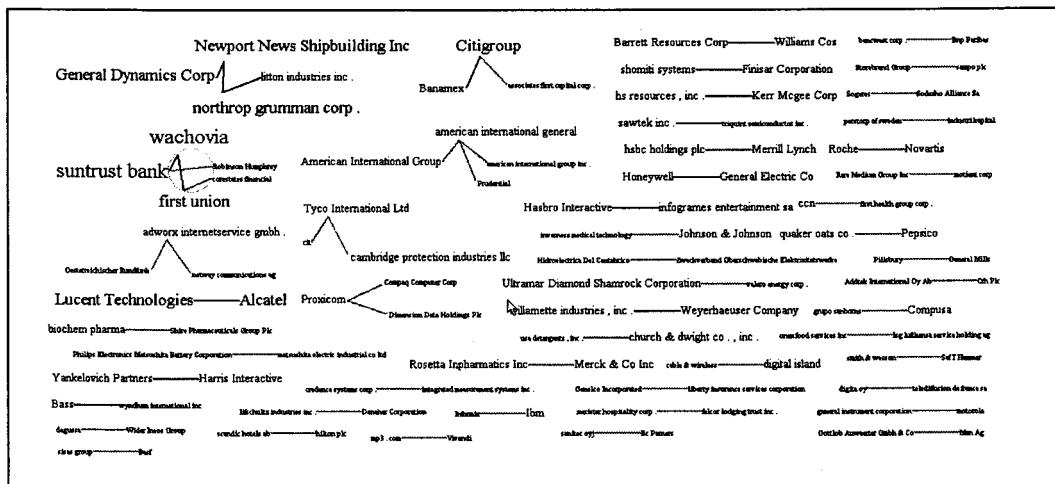


FIG. 21.25. A relationship map among all companies connected by any kind of acquisition relationship.

entire collection in one visualization. The user can appreciate the overall structure of the connections between companies in this context even before reading a single document.

A different type of visualization of relationships is shown in Fig. 21.25, which shows relationships between companies that are related by some type of acquisition relationship (planned, historic, or actual acquisition).

A spring graph is a two-dimensional graph in which the distance between two elements should reflect the strength of the relationships between the elements. The stronger the relationship the closer the two elements should be. An example of a spring graph is shown in Fig. 21.26. The graph represents the relationships between the people in a document

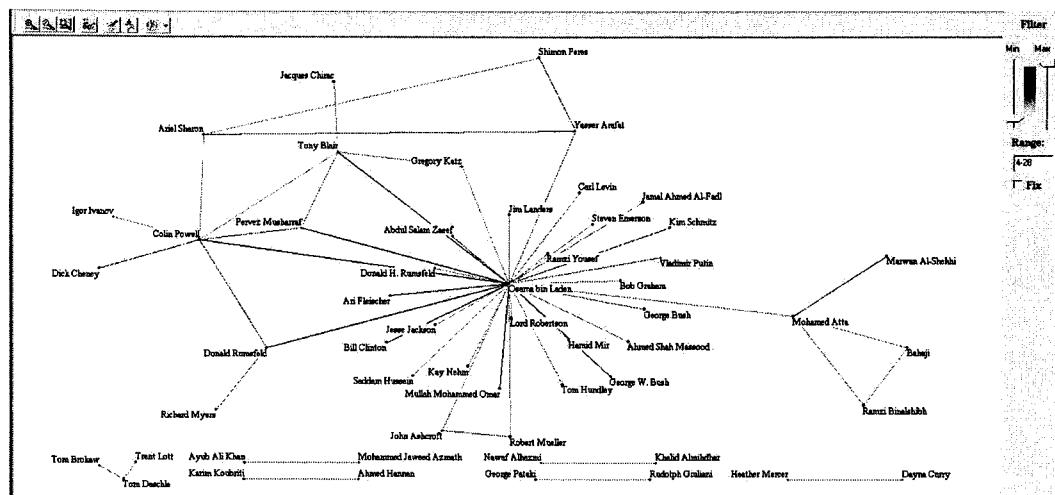


FIG. 21.26. Spring graph that shows the relationship between people around the 9/11 event (source: Yahoo News).

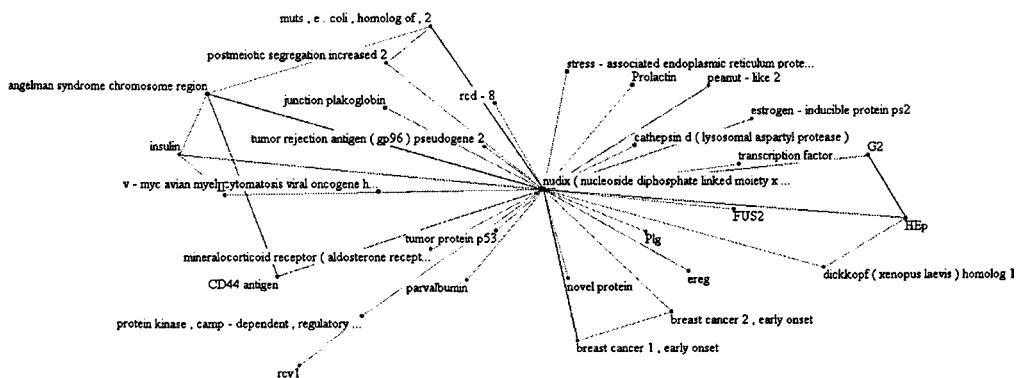


FIG. 21.27. Spring graph that shows the cooccurrence relationship between gene phrases (in the context of various cancer types). (Source: Medline.)

collection containing 2,210 news articles focusing on the 9/11 events. We can see that Osama bin Laden is at the center connected to many of the other key players related to the tragic events.

Another example of a spring graph is shown in Fig. 21.27. This figure depicts the co-occurrence (within the same sentence) relationships between gene phrases in the context of any type of cancer. The document collection is 50,000 Medline articles.

Clustering. For relationship map we use clustering to identify clusters of nodes that are strongly interrelated in the given context. In the example of Fig. 21.24, the system identified six separate clusters. The edges between members of each cluster are depicted in a separate small relationship map, adjacent to the center graph. The center graph shows connections between terms of different clusters and those with terms that are not in any cluster. We now describe the algorithm for determining the clusters.

Note that the clustering problem here is different from the classic clustering problem. In the classic problem we are given points in some space and seek to find clusters of points that are close to each other. Here, we are given a graph in which we are seeking to find dense subgraphs of the graph. Thus, a different type of clustering algorithm is necessary.

The algorithm is composed of two main steps. In the first step we assign weights to edges in the graph. The weight of an edge reflects the strength of the connection between the vertices. Edges incident to vertices in the same cluster should be associated with high weights.

In the next step we identify sets of vertices that are dense subgraphs. This step uses the weights assigned to the edges in the previous step.

We first describe the weight assignment method. To evaluate the strength of a link between a pair of vertices u and v , we consider the following two criteria.

Let u be a vertex in the graph. We use the notation $\Gamma(u)$ to represent the neighborhood of u . The *cluster weight* of (u, v) is affected by the similarity of $\Gamma(u)$ and $\Gamma(v)$. We assume that vertices within the same clusters have many common neighbors.

Existence of many common neighbors is not a sufficient condition, because in dense graphs any two vertices may have some common neighbors. Thus, we emphasize the neighbors that are close to u and v in the sense of *cluster weight*. Suppose $x \in \Gamma(u) \cap \Gamma(v)$; if the *cluster weights* of (x, u) and (x, v) are high, there is a good chance that x belongs to the same cluster as u and v .

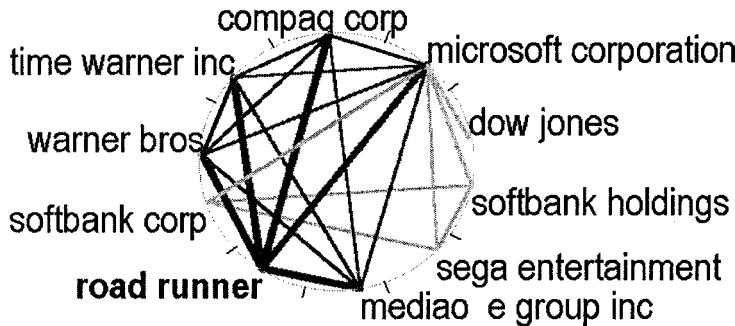


FIG. 21.28. Cluster details (details of one of the clusters from Fig. 21.24).

We can now define an *update operation* on an edge (u, v) that takes into account both criteria:

$$w(u, v) = \sum_{x \in \Gamma(u) \cap \Gamma(v)} w(x, u) + \sum_{x \in \Gamma(u) \cap \Gamma(v)} w(x, v)$$

The algorithm starts with initializing all weights to be equal, $w(u, v) = 1$ for all u, v . Next, the update operation is applied to all edges iteratively. After a small number of iterations (set to five in the current implementation) it stops and outputs the values associated with each edge. We call this the *cluster weight* of the edge.

The cluster weight has the following characteristic. Consider two vertices u and v within the same dense subgraph. The edges within this subgraph mutually affect each other. Thus, the iterations drive cluster weight $w(u, v)$ up. If, however, u and v do not belong to the dense subgraph, the majority of edges affecting $w(u, v)$ will have lower weights, resulting in a low *cluster weight* assigned to (u, v) .

After computing the weights, the second step of the algorithm finds the clusters. We define a new graph with the same set of vertices. In the new graph we consider only a small subset of the original edges, the weights of which were the highest. In our experiments we took the top 10% of the edges. Because now almost all of the edges are likely to connect vertices within the same dense subgraph, we thus separate the vertices into clusters by computing the connected components of the new graph and considering each component as a cluster.

Fig. 21.24 shows a circle context graph with six clusters. The clusters are depicted around the center graph. Each cluster is depicted in a different color. Nodes that are not in any cluster are colored gray. Note that the nodes of a cluster appear both in the central circle and in the separate cluster graph. Edges within the cluster are depicted in the separate cluster graph. Edges between clusters are depicted in the central circle. Fig. 21.28 shows the details of one of the clusters (the cluster on the right). This cluster represents a group of companies with a strong interrelationship in the context of joint venture.

Interactive Features. The graphs are interactive. A double-click on any line segment brings up the list of documents that support the connection, together with the relevant sentence within each document. Fig. 21.29 shows this list of documents supporting the edge connecting “road runner” and “Microsoft” in Fig. 21.28. The system highlights the main terms relevant for the query. In this case these are “road runner,” “Microsoft,” and the context “joint venture.” Another double-click on any document in the document list brings up the full text of the document. The most relevant sentence from each document is presented, and the relevant terms are highlighted.

Titles Browser

microsoft corporation , road runner And One Of: joint venture

4 Documents Found

ID	Date	Title
791	19/01/99	Road Runner Ends 1998 With 180,000 Customers; Growth Co... Road Runner is a joint venture among affiliates of Time Warner Inc., MediaOne Group, Microsoft Corp., Compaq Corp. and Advance/Newhouse.
952	20/01/99	Road Runner Service to Feature SegaSoft's HEAT.NET HEAT.NET to be Premiere Multi-Player Gaming Service on Road Runner. Hundreds of games, fastest connections, equal ultimate competitive arena Road Runner, the joint venture between Time Warner, MediaOne, Microsoft, Compaq and Advance Newhouse, announced today that it
1860	27/01/99	Road Runner Launches On Fanch CABLECOMM Systems in P... Road Runner is a joint venture among affiliates of Time Warner Inc., MediaOne Group, Microsoft Corp., Compaq Corp. and Advance/Newhouse.
2366	01/02/99	Multimedia Cablevision Signs With Road Runner Service to La... Road Runner has in excess of 180,000 customers within the 50 plus communities that it serves within the U.S. It is a joint venture among affiliates of Time Warner Inc., MediaOne Group, Microsoft Corp., Compaq Corp., and Advance/ Newhouse.

Exit

FIG. 21.29. Document list (the list of documents supporting the connection between "road runner" and "microsoft" in Fig. 21.28).

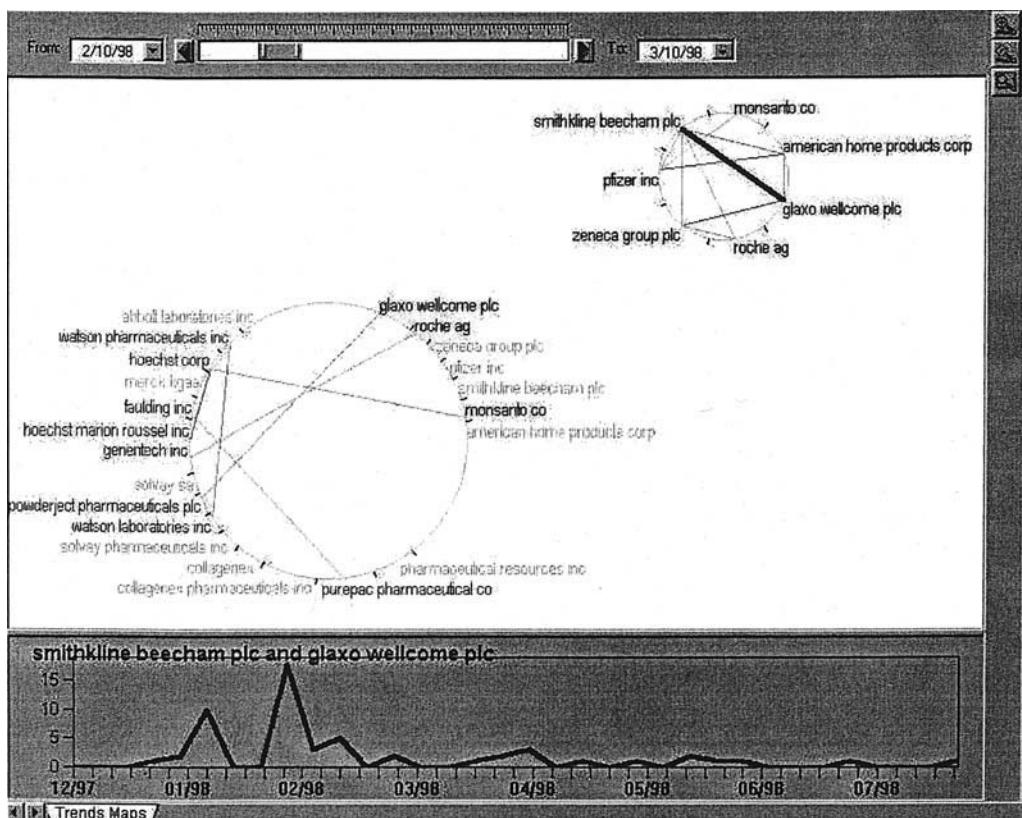


FIG. 21.30. Trend maps. Connections between pharmaceutical companies. Based on 64,000 Reuters news articles from 1998.

Titles Browser		
smithkline beecham plc, glaxo wellcome plc In: [1/2/1998 , 7/2/1998]		
10 Documents Found		
ID	Date	Title
2714	02/02/98	BEFORE THE BELL - SMITHKLINE, GLAXO ADRS JUMP. American Depository Receipts (ADRs) in Britain's Glaxo Wellcome Plc and SmithKline Beecham Plc rocketed up Monday on news the pharmaceutical giants were in
2912	02/02/98	EUROPEAN REGULATORS TO EYE ANY GLAXO DEAL CLOS... 02/02/98 BELGIUM-BRUSSELS EUROPEAN REGULATORS TO EYE ANY GLAXO DEAL CLOSELY. by Fredrik Dahl Europe's top antitrust regulator said Monday he would look closely at any deal between Glaxo Wellcome Plc and SmithKline Beecham Plc, which shocked the
2932	02/02/98	ASTRA, P&U JUMP ON MERGER HOPES. S. group Pharmacia &Uppjohn rose sharply at market open on Monday on news that Glaxo Wellcome PLC and SmithKline Beecham Plc plan to merge.
2852	02/02/98	RESEARCH ALERT - GLAXO RAISED TO BUY. Salomon Smith Barney said Monday it raised its rating on Glaxo Wellcome Plc to buy-medium risk from outperform-medium risk based on the company's proposed merger with SmithKline Beecham Plc. -- SmithKline's rating remained a buy-medium risk.
2883	02/02/98	U.S. HEADLINE STOCKS - ISSUES TO WATCH FEBRUARY 2. Stocks to watch this morning: -- U.S. stocks in London were boosted by news of merger talks between Glaxo Wellcome and SmithKline Beecham Plc, and a strong performance by Asian
2773	02/02/98	RESEARCH ALERT - GLAXO, SMITHKLINE RAISED. HSBC James Capel said Monday it raised its rating on shares of Glaxo Wellcome Plc to buy from reduce and SmithKline Beecham Plc to buy from add on news the two drug companies are in merger talks.

Titles Browser		
smithkline beecham plc, glaxo wellcome plc In: [22/2/1998 , 28/2/1998]		
18 Documents Found		
ID	Date	Title
49962	23/02/98	P&U SAYS HAD NO MERGER/ACQUISITION TALKS. Speculation about a possible merger or acquisition of P&U emerged in the market in the wake of a merger of Glaxo Wellcome and SmithKline Beecham Plc. P&U's performance has been disappointing since it was formed in November 1995 by a merger of Sweden's
49758	23/02/98	SMITHKLINE UNABLE TO REACH GLAXO MERGER. SmithKline Beecham Plc said on Monday that merger talks with Glaxo Wellcome Plc have ended because the two companies were unable to complete the terms of the deal, citing "insurmountable" differences between the two companies.
51412	24/02/98	DRUGS STOCKS SLIP AFTER MEGAMERGER STALLS. Shares of U.S. drug-makers edged lower Tuesday after news that the biggest proposed corporate merger ever, between British drug companies Glaxo Wellcome Plc and SmithKline Beecham Plc, had been scrapped.
51528	24/02/98	GLAXO SAYS NOT ACTIVELY SEEKING MERGER PARTNER. Glaxo Wellcome Plc said late Monday it was not actively seeking a merger partner in the wake of collapsed talks between it and fellow British drugmaker SmithKline Beecham Plc. "We are not actively looking for a merger partner," said Glaxo spokesman Rick Sluder.
51554	24/02/98	SMITHKLINE HAS NO PLANS TO RESTART AHP TALKS. British-based SmithKline Beecham Plc said on Tuesday it had no plans to resume negotiations with American Home Products Corp of the U.S. in the light of the failure of merger talks with Glaxo Wellcome Plc. "We have no current plans to resume talks with them," a spokeswoman
51553	24/02/98	GLAXO/SMITHKLINE THE COMPANY THAT GOT AWAY

FIG. 21.31. Document lists for the two peaks of the graph in Fig. 21.30. The top list is for the first peak, the bottom list for the second.

Thus, with the context connection graphs the user first gets the global picture of all connections in a given context, then can zoom-in on any of the clusters or drill-down to the supporting documents. This allows for an effective and efficient discovery process.

Trend Graphs

Trend graphs are designed to allow the user to view changes in relationships over time, thus identifying trends and patterns. The basis for the definition of trend graphs is that of context connection graphs. Given a context connection graph, the associated trend graph is a dynamic graph that enables one to see the connections within each time period. Fig. 21.30 depicts a trend graph for the connection between pharmaceutical companies (no context). The circle in the upper right corner is the only cluster in the graph. The main graph is in the lower left corner. This graph is based on 64,000 news articles of Reuters from 1998.

Note the slider and the dates on the top part of the screen. The dates indicate that the edges shown in the current picture represent only connections based on articles published between 2.10.98 and 3.10.98. Moving the slider changes the time period, thus showing how the connections change over time. In addition, the length of the time interval can be changed.

Clicking on any single edge shows the trend for that pair. In the example in Fig. 21.30 the trend for the “smithkline beecham plc” and “glaxo wellcome plc” is shown. Notice the two peaks in the graph, one in January 1998 and one a month later in February 1998. Fig. 21.31 gives the list of articles supporting each peak. A quick look at these lists shows that the first peak is related to a merger announced between the two companies. The second peak is when the merger was called off! Thus, the graph helps to discover instantly the key elements of dynamics of the relationship between the two companies.

SUMMARY

Due to the abundance of available textual data, there is a growing need for efficient tools for text mining. Unlike structured data, in which the data mining algorithms can be performed directly on the underlying data, textual data requires some preprocessing before the data mining algorithm can be successfully applied. Information extraction has proved to be an efficient method for this first preprocessing phase. Text mining based on information extraction attempts to hit a midpoint, reaping some benefits from each of the extremes while avoiding many of their pitfalls. On the one hand, there is no need for human effort in labeling documents, and we are not constrained to a smaller set of labels that lose much of the information present in the documents. Thus, the system has the ability to work on new collections without any preparation, as well as the ability to merge several distinct collections into one (even though they might have been tagged according to different guidelines that would prohibit their merger in a tagged-based system). On the other hand, the number of meaningless results is greatly reduced, and the execution time of the mining algorithms is also reduced relative to pure word-based approaches. Text mining using information extraction thus hits a useful middle ground in the quest for tools for understanding the information present in the large amount of data that is only available in textual form. The powerful combination of precise analysis of the documents and a set of visualization tools enables the user to easily navigate and utilize very large document collections.

REFERENCES

- Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., & Tyson, M. (1993a). FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings of IJCAI-93* (pp. 1172–1178).
- Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D., Kameyama, M., & Tyson, M. (1993b). The SRI MUC-5 JV-FASTUS information extraction system. In *Proceedings of the Fifth Message Understanding Conference (MUC-5)*.
- Aumann, Y., Feldman, R., Ben Yehuda, Y., Landau, D., Lipshtat, O., & Schler, Y. (1999). Circle graphs: New visualization tools for text-mining. In *Principles of Data Mining and Knowledge Discovery, Third European Conference (PKDD'99)* (pp. 277–282). Prague: Springer.
- Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21, 543–565.
- Cohen, W. (1992). *Compiling prior knowledge into an explicit bias*. Working notes of the 1992 AAAI Spring Symposium on Knowledge Assimilation, Stanford, CA, March.
- Cohen, W., & Singer, Y. (1996). Context Sensitive Learning Methods for Text categorization. In Proceedings of SIGIR'96.
- Cowie, J., & Lehnert, W. (1996). Information extraction. *Communications of the Association of Computing Machinery*, 39, 80–91.
- Daille, B., Gaussier, E., & Lange, J. M. (1994). Towards automatic extraction of monolingual and bilingual terminology. In *Proceedings of the International Conference on Computational Linguistics* (pp. 515–521).
- Dumais, S. T., Platt, J., Heckerman, D., & Sahami, M. (1998) Inductive learning algorithms and representations for text categorization. In *Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM'98)* (pp. 148–155). New York: ACM Press.
- Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19 (1).
- Feldman, R., Aumann, Y., Amir, A., Klösgen, W., & Zilberstien, A. (1997). Maximal association rules: A new tool for mining for keyword co-occurrences in document collections. *Proceedings of the Third International Conference on Knowledge Discovery 1997* (pp. 167–170). Menlo Park, CA: AAAI Press.
- Feldman, R., Aumann, Y., Finkelstein-Landau, M., Hurvitz, E., Regev, Y., & Yaroshevich, A. (2002). “A comparative study of information extraction strategies.” *CICLing 2002*, (pp. 349–359). Lecture Notes in Computer Science 2276, Berlin: Springer.
- Feldman, R., Aumann, Y., Liberzon, Y., Ankori, K., Schler, J., & Rosenfeld, B. (2001). A domain independent environment for creating information extraction modules. *CIKM 2001*, (pp. 586–588). New York: ACM Press.
- Feldman, R., & Dagan, I. (1995). KDT—knowledge discovery in texts. In *Proceedings of the First International Conference on Knowledge Discovery*, (pp. 112–117). Menlo Park, CA: AAAI Press.
- Feldman, R., & Hirsh, H. (1997). Exploiting background information in knowledge discovery from text. *Journal of Intelligent Information Systems* 9, 83–97. Norwell, MA: Kluwer Academic Publishers.
- Feldman, R., Rosenfeld, B., Stoppa, J., Liberzon, Y., & Schler, J. (2000). A framework for specifying explicit bias for revision of approximate information extraction rules. *Proceedings of the Sixth International Conference on Knowledge Discovery 2000* (pp. 189–199). New York: ACM Press.
- Fisher, D., Soderland, S., McCarthy, J., Feng, F., & Lehnert, W. (1995). Description of the UMass systems as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference* (pp. 127–140).
- Frantzi, T. K. (1997). Incorporating context information for the extraction of terms. In *Proceedings of ACL-EACL'97* (pp. 501–503). San Francisco: Morgan Kaufmann.
- Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1991). Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro & W. J. Frawley (Eds.), *Knowledge Discovery in Databases* (pp. 1–27). Cambridge, MA: MIT Press.
- Grishman, R. (1996). The role of syntax in information extraction. In *Advances in text processing: Tipster program phase II*. San Francisco: Morgan Kaufmann.
- Hahn, U., & Schnattinger, K. (1997). Deep knowledge discovery from natural language texts. *Proceedings of the Third International Conference on Knowledge Discovery 1997* (pp. 175–178). Menlo Park, CA: AAAI Press.
- Hayes, P. (1992). Intelligent high-volume processing using shallow, domain-specific techniques. In *Text-based intelligent systems: current research and practice in information extraction and retrieval* (pp. 227–242).
- Hayes, P. J., & Weinstein, S. P. (1990). CONTRUE: A system for content-based indexing of a database of news stories. In *Proceedings of Second Conference on Innovative Applications of Artificial Intelligence* (pp. 49–64). Menlo Park, CA: AAAI Press.
- Hobbs, J. (1986). Resolving pronoun references, In B. J. Grosz, K. Sparck Jones, & B. L. Webber (Eds.), *Readings in Natural Language Processing* (pp. 339–352). San Francisco: Morgan Kaufmann.

- Hobbs, J. R., Appelt, D. E., Bear, J., Israel, D., Kameyama, M., & Tyson, M., FASTUS: A system for extracting information from text. In *Proceedings of Human Language Technology* (pp. 133–137). Princeton, N.J.
- Hobbs, J. R., Stickel, M., Appelt, D., & Martin P. (1993). Interpretation as abduction. *Artificial Intelligence*, 63, 69–142.
- Hull, D. (1996). Stemming algorithms—a case study for detailed evaluation. *Journal of the American Society for Information Science*, 47, 70–84.
- Ingraham, R., & Stallard, D. (1989). A computational mechanism for pronominal reference. *Proceedings of the Association for Computational Linguistics* (pp. 262–271). San Francisco: Morgan Kaufmann.
- Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of Tenth European Conference on Machine Learning* (pp. 137–142). New York: ACM Press.
- Justeson, J. S., & Katz, S. M. (1995). Technical terminology: some linguistic properties and an algorithm for identification in text. In *Natural Language Engineering*, 1, 9–27.
- Klösgen, W. (1992). Problems for knowledge discovery in databases and their treatment in the statistics interpreter EXPLORA. *International Journal for Intelligent Systems*, 7, 649–673.
- Lappin, S., & McCord, M. (1990). A syntactic filter on pronominal anaphora for slot grammar. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics* (pp. 135–142).
- Larkey, L. S., & Croft, W. B. (1996). Combining classifiers in text categorization. In *Proceedings of SIGIR-96* (pp. 289–297). New York: ACM Press.
- Lehnert, W., Cardie, C., Fisher, D., Riloff, E., & Williams, R. (1991). Description of the CIRCUS system as used for MUC-3. In *Proceedings of the Third Message Understanding Conference* (pp. 223–233).
- Lent, B., Agrawal, R., & Srikant, R. (1997). Discovering trends in text databases. In *Proceedings of the Third International Conference on Knowledge Discovery* (pp. 227–230). Menlo Park, CA: AAAI Press.
- Lewis, D. D. (1995). Evaluating and optimizing autonomous text classification systems. In *Proceedings of SIGIR-95* (pp. 246–254).
- Lewis, D. D., & Hayes, P. J. (1994). Guest editorial (Special issue). *ACM Transactions on Information Systems*, 12, 231.
- Lewis, D. D., & Ringette, M. (1994). A comparison of two learning algorithms for text categorization. In *Proceedings of Symposium on Document Analysis and Information Retrieval* (pp. 81–93).
- Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training algorithms for linear text classifiers. In *Proceedings of SIGIR '96* (pp. 298–306). New York: ACM Press.
- Lin, D. (1995). University of Manitoba: Description of the PIE system as used for MUC-6. In *Proceedings of the Sixth Conference on Message Understanding*.
- Piatetsky-Shapiro, G., & Frawley, W. (Eds.) (1991). *Knowledge Discovery in Databases*. Menlo Park, CA: AAAI Press.
- Rajman, M., & Besançon, R. (1997). Text mining: Natural language techniques and text mining applications. In *Proceedings of the Seventh IFIP 2.6 Working Conference on Database Semantics*. London: Chapman & Hall.
- Riloff, E., & Lehnert, W. (1994). Information extraction as a basis for high-precision text classification, (Special issue). *ACM Transactions on Information Systems* (pp. 296–333). New York: ACM Press.
- Sebastiani, F. (2002). Machine learning in automated text categorization, *ACM Computing Surveys*, 34, 1–47.
- Soderland, S., Fisher, D., Aseltine, J., & Lehnert, W. (1995). Issues in inductive learning of domain-specific text extraction rules. In *Proceedings of the Workshop on New Approaches to Learning for Natural Language Processing at the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1314–1321). Menlo Park, CA: AAAI Press.
- Srikant, R., & Agrawal, R. (1995). Mining generalized association rules. In *Proceedings of the 21st VLDB Conference* (pp. 407–419). San Francisco: Morgan Kaufmann.
- Sundheim, B. (Ed.). (1993). *Proceedings of the Fifth Message Understanding Conference (MUC-5)*. San Mateo, CA: Morgan Kaufmann.
- Tejada, S., Knoblock, C. A., & Minton, S. (2001). Learning object identification rules for information integration. *Information Systems*, 26, 607–633.
- Tipster Text Program (Phase I), (1993). In *Proceedings, Advanced Research Projects Agency*.
- Vapnik, V. (1979). *Estimation of Dependencies Based on Data* (Springer-Verlag, Trans.). Moscow: Nauka.
- Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Berlin: Springer-Verlag.
- Yang, Y., & Chute, C. G. (1994). An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12, 252–277.
- Yang, Y., & Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of SIGIR '99* (pp. 42–49).

22

Mining Geospatial Data

Shashi Shekhar and Ranga Raju Vatsavai
University of Minnesota

Introduction	520
Spatial Outlier Detection Techniques	521
Illustrative Examples and Application Domains	521
Tests for Detecting Spatial Outliers	522
Solution Procedures	525
Spatial Colocation Rules	525
Illustrative Application Domains	526
Colocation Rule Approaches	527
Solution Procedures	530
Location Prediction	530
An Illustrative Application Domain	530
Problem Formulation	532
Modeling Spatial Dependencies Using the SAR and MRF Models	533
Logistic SAR	534
MRF Based Bayesian Classifiers	535
Clustering	537
Categories of Clustering Algorithms	539
K-Medoid: An Algorithm for Clustering	540
Clustering, Mixture Analysis, and the EM Algorithm	541
Summary	544

This work was supported in part by the Army High-Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement DAAD19-01-2-0014, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred.

Acknowledgments	545
References	545

INTRODUCTION

Explosive growth in geospatial data and the emergence of new spatial technologies emphasize the need for the automated discovery of spatial knowledge. Spatial data mining is the process of discovering interesting and previously unknown but potentially useful patterns from spatial databases. The complexity of spatial data and intrinsic spatial relationships limits the usefulness of conventional data mining techniques for extracting spatial patterns. In this chapter we explore the emerging field of spatial data mining and applications, especially the areas of outlier detection, colocation rules, classification/prediction, and clustering techniques that explicitly model spatial neighborhood relationships. We present several case studies utilizing a wide variety of spatial data sets.

Spatial data mining is the process of exploring large spatial data sets for extracting interesting, potentially useful, and recurrent patterns and relationships. Widespread use of spatial databases (Guting, 1994; Shekhar & Chawla, 2002; Shekhar et al., 1999; Worboys, 1995) is leading to an increasing interest in mining interesting and useful, but implicit, spatial patterns (Greenman, 2000; Koperski, Adhikary, & Han, 1996; Mark, 1999; Roddick & Spiliopoulou, 1999; Stolorz et al., 1995). Spatial data sets and patterns are abundant in many application domains related to the National Aeronautics and Space Administration, the National Imagery and Mapping Agency, the National Cancer Institute, and the United States Department of Transportation. Efficient tools for extracting information from geospatial data are crucial to organizations that make decisions based on large spatial data sets. These applications are spread across many domains including ecology and environmental management, public safety, transportation, public health, business, and travel and tourism (Albert & McShane, 1995; Haining, 1989; Hohn, Gribki, & Liebhold, 1993; Issaks, Edward, & Srivastava, 1989; Krugman, 1995; Shekhar, Yang, & Hancock, 1993; Stolorz et al., 1995; Yasui & Lele, 1997).

General purpose data mining tools such as Clementine (SPSS), See5/C5.0 (Research), and Enterprise Miner (SAS) are designed for the purpose of analyzing large commercial databases. Although these tools were primarily designed to identify customer-buying patterns in market-basket data, they also have been used in analyzing scientific and engineering, astronomical, multimedia, genomic, and Web data. Extracting interesting and useful patterns from spatial data sets is more difficult than extracting corresponding patterns from traditional numeric and categorical data due to the complexity of spatial data types, spatial relationships, and spatial autocorrelation. Specific features of geographical data that preclude the use of general purpose data mining algorithms are (a) spatial relationships among the variables, (b) spatial structure of errors, (c) mixed distributions as opposed to commonly assumed normal distributions, (d) observations that are not independent, (e) spatial autocorrelation among the features, and (f) nonlinear interaction in feature space. Of course, one can apply conventional data mining algorithms, but it is often observed that these algorithms perform more poorly on spatial data. Many supportive examples can be found in the literature; for instance, parametric classifiers such as maximum likelihood perform more poorly than nonparametric classifiers when the assumptions about the parameters (e.g., normal distribution) are violated, and Markov random fields (MRFs) perform better than per-pixel based classifiers when the features are autocorrelated.

Now the question arises whether we really need to invent new algorithms or extend the existing approaches to explicitly model spatial properties and relationships. Although it is difficult to tell the direction of future research, for now it seems both approaches are gaining

momentum. In this chapter we present techniques that are specifically designed for analyzing large volumes of spatial data as well as extensions of conventional data mining techniques that explicitly model spatial concepts.

This chapter is organized as follows. In the next section we introduce spatial outlier detection techniques and their use in finding spatiotemporal outliers in traffic data. In the section after the heading “Spatial Colocation Rules” we present a new approach, called colocation mining, which finds the subsets of features frequently located together in spatial databases. The section after the heading “Location Prediction” presents extensions of classification and prediction techniques that model spatial context. Here we present a case study on a bird nest data set using MRFs and spatial autoregressive regression (SAR) techniques. The section after the heading “Clustering” explores expectation maximization (EM) and its spatial extension, neighborhood expectation maximization (NEM), and provides bird nest prediction results. Finally, this chapter concludes with a summary of techniques and results.

Spatial Outlier Detection Techniques

Global outliers have been defined informally as observations in a data set that appear to be inconsistent with the remainder of that set of data (Barnett & Lewis, 1994), or that deviate so much from other observations so as to arouse suspicions that they were generated by a different mechanism (Hawkins, 1980). The identification of global outliers can lead to the discovery of unexpected knowledge and has a number of practical applications in areas such as credit card fraud, athlete performance analysis, voting irregularity, and severe weather prediction. This section focuses on spatial outliers, that is, observations that appear to be inconsistent with their neighborhoods. Detecting spatial outliers is useful in many applications of geographic information systems and spatial databases. These application domains include transportation, ecology, public safety, public health, climatology, and location based services.

We model a spatial data set to be a collection of spatially referenced objects such as houses, roads, and traffic sensors. Spatial objects have two distinct categories of dimensions along which attributes may be measured. Categories of dimensions of interest are spatial and non-spatial. Spatial attributes of a spatially referenced object include location, shape, and other geometric or topological properties. Nonspatial attributes of a spatially referenced object include traffic sensor identifiers, manufacturer, owner, age, and measurement readings. A spatial neighborhood of a spatially referenced object is a subset of the spatial data based on a spatial dimension, for example, location. Spatial neighborhoods may be defined based on spatial attributes such as location, using spatial relationships such as distance or adjacency. Comparisons between spatially referenced objects are based on nonspatial attributes.

A spatial outlier is a spatially referenced object the nonspatial attribute values of which differ significantly from those of other spatially referenced objects in its spatial neighborhood. Informally, a spatial outlier is a local instability (in values of nonspatial attributes) or a spatially referenced object the nonspatial attributes of which are extreme relative to its neighbors, even though the attributes may not be significantly different from the entire population. For example, a new house in an old neighborhood of a growing metropolitan area is a spatial outlier based on the nonspatial attribute house age.

Illustrative Examples and Application Domains

We use an example to illustrate the differences among global and spatial outlier detection methods. In Fig. 22.1(a), the x -axis is the location of data points in one-dimensional space; the y -axis is the attribute value for each data point. Global outlier detection methods ignore the

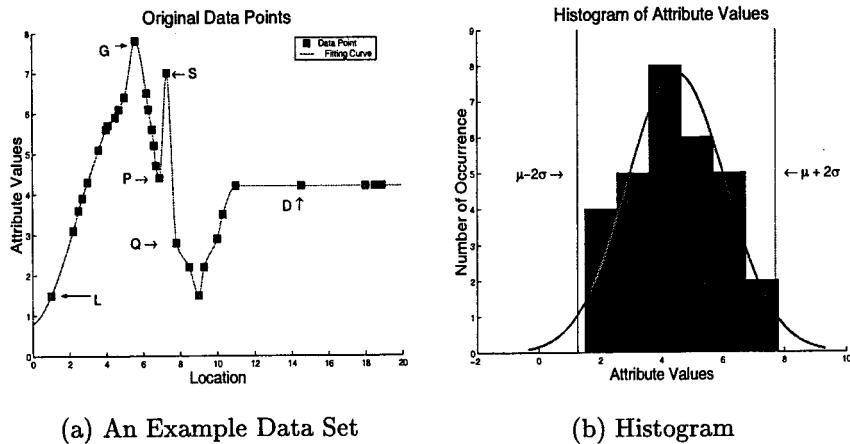


FIG. 22.1. A data set for outlier detection.

spatial location of each data point and fit the distribution model to the values of the nonspatial attribute. The outlier detected using this approach is the data point G , which has an extremely high attribute value of 7.9, exceeding the threshold of $\mu + 2\sigma = 4.49 + 2 * 1.61 = 7.71$, as shown in Fig. 22.1(b). This test assumes a normal distribution for attribute values. On the other hand, S is a spatial outlier with an observed value significantly different from its neighbors P and Q .

As another example we use a spatial database consisting of measurements from the Minneapolis-St. Paul freeway traffic sensor network. The sensor network includes about 900 stations, each of which contains one to four loop detectors, depending on the number of lanes. Sensors embedded in the freeways and interstate monitor the occupancy and volume of traffic on the road. At regular intervals this information is sent to the Traffic Management Center for operational purposes, for instance, ramp meter control, as well as for experiments and research on traffic modeling.

In this application we are interested in discovering the location of stations with measurements that are inconsistent with those of their spatial neighbors and the time periods when those abnormalities arise. The outlier detection tasks are to (a) build a statistical model for a spatial data set; (b) check whether a specific station is an outlier; and (c) check whether stations on a route are outliers.

Tests for Detecting Spatial Outliers

Tests to detect spatial outliers separate spatial attributes from nonspatial attributes. Spatial attributes are used to characterize location, neighborhood, and distance. Nonspatial attribute dimensions are used to compare a spatially referenced object to its neighbors. Spatial statistics literature provides two kinds of bipartite multidimensional tests, namely graphical tests and quantitative tests. Graphical tests, which are based on the visualization of spatial data, highlight spatial outliers. Example methods include variogram clouds and Moran scatterplots. Quantitative methods provide a precise test to distinguish spatial outliers from the remainder of data. Scatterplots (Luc, 1994) are a representative technique from the quantitative family.

A variogram cloud displays data points related by neighborhood relationships. For each pair of locations the square root of the absolute difference between attribute values at the locations

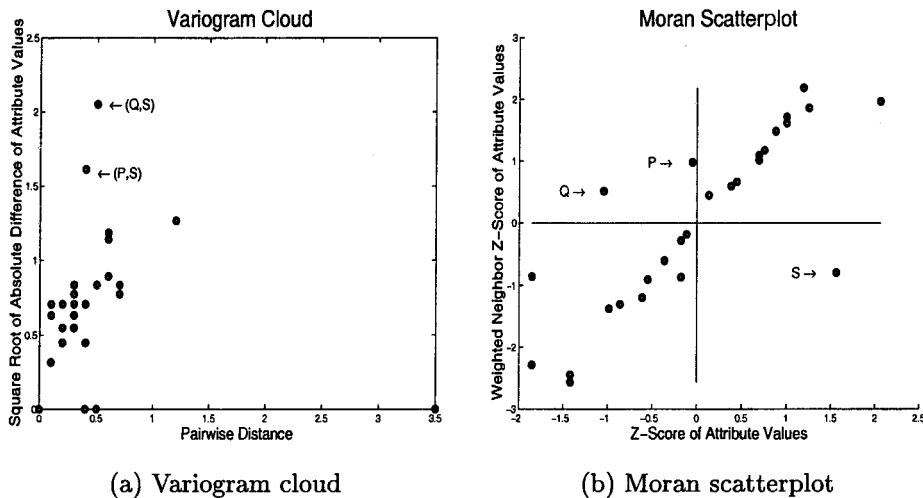


FIG. 22.2. Variogram cloud and Moran scatterplot to detect spatial outliers.

versus the Euclidean distance between the locations is plotted. In data sets exhibiting strong spatial dependence the variance in the attribute differences increase with increasing distance between locations. Locations that are near to one another, but with large attribute differences, might indicate a spatial outlier even though the values at both locations may appear to be reasonable when examining the data set nonspatially. Figure 22.2(a) shows a variogram cloud for the example data set shown in Fig. 22.1(a). This plot shows that two pairs (P, S) and (Q, S) on the left-hand side lie above the main group of pairs and are possibly related to spatial outliers. The point S may be identified as a spatial outlier because it occurs in both pairs (Q, S) and (P, S). However, graphical tests of spatial outlier detection are limited by the lack of precise criteria to distinguish spatial outliers. In addition, a variogram cloud requires nontrivial postprocessing of highlighted pairs to separate spatial outliers from their neighbors, particularly when multiple outliers are present or density varies greatly.

A Moran scatterplot (Luc, 1995) is a plot of normalized attribute value ($Z[f(i)] = \frac{f(i) - \mu_f}{\sigma_f}$) against the neighborhood average of normalized attribute values ($W \cdot Z$), where W is the row-normalized (i.e., $\sum_j W_{ij} = 1$) neighborhood matrix, (i.e., $W_{ij} > 0$ iff neighbor (i, j)). The upper left and lower right quadrants of Fig. 22.2(b) indicate a spatial association of dissimilar values: low values surrounded by high-value neighbors (e.g., points P and Q), and high values surrounded by low values (e.g., point S). Thus, we can identify points (nodes) that are surrounded by unusually high- or low-value neighbors. These points can be treated as spatial outliers.

Definition. A *Moran_{outlier}* is a point located in the upper left and lower right quadrants of a Moran scatterplot. This point can be identified by $(Z[f(i)]) \times (\sum_j (W_{ij} Z[f(j)])) < 0$.

A scatterplot (Luc, 1994) shows attribute values on the x -axis and the average of the attribute values in the neighborhood on the y -axis. A least square regression line is used to identify spatial outliers. A scatter sloping upward to the right indicates a positive spatial autocorrelation (adjacent values tend to be similar); a scatter sloping upward to the left indicates a negative spatial autocorrelation. The residual is defined as the vertical distance (y -axis) between a point P with location (X_p, Y_p) to the regression line $Y = mX + b$, that is, residual $\epsilon = Y_p - (mX_p + b)$. Cases with standardized residuals, $\epsilon_{\text{standard}} = \frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon}$, greater than 3.0 or

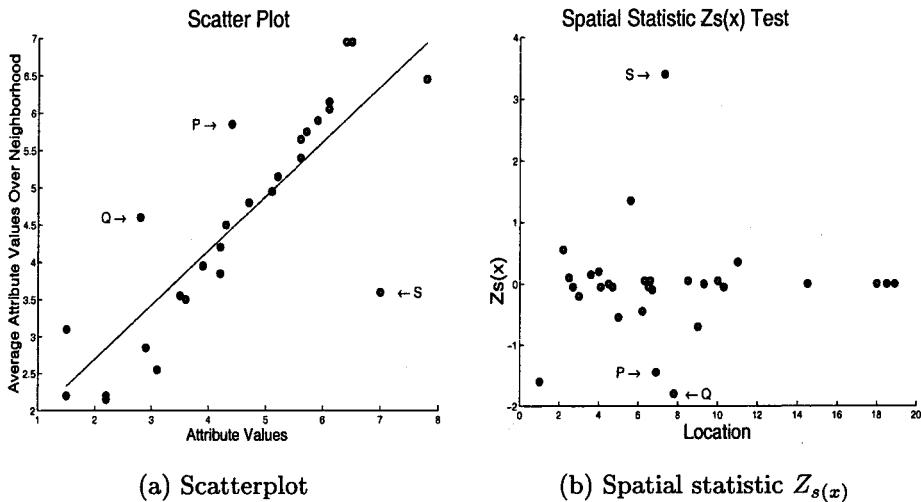


FIG. 22.3. Scatterplot and spatial statistic $Z_{s(x)}$ to detect spatial outliers.

less than -3.0 are flagged as possible spatial outliers, where μ_ϵ and σ_ϵ are the mean and standard deviation of the distribution of the error term ϵ . In Fig. 22.3(a), a scatterplot shows the attribute values plotted against the average of the attribute values in neighboring areas for the data set in Fig. 22.1(a). The point S turns out to be the farthest from the regression line and may be identified as a spatial outlier.

Definition. A *Scatterplot_{outlier}* is a point with significant standardized residual error from the least square regression line in a scatterplot. Assuming errors are normally distributed, then $\epsilon_{\text{standard}} = \left| \frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon} \right| > \theta$ is a common test. Nodes with standardized residuals $\epsilon_{\text{standard}} = \frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon}$ from regression line $Y = mX + b$ and greater than θ or less than $-\theta$ are flagged as possible spatial outliers. The μ_ϵ and σ_ϵ are the mean and standard deviation of the distribution of the error term ϵ .

A location (sensor) is compared with its neighborhood using the function $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$, where $f(x)$ is the attribute value for a location x , $N(x)$ is the set of neighbors of x , and $E_{y \in N(x)}(f(y))$ is the average attribute value for the neighbors of x . The statistical function $S(x)$ denotes the difference of the attribute value of a sensor located at x and the average attribute value of x 's neighbors.

Spatial statistic $S(x)$ is normally distributed if the attribute value $f(x)$ is normally distributed. A popular test for detecting spatial outliers for normally distributed $f(x)$ can be described as follows: Spatial statistic $Z_{s(x)} = \left| \frac{S(x) - \mu_s}{\sigma_s} \right| > \theta$. For each location x with an attribute value $f(x)$, the $S(x)$ is the difference between the attribute value at location x and the average attribute value of x 's neighbors, μ_s is the mean value of $S(x)$, and σ_s is the value of the standard deviation of $S(x)$ over all stations. The choice of θ depends on a specified confidence level. For example, a confidence level of 95% will lead to $\theta \approx 2$.

Figure 22.3(b) shows the visualization of the spatial statistic $Z_{s(x)}$ method described earlier and in example 1. The x -axis is the location of data points in one-dimensional space; the y -axis is the value of spatial statistic $Z_{s(x)}$ for each data point. We can easily observe that point S has a $Z_{s(x)}$ value exceeding 3 and will be detected as a spatial outlier. Note that the two neighboring points P and Q of S have $Z_{s(x)}$ values close to -2 due to the presence of spatial outliers in their neighborhoods. Example 1 has already shown that $Z_{s(x)}$ is a special case of an S -outlier.

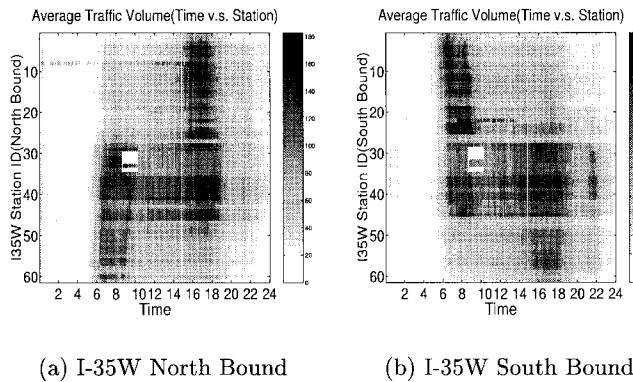


FIG. 22.4. Spatial outliers in traffic volume data.

Solution Procedures

Given the components of the S -outlier definition, the objective is to design a computationally efficient algorithm to detect S -outliers. We presented scalable algorithms for spatial outlier detection in [Shekhar, Lu, & Zhang (2001)], where we showed that almost all statistical tests are “algebraic” aggregate functions over a neighborhood join. The spatial outlier detection algorithm has two distinct tasks: The first deals with model building and the second involves a comparison (test statistic) with spatial neighbors. During model building, algebraic aggregate functions (e.g., mean and standard deviation) are computed in a single scan of a spatial-join using a neighbor relationship. In the second step a neighborhood aggregate function is computed by retrieving the neighboring nodes, and then a difference function is applied over the neighborhood aggregates and algebraic aggregates. This study showed that the computational cost of outlier detection algorithms are dominated by the disk page access time (i.e., the time spent on accessing neighbors of each point). In this study we utilized three different data page clustering schemes: the connectivity-clustered access method (CCAM; Shekhar & Liu, 1997), Z-ordering (Orenstein & Merrett, 1984), and cell-tree (Gunther, 1989) and found that CCAM produced the lowest number of data page accesses for outlier detection.

The effectiveness of the $Z_{s(x)}$ method on a Minneapolis-St. Paul traffic data set is illustrated in the following example. Figure 22.4 shows one example of traffic flow outliers. Figures 22.4(a) and (b) are the traffic volume maps for I-35W north bound and south bound, respectively, on January 21, 1997. The x -axis is a 5-minute time slot for the whole day, and the y -axis is the label of the stations installed on the highway, starting from 1 on the north end to 61 on the south end. The abnormal white line at 2:45 p.m. and the white rectangle from 8:20 a.m. to 10:00 a.m. on the x -axis and between stations 29 to 34 on the y -axis can be easily observed from both (a) and (b). The white line at 2:45 p.m. is an instance of temporal outliers, whereas the white rectangle is a spatial-temporal outlier. Both represent missing data. Moreover, station 9 in Fig. 22.4(a) exhibits inconsistent traffic flow compared with its neighboring stations and was detected as a spatial outlier. Station 9 may be a malfunctioning sensor.

SPATIAL COLOCATION RULES

Association rule finding (Hipp, Guntzer, & Nakaeizadeh, 2000) is an important data mining technique that has helped retailers interested in finding items frequently bought together to make store arrangements, plan catalogs, and promote products together. In market-basket

data a transaction consists of a collection of item types purchased together by a customer. Association rule mining algorithms (Agrawal & Srikant, 1994) assume that a finite set of disjoint transactions are given as input to the algorithms. Algorithms such as the a priori (Agrawal & Srikant, 1994) can efficiently find the frequent item sets from all the transactions, and association rules can be found from these frequent item sets. Many spatial data sets consist of instances of a collection of Boolean spatial features (e.g., drought, needle leaf vegetation). Although Boolean spatial features can be thought of as item types, there may not be an explicit finite set of transactions due to the continuity of underlying spaces. In this section we define colocation rules, a generalization of association rules to spatial data sets.

Illustrative Application Domains

Many ecological data sets (Li, Cihlar, Moreau, Huang, & Lee, 1997; Nepstad et al., 1999) consist of raster maps of the Earth at different times. Measurement values for a number of variables (e.g., temperature, pressure, and precipitation) are collected for different locations on Earth. Maps of these variables are available for different time periods ranging from 20 to 100 years. Some variables are measured using sensors, whereas others are computed using model predictions.

A set of events, that is, Boolean spatial features, are defined on these spatial variables. Example events include drought, flood, fire, and smoke. Ecologists are interested in a variety of spatiotemporal patterns including colocation rules. Colocation patterns represent frequent cooccurrences of a subset of Boolean spatial features. Examples of interesting colocation patterns in ecology are shown in Table 22.1. Net primary production (NPP) is a key variable for understanding the global carbon cycle and the ecological dynamics of the Earth.

The spatial patterns of ecosystem data sets include the following:

1. Local colocation patterns that represent relationships among events in the same grid cell, ignoring the temporal aspects of the data. Examples from the ecosystem domain include patterns P1 and P2 of Table 22.1. These patterns can be discovered using algorithms (Agrawal & Srikant, 1994) for mining classical association rules.
2. Spatial colocation patterns that represent relationships among events happening in different and possibly nearby grid cells. Examples from the ecosystem domain include patterns P3 and P4 of Table 22.1.

Additional varieties of colocation patterns may exist. Furthermore, the temporal nature of ecosystem data gives rise to many other time related patterns. We focus on these colocation patterns in the following sections.

TABLE 22.1
Examples of Interesting Spatiotemporal Ecological Patterns

Pattern No.	Variable A	Variable B	Examples of Interesting Patterns
P1	Cropland area	Vegetation	Higher cropland area alters NPP
P2	Precipitation drought index	Vegetation	Low rainfall events lead to lower NPP
P3	Smoke aerosol index	Precipitation	Smoke aerosols alter the likelihood of rainfall in a nearby region
P4	Sea surface temperature	Land surface climate and NPP	Surface ocean heating affects regional terrestrial climate and NPP

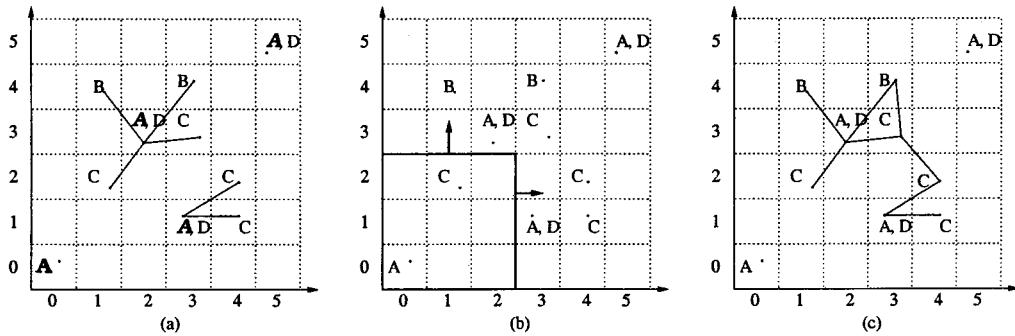


FIG. 22.5. Spatial data set to illustrate different colocation models.

Colocation Rule Approaches

Given the difficulty in creating explicit disjoint transactions from continuous spatial data, this section defines several approaches to model colocation rules. We use Fig. 22.5 as an example spatial data set to illustrate the different models. In this figure a uniform grid is imposed on the underlying spatial framework. For each grid l , its neighbors are defined to be the nine adjacent grids (including l). Spatial feature types are labeled beside their instances. Figure 22.5(a) shows a reference feature-centric model. The instances of A are connected with their neighboring instances of B and C by edges. Figure 22.5(b) shows a window-centric model. Each 3×3 window corresponds to a transaction. Figure 22.5(c) shows an event-centric model. Neighboring instances are joined by edges. We define the following basic concepts to facilitate the description of different models.

Definition 1. *A colocation is a subset of boolean spatial features or spatial events.*

Definition 2. *A colocation rule is of the form $C_1 \rightarrow C_2 (p, cp)$ where C_1 and C_2 are colocations, p is a number representing the prevalence measure, and cp is a number measuring conditional probability.*

The prevalence measure and the conditional probability measure are called interest measures and are defined differently in different models.

The reference feature-centric model is relevant to application domains focusing on a specific Boolean spatial feature, for instance, cancer. Domain scientists are interested in finding the colocations of other task-relevant features (e.g., asbestos, other substances) to the reference feature. This model enumerates neighborhoods to “materialize” a set of transactions around instances of the reference spatial feature. A specific example is provided by the spatial association rule (Koperski & Han, 1995).

For example, in Fig. 22.5(a), let the reference feature be A , the set of task-relevant features be B and C , and the set of spatial predicates include one predicate named “*close_to*.” Let us define $\text{close_to}(a, b)$ to be true if and only if b is a ’s neighbor. Then for each instance of spatial feature A , a transaction that is a subset of relevant features $\{B, C\}$ is defined. For example, for the instance of A at $(2,3)$, transaction $\{B, C\}$ is defined because the instance of B at $(1,4)$ (and at $[3,4]$) and instance of C at $(1,2)$ (and at $[3,3]$) are *close_to* $(2,3)$. The transactions defined around instances of feature A are summarized in Table 22.2.

With “materialized” transactions the support and confidence of the traditional association rule problem (Agrawal & Srikant, 1994) may be used as prevalence and conditional probability measures as summarized in Table 22.3. Because one out of two nonempty transactions contains instances of both B and C and one out of two nonempty transactions contain C in

TABLE 22.2
 Reference Feature-Centric View: Transactions
 Are Defined Around Instances of Feature A Relevant
 to B and C in Figure 22.5(a)

<i>Instance of A</i>	<i>Transaction</i>
(0,0)	\emptyset
(2,3)	$\{B, C\}$
(3,1)	$\{C\}$
(5,5)	\emptyset

Table 22.2, an association rule example is: $is_type(i, A) \wedge \exists j is_type(j, B) \wedge close_to(j, i) \rightarrow \exists k is_type(k, C) \wedge close_to(k, i)$ with $\frac{1}{1} * 100\% = 100\%$ probability.

The window-centric model is relevant to applications like mining, surveying, and geology, which focus on land parcels. A goal is to predict sets of spatial features likely to be discovered in a land parcel given that some other features have been found there. The window-centric model enumerates all possible windows as transactions. In a space discretized by a uniform grid, windows of size $k \times k$ can be enumerated and materialized, ignoring the boundary effect. Each transaction contains a subset of spatial features of which at least one instance occurs in the corresponding window. The support and confidence of the traditional association rule problem may again be used as prevalence and conditional probability measures as summarized in Table 22.3. There are 16 3×3 windows corresponding to 16 transactions in Figure 22.5(b). All of them contain A , and 15 of them contain both A and B . An example of an association rule of this model is: *an instance of type A in a window \rightarrow an instance of type B in this window* with $\frac{15}{16} = 93.75\%$ probability. A special case of the window-centric model relates to the case when windows are spatially disjoint and form a partition of space. This case is relevant when analyzing spatial data sets related to the units of political or administrative boundaries (e.g., country, state, zip code). In some sense this is a local model because we treat each arbitrary partition as a transaction to derive colocation patterns without considering any patterns cross partition boundaries. The window-centric model “materializes” transactions in a different way from the reference feature-centric model.

The event-centric model is relevant to applications such as ecology in which there are many types of Boolean spatial features. Ecologists are interested in finding subsets of spatial

TABLE 22.3
 Interest Measures for Different Models

<i>Model</i>	<i>Items</i>	<i>Transactions Defined By</i>	<i>Interest Measures for $C_1 \rightarrow C_2$</i>	
			<i>Prevalence</i>	<i>Conditional Probability</i>
Local	Boolean feature types	Partitions of space	Fraction of partitions with $C_1 \cup C_2$	$Pr(C_2 \text{ in a partition given } C_1 \text{ in the partition})$
Reference feature-centric	Predicates on reference and relevant features	Instances of reference feature C_1 and C_2 involved with	Fraction of instances of reference feature with $C_1 \cup C_2$	$Pr(C_2 \text{ is true for an instance of reference features given } C_1 \text{ is true for that instance of reference feature})$
Window-centric	Boolean feature types	Possibly infinite set of distinct overlapping windows	Fraction of windows with $C_1 \cup C_2$	$Pr(C_2 \text{ in a window given } C_1 \text{ in that window})$
Event centric	Boolean feature types	Neighborhoods of instances of feature types	Participation index of $C_1 \cup C_2$	$Pr(C_2 \text{ in a neighborhood of } C_1)$

features likely to occur in a neighborhood around instances of given subsets of event types. For example, let us determine the probability of finding at least one instance of feature type B in the neighborhood of an instance of feature type A in Figure 22.5(c). There are four instances of type A and only one of them has some instance(s) of type B in its nine-neighbor adjacent neighborhoods. The conditional probability for the colocation rule is: *spatial feature A at location l → spatial feature type B in 9-neighbor neighborhood is 25%*.

Neighborhood is an important concept in the event-centric model. Given a reflexive and symmetric neighbor relation R , we can define neighborhoods of a location l that satisfy the definition of neighborhood in topology [Worboys, 1995] as seen in the following definitions:

Definition 3. *A neighborhood of l is a set of locations $L = \{l_1, \dots, l_k\}$ such that l_i is a neighbor of l , that is, $(l, l_i) \in R (\forall i \in 1, \dots, k)$.*

We generalize the neighborhood definition to a collection of locations.

Definition 4. *For a subset of locations L' if L' is a neighborhood of every location in $L = \{l_1, \dots, l_k\}$ then L' is a neighborhood of L .*

In other words, if every l_1 in L' is a neighbor of every l_2 in L , then L' is a neighborhood of L .

The definition of neighbor relation R is an input and is based on the semantics of application domains. The neighbor relation R may be defined using topological relationships (e.g., connected, adjacent), metric relationships (e.g., Euclidean distance) or a combination (e.g., shortest path distance in a graph such as a road map). In general, there are infinite neighborhoods over continuous space, and it may not be possible to materialize all of them. But we are interested in only the locations where instances of spatial feature types (events) occur. Even confined to these locations, enumerating all the neighborhoods incurs substantial computational cost because support based pruning cannot be carried out before the enumeration of all the neighborhoods is completed and the total number of neighborhoods is obtained. Furthermore, this support based prevalence measure definition may not be meaningful because the value of the prevalence may be extremely small due to the fact that many neighborhoods are contained in bigger neighborhoods and counted multiple times. Thus, the participation index is proposed to be a prevalence measure as defined below.

Definition 5. *For a colocation $C = \{f_1, \dots, f_k\}$ and a set of locations $I = \{i_1, \dots, i_k\}$ where i_j is an instance of feature $f_j (\forall j \in 1, \dots, k)$ if I is a neighborhood of I itself then I is an instance of C .*

In other words, if elements of I are neighbors to each other, then I is an instance of C . For example, $\{(3,1), (4,1)\}$ is an instance of colocation $\{A, C\}$ in Fig. 22.5(c) using a nine-neighbor adjacent neighbor definition.

Definition 6. *The participation ratio $pr(C, f_i)$ for feature type f_i of a colocation $C = \{f_1, f_2, \dots, f_k\}$ is the fraction of instances of f_i that participate in the colocation C . It can be formally defined as $\frac{|\text{distinct}(\pi_{f_i}(\text{all instances of colocation } C))|}{|\text{instances of } f_i|}$ where π is a projection operation.*

For example, in Fig. 22.5(c), instances of colocation $\{A, B\}$ are $\{(2,3), (1,4)\}$ and $\{(2,3), (3,4)\}$. Only one instance (2,3) of spatial feature A out of four participates in colocation $\{A, B\}$. So $pr(\{A, B\}, A) = \frac{1}{4} = .25$.

Definition 7. *The participation index of a colocation $C = \{f_1, f_2, \dots, f_k\}$ is $\prod_{i=1}^k pr(C, f_i)$.*

In Fig. 22.5(c), participation ratio $\text{pr}(\{A, B\}, A)$ of feature A in colocation $\{A, B\}$ is .25 as calculated above. Similarly $\text{pr}(\{A, B\}, B)$ is 1.0. The participation index for colocation $\{A, B\}$ is $.25 \times 1.0 = .25$.

The conditional probability of a colocation rule $C_1 \rightarrow C_2$ in the event-centric model is the probability of finding C_2 in a neighborhood of C_1 or it can be formally defined as follows.

Definition 8. *The conditional probability of a colocation rule $C_1 \rightarrow C_2$ is*

$$\frac{\text{distinct}(\pi_{C_1}(\text{all instances of colocation } C_1 \cup C_2)))}{|\text{instances of } C_1|}$$
 where π is a projection operation.

For details of algorithms that mine colocation rules in the event centric model, refer to Shekhar and Huang (2001).

Solution Procedures

Colocation mining is a complex task. It consists of two tasks, schema level pruning and instance level pruning. At schema level pruning the apriori algorithm (Agrawal & Srikant, 1994) can be used. However, instance level pruning involves neighborhood (i.e., colocation row instance) enumeration, which is a computationally intense task. Shekhar and Huang (2001) developed pure geometric, pure combinatorial, hybrid, and multiresolution algorithms for instance level pruning. Experimental analysis shows that the pure geometric algorithm performs much better than the pure combinatorial approach. The hybrid algorithm, which is a combination of geometric and combinatorial methods, performed better than both of these approaches. On the other hand, the multiresolution algorithm outperforms all these methods when the data is “clumped.” The results also show that the colocation miner algorithm is complete and correct.

LOCATION PREDICTION

The prediction of events occurring at particular geographic locations is very important in several application domains. Crime analysis, cellular networks, and natural disasters such as fires, floods, droughts, vegetation diseases, and earthquakes are all examples of problems that require location prediction. In this section we provide two spatial data mining techniques, namely the SAR and MRF and analyze their performance in an example case, the prediction of the location of bird nests in the Darr and Stubble wetlands in Ohio.

An Illustrative Application Domain

We now introduce an example to illustrate the different concepts in spatial data mining. We are given data about two wetlands, named Darr and Stubble, on the shores of Lake Erie in Ohio to predict the spatial distribution of a marsh-breeding bird, the red-winged blackbird (*Agelaius phoeniceus*). The data was collected from April to June in two successive years, 1995 and 1996.

A uniform grid was imposed on the two wetlands, and different types of measurements were recorded at each cell or pixel. In total, values of seven attributes were recorded at each cell. Domain knowledge is crucial in deciding which attributes are important and which are not. For example, vegetation durability was chosen over vegetation species because specialized knowledge about the bird-nesting habits of the red-winged blackbird suggested that the choice of nest location is more dependent on plant structure, plant resistance to wind, and wave action than on the plant species.

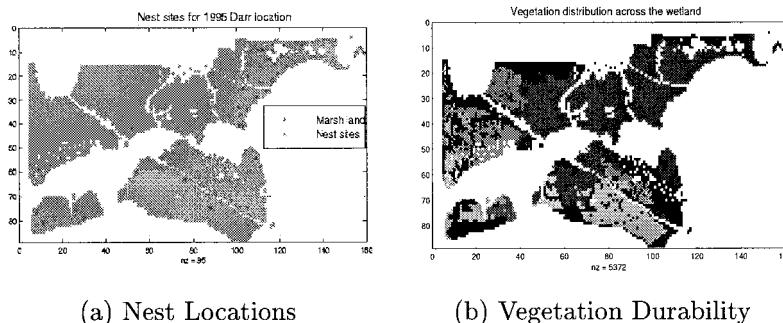


FIG. 22.6. (a) Learning data set: the geometry of the wetland and the locations of the nests; (b) the spatial distribution of vegetation durability over the marshland.

Our goal is to build a model for predicting the location of bird nests in the wetlands. Typically the model is built using a portion of the data, called the *learning* or *training* data, and then tested on the remainder of the data, called the *testing* data. In the learning data all the attributes are used to build the model, and in the testing data one value is *hidden*, in our case the location of the nests.

We focus on three independent attributes, namely vegetation durability, distance to open water, and water depth. The spatial distribution of vegetation durability and the actual nest locations for the Darr wetland in 1995 are shown in Fig. 22.6. These maps illustrate the following two important properties inherent in spatial data.

1. The values of vegetation durability and the actual nest locations that are referenced by spatial location tend to vary gradually over space (values of distance to open water and water depth are similar). Although this may seem obvious, classical data mining techniques either explicitly or implicitly assume that the data is *independently* generated. For example, the maps in Fig. 22.7 show the spatial distribution of attributes if they were independently generated. Classical data mining techniques such as logistic regression (Ozesmi & Mitch, 1997) and neural networks (Ozesmi & Ozesmi, 1999) were applied to build spatial habitat models. These models predict the bird nesting locations by considering the spatial interactions

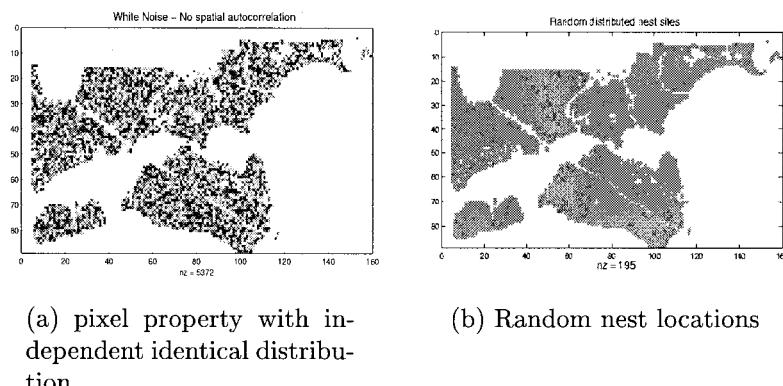


FIG. 22.7. Spatial distribution satisfying random distribution assumptions of classical regression.

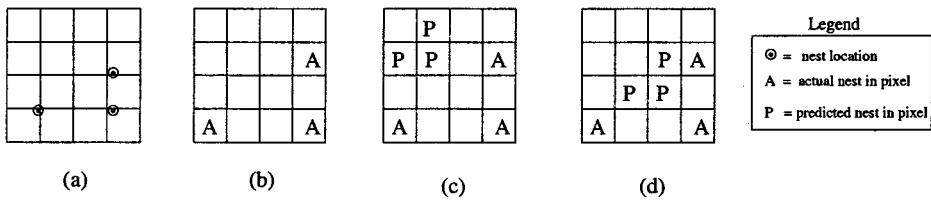


FIG. 22.8. (a) The actual locations of nests; (b) pixels with actual nests; (c) location predicted by a model; (d) location predicted by another model. Prediction (d) is spatially more accurate than (c).

among the dependent variables. Logistic regression was used because the dependent variable is binary (nest/no-nest) and the logistic function “squashes” the real line onto the unit interval. The values in the unit interval can then be interpreted as probabilities. The study concluded that with the use of logistic regression, the nests could be classified at a rate 24% better than random (Ozesmi & Ozesmi, 1999).

2. The spatial distributions of attributes sometimes have distinct local trends that contradict the global trends. This is seen most vividly in Fig. 22.6(b), in which the spatial distribution of vegetation durability is jagged in the western section of the wetland as compared with the overall impression of uniformity across the wetland. This property is called spatial heterogeneity.

The fact that classical data mining techniques ignore spatial autocorrelation and spatial heterogeneity in the model-building process is one reason these techniques do a poor job. A second more subtle but equally important reason is related to the choice of the objective function to measure classification accuracy. For a two-class problem the standard way to measure classification accuracy is to calculate the percentage of correctly classified objects. However, this measure may not be the most suitable in a spatial context. *Spatial accuracy*—how far the predictions are from the actuals—is just as important in this application domain due to the effects of the discretizations of a continuous wetland into discrete pixels, as shown in Fig. 22.8. Figure 22.8(a) shows the actual locations of nests and 22.8(b) shows the pixels with actual nests. Note the loss of information during the discretization of continuous space into pixels. Many nest locations barely fall within the pixels labeled “A” and are quite close to other blank pixels, which represent “no-nest.” Now consider two predictions shown in Fig. 22.8(c) and 22.8(d). Domain scientists prefer prediction 22.8(d) over 22.8(c), because the predicted nest locations are closer on average to some actual nest locations. The classification accuracy measure cannot distinguish between 22.8(c) and 22.8(d), and a measure of spatial accuracy is needed to capture this preference.

Problem Formulation

The location prediction problem is a generalization of the nest location prediction problem. It captures the essential properties of similar problems from other domains including crime prevention and environmental management. The problem is formally defined as follows.

Given:

- A spatial framework S consisting of sites $\{s_1, \dots, s_n\}$ for an underlying geographic space G .
- A collection X of explanatory functions $f_{X_k} : S \rightarrow R^k, k = 1, \dots, K$. R^k is the range of possible values for the explanatory functions. Let $X = [1, X]$, which also includes a constant vector along with explanatory functions.

- A dependent class variable $f_C : S \rightarrow C = \{c_1, \dots, c_M\}$
- A value for parameter α , relative importance of spatial accuracy

Find: Classification model: $\hat{f}_C : R^1 \times \dots R^k \rightarrow C$.

Objective: Maximize similarity $(map_{s_i \in S}(\hat{f}_C(f_{x_1}, \dots, f_{x_k})), map(f_C)) = (1 - \alpha) \text{classification_accuracy}(\hat{f}_C, f_C) + (\alpha) \text{spatial_accuracy}(\hat{f}_C, f_C)$.

Constraints:

1. Geographic space S is a multidimensional Euclidean space.¹
2. The values of the explanatory functions, f_{x_1}, \dots, f_{x_k} and the dependent class variable, f_C , may not be independent with respect to the corresponding values of nearby spatial sites (i.e., spatial autocorrelation exists).
3. The domain R^k of the explanatory functions is the one-dimensional domain of real numbers.
4. The domain of dependent variable, $C = \{0, 1\}$.

The above formulation highlights two important aspects of location prediction. It explicitly indicates that (a) the data samples may exhibit spatial autocorrelation and, (b) an objective function (i.e., a map similarity measure) is a combination of classification accuracy and spatial accuracy. The *similarity* between the dependent variable f_C and the predicted variable \hat{f}_C is a combination of the “traditional classification” accuracy and representation-dependent “spatial classification” accuracy. The regularization term α controls the degree of importance of *spatial accuracy* and is typically domain dependent. As $\alpha \rightarrow 0$, the map similarity measure approaches the traditional classification accuracy measure. Intuitively, α captures the spatial autocorrelation present in spatial data.

Modeling Spatial Dependencies Using the SAR and MRF Models

Several previous studies (Jhung & Swain, 1996; Solberg, Taxt, & Jain, 1996) have shown that the modeling of spatial dependency (often called context) during the classification process improves overall classification accuracy. Spatial context can be defined by the relationships between spatially adjacent pixels in a small neighborhood. The spatial relationship among locations in a spatial framework often is modeled via a contiguity matrix. A simple contiguity matrix may represent a neighborhood relationship defined using adjacency, Euclidean distance, and so forth. Example definitions of neighborhood using adjacency include a four-neighborhood and an eight-neighborhood. Given a gridded spatial framework, a four-neighborhood assumes that a pair of locations influence each other if they share an edge. An eight-neighborhood assumes that a pair of locations influence each other if they share either an edge or a vertex.

Figure 22.9(a) shows a gridded spatial framework with four locations, A, B, C, and D. A binary matrix representation of a four-neighborhood relationship is shown in Fig. 22.9(b). The row-normalized representation of this matrix is called a contiguity matrix, as shown in Fig. 22.9(c). Other contiguity matrices can be designed to model neighborhood relationship based on distance. The essential idea is to specify the pairs of locations that influence each other along with the relative intensity of interaction. More general models of spatial relationships using cliques and hypergraphs are available in the literature (Warrender & Augusteijn, 1999).

¹The entire surface of the Earth cannot be modeled as a Euclidean space but locally the approximation holds true.

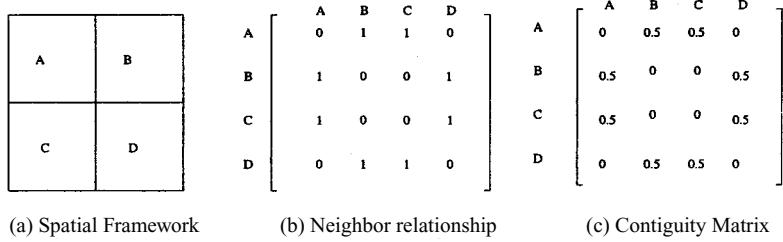


FIG. 22.9. A spatial framework and its four-neighborhood contiguity matrix.

Logistic SAR

Logistic SAR decomposes a classifier \hat{f}_c into two parts, namely spatial autoregression and logistic transformation. We first show how spatial dependencies are modeled using the framework of logistic regression analysis. In the spatial autoregression model the spatial dependencies of the error term, or the dependent variable, are directly modeled in the regression equation (Anselin, 1988). If the dependent values y_i are related to each other, then the regression equation can be modified as

$$y = \rho Wy + X\beta + \epsilon. \quad (1)$$

Here W is the neighborhood relationship contiguity matrix and ρ is a parameter that reflects the strength of the spatial dependencies between the elements of the dependent variable. After the correction term ρWy is introduced, the components of the residual error vector ϵ are then assumed to be generated from independent and identical standard normal distributions. As in the case of classical regression, the SAR equation has to be transformed via the logistic function for binary dependent variables.

We refer to this equation as the SAR model. Notice that when $\rho = 0$, this equation collapses to the classical regression model. The benefits of modeling spatial autocorrelation are many: The residual error will have much lower spatial autocorrelation (i.e., systematic variation). With the proper choice of W , the residual error should, at least theoretically, have no systematic variation. If the spatial autocorrelation coefficient is statistically significant, then SAR will quantify the presence of spatial autocorrelation. It will indicate the extent to which variations in the dependent variable (y) are explained by the average of neighboring observation values. Finally, the model will have a better fit (i.e., a higher R-squared statistic). We compare SAR with linear regression for predicting nest location in a later section.

A mixed model extends the general linear model by allowing a more flexible specification of the covariance matrix of ϵ . The SAR model can be extended to a mixed model that allows for explanatory variables from neighboring observations (LeSage & Pace, 2001). The new model (MSAR) is given by

$$y = \rho Wy + X\beta + WX\gamma + \epsilon. \quad (2)$$

The marginal impact of the explanatory variables from the neighboring observations on the dependent variable y can be encoded as a $k * 1$ parameter vector γ .

Solution Procedures

The estimates of ρ and β can be derived using maximum likelihood theory or Bayesian statistics. We have carried out preliminary experiments using the spatial econometrics Matlab

package,² which implements a Bayesian approach using sampling-based Markov chain Monte Carlo methods (LeSage, 1997). Without any optimization, likelihood-based estimation would require $O(n^3)$ operations. Recently, Pace and Barry (1997a, 1997b) and LeSage and Pace (2001) proposed several efficient techniques to solve SAR. The techniques studied include divide and conquer, and sparse matrix algorithms. Improved performance is obtained by using LU decompositions to compute the log-determinant over a grid of values for the parameter ρ by restricting it to $[0, 1]$.

MRF Based Bayesian Classifiers

MRF based Bayesian classifiers estimate classification model \hat{f}_C using MRF and Bayes' rule. A set of random variables the interdependency relationship of which is represented by an undirected graph (i.e., a symmetric neighborhood matrix) is called an MRF (Li, 1995). The Markov property specifies that a variable depends only on its neighbors and is independent of all other variables. The location prediction problem can be modeled in this framework by assuming that the class label, $l_i = f_C(s_i)$, of different locations, s_i , constitutes an MRF. In other words, random variable l_i is independent of l_j if $W(s_i, s_j) = 0$.

The Bayesian rule can be used to predict l_i from feature value vector X and neighborhood class label vector L_i as follows:

$$Pr(l_i | X, L_i) = \frac{Pr(X | l_i, L_i)Pr(l_i | L_i)}{Pr(X)} \quad (3)$$

The solution procedure can estimate $Pr(l_i | L_i)$ from the training data, where L_i denotes a set of labels in the neighborhood of s_i excluding the label at s_i , by examining the ratios of the frequencies of class labels to the total number of locations in the spatial framework. $Pr(X | l_i, L_i)$ can be estimated using kernel functions from the observed values in the training data set. For reliable estimates even larger training data sets are needed relative to those needed for the Bayesian classifiers without spatial context, because we are estimating a more complex distribution. An assumption on $Pr(X | l_i, L_i)$ may be useful if the training data set available is not large enough. A common assumption is the uniformity of influence from all neighbors of a location. For computational efficiency it can be assumed that only local explanatory data $X(s_i)$ and neighborhood label L_i are relevant in predicting class label $l_i = f_C(s_i)$. It is common to assume that all interaction between neighbors is captured via the interaction in the class label variable. Many domains also use specific parametric probability distribution forms, leading to simpler solution procedures. In addition, it is frequently easier to work with a Gibbs distribution specialized by the locally defined MRF through the Hammersley-Clifford theorem (Besag, 1974).

Solution Procedures

Solution procedures for the MRF Bayesian classifier include stochastic relaxation (Geman & Geman, 1984), iterated conditional modes (Besag, 1986), dynamic programming (Derin & Elliott, 1987), highest confidence first (Chou, Cooper, Swain, Brown, & Wixson, 1993) and graph cut (Boykov, Veksler, & Zabih, 1999). We follow the approach suggested in Boykov et al. (1999), where it is shown that the maximum a posteriori estimate of a particular configuration of an MRF can be obtained by solving a suitable min-cut multiway graph partitioning problem. We illustrate the underlying concept with some examples.

²We would like to thank James Lesage (<http://www.spatial-econometrics.com/>) for making the Matlab toolbox available on the Web.

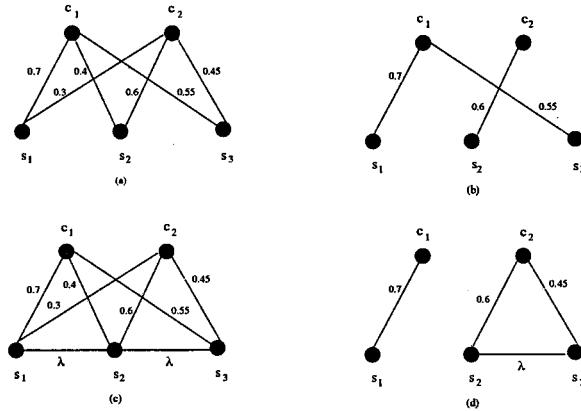


FIG. 22.10. MRF solution with graph-cut method.

Example 1. A classification problem with no spatial constraints. Even though MRFs are inherently multidimensional, we will use a simple one-dimensional example to illustrate the main points. Consider the graph $G = (V, E)$ shown in Fig. 22.10(a). The node set V itself consists of two disjoint sets, S and C . The members of S are $\{s_1, s_2, s_3\}$, and the members of C are $\{c_1, c_2\}$. Typically, the $X(s_i)$'s are the feature values at site s_i and the c_i 's are the labels such as nest or no-nest. There is an edge between each pair of members of the set S and each pair of members of set C . Here we interpret the edge weights as probabilities. For example, $p_1 = \Pr(X(s_1) = c_1) = 0.7$ and $p_2 = \Pr(X(s_1) = c_2) = 0.3$; $p_1 + p_2 = 1$.

Our goal is to provide a label for each location s_i in S using explanatory feature $X(s_i)$. This is done by partitioning the graph into two disjoint sets (not S and C) by removing certain edges such that:

1. There is a many-to-one mapping from the set S to C . Every element of S must be mapped to one and only one element of C .
2. Multiple elements of C cannot belong to a single partition. Thus, there are no edges between elements of C , and therefore, the number of partitions is equal to the cardinality of C .
3. The sum of the weights of the edges removed (the cut-set) is the minimum of all possible cut-sets.

In this example the cut-set is easily determined. For example, of the two edges connecting each element of S and an element of C , remove the edge with the smaller weight. Figure 22.10(b) shows the graph with the cut-set removed. Thus, we have just shown that when the weights of the edges are interpreted as probabilities, the min-cut graph partition induces a maximum a posterior estimate for the pixel labels. We prefer to say that the min-cut induces a Bayesian classification on the underlying pixel set. This is because we will use Bayes' theorem to calculate the edge weights of the graphs.

Example 2. Adding spatial constraints. In the previous example we did not use any information about the spatial proximity of the pixels relative to each other. We do that now by introducing additional edges in the graph structure.

Consider the graph shown in Fig. 22.10(c) in which we have added two extra edges (s_1, s_2) and (s_2, s_3) with a weight λ . In this example we have chosen $\lambda = 0.2$.

Now if we want to retain the same partitions of the graph as in example 1, then the cut-set has two extra edges, namely (s_1, s_2) and (s_2, s_3) . Thus, the sum of the weights of the edges in the cut-set, W_{C1} , is

$$W_{C1} = 0.3 + 0.4 + 0.45 + 2\lambda$$

But now, depending on λ , the cut-set weight may not be minimal. For example, if $\lambda = 0.2$ then the weight of the cut-set, W_{C2} , consisting of the edges $\{(s_1, c_2), (s_2, c_1), (s_3, c_1), (s_1, s_2)\}$ is

$$W_{C2} = 0.3 + 0.4 + 0.55 + 0.2$$

Thus, $W_{C2} < W_{C1}$. In other words, if two neighboring pixels are assigned to different labels, then the edge between the two neighbors is added to the cut-set. This shows that there is a penalty associated with two neighboring nodes being assigned to different labels every time. Thus, we can model spatial autocorrelation by adding edges between the pixel nodes of the graph. We can also model spatial heterogeneity by assigning different *weights*, the λ 's, to the pixel edges. Figure 22.10(d) shows that a min-cut graph partitioning does not necessarily induce a labeling where the labeling with maximum probabilities is retained. If two neighboring pixels are assigned different labels, then the edge connecting the pixels is added to the cut-set.

Here we briefly provide theoretical and experimental comparisons; more details can be found in Shekhar, Schrater, Vatsavai, Wu, and Chawla (2002). Although MRF and SAR classification have different formulations, they share a common goal, estimating the posterior probability distribution: $p(l_i \mid X)$. However, the posterior for the two models is computed differently with different assumptions. For MRF the posterior is computed using Bayes' rule. On the other hand, in logistic regression the posterior distribution is directly fit to the data. One important difference between logistic regression and MRF is that logistic regression assumes no dependence on neighboring classes. Logistic regression and logistic SAR models belong to a more general exponential family. The exponential family is given by $Pr(u \mid v) = e^{A(\theta_v) + B(u, \pi) + \theta_v^T u}$, where u, v are location and label, respectively. This exponential family includes many of the common distributions such as Gaussian, binomial, Bernoulli, and Poisson as special cases. Experiments were carried out on the Darr and Stubble wetlands to compare the classical regression, SAR, and the MRF-based Bayesian classifiers. The results showed that the MRF models yield better spatial and classification accuracies over SAR in the prediction of the locations of bird nests. We also observed that SAR predictions are extremely localized, missing actual nests over a large part of the marsh lands.

CLUSTERING

Clustering is a process for discovering “groups,” or clusters, in a large database. Unlike classification, clustering involves no a priori information either on the number of clusters or what the cluster labels are. Thus, there is no concept of training or test data in clustering. This is the reason that clustering is also referred as *unsupervised learning*.

The clusters are formed on the basis of a “similarity” criterion, which is used to determine the relationship between each pair of tuples in the database. Tuples that are similar are usually grouped together, and then the group is labeled. For example, the pixels of satellite images are often clustered on the basis of the spectral signature. This way a remotely sensed image can be quickly segmented with minimal human intervention. Of course, a domain expert does have to examine, verify, and possibly refine the clusters. A famous example of population

segmentation occurred in the 1996 U.S. presidential election when political pundits identified “soccer moms” as the swing electorate who were then assiduously courted by major political parties. Clustering is also used to determine the “hot spots” in crime analysis and disease tracking.

Clustering is a very well-known technique in statistics and the data mining role is to scale a clustering algorithm to deal with the large data sets that are now becoming the norm rather than the exception. The size of the database is a function of the number of records in the table and also the number of attributes (the dimensionality) of each record. Besides the volume, the type of data, whether it is numeric, binary, categorical, or ordinal, is an important determinant in the choice of the algorithm employed.

It is convenient to frame the clustering problem in a multidimensional attribute space. Given n data objects described in terms of m variables, each object can be represented as a point in an m -dimensional space. Clustering then reduces to determining high-density groups of points from a set of nonuniformly distributed points. The search for potential clusters within the multidimensional space is then driven by a suitably chosen similarity criterion.

For example, the counties in the United States can be clustered on the basis of four attributes: rate-of-unemployment, population, per-capita-income, and life-expectancy. Counties with similar values for these attributes will be grouped or clustered together.

When dealing with attribute data that is referenced in physical space, the clustering problem can have two interpretations. Consider the plot shown in Fig. 22.11, which shows the variation of an attribute value (e.g., population density) as a function of location shown on the x -axis. Now what are the clusters, and how do we interpret them? For example, if our goal is to identify central cities and their zones of influence from a set of cities that dominate other cities as measured by the variance of an attribute value across the landscape, then we are looking for spatial clusters marked S1 and S2 in Fig. 22.11. On the other hand, if our goal is to identify

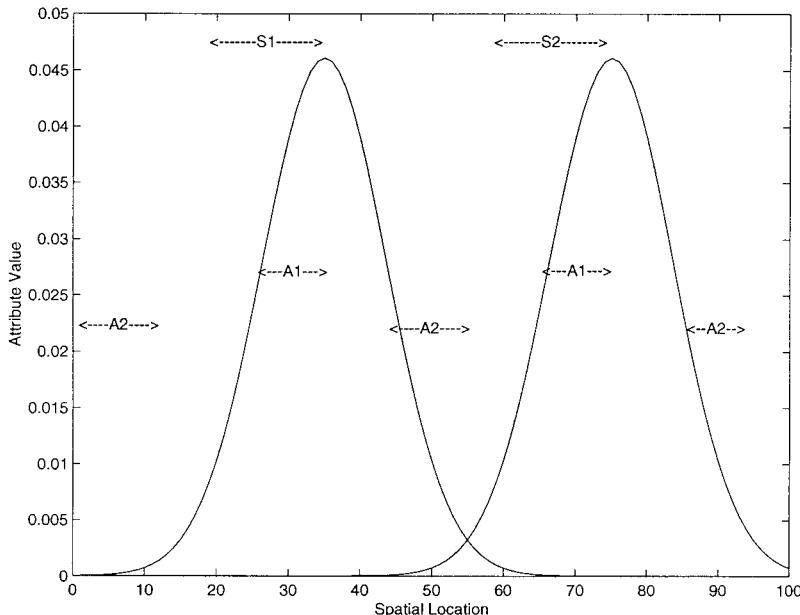


FIG. 22.11. Two interpretations of spatial clustering. If the goal is to identify locations that dominate the surroundings (in terms of influence), then the clusters are S1 and S2. If the goal is to identify areas of homogeneous values, the clusters are A1 and A2.

pockets in the landscape where an attribute (or attributes) is homogeneously expressed, then we are looking for clusters marked A1 and A2. Although the second interpretation is essentially nonspatial, the spatial aspects exist because of the spatial autocorrelation that may exist in the attribute data. The clusters identified should be spatially homogeneous and not “speckled.” These two interpretations of the clustering problem are formally defined as follows:

Definition 1. Given a set $S = \{s_1, \dots, s_n\}$ of spatial objects (e.g., points) and a real-valued, nonspatial attributes f evaluated on S (i.e., $f : S \rightarrow R$), find two disjoint subsets of S , C , and $NC = S - C$, where $C = \{s_1, \dots, s_k\}$, $NC = \{nc_1, \dots, nc_l\}$, and $k < n$. Objective $\min_{C \subset S} \sum_{j=1}^l |f(nc_j) - \sum_{i=1}^k \frac{f(c_i)}{dist(nc_j, c_i)}|^2$, where $dist(a, b)$ is the Euclidean or some other distance measure.

The constraints are as follows:

1. The data set conforms to the theory of central places, which postulates that the influence of a central city decays as the square of the distance.
2. There is at most one nonspatial attribute.

Definition 2. Given (1) A set $S = \{s_1, \dots, s_n\}$ of spatial objects (e.g., points) and a set of real-valued, nonspatial attributes f_i $i = 1, \dots, I$ defined on S , (i.e., for each i , $f_k : S \rightarrow R$); and (2) neighborhood structure E on S , find K subsets $C_k \subset S$, $k = 1, \dots, K$ such that $\min_{C_k \subset S} \sum_{C_k, s_i \in C_k, s_j \in C_k} dist(F(s_i), F(s_j)) + \sum_{i,j} nbddist(C_i, C_j)$ where (a) F is the cross-product of the f_i 's, $i = 1, \dots, n$; (b) $dist(a, b)$ is the Euclidean or some other distance measure, and (c) $nbddist(C, D)$ is the number of points in C and D that belong to E . Constraints $|C_k| > 1$ for all $k = 1, \dots, K$.

Categories of Clustering Algorithms

Cluster analysis is one of most often performed data analysis techniques in many fields. Because this has resulted in a multitude of clustering algorithms, it is useful to categorize them into groups. Based on the technique adopted to define clusters, the clustering algorithms can be divided into four broad categories:

1. *Hierarchical* clustering methods, which start with all patterns as a single cluster and successively perform splitting or merging until a stopping criterion is met. This results in a tree of clusters called *dendograms*. The dendrogram can be cut at different levels to yield desired clusters. Hierarchical algorithms can further be divided into *agglomerative* and *divisive* methods. The hierarchical clustering algorithms include balanced iterative reducing and clustering using hierarchies (BIRCH), clustering using representatives (CURE), and robust clustering using links (ROCK).

2. *Partitional* clustering algorithms, which start with each pattern as a single cluster and iteratively reallocate data points to each cluster until a stopping criterion is met. These methods tend to find clusters of spherical shape. *K-means* and *K-medoids* are commonly used partitional algorithms. Squared error is the most frequently used criterion function in partitional clustering. The recent algorithms in this category include partitioning around medoids, clustering large applications, clustering large applications based on randomized search, and EM.

3. *Density based* clustering algorithms, which try to find clusters based on the density of data points in a region. These algorithms treat clusters as dense regions of objects in the data space. The density based clustering algorithms include density based spatial clustering of applications with noise and density based clustering.

4. *Grid based* clustering algorithms, which first quantize the clustering space into a finite number of cells and then perform the required operations on the quantized space. Cells that contain more than a certain number of points are treated as dense. The dense cells are connected to form the clusters. Grid based clustering algorithms are primarily developed for analyzing large spatial data sets. The grid based clustering algorithms include the statistical information grid based method (STING), STING+, WaveCluster, BANG-clustering, and clustering-in-quest (CLIQUE).

Sometimes the distinction among these categories diminishes, and some algorithms can even be classified into more than one group. For example, CLIQUE can be considered as both a density based and grid based clustering method. More details on various clustering methods can be found in a recent survey paper (Halkidi, Batistakis, & Vazirgiannis, 2001).

We now describe two well-known approaches to clustering, the K -medoid algorithm and mixture analysis using the expectation-maximization (EM) algorithm. We will also briefly discuss how the EM algorithm can be modified to account for the special nature of spatial data.

K -Medoid: An Algorithm for Clustering

We restrict our attention to points in the two-dimensional space R^2 , although the technique can be readily generalized to a higher dimensional space. Given a set P of n data points, $P = \{p_1, p_2, \dots, p_n\}$ in R^2 , the goal of K -medoid clustering is to partition the data points into k clusters such that the following objective function is minimized:

$$J(M) = J(m_1, \dots, m_k) = \sum_{i=1}^k \sum_{p \in C_i} d(p, m_i)$$

In $J(C)$, m_i is the representative point of a cluster C_i . If m_i is restricted to be a member of P , then it is called a *medoid*. On the other hand, if m_i is the average of the cluster points and not necessarily a member of P , then it is called the *mean*. Thus, the K -mean and the K -medoid approaches are intimately related. Even though the K -mean algorithm is better known, we focus on the K -medoid approach because the medoid, like the median, is less sensitive to outliers.

The K -medoids characterize the K clusters, and each point in P belongs to its nearest medoid. Because we have restricted the ambient space to be R^2 , the distance function d is the usual Euclidean distance.

$$d(p, m_i) = ((p(x) - m_i(x))^2 + (p(y) - m_i(y))^2)^{\frac{1}{2}}.$$

Thus, the K -medoid approach transforms the clustering problem into a search problem. The search space X is the set of all k -subsets M of P (i.e., $|M| = k$), and the objective function is $J(M)$. X can be modeled as a graph, where the nodes of the graph are the elements of X . Two nodes M_1 and M_2 are *adjacent* if $|M_1 \cap M_2| = k - 1$ (i.e., they differ by one and only one data point).

The K -medoid algorithm consists of the following steps:

1. Choose an arbitrary node M_o in X .
2. Iteratively move from current node M_t to an adjacent node M_{t+1} such that $J(M_{t+1}) < J(M_t)$. The move from current node to adjacent node consists of replacing a current medoid m with a data point $p \in P$. Thus, $M_{t+1} = M_t \cup \{p\} - \{m\}$.
3. Stop when $J(M_{t+1}) \geq J(M_t)$ for all adjacent nodes.

TABLE 22.4
Four Options for Local Search in Clustering

Local Search	Strategy to Move from M_t to $M_{t+1} = M_t \cup \{p\} - \{m\}$	Guarantee Local Optima
Global hill climbing (HC)	Move to the best neighbor	Yes
Randomized HC	Move to best of sampled neighbors	No
Local HC	Move to a new neighbor as soon as it is found	Yes
Distance-restricted HC	Move to best neighbor within a specified distance	No

Step 2 is the heart of the algorithm. There are many options available to move from a node to its adjacent node. Table 22.4 lists some of the options. The table includes the name of each option as it is referred to in the literature, the strategy for moving, and whether the option will guarantee a local optima. All the options are examples of local search because only the adjacent nodes are explored.

Clustering, Mixture Analysis, and the EM Algorithm

One drawback of the K -medoid (or K -mean) approach is that it produces “hard” clusters; that is, each point is uniquely assigned to one and only one cluster. This can be a serious limitation because it is not known a priori what the actual clusters are. In the statistics literature the clustering problem is often recast in terms of *mixture models*. In a mixture model the data is assumed to be generated by a sequence of probability distributions in which each distribution generates one cluster. The goal then is to identify the parameters of the probability distributions and their weights in the overall mixture distribution. In a mixture model each instance of the database belongs to all the clusters but with a different grade of membership, which is quantified by the weights of the individual distributions in the mixture model. Thus, the mixture model framework is more flexible than the K -medoid approach. Typically, each probability distribution is represented as a normal distribution, and the challenge is to determine the mean, variance, and weight of each distribution. The assumption of normality is not as restrictive as it might appear because a statistics theorem guarantees that any probability distribution can be expressed as a finite sum of normal distributions.

A Finite Mixture Example

Consider the gray-scale 4×4 image shown in Fig. 22.12. Assume we want to partition the set of pixels into two clusters, A and B, where each cluster is modeled as a Gaussian distribution. The finite mixture problem is to calculate the parameters $\mu_A, \mu_B, \sigma_A, \sigma_B, p_A, p_B$.

For the moment, assume the cluster membership of each pixel is given as shown in Fig. 22.12b. Then all the parameters can be easily calculated. For example,

$$\mu_A = \frac{12 + 10 + 2 + 18 + 11 + 5 + 7 + 9 + 13}{9} = 9.7$$

$$\sigma_A = \frac{(12 - \mu_1)^2 + (10 - \mu_1)^2 + \dots + (13 - \mu_1)^2}{8} = 4.7$$

$$p_A = \frac{9}{16}$$

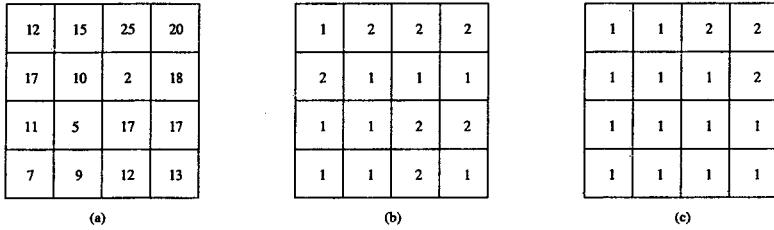


FIG. 22.12. (a) A gray-scale 4×4 image. (b) The labels of the image generated using the EM algorithm and (c) the labels generated for the same image using the NEM algorithm. Notice the spatial smoothing attained by modifying the objective function.

Similarly, $\mu_B = 17.6$, $\sigma_B = 4.1$, and $p_A = \frac{5}{16}$. Computing the probability of a given pixel value belonging to a cluster is then a simple exercise using Bayes' theorem. For example, given a pixel value x , the probability that it belongs to cluster A is

$$\begin{aligned} P(A | x) &= \frac{P(x | A)p_A}{P(x)} \\ &= \frac{P(x | A)p_A}{P(x | A)p_A + P(x | B)p_B} \\ &= \frac{N(x, \mu_A, \sigma_A)p_A}{N(x, \mu_A, \sigma_A)p_A + N(x, \mu_B, \sigma_B)p_B} \end{aligned}$$

where

$$N(x, \mu_A, \sigma_A) = \frac{1}{\sqrt{(2\pi)\sigma_A^2}} \exp^{-\frac{(x-\mu_A)^2}{2\sigma_A^2}}$$

In our case neither the cluster labels nor the the distribution parameters are known. All we know is that there are two clusters and that each cluster is modeled as a Gaussian distribution. At first this problem may appear to be unsolvable because there are too many unknowns: cluster labels for each pixel and the distribution parameters of the cluster. Problems of this type can be solved using the EM algorithm. The EM algorithm, like the K -medoid algorithm, is an iterative algorithm that begins with the guess estimate of the distribution parameters. It then computes the “expected values” of the data given the initial parameters. The new expected data values are then used to calculate the maximum likelihood estimate for the distribution parameters. This procedure is iterated until some convergence criterion is met. The EM algorithm guarantees that the maximum likelihood estimate will improve after each iteration, although the convergence can be slow. The steps of the EM algorithm follow:

1. Guess the initial model parameters: μ_A^0 , Σ_A^0 and p_A^0 and μ_B^0 , Σ_B^0 and p_B^0 .
2. At each iteration j , calculate the probability that the data object x belongs to clusters A and B :

$$P(A | x) = \frac{p_A^j P^j(x | A)}{P^j(x)} \quad P(B | x) = \frac{p_B^j P^j(x | B)}{P^j(x)}.$$

3. Update the mixture parameters on the basis of the new estimate:

$$\begin{aligned} p_A^{j+1} &= \frac{1}{n} \sum_x P(A | x) & p_B^{j+1} &= \frac{1}{n} \sum_x P(B | x) \\ \mu_A^{j+1} &= \frac{\sum_x x P(A | x)}{\sum_x P(A | x)} & \mu_B^{j+1} &= \frac{\sum_x x P(B | x)}{\sum_x P(B | x)} \\ \sigma_A^{j+1} &= \frac{\sum_x P(A | x)(x - \mu_A^{j+1})^2}{\sum_x P(A | x)} & \sigma_B^{j+1} &= \frac{\sum_x P(B | x)(x - \mu_B^{j+1})^2}{\sum_x P(B | x)} \end{aligned}$$

4. Compute the log estimate $E_j = \sum_x \log(P^j(x))$. If for some fixed stopping criterion ϵ , $|E_j - E_{j+1}| \leq \epsilon$, then stop; else set $j = j + 1$.

The NEM Algorithm

A careful reader may have noticed that the EM algorithm completely ignores the spatial distribution of the pixel; it only works with the pixel values. Thus, if we rearrange the pixel values shown in Fig. 22.12(b), the EM algorithm will still come up with the same cluster labeling and the same values of the distribution parameters.³ Such a solution, as we know, does not take into account the spatial autocorrelation property inherent in spatial data. As we have mentioned before, the search space for spatially referenced data is a combination of a conceptual attribute space and the physical (geographic) space. The spatial autocorrelation property then implies that the clusters should vary gradually in the physical space.

To make the EM algorithm spatially sensitive, we first follow the recipe proposed by Ambroise, Dang, and Govaert (1997).

Step 1. The EM algorithm for mixture models is equivalent to the optimization of the following objective function:

$$D(c, \mu_k, \sigma_k, p_k) = \sum_{k=1}^K \sum_{i=1}^n c_{ik} \log(p_k N(x_i, \mu_k, \sigma_k)) - \sum_{k=1}^K \sum_{i=1}^n c_{ik} \log(c_{ik})$$

where $c = c_{ik}$, $i = 1, \dots, n$ and $k = 1, \dots, K$ define a fuzzy classification representing the grade of membership of data point x_i into cluster k . The c_{ik} 's satisfy the constraints ($0 < c_{ik} < 1$, $\sum_{k=1}^K c_{ik} = 1$, $\sum_{i=1}^n c_{ik} > 0$). Again we have two clusters $k = 1, 2$, and there are n data points.

Step 2. To account for spatial autocorrelation, we introduce a new term,

$$G(c) = \frac{1}{2} \sum_{k=1}^2 \sum_{i=1}^n \sum_{j=1}^n c_{ik} c_{jk} w_{ij}$$

where $W = (w_{ij})$ is the contiguity matrix as defined before.

The new “spatially weighted” objective function is

$$U(c, \mu, \sigma) = D(c, \mu, \sigma) + \beta G(c)$$

where $\beta \geq 0$ is a parameter to control the spatial homogeneity of the data set.

³Actually, because of the randomness of the initial parameters, each run of the EM algorithm can result in a different solution.

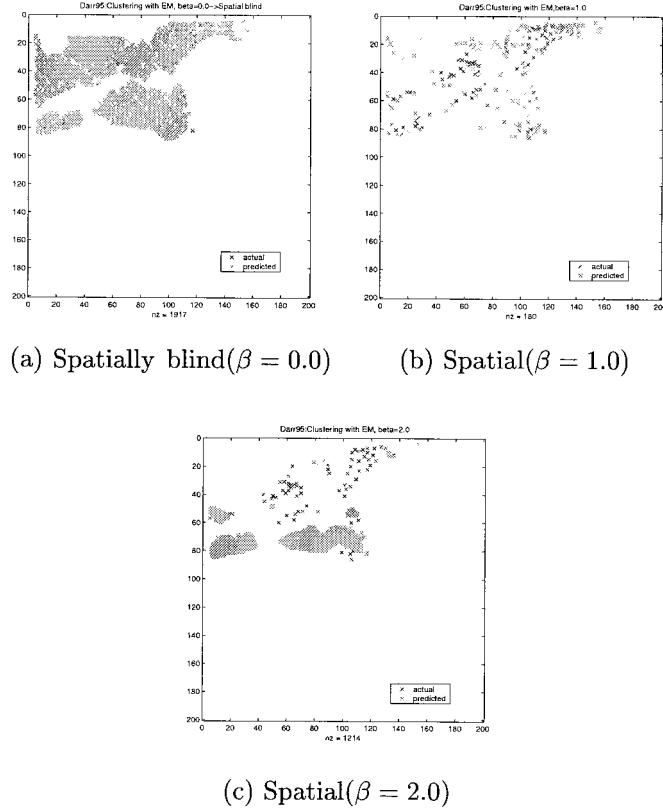


FIG. 22.13. Using the NEM algorithm. (a) As expected, clustering without any spatial information leads to poor results; (b) including spatial information ($\beta = 1.0$) leads to dramatic improvement of results; (c) overemphasizing spatial information ($\beta = 2.0$) again leads to poor results.

Step 3. Except for the new parameter c , which is an $n \times 2$ matrix, all the parameters are calculated exactly as before. The formula for c'_{ik} s is

$$c'_{ik} = \frac{p_k^m N(x_i, \mu_k, \sigma_k) \exp \left\{ \beta \sum_{j=1}^{d=n} c_{jk}^{m+1} w_{ij} \right\}}{\sum_{l=1}^2 p_l^m N(x_i, \mu_l^m, \sigma_l^m) \exp \left\{ \beta \sum_{j=1}^n c_{jl}^{m+1} w_{ij} \right\}}.$$

At each iteration m , the c'_{ik} can be solved using a fixed point iterative scheme.

We carried out experiments using the NEM algorithm on the bird data set. We assumed two clusters corresponding to the presence/absence of nests. When $\beta = 0$, the NEM reduces to the classical EM algorithm. We varied the β parameters, and the results are shown in Fig. 22.13. The results show that including spatial information in the clustering algorithm leads to a dramatic improvement in accuracy (Fig. 22.13(b) compared with Fig. 22.13(a)), but overemphasizing spatial information leads to “oversmoothing” and degradation in accuracy.

SUMMARY

In this chapter we presented techniques that are specifically designed to analyze large volumes of spatial data to find spatial outliers, colocation association rules, location prediction, and spatial clustering. We applied these techniques on traffic network sensors, wetlands, and

synthetic data sets. We compared the SAR and MRF models using a common probabilistic framework. Our study shows that the SAR model makes more restrictive assumptions about the distribution of features and class shapes (or decision boundaries) than MRF. We also observed an interesting relationship between classical models that do not consider spatial dependence and modern approaches that explicitly model spatial context. The relationship between SAR and MRF is analogous to the relationship between logistic regression and Bayesian classifiers. The analysis of spatial outlier detection algorithms showed the need for good clustering of data pages. The CCAM method yielded the best overall performance. We showed that the colocation miner algorithm is complete and correct and performs better than the well known a priori algorithm.

ACKNOWLEDGMENTS

We would like to thank our collaborators Prof. Paul Schrater, Dr. Sanjay Chawla, and Prof. Uygar Ozesmi for their various contributions, and group members Yan Huang, Chang-Tien Lu, Weili Wu, and Pusheng Zang for their contributions in developing these techniques and software. The comments of Kim Koffolt have greatly improved the readability of this chapter.

REFERENCES

- Agrawal, R., & Srikant, R. (1994). Fast algorithms for Mining Association Rules. In *Proceedings of 20th International Conference on Very Large Databases* (pp. 487–499). San Francisco: Morgan Kaufmann.
- Albert, P., & McShane, L. (1995). A generalized estimating equations approach for spatially correlated binary data: Applications to the analysis of neuroimaging data. *Biometrics*, 1, 627–638.
- Ambroise, C., Dang, V., & Govaert, G. (1997). Clustering of spatial data by the EM algorithm. *Quantitative Geology and Geostatistics*, 9, 493–504.
- Anselin, L. (1988). *Spatial econometrics: methods and models*. Dordrecht, Netherlands: Kluwer.
- Barnett, V., & Lewis, T. (1994). *Outliers in statistical data* (3rd ed.). New York: Wiley.
- Besag, J. (1974). Spatial interaction and statistical analysis of lattice systems. *Journal of the Royal Statistical Society (Series B)* 36, 192–236.
- Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society*, 48, 259–302.
- Boykov, Y., Veksler, O., & Zabih, R. (1999). Fast approximate energy minimization via graph cuts. In *Proceedings of the International Conference on Computer Vision* (pp. 377–384). Los Alamitos, CA: IEEE Computer Society Press.
- Chou, P., Cooper, P., Swain, M. J., Brown, C., & Wixson, L. (1993). Probabilistic network inference for cooperative high and low level vision. In R. Chellappa & A. K. Jain (Eds.) *Markov Random Field, Theory and Applications*. (pp. 211–243). New York: Academic Press.
- Derin, H., & Elliott, H. (1987). Modeling and segmentation of noisy and textured images using Gibbs random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9, 39–55.
- Geman, S., & Geman, D. (1984). Stochastic relaxation, gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 721–741.
- Greenman, G. (2000). Turning a map into a cake layer of information. *New York Times*. Retrieved [2000, January] from <http://www.nytimes.com/library/tech/00/01/circuits/articles/20giss.html>
- Gunther, O. (1989). The design of the cell tree: An object-oriented index structure for geometric databases. In *Proceedings of the Fifth International Conference on Data Engineering* (pp. 598–605). Los Alamitos, CA: IEEE Computer Society Press.
- Guting, R. (1994). An introduction to spatial database systems. In *Very Large Data Bases Journal*, 3, 357–399.
- Haining, R. (1989). Spatial data analysis in the social and environmental sciences. Cambridge, UK: Cambridge University Press.
- Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). Clustering algorithms and validity measures. *Proceedings of the 13th International Conference on Scientific and Statistical Database Management* (pp. 3–22). Los Alamitos, CA: IEEE Computer Society Press.
- Hawkins, D. (1980). *Identification of outliers*. London: Chapman & Hall.

- Hipp, J., Guntzer, U., & Nakaeizadeh, G. (2000). Algorithms for association rule mining—a general survey and comparison. In *SIGKDD Explorations*, 2, 58–64.
- Hohn, M., Gribki, L., & Liebhold, A. (1993). A geostatistical model for forecasting the spatial dynamics of defoliation caused by the Gypsy moth *Lymantria dispar* (Lepidoptera: Lymantriidae). *Environmental Entomology* 22, 1066–1075.
- Issaks, E. H., & Srivastava, R. M. (1989). Applied geostatistics. Oxford: Oxford University Press.
- Jhung, Y., & Swain, P. H. (1996). Bayesian contextual classification based on modified M-estimates and Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34, 67–75.
- Koperski, K., Adhikary, J., & Han, J. (1996). Knowledge discovery in spatial databases: Progress and challenges. In *Proceedings of the ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery* (pp. 55–70). Montreal, Canada.
- Koperski, K., & Han, J. (1995). Discovery of spatial association rules in geographic information databases. In *Proceedings of the Fourth International Symposium on Large Spatial Databases* (Vol. 951 of Lecture Notes in Computer Science, pp. 47–66). Berlin: Springer-Verlag.
- Krugman, P. (1995). *Development, geography, and economic theory*. Cambridge, MA: MIT Press.
- LeSage, J. (1997). Bayesian estimation of spatial autoregressive models. *International Regional Science Review*, 20, 113–129.
- LeSage, J. P., & Pace, R. (2001). Modeling spatial dependencies for mining geospatial data: An introduction. In *Geographic Data Mining and Knowledge Discovery*. Taylor & Francis.
- Li, S. (1995). Markov random field modeling in computer vision. New York: Springer-Verlag.
- Li, Z., Cihlar, J., Moreau, L., Huang, F., & Lee, B. (1997). Monitoring fire activities in the Boreal Ecosystem. *Journal of Geophysical Research*, 102, 611–629.
- Luc, A. (1994). Exploratory spatial data analysis and geographic information systems. In M. Painho (Ed.), *New tools for spatial analysis* (pp. 45–54).
- Luc, A. (1995). Local indicators of spatial association: LISA. *Geographical Analysis*, 27, 93–115.
- Mark, D. (1999). Geographical information science: Critical issues in an emerging cross-disciplinary research domain. *URISA Journal*, 12, 45–54.
- Nepstad, D., Veríssimo, A., Alencar, A., Nobre, C., Lima, E., Lefebvre, P., Schlesinger, P., Potter, C., Moutinho, P., Mendoza, E., Cochrane, M., & Brooks, V. (1999). Large-scale impoverishment of Amazonian forests by logging and fire. *Nature*, 398, 505–508.
- Orenstein, A., & Merrett, T. (1984). A class of data structures for associative searching. In *Proceedings of the Symposium on Principles of Database Systems* (pp. 181–190). New York: ACM Press.
- Ozesmi, U., & Mitsch, W. (1997). A spatial habitat model for the marsh-breeding red-winged black-bird (*Agelaius phoeniceus* l) in coastal Lake Erie wetlands. *Ecological Modelling*, 101, 139–152.
- Ozesmi, S., & Ozesmi, U. (1999). An artificial neural network approach to spatial habitat modeling with interspecific interaction. *Ecological Modelling*, 116, 15–31.
- Pace, R., & Barry, R. (1997a). Quick computation of regressions with a spatially autoregressive dependent variable. *Geographic Analysis*, 29, 232–247.
- Pace, R., & Barry, R. (1997b). Sparse spatial autoregressions. *Statistics and Probability Letters*, 33, 291–297.
- RuleQuest Research, Rulequest data mining tools. Retrieved [2002] from <http://www.rulequest.com/>.
- Roddick, J.-F., & Spiliopoulou, M. (1999). A bibliography of temporal, spatial and spatio-temporal data mining research. *SIGKDD Explorations* 1, 34–38.
- SAS. Enterprise miner. Retrieved [2002] from <http://www.sas.com/products/miner/index.html>
- Shekhar, S., & Chawla, S. (2002). *A tour of spatial databases*. Prentice Hall.
- Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., Liu, X., & Lu, C.-T. (1999). Spatial databases—accomplishments and research needs. *Transactions on Knowledge and Data Engineering* 11, 45–55.
- Shekhar, S., & Huang, Y. (2001). Colocation rules mining: A summary of results. In *Proceedings of the Spatio-Temporal Symposium on Databases* (Vol. 2121 Lecture Notes of Computer Science). Berlin: Springer-Verlag.
- Shekhar, S., & Liu, D.-R. (1997). CCAM: A connectivity-clustered access method for aggregate queries on transportation networks. *IEEE Transactions on Knowledge and Data Engineering*, 9, 102–119.
- Shekhar, S., Lu, C., & Zhang, P. (2001). A unified approach to spatial outliers detection. *Technical Report TR01-045 of Department of Computer Science, University of Minnesota*. Retrieved [2001] from http://www.cs.umn.edu/research/shashi-group/paper_list.html
- Shekhar, S., Schrater, P. R., Vatsavai, R. R., Wu, W., & Chawla, S. (2002). Spatial contextual classification and prediction models for mining geospatial data. *IEEE Transactions on Multimedia*, 4, 174–188.
- Shekhar, S., Yang, T., & Hancock, P. (1993). An intelligent vehicle highway information management system. *International Journal on Microcomputers in Civil Engineering*, 8.
- Solberg, A. H., Taxt, T., & Jain, A. K. (1996). A Markov random field model for classification of multisource satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 34, 100–113.

- SPSS. Clementine home. Retrieved [2002] from <http://www.spss.com/clementine/>
- Stolorz, P., Nakamura, H., Mesrobian, E., Muntz, R., Shek, E., Santos, J., Yi, J., Ng, K., Chien, S., Mechoso, R., & Farrara, J. (1995). Fast spatio-temporal data mining of large geophysical datasets. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (pp. 300–305). Menlo Park, CA: AAAI Press.
- Warrender, C. E., & Augusteijn, M. F. (1999). Fusion of image classifications using Bayesian techniques with Markov random fields. *International Journal of Remote Sensing*, 20, 1987–2002.
- Worboys, M. (1995). *GIS: A Computing Perspective*. Bristol, PA: Taylor and Francis.
- Yasui, Y., & Lele, S. (1997). A regression method for spatial disease rates: An estimating function approach. *Journal of the American Statistical Association*, 94, 21–32.

23

Mining Science and Engineering Data

Chandrika Kamath

Lawrence Livermore National Laboratory

Introduction	550
Motivation for Mining Scientific Data	551
Data Mining Examples in Science and Engineering	552
Data Mining in Astronomy	552
Data Mining in Earth Sciences	555
Data Mining in Medical Imaging	557
Data Mining in Nondestructive Testing	557
Data Mining in Security and Surveillance	558
Data Mining in Simulation Data	558
Other Applications of Scientific Data Mining	561
Common Challenges in Mining Scientific Data	561
Potential Solutions to Some Common Problems	562
Data Registration	564
De-Noising Data	565
Object Identification	566
Dimensionality Reduction	567
Generating a Good Training Set	568
Software for Scientific Data Mining	568
Summary	569
References	569

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48.

INTRODUCTION

Data mining techniques have long been applied to the analysis of scientific data in various domains such as astronomy and remote sensing. However, with advances in sensor technology these data sets are becoming not only more massive, but also more complex. In many cases a single problem may need to exploit data obtained from different sensors, at different wavelengths, at different resolutions, at different times, and from different viewpoints. In addition, new sources of data are becoming available such as the data from computer simulations of complex physical phenomena. As a result, data sets from science and engineering applications are providing a rich environment for the application of data mining techniques. This chapter provides an overview of data mining applications in scientific domains, identifies some of the common themes and challenges in mining such data, and briefly discusses potential solutions to these common problems.

Advances in technology have enabled us to collect data far more rapidly than we can analyze it. This is especially true in science and engineering domains, where vast amounts of data are being collected through observations, experiments, and simulations. These data sets are not only massive, being measured in terabytes and beyond, they are also very complex. As a result, data mining techniques are becoming invaluable in helping scientists and engineers find useful information in their data. However, given the focus on commercial and business data, the advances made in the analysis of scientific data, as well as the opportunities that await a data mining practitioner analyzing such data, are often overlooked. In this chapter I hope to address this oversight by providing an overview of the scientific applications that benefit from data mining, identifying the common challenges in these applications, and discussing some of the solutions to these problems.

There are several aspects of data mining that are common across both commercial and scientific data sets, such as the use of algorithms like decision trees or neural networks for classification, and k -means for clustering. However, there are several differences as well; these form the focus of this chapter. For example, scientific data is usually available in the raw form as noisy images, or as unstructured mesh data with physical variables at each mesh point, or as the output of different sensors observing a scene. As a result, a substantial amount of preprocessing is needed to bring this raw data into a form suitable for the detection of patterns. This preprocessing could even vary as different types of analyses are done with a given data set. In addition, the validation of the patterns (or information) identified by data mining, is an important, though often overlooked, postprocessing step in mining scientific data.

This chapter is organized as follows. First, in the following section I briefly describe the motivation behind scientific data mining. Next, I illustrate the diversity of problems in the different application areas where data mining techniques are being applied. In the fourth section I use these examples to identify the common themes and challenges in the analysis of scientific data. The following section discusses some ways of solving these challenges. The focus in this section is on the aspects of scientific data mining that differentiate it from commercial and business data mining. The issues common to both types of data are covered elsewhere in this book. Given the limited space and the breadth of the topics being covered, I envision the section as a first introduction rather than a comprehensive survey of all possible solutions to a problem. I conclude with a brief summary. Throughout this chapter I use the term “scientific data” to imply both science and engineering data.

MOTIVATION FOR MINING SCIENTIFIC DATA

The motivation behind mining data, whether commercial or scientific, is the same—the need to find useful information in data to enable better decision making or a better understanding of the world around us. Traditionally, data analysts have turned to data mining techniques when the size of their data has become too large for manual or visual analysis. In the science and engineering domains the size of the data is only one reason why data mining techniques are gaining popularity. Science data in areas such as remote sensing, astronomy, and computer simulations is routinely being measured in terabytes and petabytes. However, what makes the analysis of these data sets challenging is not just the size, but the complexity of the data. This complexity is the result of several advances in technology, including the following:

- Improved sensor technology. With sensors becoming ubiquitous and more powerful than before, the sources of scientific data also have become more varied. For example, in the past astronomical telescopes were typically ground-based; but now, with the Hubble telescope and the Chandra X-ray telescope, as well as missions to other planets, we have space-based data collecting abilities to complement the ground-based systems. In addition, satellites are collecting data at different resolutions and at different frequencies. For example, we now have systems operating at several different wavelengths, including visible (photographic and electro-optical), infrared, and microwave. In addition, multi- and hyperspectral imagers are now routine, enabling a more precise discrimination of the materials being observed.
- Improved data storage and compression capabilities. With memory sizes increasing rapidly, disk space becoming cheaper, and data archiving capabilities becoming more accessible, much of the science data being collected is now either online or relatively easily accessible in a digital form. In addition, data compression techniques such as those based on wavelets have made it relatively easy to transmit this digitized data without any visible loss of information. As a result, computational data analysis techniques are now being applied to data previously analyzed using visual means simply because the data were not in a digitized form. For example, doctors are now more comfortable with the digitized versions of mammograms, and medical imaging techniques are increasing becoming more sophisticated as they gain acceptance. In addition, as many diverse sources of data in a field come online, scientists increasingly want to exploit this new information. A classic example is the National Virtual Observatory, where astronomers are interested in mining data spread across several different surveys.
- Improved computing power. As computers become faster, they not only help in the analysis of data, but also give rise to new sources of data, such as those arising from computer simulations. Simulations are increasingly being seen as the third mode of science, complementing theory and experiments. Petabytes of data are routinely produced in domains such as structural mechanics and computational fluid dynamics as a result of simulations of car crash tests or the flow around an airplane. This source of data gives rise to new types of analysis problems such as comparing simulations to one another or to experiments.

These advances in technology have introduced complexity in scientific data, complexity that can take various forms such as multisensor, multispectral, multiresolution data; spatiotemporal data; high dimensional data; structured and unstructured mesh data from simulations; data contaminated with different types of noise; three-dimensional data, and so on. As a result of this complexity, visual data analysis, given its subjective nature and the human limitations in absorbing details, is becoming impractical even for moderate-sized scientific data sets. For

large to massive data sets visual analysis is practically impossible. As a result, science and engineering data sets provide a very rich environment for the application of data mining, one in which the diversity of problems is matched only by the potential benefits obtained when knowledge is discovered in the data. In later sections I briefly describe ways in which this complexity can be addressed. First, I describe in more detail some of the problems that are being addressed in various scientific domains.

DATA MINING EXAMPLES IN SCIENCE AND ENGINEERING

There are many ways in which data mining techniques are being applied in science and engineering domains. In this section I describe several of these applications with the goal of illustrating the challenges in mining scientific data. In some of the domains I cover one or two representative problems in detail; in others, I describe, in general, the problems being addressed through data mining. Readers who are interested in additional details can consult the references as well as other resources (Grossman, Kamath, Kegelmeyer, Kumar, & Namburu, 2001; Institute for Pure and Applied Mathematics, 2002). I also focus on domains that are not covered elsewhere in this handbook. For example, applications in mining geo-spatial data, manufacturing data, and bioinformatic data are not covered, though they are valid applications of data mining in science and engineering.

Data Mining in Astronomy

Astronomy has had a long history of being awash in data, and therefore has been a good candidate for the application of data mining techniques (Fayyad, Smyth, Burl, & Perona, 1996; Jarvis & Tyson, 1981; Odewahn, Stockwell, Pennington, Humphreys, & Zumach, 1992; Storrie-Lombardi, Lahav, Sodre, & Storrie-Lombardi, 1992). As case studies, I describe two recent applications, one focusing on optical data and one on data at radio frequencies.

Recognizing Volcanoes on Venus. The JARtool software system (Burl et al., 1998) at the Jet Propulsion Laboratory was developed to learn to recognize volcanoes in a large data set of Venusian imagery. Volcanism is the most widespread and important geologic phenomenon on Venus. It is of particular interest to geologists studying the planet, because by understanding the global distribution and clustering characteristics of the volcanoes, they can understand the geologic evolution of the planet. As part of the Magellan mission to Venus, more than 30 thousand 1024×1024 pixel images covering 98% of the surface were obtained. Based on previous low-resolution imagery, it was estimated that there are approximately 1 million small volcanoes of diameter less than 20 km in this imagery. Although a catalog of larger Venusian volcanoes has been completed manually, it was expected that a comprehensive catalog of the small volcanoes would take 10 to 20 person-years. Data mining techniques therefore presented an approach to creating this catalog in an objective manner.

I selected the JARtool effort as a case study instead of its more well-known predecessor SKICAT (Fayyad, Smyth, Burl, & Perona, 1996) because it is more representative of the problems faced in mining scientific data. In SKICAT a good set of features for the objects had already been obtained by the astronomy community, and so much of the effort focused on classification performance. In contrast, in the Venus imagery, separating the volcanoes from the background is quite difficult (see Fig. 23.1 for an illustrative image) and there is no established set of pixel-level features for the volcanoes, thus making the problem far more complicated.

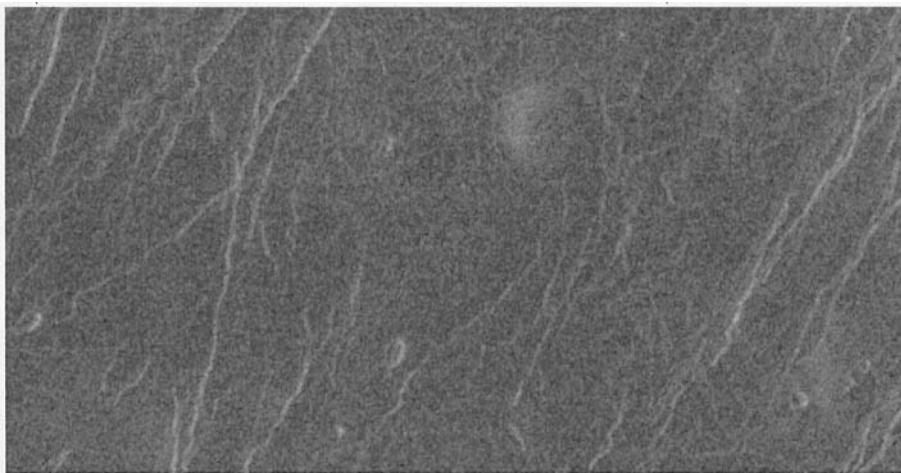


FIG. 23.1. Venusian imagery from the UCI KDD archive (Source: <http://kdd.ics.uci.edu/>).

The approach used in JARtool for the detection of volcanoes, and described in detail in Burl et al. (1988), can be summarized as follows. Burl et al. first used a matched filter derived from training examples to focus attention on regions of interest within an image. Principal components analysis of the training examples provided a set of domain-specific features that mapped the low-level pixel data to a higher-level feature space. An extensive set of supervised machine learning techniques, including quadratic (or Gaussian) classifiers, decision trees, feedforward neural networks, linear discriminant analysis, nearest neighbors, Gaussian mixture models, and so forth were used. For an initial test set of images, it was found that all methods (with the exception of linear discriminant analysis) yielded similar performance, indicating that the critical design choices in the feature learning stage had already been made.

The experiences of Burl and colleagues with the JARtool system and the recognition of volcanoes on Venus provides some interesting insights into data mining applied to science data. They observed that training and testing classifiers took less than 20% of effort on the project. In contrast, a large fraction of the effort (at least 30%) was spent on trying to find an effective feature space in which to apply the classification algorithms. They also found that in real-world systems each component had its own set of parameters, making overall system optimization difficult if not impossible. In addition, the labels assigned by astronomers to the objects were often noisy and subjective. Adding to the problems, spatial context was important in image analysis problems, and training a model on one part of the planet could lead to poor performance elsewhere. These observations, though made in the context of one application, are important in mining scientific data; I shall revisit them again later in this chapter.

Classification of Bent-Double Galaxies. As part of the Sapphire project (<http://www.llnl.gov/casc/sapphire>), my colleagues and I have been involved in the classification of radio-emitting galaxies with a bent-double morphology. This work was done using the data from the FIRST (Faint Images of the Radio Sky at Twenty-cm) astronomical survey (Becker, White, & Helfand, 1995). The FIRST survey was started in 1993 with the goal of producing the radio equivalent of the Palomar Observatory Sky Survey. Using the Very Large Array at the National Radio Astronomy Observatory, FIRST is scheduled to cover more than 10,000 square degrees of the northern and southern galactic caps, to a flux density limit of

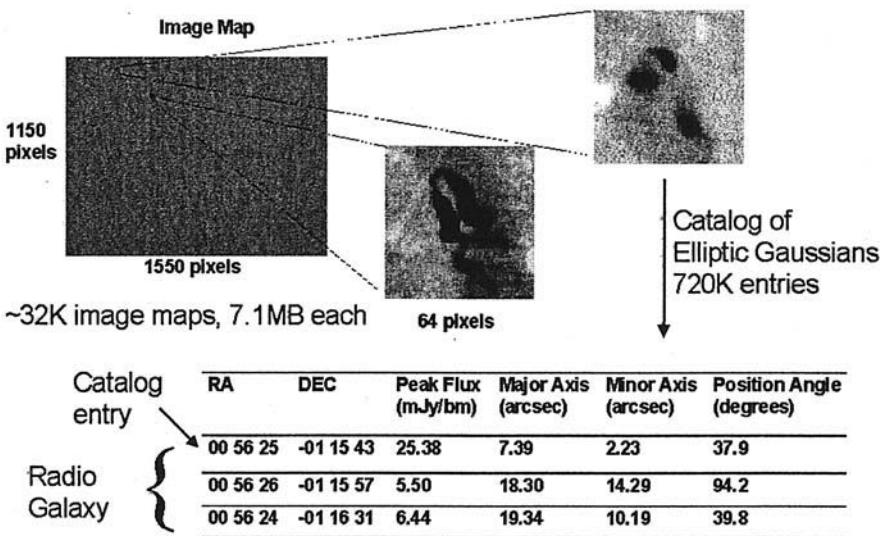


FIG. 23.2. FIRST data: Images, maps, and catalog entries.

1.0 mJy (milli-Jansky). With the data from the 1993 through 1999 observations, FIRST has covered about 8,000 square degrees, producing more than 32,000 two-million pixel images. At a threshold of 1 mJy, there are approximately 90 radio-emitting galaxies, or radio sources, in a typical square degree. The survey, when complete, will have nearly a million galaxies.

Radio sources exhibit a wide range of morphological types that provide clues to the source class, emission mechanism, and properties of the surrounding medium. Of particular interest are sources with a bent-double morphology, because they indicate the presence of clusters of galaxies, a key project within the FIRST survey. FIRST scientists currently use a manual approach to detect bent-double galaxies. If a visual inspection of an image indicates that a galaxy is possibly a bent-double, then additional observations are carried out to study it in more detail. This visual inspection of the radio images, besides being very subjective, has also become increasingly infeasible as the survey has grown in size. Our goal was to bring automation to this process of classifying galaxies and replace the visual inspection by data mining techniques.

The data from the FIRST survey (<http://sundog.stsci.edu>) is available as image maps and a catalog. Figure 23.2 shows an image map and the three catalog entries corresponding to one of the bent-doubles present in the image map. These large image maps are mostly “empty,” that is, composed of background noise that appears as streaks in the image. The FIRST catalog (White, Becker, Helfand, & Gregg, 1997) is obtained by processing an image map to fit two-dimensional elliptic Gaussians to each radio source. Each entry in the catalog corresponds to the information on a single Gaussian. This includes, among other things, the coordinates for the center of the Gaussian, the major and minor axes, the peak flux, and the position angle of the major axis (degrees counterclockwise from north).

Our approach to mining the FIRST data for bent-doubles is described in detail elsewhere (Fodor, Cantú-Paz, Kamath, & Tang, 2000; Kamath, Cantú-Paz, Fodor, & Tang, 2001). Our initial focus was on the catalog data because it was easy to work with and a good representation of all but the most complex of radio-emitting galaxies. We first grouped the catalog entries that were close to each other and then focused on those groups that consisted of two or three catalog entries. This was based on the observation that a single entry galaxy was unlikely to be a bent-double, whereas four or more entries in a galaxy would make it complex enough to be of interest

to astronomers. We next extracted a separate set of features for the two- and three-entry galaxies, focusing on features such as relative distances and angles between entries. These features were likely to be robust and invariant to rotation, scaling, and translation. Separating the two- and three-entry galaxies enabled us to have uniform length feature vectors for each case. However, it also meant that a small training set (313 examples) was split further into smaller training sets of 118 examples for two-entry and 195 examples for three-entry sources, respectively. Our focus thus far has been on the three-entry sources, with the training set consisting of 167 bent-doubles and 28 nonbent-doubles. Note that the training set is unbalanced, with far more bent-doubles than nonbent-doubles.

Once we had extracted an initial set of features, we continued refining them until the cross-validation error for a decision tree classifier was reduced to about 10%, a number the astronomers felt was sufficient for their use. The tree created using this set of features was then used to classify unlabeled galaxies. Several of these galaxies were shown to the astronomers for validation. Because we wanted to use this new set to enhance our training set, we selected a higher percentage of galaxies that had been classified as nonbents. This process of validation is rather tedious and has the drawback of being not only subjective, but somewhat inconsistent because the labels assigned by an astronomer are subject to the drift common to human labelers. Therefore, we were able to validate only 290 galaxies, of which 92 were bents and 198 nonbents. These were combined with the original training set of 195 instances, and the combined set of 485 instances was used as a new training set.

Decision trees created with the new training set increased the cross-validation error to 20%. A closer inspection of the misclassified instances indicated that they belonged to the borderline cases, the labels of which were often found to be subjective and inconsistent. Therefore, we went back to the original 195-item training set and proceeded along two directions. First, we tried to reduce the number of features using techniques such as principal component analysis and exploratory data analysis. Next, in addition to decision trees, we tried generalized linear models (Fodor & Kamath, *in press*). The six models created using different features and classifiers were then used to classify the unlabeled galaxies. If all six models agreed with a label, we considered the galaxy to be a bent or a nonbent with high probability. As fewer models agreed with a label, the probability of the galaxy being bent or nonbent was reduced. This enabled the astronomers to first focus on those galaxies that were bent with a high probability.

Our experiences with the problem of classification of bent-double galaxies led to several observations. The existence of the catalog, the easy access to the data, and the availability of software to read, write, and display the astronomical images all were an immense help in getting started on the problem. We also found that we had to interact extensively with the astronomers, especially at the beginning, to understand the data, how it was collected, identify the features that might be relevant to the classification of bent-doubles, and so forth. We also learned that the labels assigned by the astronomers to the galaxies could be inconsistent, especially in the borderline cases, which also turned out to be the hardest to classify. This, in addition to the lack of ground truth, made it difficult to obtain a good training set. Also, the data in the survey was in the process of being collected. This meant that as new data was added, galaxies at the edge of the survey that appeared in the catalog in one year could fall below the threshold of processing as new data was added the following year. As a result, we had to be careful when we incorporated new data into our existing data set.

Data Mining in Earth Sciences

Ever since the first balloonists took pictures of the ground using a simple camera, remote sensing has been a source of vast amounts of data (Canada Center for Remote Sensing, 2001; NASA, 1999a, 1999b; Short, 2002). Of particular interest are images taken of the earth, because

they lead to a very rich set of applications. For example, remote sensing data has been used to evaluate dynamic changes from natural events such as floods and volcanoes; seek surface clues such as existence of a particular type of vegetation to indicate subsurface mineral deposits; monitor crops in terms of their identity, stage of growth, and predicted yield; understand land use by identifying and categorizing the various natural and man-made features around large cities; observe various properties of the terrestrial atmosphere and climate changes; and identify petroleum fields.

Once the data has been collected, some of the common tasks in the above applications include data registration to align data taken at different times; change detection to identify changes over time; classification to identify different types of minerals, vegetation, or man-made objects; image processing to separate areas of interest from the background, and so forth. Remote sensing data is often multispectral, multisensor, and multiresolution. It is also noisy, as a result of noise from the sensors, atmospheric conditions, and so forth. However, what is noise and what is signal may not always be clear. For example, if a study of clouds is being conducted to try to classify them into different types, the clouds would be the signal; however, in an optical image, clouds could be the noise obscuring the image of a city.

Remote sensing data often has a temporal component, because one of the main applications of such data is to understand changes in the earth and its environment over time. As a result of this temporal component, data registration is a key task in remote sensing. Another characteristic of remote sensing data is its massive size. For example, the Earth Observing System data is expected to have 11,000 terabytes of data collected, processed, and stored during the 15-year lifetime of the program (Short et al., 1995).

I next present some examples from ADaM (Algorithm Development and Mining), a project in mining earth science data (Ramachandran, Conover, Graves, & Keiser, 2000). Over the years, the ADaM system has been applied to several different problems, and these experiences have led to some valuable insights, from both scientific and data mining viewpoints (http://datamining.itsc.uah.edu/case_studies/). For example, ADaM researchers have analyzed data from GOES satellites to detect cumulus cloud fields (Nair et al., 1999). These satellites are used for monitoring and predicting weather, and the study of the formation of cloud fields has helped atmospheric scientists to better understand how land use changes such as urbanization and deforestation can affect the climate. Cloud fields often begin as boundary layer cumulus fields consisting of many small scattered clouds. Traditional cloud detection algorithms based on spectral properties often miss these clouds because they can have a diameter of 1 km or less, which is significantly less than the size of a single infrared pixel. So, the higher resolution visible channel is used. Using a variety of methods based on various texture features and edge detection techniques, ADaM researchers first identified the best method and then mined the satellite data for a certain time period. One of their important observations was that in generating labeled data, the atmospheric scientists did not always agree on the label. To address this, ADaM scientists used only those training examples for which all scientists agreed on a label. This approach is practical when there is lots of training data, but not when such data, however inconsistent, is limited. The ADaM system has also been used in other applications including rain detection using classical pattern recognition techniques such as backpropagation and discrete Bayes classifiers, and lightning detection using a small set of training examples and combining techniques from image processing and genetic algorithms.

Other examples of data mining in remote sensing include the use of neural nets to generate land cover maps (Liu, Gopal, & Woodcock, 2001), detection of earthquakes using statistical inference techniques (Stolorz and Dean, 1996), oil spill detection (Kubat, Holte, & Matwin, 1998), and other examples included in the excellent tutorials on remote sensing (Short, 2002; Canada Center for Remote Sensing, 2001).

Data Mining in Medical Imaging

A field that is becoming a rich area for the application of data mining is that of medical imaging (SPIE, 2000b). The tremendous pace of development in imaging technologies such as X-rays, computed tomography, magnetic resonance, ultrasound, positron emission tomography, and single-photon emission computed tomography has led to the generation of vast amounts of data. Scientists are interested, of course, in learning from this data, and data mining techniques are increasingly being applied in these analyses. For example, identifying tumors in mammograms has been an active area of research, and technology based on neural networks and signal processing has made its way into commercial products for the detection of breast and cervical cancers (Lewin, 2000).

In this section, as an illustrative example, I briefly describe the work being done in mining brain images. As mentioned in Thompson et al. (2000), a number of brain imaging investigations are being conducted with the goal of analyzing how the dynamically changing brain varies across age, gender, disease, multiple imaging modalities, and within large human populations. These studies require the use of sophisticated algorithms for brain image analysis using techniques based on computer vision, image analysis, computer graphics, and artificial intelligence. Novel image analysis algorithms are uncovering new patterns of altered structure and function in individuals and clinical populations. The problems that are being addressed through brain imaging are based on the extraction of shape parameters that are subjected to multivariate analysis of variance to test hypotheses about disease-specific changes. These studies include the identification of patterns in patients at risk for Alzheimer's disease, adult and childhood-onset schizophrenia, and fetal alcohol syndrome, as well as interactions between gender and disease and subtle developmental shape changes in childhood.

An excellent overview of the current challenges in brain image analysis focusing on the algorithms, their technical foundations, and their scientific and clinical applications is given in Thompson et al. (2000). These algorithms typically include methods for image segmentation, anatomical parameterization and modeling, tissue classification and shape analysis, and pathology detection in individuals and groups. In addition, a key task in brain image analysis is the creation of digital brain atlases, that is, annotated representations of anatomy in a three-dimensional coordinate system that serve as standardized templates over which other brain maps can be overlaid for subsequent comparison and integration. As a result, image registration is a key problem in brain image analysis, and robust techniques, particularly those based on nonlinear approaches, are an active area of research.

Data Mining in Nondestructive Testing

An application area often overlooked in data mining of scientific data sets is that arising from nondestructive testing. These techniques are used to study the inside of an object without affecting it (e.g., breaking it apart). This might include situations in which the contents of an object might be dangerous, or it might be more cost-effective to analyze the contents without taking the object apart. For example, if there is a container of unknown but potentially dangerous chemical, a nondestructive evaluation technique might help identify the chemical so appropriate measures could be taken to handle the container. These techniques can also be used to detect flaws in components such as air bubbles in poured concrete, corrosive parts in airplanes (Brown & Brence, 2001), damage in bridges (HERMES Project Web page, 2000), and land mines buried in a field.

In nondestructive testing the object of interest is analyzed using waves, either mechanical or electrical, depending on the situation. Once the waves have propagated through the material,

the output response is observed and analyzed. The frequency of the wave must be chosen appropriately to “see” defects in the material or determine the signature of a chemical in a container. The output response then can be analyzed effectively using data mining techniques.

Data Mining in Security and Surveillance

The increased need for security and surveillance is creating a rich environment for data mining. The applications of data mining include human face detection and recognition (Frischholz, 2002; Kruizinga, 2002), network intrusion detection (Marchette, 2001), biometric identification including fingerprint and retinal identification, and various applications in the military such as automated target recognition (SPIE, 2000a). These applications typically require real time turnaround and often involve security and privacy issues—a characteristic that is not frequently seen in scientific data mining, where data sets in most domains are openly shared. In addition, the data sets are also very large; for example, the Federal Bureau of Investigation fingerprint database is over 200 terabytes, and compression techniques based on wavelets are used to store and transmit the data (Brislawn, 2002; Candela & Chellappa, 1993). Security and surveillance is another domain in which there is an increasing need to combine data from multiple and diverse sensors and sources to make inferences about events, activities, and situations.

Data Mining in Simulation Data

A new source of data that is ripe for mining is that arising from computer simulations of complex physical phenomena. Computer simulations are being seen as the third mode of science, complementing theory and experiment. The idea behind such simulations is to understand complex phenomena by analyzing mathematical models on high-performance computers. This approach is often taken when the phenomena are too complex to be solved by analytical methods or too expensive, impractical, or dangerous to be studied using experimental means. Examples include computational fluid dynamics simulations to study the flow around an airplane, environmental simulations to understand the effects of volcanic eruptions on global climate, or structural mechanics simulations to study the effect of car crash tests.

Computer simulations are being used in several different fields for a variety of problems such as astrophysics for modeling how stars evolve, computational fluid dynamics to understand turbulence, combustion to understand the interaction between turbulent flow fields and chemical reactions, structural mechanics to model the stability of bridges, and so on. In these problems the physical phenomenon of interest is modeled using partial differential equations (PDEs) (Vemuri & Karplus, 1981). In many fields (turbulence is a notable exception), the physical phenomena are well understood, and hence the PDE represents the model of the data output from the simulations. However, data mining can still be useful in the analysis of simulation output as a technique to complement visualization. Other areas in which data mining techniques are being applied are the identification of coherent structures in turbulent flow (Marusic, 2000; Marusic, Candler, Interrante, Subbareddy, & Moss, 2001), verification and validation involving the comparison of simulation results with other simulation results or with experimental data, identification of damaged structures in a structural mechanics simulation (Sandhu, Kanapady, Tamma, Kamath, & Kumar, 2001), generating meshes for discretization of the simulation (Chedid & Najjar, 1996), and understanding the design parameter space, in which a model is built to understand how the output of a simulation depends on the input parameters.

Because data from computer simulations are a new source of data for mining, perhaps a brief description of such data might be helpful. Mesh data from simulations can be broadly

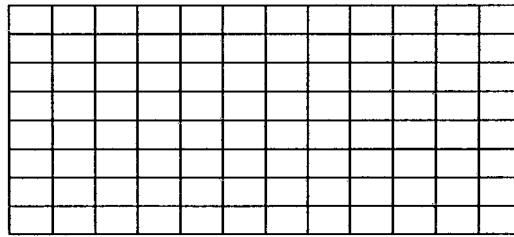


FIG. 23.3. An example of an image or a regularly spaced Cartesian mesh.

categorized into structured and unstructured data. In structured data types the data can be either regular in the form of a Cartesian mesh (Fig. 23.3) or curvilinear (as shown in the left panel in Fig. 23.4). In the case of a regularly spaced Cartesian mesh, techniques from image processing can be applied to preprocess the mesh data. In the unstructured mesh data category, for example, the right panel in Fig. 23.4, the data can be composed of either homogeneous elements (in this case triangles) or heterogeneous elements as in Fig. 23.5. In addition to structured and unstructured meshes there are also meshes that are a hybrid of the two—these include hierarchical meshes, which are globally unstructured but locally structured (Fig. 23.6), and meshes that are structured in one part of the problem domain and unstructured in another. In many problems the meshes are not static—as the physical features in the simulation evolve, the mesh too must adaptively change to accurately model these features. There are also meshless methods used in simulations that result in output data that is irregularly spaced. Often a series of two- or three-dimensional data sets is produced as the simulation evolves in time and data is output at each time step.

For simulation data, especially data generated using non-Cartesian meshes, a key problem is the extraction of features. This is because the data values are not available on a regular grid as in an image, but at data points irregularly spaced in two- or three-dimensions. One solution to this is to use domain information that may not require the spatial distribution of the grid points. Another is to project the irregularly spaced points on to a regular grid, but this introduces errors due to the projection. One could also borrow ideas from image analysis techniques that use

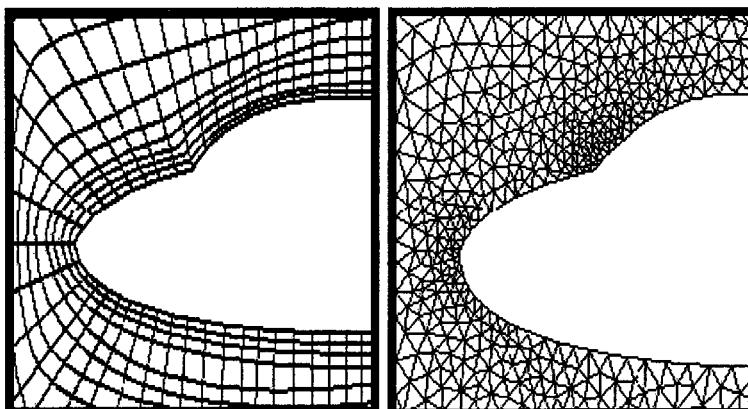


FIG. 23.4. An example of two-dimensional structured mesh (left) and an unstructured mesh (right) around the front of an aircraft (http://www.nas.nasa.gov/Pubs/Docs/FAST/chp_16.surferu.html).

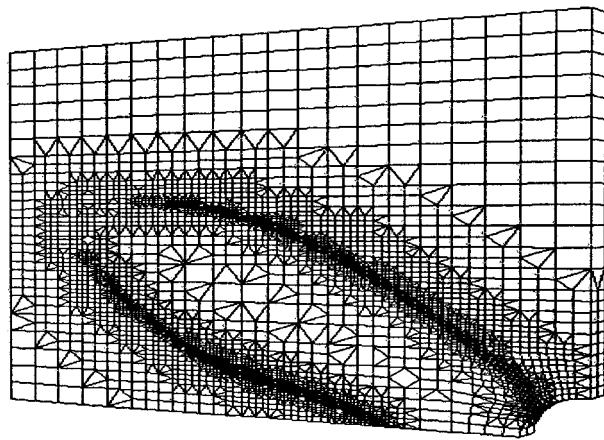


FIG. 23.5. An example of a three-dimensional unstructured mesh with heterogeneous elements (http://cox.iwr.uni-heidelberg.de/ug/Images/benchmark_grid.gif).

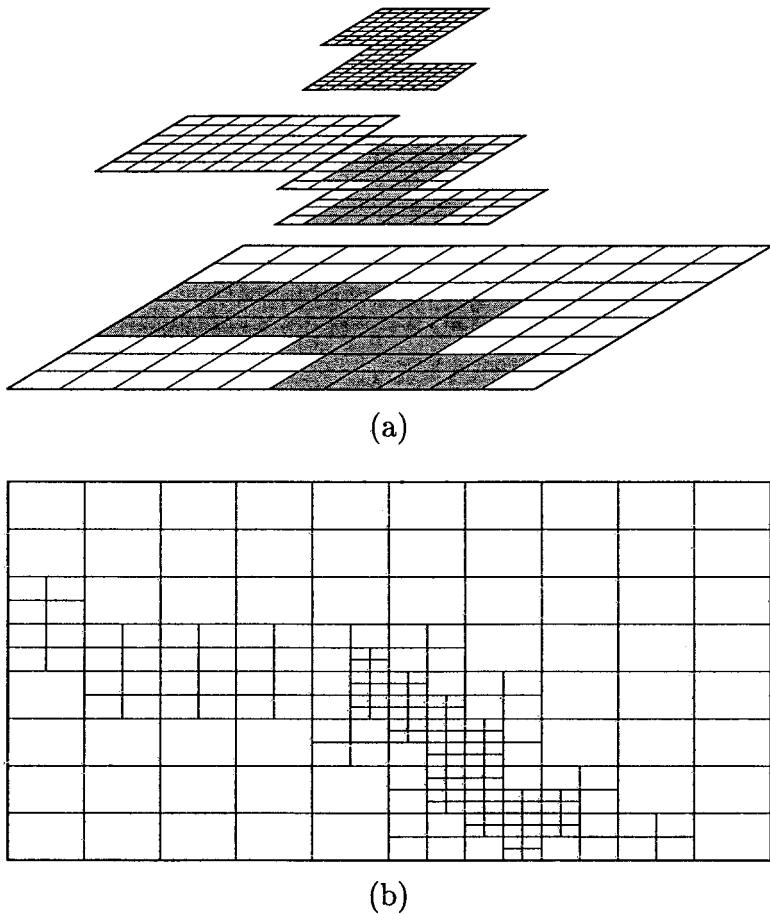


FIG. 23.6. An example of a composite mesh made up of regular meshes illustrating (a) the hierarchy of regular meshes and (b) the composed “unstructured mesh” (<http://www.llnl.gov/casc/samrai>).

PDEs and implement them using the mesh underlying the simulation. Preliminary details on this approach are outlined in a later section.

Other Applications of Scientific Data Mining

In addition to the application areas already discussed, there are several other areas of science and engineering in which data mining is being applied or has the potential of being a useful tool in the analysis of data. Examples of such areas include bioinformatics (both genomics and proteomics) (Venter et al., 2001; International Human Genome Sequencing Consortium, 2001), which is discussed elsewhere in this book; combinatorial chemistry (Indiana University, 2000); computer vision (Carnegie Mellon University, 2000); high-energy physics (STAR Project Web page, 2002); video and audio mining (Video Mining Web Page, 2002), mining of distributed sensor data, and so forth.

COMMON CHALLENGES IN MINING SCIENTIFIC DATA

Given the foregoing descriptions of practical data mining applications in science and engineering, certain common themes start to emerge:

- Scientific data is often in the form of images, meshes, or sensor outputs, and a substantial amount of processing is needed to convert this low-level data into higher-level features prior to pattern recognition. For this reason, I prefer the definition of data mining that treats it as the “overall process of extracting high-level information from low-level data” (Simoudis, 1996), rather than the one that sees it as a single step in the multistep process of knowledge discovery in databases (KDD) (Fayyad, Piatetsky-Shapiro, Smyth, & Uthurusamy, 1996a, p. 6). In the latter definition, KDD is defined as the “non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable pattern in data,” with the data mining component seen as the means by which patterns are extracted and enumerated from the data.

There are several other reasons I prefer the definition of data mining that includes both data preprocessing and pattern recognition instead of the one that focuses on pattern recognition alone. First, scientific data are rarely in databases, making the KDD terminology inapplicable. Second, the preprocessing step takes up much of the time in the end-to-end process of analysis and is crucial to the success of the pattern recognition step. In addition, once the features have been extracted, they have to be refined until the desired results are obtained with the pattern recognition algorithms. A close coupling between the data preprocessing part and the pattern recognition part enables this refinement. Further, in scientific data mining an active collaboration with the domain scientists is essential during all stages of the data mining process. Because the data preprocessing is typically domain and problem dependent, such a collaboration often results automatically when the preprocessing and pattern recognition parts are closely linked.

The important role played by the proper formulation of the problem and processing of the data has been observed by several data miners. For example, Burl et al. (1989) in their work on cataloguing volcanoes on Venus comment that “perhaps as little as 10% of the effort was spent on classification aspects of the problem,” (p. 190) and Brodley and Smyth (1995) observe that “in practical applications, it is often the data and the human issues which ultimately dictate success or failure of a project rather than algorithmic and model issues” (p. 12).

- Scientific data frequently has both a spatial and a temporal aspect. This is the case whether we are looking at two-dimensional images of a scene evolving over time, the

three-dimensional image of a human brain before and after treatment for a tumor, or the evolution of a two- or three-dimensional complex physical phenomena being simulated on a massively parallel computer. This temporal aspect implies that we are frequently interested in changes of the variables being observed over time.

- There is often a desire by scientists to exploit data from different sources. For example,

astronomers frequently cross-validate what they find in one survey with information from other surveys. This will become more feasible with the creation of the National Virtual Observatory (NVO Web page, 2000). In addition, in many cases the problem being addressed is one of change detection, for example, identifying an object such as an asteroid that appears in one astronomical image but not another, or understanding the effects of a volcanic eruption on the vegetation in an area. All these require that the data from different sources must somehow be “registered” before any comparison or data fusion can be done.

- Scientific data can be noisy with missing values. Typically, the noise can cause problems in the analysis of the data, and effects of the noise must be reduced before any analysis can be done.

In some cases data may be missing either due to sensor problems (a part of a sensor, or one sensor among many, was inoperable for part of the period during which the data was collected), or because data was not collected over a period of time due to extraneous reasons (e.g., the percentage of the earth’s surface over which surface temperatures were measured is relatively small during the years of the two world wars). When it is necessary to assign values to such missing data, it must be done with careful justification, and a simple averaging or interpolation, which is the technique often used for commercial data, might not be appropriate.

- The data can be high-dimensional, with each object represented by a large number of features or attributes. For example, the high-energy physics data generated during the STAR experiment (STAR Project Web page, 2002) have 100–200 features describing each event.

Although this number is not as large as, say, the number of features in text mining, it is still large enough to warrant a reduction in the number of features prior to the application of many pattern recognition algorithms.

- Scientific data is often compressed to make it easier to store and transmit. This is true of fingerprint data, some astronomical data, and much of the large-scale simulation data, with wavelet based compression being the technique of choice. This raises the interesting question of whether one can mine the compressed data as is, without uncompressing it.

• In classification problems in science and engineering domains, there is often a scarcity of good labeled data. This is because such data is usually generated manually. In contrast, in commercial data a labeled set might be generated historically, for example, in the case of targeted marketing or customer churn. In addition, scientists may often disagree on a label or, when labeling several images, be inconsistent in their labeling due to the drift associated with such visually intensive tasks.

- For the patterns found in scientific data to be useful eventually, it is important that they be validated by the scientists. This validation must be done keeping in mind the uncertainty inherently present in the raw data and that introduced during the entire process of data mining.

POTENTIAL SOLUTIONS TO SOME COMMON PROBLEMS

Based on the previous description of the application areas and the challenges in mining scientific data, we need to consider data mining as the end-to-end process of starting with the raw data in the form of meshes, images, or other sensor outputs and finding useful information in this data. One approach, among many, is the one I have taken in the Sapphire project

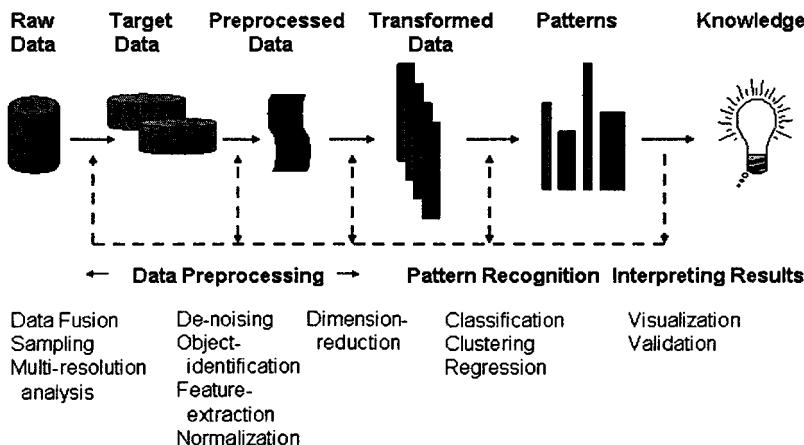


FIG. 23.7. Data mining: An iterative and interactive process.

(<http://www.llnl.gov/casc/sapphire>). Figure 23.7 describes the data flow diagram we use for mining scientific data, wherein we clearly identify the phases of data preprocessing, pattern recognition, and interpretation of the results. If the raw data are very large, we could use sampling and work with fewer instances or use multiresolution techniques and work with data at a coarser resolution. Such a multiresolution analysis could also help us identify patterns at different scales in the data. In addition, if the data were obtained from different sensors, data fusion might be required. Next, noise is removed and objects in the data are identified. How we define an object in the data depends very much on the data and the problem being solved. Next, relevant features that describe the objects of interest are extracted from the data. These features must be not only robust, that is, insensitive to small changes in the data, but also invariant to scaling, translation, and rotation. At the end of this step we have a vector of features for each object or data item. These vectors are normalized to remove any skew resulting from possibly different units used for the different features. Next, depending on the problem and the data, we may need to reduce the number of features using dimension reduction techniques such as principal component analysis (PCA) or its nonlinear variants. After this preprocessing the data is ready for the detection of patterns, through the use of traditional techniques such as classification, clustering, regression, and so forth. The patterns obtained are then displayed to the domain scientist for validation. As the application of data mining to scientific data becomes an established field, increasing attention will be paid to this last step, and statistical techniques for validating the information found in the data will become more important.

The data mining process is iterative and interactive; any step may lead to a refinement of one or more of the previous steps. To ensure the success of data mining, the domain scientist should be actively involved in all stages, starting from the initial description of the data and the problem and going through the extraction of potentially relevant features, the identification of the training set (where necessary), and the validation of the results.

It must be noted that several of the tasks in scientific data mining are independent of the data or the problem (e.g., the pattern recognition steps), whereas others (such as the data preprocessing and the validation) are more specific to the domain. However, for scientific data in the form of signals, images, or meshes there is ample opportunity for reuse of algorithms and software. For example, although wavelet based de-noising techniques can be used in general to de-noise signals and images, the particular type of wavelet used might depend on the statistical

properties of the data and noise, which in turn is determined by the sensors that were used to collect the data. Similarly, although decision trees may be used in the classification of both astronomy and simulation data, it may turn out that neural networks are better at classification for a particular problem in astronomy. In addition, techniques that may have been developed for one kind of data, such as segmentation for images, can be applied to a similar task in another kind of data, such as segmentation in mesh data. However, tasks such as reading, writing, and displaying data are typically very specific to the format used for the data.

Next, I discuss the approaches commonly used to address the challenges described in the previous section. Some of the methods have been in use a long time, whereas others have been the subject of active research in the last decade. The latter techniques are typically more complex and have evolved in response to the growing diversity and complexity in the data obtained from different sources, their size, and the application of data mining for domains in which accuracy is a major issue. Discussing all of these techniques in detail is beyond the scope of this chapter; instead, I will briefly describe some of the simpler techniques and provide references for the more complex ones. Needless to say, this description is not exhaustive; it is not possible to cover all the techniques developed to solve a problem in the many different domains and application areas in which the problem occurs. However, the information in this section should provide the reader with a starting point.

In the following I focus on the data preprocessing part for two reasons. First, it is domain specific, and therefore the techniques described here are specific to science and engineering data, which is the focus of this chapter. Second, the pattern recognition techniques that are applicable across several domains have been covered elsewhere in this book.

Data Registration

Registration is the process of alignment of data, such as images, to relate the information in one image to information in another. It is typically used in change detection, when data has been acquired over time and must first be aligned before any changes can be detected, and in data fusion, when data from different sources must be fused. Registration is typically a two-step process—first, characteristic features in the two images are identified, and then, a transform is computed that brings the features into alignment. Registration can be a difficult task, given the diversity of sensors that generate data, the presence of noise in the data, the absence of fiducial markers, the sheer volume of data even from a single sensor, and the technical challenges associated with the identification of characteristic features and optimal transforms.

An excellent survey of registration techniques has been provided in Brown (1992). Registration can be broadly viewed as the combination of four components:

- A feature space, which is the set of characteristic features used to perform the matching, for instance pixels, curves, edges, and so forth. These must be extracted from the data in a robust manner.
- A search space, which is the set of potential transforms that can establish the correspondence between the images, for example, rotation, translation, scaling, polynomials, and so forth.
- A search strategy, which determines which transformations are computed and evaluated; for example, this could be an exhaustive search to evaluate all possible rotations and translations, or it could be a selective search done using evolutionary algorithms.
- A similarity metric, which evaluates the match between images for a given transformation in the search space.

Several new approaches for registration have been proposed in fields such as remote sensing and medical imaging, where accuracy and turnaround time have become important issues for both monomodality (only one type of image) and multimodality (different types of images such as X-rays and computed tomography scans) data. For example, wavelets and other multiresolution techniques are being used to register remote sensing data (Le Moigne, 1994), genetic algorithms are being used to perform a more efficient search (Mandava, Fitzpatrick, & Pickens, 1989), and nonlinear transformations are being used in medical imaging (Thompson et al., 2000). Other excellent overviews of registration techniques in the context of medical images are given in the survey papers by Maurer and Fitzpatrick (1993) and by Maintz and Viergever (1998).

De-Noising Data

Reducing noise in the data is often a key step in scientific data mining. This noise could be the result of the data acquisition process or due to natural phenomena such as atmospheric turbulence. The techniques used for reducing noise depend on what is considered to be noise in the data, and what are the statistical properties of the noise. For example, as mentioned previously, a remote sensing application might consider clouds in the image to be noise, whereas a different application that uses similar data for the classification of clouds would consider the clouds to be the signal. In addition, depending on the sensor different statistical models can be used for the noise, such as additive Gaussian noise or multiplicative speckle noise.

The traditional approaches to reducing noise include the use of simple spatial filters, as given in Fig. 23.8. Convolution of an image with such filters smooths the noise; more smoothing is obtained with a wider filter, but the entire image, including the edges, are smoothed as well. Other filters, such as the minimum mean squared error filter (Umbaugh, 1998), determine the amount of smoothing based on the variance of the image in a small window, thus reducing the smoothing near the edges. As a result, the areas of the image that are homogeneous are de-noised more than the area near the edges, where the image often remains as noisy as before. Another simple technique based on domain knowledge is to de-noise using a simple thresholding (Fodor and Kamath, 2001b). However, this technique can be applied only in very specific cases because domain knowledge is required to identify the threshold and to ensure that what remains is the required signal.

To overcome problems such as smoothing and moving of the edges associated with these traditional de-noising techniques, several researchers have proposed alternative approaches, including:

- Wavelet based thresholding. Here, a forward wavelet transform first decomposes the image into its high- and low-frequency components, also referred to as the detail and smooth coefficients. Next, an appropriate threshold is calculated and applied to the detail coefficients. An inverse wavelet transform on the thresholded coefficients results in the de-noised image.

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \frac{1}{32} \begin{bmatrix} 1 & 4 & 1 \\ 4 & 12 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

FIG. 23.8. Simple filters for smoothing images: A 3×3 mean filter and a 3×3 Gaussian filter.

The basic idea is that the noise in an image is usually at a higher frequency than the background but not as high as the edges in the image. The thresholding of the detail coefficients reduces the noise while preserving the edges. There are several different ways in which the thresholds can be obtained and applied; a survey is available in Fodor and Kamath (2001a). However, recent research has shown that wavelet thresholding might not be the best way of representing the de-noising done by the human visual system. This is because in two-dimensions interesting phenomena is organized along lines and curves, for which wavelets are poorly adapted. Various solution approaches based on transforms such as curvelets, ridgelets, wedgelets, and beamlets have been proposed to solve this problem. For those interested in exploring this rapidly evolving field, the Web page listed in Donoho (2002) is worthy of exploration.

- Nonlinear diffusion based de-noising. This approach exploits the connection between the diffusion equation and the use of the Gaussian filter in multiscale image analysis. Convolving an image with a Gaussian filter K_σ :

$$K_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{|x|^2 + |y|^2}{2\sigma^2}\right) \quad (1)$$

with standard deviation σ , is equivalent to the solution of the diffusion equation in two dimensions

$$\frac{\partial}{\partial t} I(x, y, t) = \nabla^2 I(x, y, t) = \frac{\partial^2}{\partial x^2} I(x, y, t) + \frac{\partial^2}{\partial y^2} I(x, y, t) \quad (2)$$

where $I(x, y, t)$ is the two-dimensional image $I(x, y)$ at time $t = 0.5\sigma^2$, with initial conditions $I(x, y, 0) = I_0(x, y)$, where I_0 is the original image. We can write Equation 2 in a general form as

$$\begin{aligned} \frac{\partial}{\partial t} I(x, y, t) &= \nabla \cdot (c(x, y, t) \nabla I(x, y, t)) \\ I(x, y, 0) &= I_0(x, y) \end{aligned} \quad (3)$$

where $c(x, y, t)$ is the diffusivity of the equation, ∇ is the gradient operator, and $\nabla \cdot$ is the divergence operator. By suitably controlling the function $c(x, y, t)$, we can control the diffusivity at different points in the image as it evolves with time, thus reducing the noise without smoothing the edges in the image. Since the original article of Perona and Malik (1990) much work has been done to extend this basic idea to include several different diffusivity functions, several ways of modifying the diffusivity as the image evolves, and several techniques to solve the equation numerically in a fast yet accurate manner (Weeratunga & Kamath, 2002; Weickert, 1998). One of the interesting aspects of the diffusion based approach is that it can be used in the case of mesh data from simulations by using the underlying mesh as the grid on which to solve the partial differential equation (PDE) in Equation 3. Such applications of PDE-based techniques for use in image analysis are an active research area (Malladi & Sethian, 1996; Sapiro, 2001).

Object Identification

Once the data has been de-noised, the next step is to identify the objects of interest in the data. This is a difficult problem for several reasons. The processing steps prior to object identification, such as de-noising, may have moved the edges of the objects or smoothed them so that it is difficult to identify the boundary. In addition, in some data sets, especially those

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

FIG. 23.9. Simple filters for detecting edges in images: A 3×3 Laplacian filter and the two Sobel filters for the x and y direction, respectively.

containing natural objects, there is tremendous variability in the objects themselves, making their separation from the background very difficult. For example, in the images from the FIRST astronomical survey in Fig. 23.2, the two galaxies have very different shapes, and the boundaries of the galaxies are complex curves rather than straight lines, with some parts of the galaxy not clearly demarcated from the background. The situation is less difficult in the case of man-made objects, such as houses, streets, and semiconductor masks, where the object boundaries are often straight lines. The problem of object identification is made even harder by the fact that image data often varies in quality, making it difficult to identify a continuous boundary for the object.

The simplest approach to identifying edges or boundaries in data is to use convolution with filters, such as the ones shown in Fig. 23.9. These filters exploit the fact that an edge is a change in the intensity of the neighboring pixels. By appropriately choosing the filter values, it is possible to determine the location and direction of an edge pixel. Once an edge (or parts of an edge) has been found, further processing is usually required to complete the edge, remove “edge” pixels in the image that are not really part of an edge, and perhaps “thin” the edge so it is just a couple of pixels wide. It is also possible to detect an object by focusing on the object itself instead of the edges of the object. These image segmentation techniques (Sonka, Hlavac, & Boyle, 1999; Umbaugh, 1998) can be applied using either a top-down or bottom-up approach. In the former the image is split sequentially into regions until each region can be considered homogeneous based on some criterion. Next, neighboring regions are combined as appropriate to form larger regions, again ensuring that the homogeneity criterion is met. In the bottom-up approach each pixel is initially considered to be a region, and the region is grown by combining neighboring pixels that satisfy the homogeneity criterion. Segmentation can also be done using the clustering techniques traditionally used in pattern recognition.

Although the techniques mentioned are relatively simple to apply, they are not very robust and accurate, especially when applied to images that are noisy, with a lot of variability within an image and across images. As a result, identifying an object in an image is still an unsolved problem. Recent approaches that address some of the existing problems include those based on graph searches (Sonka et al., 1999), level set methods (Sethian, 1999), snakes or active contours (Blake & Isard, 1999), and combinations with evolutionary algorithms and other computational intelligence techniques to make the algorithms more adaptive (Bhanu & Lee, 1994; Perry & Wong, 2001). Some of these techniques such as active contours and level sets can be applied to data resulting from mesh based simulations.

Dimensionality Reduction

Once the objects in the data have been identified and features relevant to the problem have been extracted, we have a data item-by-feature matrix, in which for each data item or object we have a set of features describing each object. As mentioned earlier, the number of features can be rather high and must be reduced prior to the application of the pattern recognition algorithms. The traditional technique used for this purpose is PCA (Jackson, 1991), in which

the data typically is projected onto the space spanned by the largest principal components (i.e., the ones that explain most of the variance). However, in some scientific domains the scientists want the features in this reduced space to have some physical interpretation instead of being a linear combination of the original features, which may not correspond to any physical quantity. To accomplish this, one approach would be that suggested by Mardia, Kent, and Bibby (1995) who used the principal components to eliminate the unimportant variables. Starting with the eigenvector corresponding to the smallest eigenvalue of the covariance matrix, they discarded the variable with the largest coefficient in absolute value. This process is repeated for the eigenvector corresponding to the next smallest eigenvalue, considering only the variables that have not been discarded so far. This approach results in the variables in reduced subspace being a subset of the original variables.

In some scientific domains it is not clear if PCA is the right dimension reduction technique to use, because it might not provide a satisfactory representation given its linearity and orthogonality. In such cases alternative techniques such as independent component analysis, projection pursuit, nonlinear PCA, or principal curves (Hyvarinen & Karhunen, & Oja, 2001; Friedman, 1987; Hastie & Stuetzle, 1989) might be more appropriate.

Generating a Good Training Set

As mentioned earlier, a common problem in classification for scientific data sets is the lack of a good training set. There are two aspects to this problem—the small size of the training set and the often poor quality of the training set. It is possible to address the former problem by using the classifier created from a small training set to classify the unlabeled examples, have some of these examples validated by the scientists, and then use the validated ones to enhance the training set. Another option is to use clustering techniques to identify new training examples. The new approach of active learning can also be used in this context.

The second problem of the quality of the training set is harder to address. To avoid the problems associated with drift in the labels, one could arrange to have the scientists label only a few examples at a time and maybe use the same examples at different times to ensure consistency. However, the problem of disagreement between scientists is harder to address as this typically occurs in the hard-to-classify cases. In our experience we found that the scientists do not object to false positives being included, but do not like false negatives. In other words, they do not like to miss any examples of interest, but do not have any problems if they have to do a visual inspection to remove any “uninteresting” examples. A partial, and somewhat unsatisfactory, solution to this problem is therefore to classify the hard-to-classify cases as positives.

Software for Scientific Data Mining

A few words on the software used in scientific data mining might be in order. For the pattern recognition tasks of classification, clustering, and so forth, one can use any software that is appropriate to the size and type of problem; several examples are listed in the chapters that describe such techniques. For data preprocessing the general trend thus far has been the development of domain-specific software. For example, the astronomy community has its set of software packages (such as FITS, or flexible image transport system [National Radio Astronomy Observatory, 2002b] or AIPS—Astronomical Image Processing System [National Radio Astronomy Observatory, 2002a]) for reading, writing, and displaying data, as well as for application of various image processing operations. Because the preprocessing steps are

very closely tied to the domain, it would make the most sense for a data miner to work with the software developed by scientists within the domain of interest.

It must also be noted that when scientific data is available in the form of images, these are usually real valued pixels, not gray-scale. This is because the pixels correspond to some physical quantity such as the radio intensity. As a result, application of many of the standard image processing software would require quantization of the original data into 8 bits for gray-scale, a conversion that scientists would like to avoid until perhaps the very end of the data preprocessing step. However, floating point versions of the image processing algorithms can be applied to such data.

SUMMARY

In this chapter, through illustrative examples, I have given an introduction to the subject of mining science and engineering data sets. I identified several common themes across these diverse applications and provided solutions to some of the challenges facing a data miner working with scientific data. These solutions are only an introductory first step, and further information is provided in the references. Even though data mining and the techniques that comprise it have been applied to scientific data for a long time, there are still several areas in which much needs to be done. For example, image processing techniques for segmenting an image have been in use for many decades, but segmentation techniques that are robust and accurate, as well as adaptive to changes in the images, are still an active area of research. As scientific data sets become larger and more complex and scientific discoveries result in potential payoffs and a better understanding of the world around us, it is clear that scientific data mining will continue to be a field rich in challenges as well as opportunities.

Finally, there are several resources for the reader interested in mining scientific data, including the workshop series on Mining Scientific Data sets (<http://www.ahpcrc.umn.edu/conferences>), the Web page for the IPAM program on Mathematical Challenges in Scientific Data Mining (IPAM, 2002), the symposia on the Interface Between Computing Science and Statistics, and several conferences organized by the International Society for Optical Engineering (www.spie.org), the Association for Computing Machinery (www.acm.org), the Society for Industrial and Applied Mathematics (www.siam.org), and the Institute for Electrical and Electronic Engineers (www.ieee.org), as well as the journals supported by these organizations.

REFERENCES

- Becker, R. H., White, R., & Helfand, D. (1995). The FIRST survey: Faint Images of the Radio Sky at Twenty-cm. *Astrophysical Journal*, 450, 559.
- Bhanu, B., & Lee, S. (1994). *Genetic learning for adaptive image segmentation*. Boston: Kluwer Academic Publishers.
- Blake, A., & Isard, M. (1999). *Active contours: The application of techniques from graphics, vision, control theory and statistics to visual tracking of shapes*. New York: Springer.
- Brislawns, C. (2002). The FBI fingerprint image compression standard. Retrieved January, 2002 from <http://www.c3.lanl.gov/~brislawns/FBI/FBI.html>
- Brodley, C., & Smyth, P. (1995). The process of applying machine learning algorithms. In *Workshop on applying machine learning in practice, IMLC* (pp. 7–13). Retrieved January, 2002 from <http://citeseer.nj.nec.com/722.html>
- Brown, L. G. (1992). A survey of image registration techniques. *ACM Computing Surveys*, 24, 325–376.
- Brown, D., & Brence, J. (2001). Discovering corrosion relationships in eddy current non-destructive test data. In *Proceedings, Fourth Workshop on Mining Scientific Data Sets* (pp. 49–55). New York: ACM Press.
- Burl, M., Asker, L., Smyth, P., Fayyad, U., Perona, P., Crumpler, L., & Aubele, J. (1998). Learning to recognize volcanoes on Venus. *Machine Learning*, 30, 165–195.

- Canada Center for Remote Sensing (2001). Fundamentals of remote sensing. Retrieved January, 2002 from <http://www.ccrs.nrcan.gc.ca/ccrs/eduref/tutorial/tutore.html>
- Candela, G., & Chellappa, R. (1993). *Comparative performance of classification methods for fingerprints* (Technical Report NISTIR 5163). Gaithersburg, MD: National Institute of Standards and Technology.
- Carnegie Mellon University. (2000). The computer vision homepage at Carnegie Mellon University. Retrieved January, 2002 from <http://www.cs.cmu.edu/afs/cs/project/cil/www/vision.html>
- Chedid, R., & Najjar, N. (1996). Automatic finite-element mesh generation using artificial neural networks—part I : Prediction of mesh density. *IEEE Transactions on Magnetics*, 32, 5173–5178.
- Donoho, D. (2002). Technical reports by David Donoho and co-authors. Retrieved January, 2002 from <http://www-stat.stanford.edu/donoho/Reports/index.html>
- Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances in knowledge discovery and data mining*. Cambridge, MA: MIT Press.
- Fayyad, U., Smyth, P., Burl, M., & Perona, P. (1996). Learning to catalog science images. In *Early Visual Learning* (pp. 237–268). Oxford, U.K.: Oxford University Press.
- Fodor, I., & Kamath, C. (2001a). *On denoising images using wavelet-based statistical techniques* (Technical report UCRL-JC-142357, Lawrence Livermore National Laboratory) Retrieved January, 2002 from <http://www.llnl.gov/casc/sapphire/>
- Fodor, I. K., & Kamath, C. (2001b). A comparison of denoising techniques for FIRST images. In *Proceedings, Third Workshop on Mining Scientific Data Sets* (pp. 13–20). Philadelphia: SIAM.
- Fodor, I., & Kamath, C. (in press). Dimension reduction techniques and the classification of bent-double galaxies. *Computational Statistics and Data Analysis*.
- Fodor, I. K., Cantú-Paz, E., Kamath, C., & Tang, N. (2000). Finding bent-double radio galaxies: A case study in data mining. In *Interface: Computer Science and Statistics* (Vol. 32, pp. 37–47).
- Friedman, J. H. (1987). Exploratory projection pursuit. *Journal of the American Statistical Association*, 82, 249–266.
- Frischholz, R. (2002). Face detection. Retrieved January, 2002 from <http://home.t-online.de/home/Robert.Frischholz/face.htm>
- Grossman, R., Kamath, C., Kegelmeyer, P., Kumar, V., & Namburu, R. (2001). *Data mining for science and engineering applications*. Boston: Kluwer Academic Publishers.
- Hastie, T., & Stuetzle, W. (1989). Principal curves. *Journal of the American Statistical Association*, 84, 502–516.
- HERMES Project. (2000). Hermes: Bridge inspection technology for the 21st century. Retrieved January, 2002 from <http://lasers.llnl.gov/lasers/hermes>
- Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis*. New York: Wiley.
- Indiana University. (2000). Chemical informatics Web page. Retrieved January, 2002 from [http://www.indiana.edu/~cheminfo/informatics/ciform_whatis.html](http://www.indiana.edu/~cheminfo/informatics/cinform_whatis.html)
- Institute for Pure and Applied Mathematics (IPAM). (2002). Mathematical challenges in scientific data mining. Retrieved January, 2002 from <http://www.ipam.ucla.edu/programs/sdm2002>
- International Human Genome Sequencing Consortium (2001). Initial sequencing and analysis of the human genome. *Nature*, 409, 860–921.
- International Society for Optical Engineering (SPIE). (2000a). *Automated target recognition: Selected SPIE papers on CD-ROM* (Vol. 6). Bellingham, WA: SPIE Press.
- International Society for Optical Engineering (SPIE). (2000b). *Medical image processing: Selected SPIE papers on CD-ROM* (Vol. 13). Bellingham, WA: SPIE Press.
- Jackson, J. E. (1991). *A user's guide to principal components*. New York: Wiley.
- Jarvis, J., & Tyson, J. (1981). FOCAS: Faint object classification and analysis system. *Astronomical Journal*, 86, 476–495.
- Kamath, C., Cantú-Paz, E., Fodor, I., & Tang, N. (2001). Searching for bent-double galaxies in the first survey. In R. Grossman, C. Kamath, W. Kegelmeyer, V. Kumar, & R. Namburu (Eds.), *Data mining for scientific and engineering applications* (pp. 95–114). Boston: Kluwer Academic Publishers.
- Kruizinga, P. (2002). *Face recognition Web page*. Retrieved January, 2002 from <http://www.cs.rug.nl/users/peterkr/FACE/face.html>
- Kubat, M., Holte, R. C., & Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30, 195–215.
- Le Moigne, J. (1994). Parallel registration of multi-sensor remotely sensed imagery using wavelet coefficients. In *Proceedings of the SPIE OE/Aerospace Sensing, Wavelet Applications Conference*. Bellingham, WA: SPIE Press.
- Liu, W., Gopal, S., & Woodcock, C. (2001). Spatial data mining for classification visualization and interpretation with ARTMAP neural network. In R. Grossman, C. Kamath, W. Kegelmeyer, V. Kumar, & R. Namburu (Eds.), *Data mining for scientific and engineering applications* (pp. 201–221). Boston: Kluwer Academic Publishers.
- Lewin, D. I. (2000). Getting clinical about neural networks. *IEEE Intelligent Systems*, 50, 2–7.

- Maintz, J., & Viergever, M. (1998). A survey of medical image registration. *Medical Image Analysis*, 2, 1–36.
- Malladi, R., & Sethian, J. (1996). A unified approach to noise removal, image enhancement, and shape recovery. *IEEE Transactions on Image Processing*, 5, 1154–1168.
- Mandava, V., Fitzpatrick, J., & Pickens, D. (1989). Adaptive search space scaling in digital image registration. *IEEE Transactions on Medical Imaging*, 8, 251–262.
- Marchette, D. J. (2001). *Computer intrusion detection and network monitoring: A statistical viewpoint*. Berlin: Springer-Verlag.
- Mardia, K. V., Kent, J., & Bibby, J. (1995). *Multivariate analysis*. London: Academic Press.
- Marusic, I. (2000). *Dynamic feature extraction and data mining for the analysis of turbulent flows: Project web page*. Retrieved January, 2002 from http://www.aem.umn.edu/research/marusic_nsf_dyn.html
- Marusic, I., Candler, G. V., Interrante, V., Subbareddy, P. K., & Moss, A. (2001). Real time feature extraction for the analysis of turbulent flows. In R. Grossman, C. Kamath, W. Kegelmeyer, V. Kumar, & R. Namburu (Eds.), *Data mining for scientific and engineering applications* (pp. 223–256). Boston: Kluwer Academic Publishing.
- Maurer, C. R., & Fitzpatrick, J. M. (1993). A review of medical image registration. In R. J. Maciunas, (Ed.), *Interactive image-guided neurosurgery* (pp. 17–44). New York: American Association of Neurological Surgeons.
- Nair, U. S., Rushing, J. F., Ramachandran, R. P., Kuo, K. S., Welch, R. M., & Graves, S. J. (1999). Detection of cumulus cloud fields in satellite imagery. In *Earth Observing Systems IV, Proceedings of SPIE* (Vol. 3750). Bellingham, WA: SPIE Press.
- National Aeronautics and Space Administration (NASA). (1999a). *NASA workshop on data fusion and data mining*. Presented at NASA Ames Research Center, Moffett Field, CA, July 9–10, 1999.
- National Aeronautics and Space Administration (NASA). (1999b). *NASA workshop on issues in the application of data mining to scientific data*. Presented at University of Alabama in Huntsville, October 19–21, 1999.
- National Radio Astronomy Observatory. (2002a). Astronomical image processing system. Retrieved January, 2002 from <http://www.aoc.nrao.edu/aips/>
- National Radio Astronomy Observatory. (2002b). Flexible image transport system. Retrieved January, 2002 from <http://www.cv.nrao.edu/fits/FITS.html>
- National Virtual Observatory. (2000). National Virtual Observatory. Retrieved January, 2002 from <http://www.srl.caltech.edu/nvo>
- Odewahn, S., Stockwell, E., Pennington, R., Humphreys, R., & Zumach, W. (1992). Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103, 318–331.
- Perona, P., & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, 629–639.
- Perry, S., & H.-S. Wong, L. G. (2001). *Adaptive image processing: A computational intelligence perspective*. Bellingham, WA: SPIE Press and CRC Press.
- Ramachandran, R., Conover, H., Graves, S., & Keiser, K. (2000). Challenges and solutions to mining earth science data. In *Data mining and knowledge discovery, SPIE Proceedings*, 4057, 259–264. Bellingham, WA: SPIE Press.
- Sandhu, S. S., Kanapady, R., Tamma, K. K., Kamath, C., & Kumar, V. (2001). Damage prediction and estimation in structural mechanics based on data mining. In *Proceedings, Fourth Workshop on Mining Scientific Data Sets* (pp. 56–63). New York: ACM Press.
- Sapiro, G. (2001). *Geometric partial differential equations and image analysis*. Cambridge, U.K.: Cambridge University Press.
- Sethian, J. (1999). *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision and materials science*. Cambridge, U.K.: Cambridge University Press.
- Short, N. (2002). The remote sensing tutorial. Retrieved January, 2002 from <http://rst.gsfc.nasa.gov/Front/overview.html>
- Short, N., Cromp, R., Campbell, W., Tilton, J., LeMoigne, J., Fekete, G., Netanyahu, N., Ligon, W., & Wichmann, K. (1995). Mission to planet earth: AI views the world (special issue). *IEEE Expert/Intelligent Systems and Their Applications*, 10, 24–34.
- Simoudis, E. (1996). Reality check for data mining. *IEEE Expert/Intelligent Systems and Their Applications*, 11, 26–33.
- Sonka, M., Hlavac, V., & Boyle, R. (1999). *Image processing, analysis, and machine vision*. PWS Publishing.
- STAR Project Web page (2002). The STAR collaboration. Retrieved January, 2002 from <http://www.rhic.bnl.gov/STAR>
- Stolorz, P., & Dean, C. (1996). Quakefinder: A scalable data mining system for detecting earthquakes from space. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 208–213). Menlo Park, CA: AAAI Press.
- Storrie-Lombardi, M., Lahav, O., Sodre, L., & Storrie-Lombardi, L. (1992). Morphological classification of galaxies by artificial neural networks. *Monthly Notices of Royal Astronomical Society*, 259, 8–12.

- Thompson, P., Mega, M., Narr, K., Sowell, E., Blanton, R., & Toga, A. (2000). Brain image analysis and atlas construction. In M. Fitzpatrick & M. Sonka (Eds.), *SPIE handbook of medical image processing and analysis*. Bellingham, WA: SPIE Press.
- Umbaugh, S. (1998). *Computer vision and image processing: A practical approach using CVIPtools*. Englewood Cliffs, NJ: Prentice Hall.
- Vemuri, V., & Karplus, W. J. (1981). *Digital computer treatment of partial differential equations*. Englewood Cliffs, NJ: Prentice Hall.
- Venter, N. C., et al. (2001). The sequence of the human genome. *Science*, 291, 1304–1351.
- Video Mining Web page. (2002). DIMACS Workshop on Video Mining, 2002. Retrieved January, 2002 from <http://dimacs.rutgers.edu/Workshops/Video/>
- Weeratunga, S. K., & Kamath, C. (2002). PDE-based non-linear diffusion techniques for denoising scientific and industrial images: An empirical study. In *Image processing: Algorithms and systems*. Bellingham, WA: SPIE Press.
- Weickert, J. (1998). *Anisotropic diffusion in image processing*. Stuttgart, Germany: Teubner–Verlag.
- White, R. L., Becker, R., Helfand, D., & Gregg, M. (1997). A catalog of 1.4 GHz radio sources from the FIRST survey. *Astrophysical Journal*, 475, 479.

24

Mining Data in Bioinformatics

Mohammed J. Zaki

Rensselaer Polytechnic Institute

Introduction	574
Background	574
Basic Molecular Biology	574
Mining Methods in Protein Structure Prediction	575
Mining Protein Contact Maps	577
Classifying Contacts Versus Noncontacts	578
Mining Methodology	578
How Much Information Is There in Amino Acids Alone?	581
Using Local Structures for Contact Prediction	582
Characterizing Physical, Protein-Like Contact Maps	587
Generating a Database of Protein-Like Structures	588
Mining Dense Patterns in Contact Maps	589
Pruning and Integration	590
Experimental Results	591
Future Directions for Contact Map Mining	593
Heuristic Rules for “Physicality”	593
Rules for Pathways in Contact Map Space	594
Summary	595
References	596

This work was supported in part by NSF CAREER Award IIS-0092978, NSF Next Generation Software Program grant EIA-0103708, and DOE Early Career PI Award DE-FG02-02ER25538.

INTRODUCTION

Bioinformatics is the science of storing, extracting, organizing, analyzing, interpreting, and utilizing information from biological sequences and molecules. It mainly has been fueled by advances in DNA sequencing and genome mapping techniques. The Human Genome Project has resulted in rapidly growing databases of genetic sequences, whereas the Structural Genomics Initiative is doing the same for the protein structure database. New techniques are needed to analyze, manage, and discover sequence, structure, and functional patterns or models from these large sequence and structural databases. High-performance data analysis algorithms also are becoming central to this task.

Bioinformatics provides opportunities for developing novel data analysis methods. Some of the grand challenges in bioinformatics include protein structure prediction, homology search, multiple alignment and phylogeny construction, genomic sequence analysis and gene finding, as well as applications in gene expression data analysis, drug discovery in pharmaceutical industry, and so forth. For example, in protein structure prediction one is interested in determining the secondary, tertiary, and quaternary structure of proteins, given their amino acid sequence. Homology search aims at detecting increasingly distant homologues, that is, proteins related by evolution from a common ancestor. Multiple alignment and phylogenetic tree construction are interrelated problems. Multiple alignment aims at aligning a whole set of sequences to determine which subsequences are conserved. This works best when a phylogenetic tree of related proteins is available. Finally, gene finding aims at locating the genes in a DNA sequence.

In this chapter I focus on how data mining methods can be utilized for protein structure prediction when one is interested in determining three-dimensional (3-D) structure of proteins given their amino acid sequence. I begin with a brief introduction to concepts of molecular biology and then focus on the protein folding problem and various mining techniques applied in this domain.

BACKGROUND

Basic Molecular Biology

There are three main building blocks of biological systems formed from nucleic or amino acids. *DNA* (deoxyribonucleic acid) acts as the information carrier/encoder, *RNA* (ribonucleic acid) acts as the bridge from DNA to proteins, and *proteins* (composed of amino acids) act as the action molecules.

DNA is an unbranched double helical polymer composed of nucleotides, which have four kinds of bases (the nucleic acids): adenine (A), thymine (T), cytosine (C), and guanine (G). A always pairs with T and G pairs with C to form complimentary base pairing in the double helical DNA strand. The primary structure of DNA can be represented simply as a string of bases, for example, *ATGAATCGTAA* · · · .

Unlike DNA, RNA is a single-stranded molecule also made up of four nucleic acids, but instead of T RNA has uracil (U). U also binds with A just like T. There are different kinds of RNA in the cell; messenger (mRNA) and transfer RNAs (tRNA) are of most relevance for protein folding.

Proteins are unbranched polymers with a peptide backbone formed from a chain of amino acids. An amino acid has a central carbon atom, called the alpha carbon (C_α). Attached to the

C_α atom is a hydrogen atom, an amino group, a carboyl group, and a side chain. There are 20 different side chains, which differentiate one amino acid from another, giving 20 different amino acids. The side chains differ in size (ranging from a single hydrogen atom to having carbon rings), polarity (polar/nonpolar), attractivity to water (hydrophilic/hydrophobic), and so forth.

The primary structure of a protein can be represented as a sequence formed from a 20 letter alphabet (1 letter per amino acid), for example, *MNRRGLNAGNTMTSQANID*.... Because proteins are 3-D molecules, they have higher-order structures. The secondary structure of a protein refers to the structures formed by short subsequences (e.g., alpha helixes and beta sheets are the main secondary structures). The global fold of the proteins is called its tertiary structure (i.e., the arrangement of secondary structures in 3-D). Multiple protein domains also arrange themselves into complexes, referred to as the quaternary structure. The shape of the protein is crucial to the function; its irregular surface allows different kinds of molecules to bind to the protein, and this facilitates various activities (regulatory, enzymatic, etc.) in the cell. This is why the protein folding problem is so important.

So how do proteins form? The *genes*, which are contiguous stretches of DNA, encode the information to manufacture proteins. Often there is a very complex regulatory network involving several genes that controls the production of proteins. A DNA strand thus consists of the genes at different positions interspersed with the intergenic regions, which do not code for any protein. A gene itself can be composed of exons (expressed segments) and introns (unexpressed segments).

Protein formation happens via two main steps: *transcription* and *translation*. First a copy of the gene is made; subsequently, the unexpressed introns are spliced out to yield an mRNA molecule that consists only of the exons. This process is called transcription. A subsequence of three mRNA bases, called a codon, codes for a single amino acid. tRNA molecules are the ones that implement the translation between the codon and the amino acids. The ribosomal assembly unit slides along the mRNA molecule, and for successive codons the tRNA molecule attracts different amino acids, with the result that as the ribosomal unit slides there is a growing polypeptide chain. This process is called translation. The protein synthesis process stops when a stop codon is encountered. As mentioned earlier, the proteins take on unique 3-D shapes, which govern their function.

Mining Methods in Protein Structure Prediction

It is well known that proteins fold spontaneously and reproducibly to a unique 3-D structure in aqueous solution. Despite significant advances in recent years, the goal of predicting the three-dimensional structure of a protein from its one-dimensional sequence of amino acids, without the aid of evolutionary information, remains one of greatest and most elusive challenges in bioinformatics. The current state of the art in structure prediction provides insights that guide further experimentation but falls far short of replacing those experiments.

Today we are witnessing a paradigm shift in predicting protein structure from its known amino acid sequence (a_1, a_2, \dots, a_n). The traditional or *ab initio* folding method employed first principles to derive the 3-D structure of proteins. However, even though considerable progress has been made in understanding the chemistry and biology of folding, the success of *ab initio* folding has been quite limited.

Instead of simulation studies, an alternative approach is to employ learning from examples using a database of known protein structures. For example, the Protein Data Bank (PDB) records the 3-D coordinates of the atoms of thousands of protein structures. Most of these

proteins cluster into approximately 700 fold-families based on their similarity. It is conjectured that there will be on the order of 1,000 fold-families for the natural proteins (Wolf, Grishin, & Koonin, 2000). The PDB thus offers a new paradigm to protein structure prediction by employing data mining methods such as clustering, classification, association rules, hidden Markov models, and so forth.

The ability to predict protein structure from the amino acid sequence will do no less than revolutionize molecular biology. All genes will be interpretable as 3-D, not one-dimensional, objects. The task of assigning a predicted function to each of these objects (arguably a simpler problem than protein folding) would then be underway. In the end, combined with proteomics data (i.e., expression arrays), we would have a flexible model for the whole cell, potentially capable of predicting emergent properties of molecular systems, such as signal transduction pathways, cell differentiation, and the immune response.

Protein Folding Pathways. Proteins are chains of amino acids having lengths ranging from 50 to 1,000 or more residues. The early work of Levinthal (Levinthal, 1968) and Anfinsen (Anfinsen & Scheraga, 1975) established that a protein chain folds spontaneously and reproducibly to a unique 3-D structure when placed in aqueous solution. The sequence of amino acids making up the polypeptide chain contains, encoded within it, the complete building instructions. Levinthal also proved that the folding process cannot occur by random conformational search for the lowest energy state, because such a search would take millions of years, whereas proteins fold in milliseconds. As a result, Anfinsen proposed that proteins must form structure in a time-ordered sequence of events, now called a “pathway.” The nature of these events, whether they are restricted to “native contacts” (defined as contacts that are retained in the final structure) or whether they might include nonspecific interactions such as a general collapse in size at the very beginning were left unanswered. Over time the two main theories for how proteins fold became known as the “molten globule” or “hydrophobic collapse” (invoking nonspecific interactions) and the “framework” or “nucleation/condensation” model (restricting pathways to native contacts only).

Over the years the theoretical models for folding have converged somewhat, in part due to a better understanding of the structure of the so-called unfolded state and due to a more detailed description of kinetic folding intermediates. The “folding funnel” model (Nolting et al., 1997) has reconciled hydrophobic collapse with the nucleation-condensation model by envisioning a distorted, funicular energy landscape and a “minimally frustrated” pathway. The view remains of a gradual, counterentropic search for the hole in the funnel as the predominant barrier to folding.

The 3-D conformation of a protein may be compactly represented in a symmetrical, square, Boolean matrix of pairwise, interresidue contacts, or “contact map.” The contact map provides a host of useful information about the protein’s structure. For example, clusters of contacts represent certain secondary structures, and it also captures nonlocal interactions giving clues to the tertiary structure.

Following sections describe how data mining can be used to extract valuable information from contact maps. The focus is on two main tasks: (a) Given a database of protein sequences and their 3-D structure in the form of contact maps, build a model to predict whether pairs of amino acids are likely to be in contact; (b) discover common (nonlocal) contact patterns or “features” that characterize physical “protein-like” contact maps. I show via experiments that proposed techniques are very effective in predicting and characterizing contacts. I further highlight promising directions of future work, for example, how mining can help in generating heuristic rules of contacts, and how one can generate plausible folding pathways in contact map conformational space.

The protein folding problem will be solved gradually, by many investigators who share their results at the biannual Critical Assessment of Protein Structure Prediction meeting (Moult, Pedersen, Judson, & Fidelis, 1995), which offers a worldwide blind prediction challenge. Here, we will investigate how mining can uncover interesting knowledge from contact maps.

MINING PROTEIN CONTACT MAPS

The contact map of a protein (see Fig. 24.1) is a particularly useful representation of protein structure. Two amino acids in a protein that come into contact with each other form a noncovalent interaction (hydrogen-bonds, hydrophobic effect, etc.). More formally, two residues (or amino acids) a_i and a_j in a protein are in *contact* if the 3-D distance $\delta(a_i, a_j)$ is at most some threshold value t (a common value is $t = 7 \text{ \AA}$; \AA stands for angstrom, $1 \text{ \AA} = 10^{-10} \text{ m}$), where $\delta(a_i, a_j) = |\mathbf{r}_i - \mathbf{r}_j|$, and \mathbf{r}_i and \mathbf{r}_j are the coordinates of the α -carbon atoms of amino acids a_i and a_j (an alternative convention uses beta-carbons for all but the glycines). *Sequence separation* is defined as the distance between two amino acids a_i and a_j in the amino acid sequence, given as $|i - j|$. A contact map for a protein with N residues is an $N \times N$ binary matrix C whose element $C(i, j) = 1$ if residues i and j are in contact, and $C(i, j) = 0$ otherwise.

The contact map provides a host of useful information. For example, clusters of contacts represent certain secondary structures: α -Helices appear as bands along the main diagonal because they involve contacts between one amino acid and its four successors; β -sheets are thick bands parallel or antiparallel to the main diagonal (see Fig. 24.1). Tertiary structure also may be obtained by reverse projecting into 3-D space using the MAP algorithm. (Vendruscolo, Kussell, & Domany, 1997) or other distance geometry methods. Vendruscolo et al. (1997) also

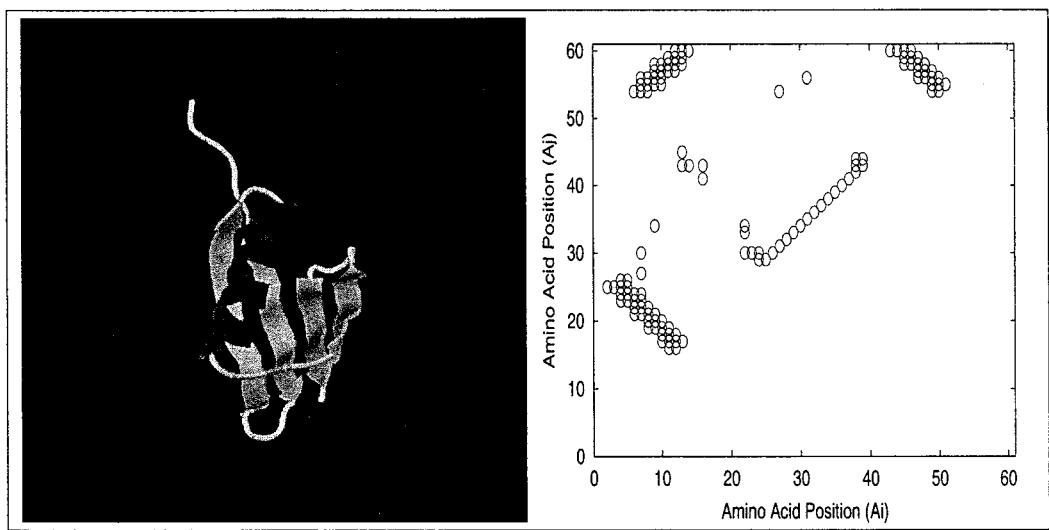


FIG. 24.1. A contact map: 3-D structure for protein G (PDB file 2igd, $N = 61$) and its contact map showing parallel (top left cluster) and antiparallel sheets (bottom left and top right cluster), and helix features (thin cluster close to main diagonal).

showed that it is possible to recover the 3-D structure from even corrupted contact maps. For predicting the elusive global fold of a protein, we are usually interested in only those contacts that are far from the main diagonal. In this chapter I thus ignore any pair of residues with sequence separation $|i - j| < 4$.

Contact Map Mining Tasks. In this chapter I focus on two main contact map mining tasks: (a) Given a database of protein sequences and their 3-D structure in the form of contact maps, build a model to predict whether pairs of amino acids are likely to be in contact or not; (b) discover common (nonlocal) contact patterns or “features” that characterize physical “protein-like” contact maps. I also highlight other complementary mining tasks, namely, how to mine heuristic rules of contacts in real proteins and how to generate folding pathways in contact map conformational space.

Classifying Contacts Versus Noncontacts

For training and testing of the classification model a nonredundant database of proteins of known structure was used, namely PDBselect:December 1998 (Hobohm & Sander, 1994) containing 691 proteins and their sequence families. The proteins in the set have $<25\%$ sequence similarity. Disordered or missing coordinates in the middle of a protein sequence were addressed by dividing the sequence at that point. This produced a set of 794 files, most of them containing an entire protein sequence, but some of these correspond to proteins that were split.

Given a PDB protein file, we have to transform the data into a format that can be easily mined, that is, we need to prepare the data in tabular format in which are multiple attributes (columns) for each example (rows) or record. Because we are interested in predicting the contact between a pair of amino acids, each pair is used as an example in the training set, associated with a special *class* attribute indicating whether it is a contact (C) or noncontact (NC); amino acids a_i and a_j are in contact if $\delta(a_i, a_j) < 7 \text{ \AA}$, that is, the distance between α -carbons of amino acids a_i and a_j is less than 7 \AA . The new database has an entry showing the two amino acids and their class for each pair of amino acids for each protein. To avoid predicting purely local contacts we ignore all pairs with sequence separation $|i - j| < 4$. Note also that the number of contacts N_C is a lot smaller than the number of noncontacts N_{NC} for any protein.

The percentage of contacts (or number of database entries with class 1) over all pairs was less than 1.7%. Across the 794 files the longest sequence had length 907, whereas the smallest had length 35. There were 17,618,115 pairs over all proteins, whereas only 292,126 pairs were in contact. This database thus corresponds to a highly biased binary classification problem. That is, we have to build a mining model that can discriminate between contacts and noncontacts between amino acids pairs, where the examples are overwhelmingly biased toward the noncontacts.

Mining Methodology

Given these databases the goal is to find high frequency and high confidence rules of the form $X \Rightarrow C$ and $X \Rightarrow NC$ that discriminate between the contact pairs and the noncontact pairs, respectively (X denotes a pair of amino acids along with possibly other attributes). *Support* is the joint probability of rule antecedent and consequent, whereas *confidence* is the conditional probability of the consequent given the antecedent. Following sections describe the mining/training and testing phases, where we learn from examples using the frequent

closed item sets (Zaki & Hsiao, 2002) and then classify unseen examples as being contacts or noncontacts, respectively.

Mining on Known Examples

The goal of the mining phase is to learn from known contact and noncontact examples and build a model or rule set that discriminates between the two classes. I selected a random 90% of the files for training out of a total of 794 files. The remaining 10% of the files were kept aside for testing the mined rule set.

Looking at all pairwise amino acids one can obtain two databases. One data set consists of all pairs that are in contact (from all the proteins), denoted as \mathcal{D}_C . The other database consists of all pairs not in contact, denoted as \mathcal{D}_{NC} . Because I am primarily interested in predicting the contacts rather than the noncontacts, I mine only on the contacts database \mathcal{D}_C . However, I do use the noncontacts database \mathcal{D}_{NC} to prune out those patterns that are frequent in both sets. Building a discriminative rule set consists of the following steps, in order:

1. *Mining.* Use CHARM (Zaki & Hsiao, 2002) to mine all the frequent closed item sets in \mathcal{D}_C . An *item set* is a set of attribute values; it is frequent if it occurs at least min_freq times in the database, and it is *closed* if there is no proper superset with the same frequency. The frequency of an item set is also called *support*, denoted $\sigma(X, \mathcal{D})$, where X is an item set and \mathcal{D} a database. Let's denote the set of mined frequent closed item sets as \mathcal{F} .
2. *Counting.* Next compute the support of all item sets in \mathcal{F} in the noncontacts database \mathcal{D}_{NC} .
3. *Pruning.* Compute the probability of occurrence of each item set in \mathcal{F} in both the contact and noncontact databases. The probability of occurrence is simply the support of the item set divided by the number of examples in the given data set. For example, if itemset $X \in \mathcal{F}$, then the probability of its occurrence in \mathcal{D}_C is given as $P(X, \mathcal{D}_C) = \sigma(X, \mathcal{D}_C)/|\mathcal{D}_C|$. As a first step in pruning we can remove all item sets $X \in \mathcal{F}$ that have a greater probability of occurrence in the noncontact database than in the contact database, that is, if $P(X, \mathcal{D}_{NC}) > P(X, \mathcal{D}_C)$. Actually, I compute the ratio of the contact probability versus the noncontact probability for X , and prune it if this ratio is less than some suitably chosen threshold ρ , that is, prune X if $P(X, \mathcal{D}_C)/P(X, \mathcal{D}_{NC}) < \rho$ (in the experiments I tried various values of ρ ranging from 2 to 10). In other words, the aim is to retain only those item sets that have a much greater chance of predicting a contact rather than a noncontact.

Testing on Unknown Examples

The goal of the testing phase is to find how accurately the mined set of rules predicts the contacts versus the noncontacts in new examples not used for training. A random 10% of the files was used in the database for testing. The test set had a total of 2,336,548 pairs, out of which 35,987 or 1.54% were contacts. Because we do know the true class of each example, it is easy to find out how good the rules are for prediction.

For testing we generate a combined database \mathcal{D}_t containing all pairs of amino acids in contact or otherwise. For each example we know the true class. We assign each example a predicted class using the following steps:

1. *Evidence calculation:* For each example E in the test data set \mathcal{D}_t , compute which item sets in the set of mined and pruned closed frequent item sets \mathcal{F} are subsets of E . Let's denote

the set of these item sets as S . Next calculate the cumulative contact and noncontact support for example E , that is, the sum of the supports of all item sets in S in the contact and noncontact database. Finally, compute the evidence for E being a contact, that is, take the ratio of the cumulative contact support over cumulative noncontact support, denoted as ρ_E , and defined as

$$\rho_E = \frac{\sum_{X \in S} \sigma(X, \mathcal{D}_C)}{\sum_{X \in S} \sigma(X, \mathcal{D}_{NC})}$$

Any example E with zero contact support is taken to be a noncontact and discarded, and only the examples or test pairs with positive contact support are retained for the next step.

2. *Prediction.* To make the final prediction of whether a test pair of residues is in contact or not, sort all test examples E (with positive cumulative contact support) in decreasing order of contact evidence ρ_E . Finally, the top γ fraction of examples in terms of ρ_E are predicted to be contacts and the remaining $1 - \gamma$ fraction of examples as noncontacts. How γ is chosen will be explained in the following section.

Model Accuracy and Coverage

In predicting contacts versus noncontacts for the test examples, we have to evaluate the mined model based on two metrics: *accuracy* and *coverage*. Furthermore, we are interested only in the prediction of contacts; thus, accuracy and coverage are considered only for contacts. Accuracy is the ratio of correct contacts to the predicted contacts, whereas coverage is the percentage of all contacts correctly predicted. Thus, accuracy tells us how good the model is, and coverage tells us the number of contacts predicted.

More formally, let N_{tc} denote the number of true contacts in the test examples, N_{pc} the number of predicted contacts, N_{tpc} the number of true predicted contacts, and let N_a denote the number of all possible contacts, that is, $N_a = (N - 3) \times (N - 2)/2$ (where N is the protein length), because the contact map is symmetric, and pairs with sequence separation less than 4 are ignored. The accuracy of the model is given as:

$$A = N_{tpc}/N_{pc}$$

The coverage of the model is given as:

$$C = N_{tpc}/N_{tc}$$

The number of contacts predicted N_{pc} , of course, depends on how we chose γ , because the top γ fraction of test examples based on evidence is predicted as contacts. Because a protein is characterized by N_{tc} true contacts, I set $\gamma = N_{tc}^*/N_a^*$ and then predict the top γ fraction of examples as contacts. Note that N_{tc}^* and N_a^* denote the actual contacts and all pairs, respectively, that have positive contact support, because I discard examples with zero contact support. By adopting this method, the number of predicted contacts is limited to those actually present in the protein. Further, this method has been used by previous approaches to contact map prediction (Fariselli & Casadio, 1999; Olmea & Valencia, 1997), and I retain it to facilitate comparison with previous results. Finally, I also compare our model against a random predictor. The accuracy of random prediction of contacts is defined as:

$$A_r = N_{tc}/N_a$$

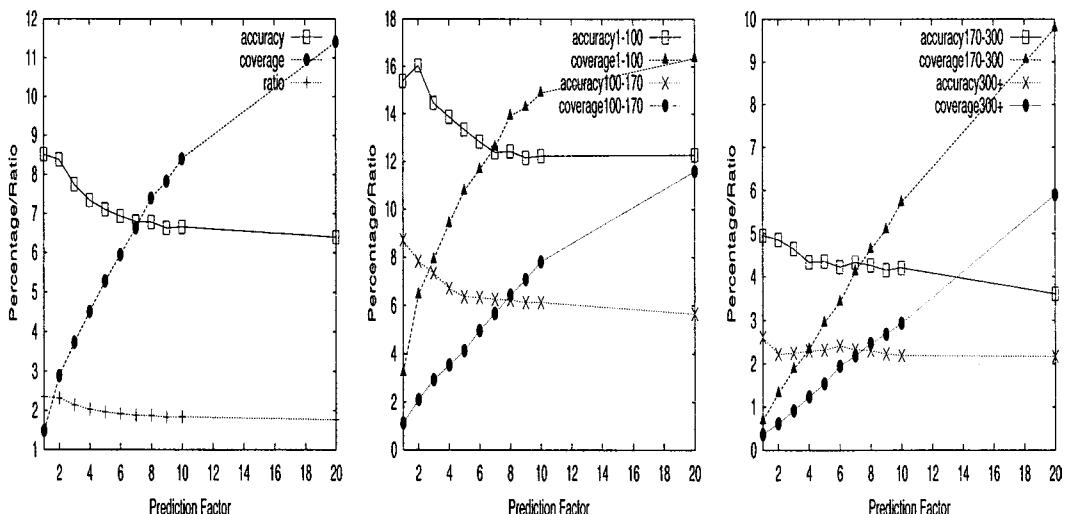


FIG. 24.2. Predicting contacts using only amino acids. Coverage, Accuracy and Improvement over random predictions for, (a) all proteins, (b) proteins with length 1–100 and 100–170, and (c) proteins with length 170–300 and 300+.

How Much Information Is There in Amino Acids Alone?

Our first goal is to test how much information is contained in the amino acids only, that is, for both training and testing, each example consisted of only the two amino acids a_i and a_j , along with their class (contact or noncontact). Figure 24.2 shows the accuracy, coverage, and improvement of the mined model over the random predictor for the test set. The accuracy and coverage is the mean value over all proteins. The figure plots the accuracy and coverage as percentages. It also plots the improvement of the model over the random predictor in ratios. The x -axis shows the *prediction factor*, which is related to the γ value (used to predict the top fraction of pairs as contacts). The prediction factor is in multiples of N_{tc}^* , the number of true contacts in the protein with positive contact evidence. For example, a value of 10 means that the top $(10 \times N_{tc}^*)/N_a^*$ fraction of the examples are predicted as contacts. As one increases the number of contacts predicted (increasing γ ratios) the coverage increases and the accuracy decreases, exhibiting the classic accuracy versus coverage tradeoff, that is, as one makes more and more predictions one clearly increases the chances of finding more true contacts, but at the same time the fraction of true predictions decreases. Looking at the ratio line, we find that the information obtained from frequent pairs of amino acids leads to a model that is around two times better than a random predictor (i.e., randomly tagging a test pair as contact or noncontact). Thus, we observe that although amino acids alone have some information that can be used to predict contacts versus noncontacts, this information is not too good.

The left-most graph in Fig. 24.2 shows the accuracy and coverage of the predictor over test proteins of all lengths. The other two figures show how accuracy and coverage change with protein length. The test proteins are divided into four bins: $1 \leq N < 100$, $100 \leq N < 170$, $170 \leq N < 300$, and $300 \leq N$.

Over all proteins the amino acids by themselves can be used to give an 8.5% accuracy, 1.5% coverage, and an improvement over a random predictor by a factor of 2.4. Note also the interesting trend in the graph. As the prediction factor increases we get better and better

coverage, but the accuracy trails off. Which value to choose for the prediction factor depends on what is more important. It has been reported in (Vendruscolo et al., 1997) that the 3D structure of proteins can be recovered quite robustly, even from corrupted contacts maps. This implies that coverage should have a higher weight than accuracy. In any case, if we had to choose a value representing the best trade-off, we can pick the point where the accuracy and coverage curves intersect. This happens for a prediction factor of 7, where we have roughly 7% accuracy and coverage, and which is 2 times better than random. For 6.3% accuracy we can increase coverage to 14%.

When we consider the results for proteins of different lengths, we find the same trade-off between accuracy and coverage. Looking at the crossover point, we get around 13% accuracy and coverage for short proteins with $N < 100$, 6% for $100 \leq N < 170$, 4.5% for $170 \leq N < 300$, and around 2% of longer proteins.

Using Local Structures for Contact Prediction

Previous work (Bystroff & Baker, 1998) showed that some small, fast-folding regions of a protein may be identified by their sequence alone. A library of 262 short sequence patterns that fold fast was compiled by cluster analysis of the database of known protein structures. Evidence from Nuclear Magnetic Resonance (NMR) and molecular dynamics simulations found that these database-derived sequence motifs are folding initiation sites, or I-sites.

Subsequent work developed HMMSTR (Bystroff, Thorsson, & Baker, 2000), a hidden Markov model (HMM) for generalized protein sequence. Generalized HMMs are directed cyclic graphs in which each node is a single symbol emitter. HMMSTR is a “parallel HMM” that emits, from one Markov state, a single amino acid and a symbol for the backbone phi and psi angles. HMMSTR was designed as condensed representation of all known local structure motifs, as defined in the I-sites database. Therefore, collectively the allowed paths through the directed graph represent all known local structure motifs, including alpha-helices, beta strands, helix caps, and a variety of loops and turns. They are represented in the model in proportion to the frequency at which they are found in the database of protein structures. The HMMSTR model from the merged I-sites motifs, results in 282 Markov states, as shown in Fig. 24.3. Each HMMSTR state can produce, or “emit,” amino acids and structure symbols according to a probability distribution specific to that state.

Using HMMSTR Output

After HMMSTR was built, the optimal HMMSTR states that agree with the observed amino acids was computed for each of the 691 proteins from PDBselect. The output probability distribution of all the states thus chosen for a protein sequence is used as input for the frequent item set mining algorithm. In fact, rather than a single state associated with a given residue, we have available the probability that the residue at the given position is associated with all the states of HMMSTR, that is, we have available $P(q_i | a_j)$ for all the 282 HMMSTR states ($1 \leq i \leq 282$) for all the residues in a given protein ($1 \leq j \leq N$), where q_i is a Markov state, a_j is a given residue, and N is the length of the protein.

For each residue we also know the amino acid at that position. Additional outputs describe the probability of observing a particular amino acid, secondary structure, backbone angle region, or structural context descriptor. We also have the spatial coordinates of the α -carbon atom $\langle x, y, z \rangle$; a distance vector of length n giving the distance of this residue from all other residues in the protein; and the 20 amino acid profiles for that position.

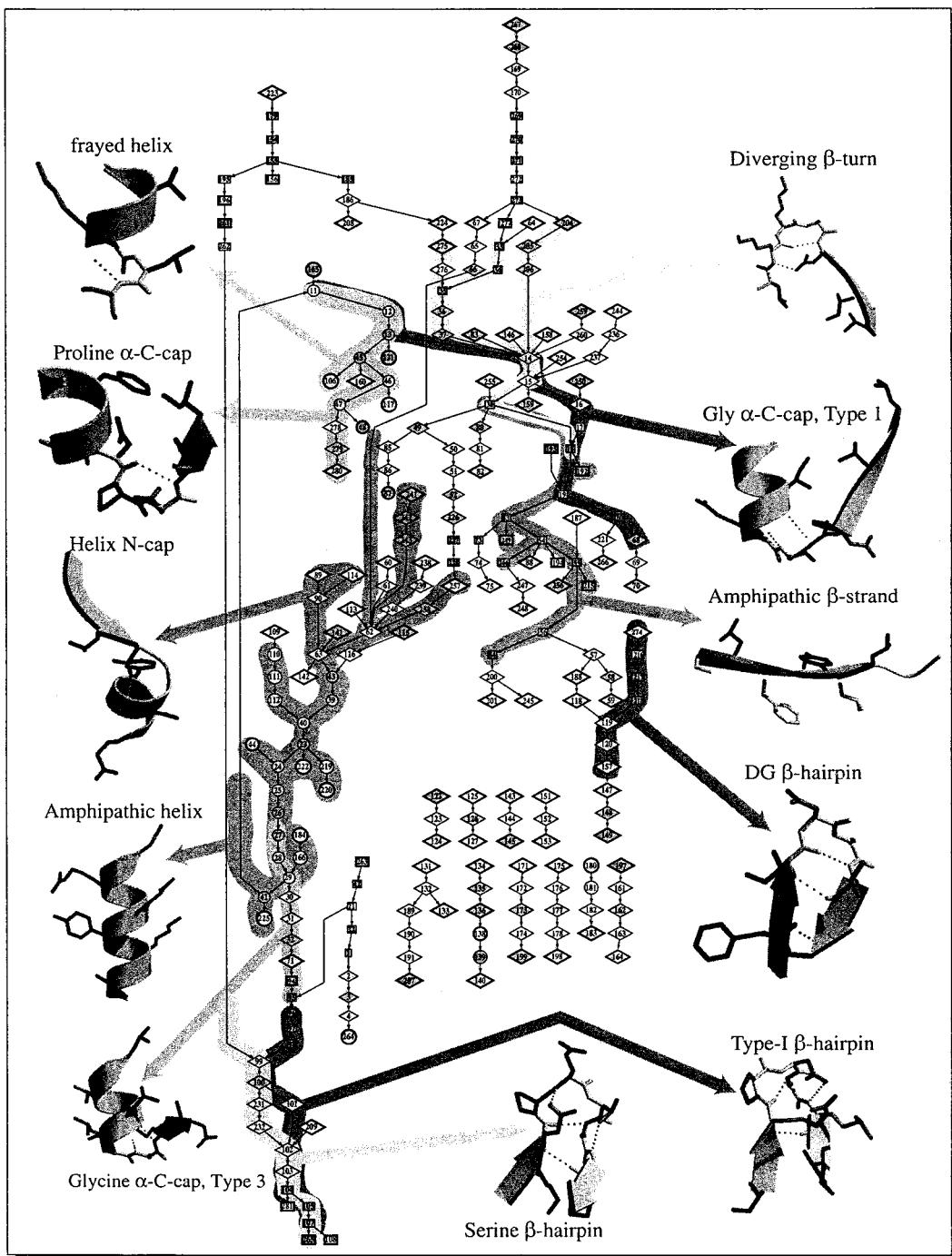


FIG. 24.3. HMMSTR hidden Markov model (Bystroff, Thorsson, & Baker, 2000).

A protein data file looks like this:

```
PDB Name: 1531_
Sequence Length: 185

Position: 1
Residue: R
Coordinates: 0.0 -73.2 177.6
Profile: 0.0 ... 1.0 ... 0.0 #20 Values
HMMSTR State Probabilities:
    0.0 ... 0.7 ... 0.3 ... 0.0 #282 Values
Distance Vector: 0 3 5 ... 18 15 13 #185 Values,
i.e., Seq Length

Position: 2
Residue: T
Coordinates: -124.4 0.2 -177.1
Profile: 0.0 ... 1.0 ... 0.0 #20 Values
HMMSTR State Probabilities:
    0.0 ... 0.9 ... 0.1 ... 0.0 #282 Values
Distance Vector: 3 0 3 ... 15 13 10 #185 Values

...
Position: 185
Residue: Y
Coordinates: -88.7 0.0 0.0
Profile: 0.0 ... 0.4 ... 0.6 ... 0.0 #20 Values
HMMSTR State Probabilities:
    0.0 ... 0.2 ... 0.5 ... 0.3 ... 0.0 #282 Values
Distance Vector: 15 13 10 ... 5 3 0 #185 Values
```

We have a file like this for all of the 691 nonredundant set of proteins from PDBSelect. Disordered or missing coordinates in the middle of a protein sequence were addressed by dividing the sequence at that point. This produced a set of 794 files, most of them containing an entire protein sequence, but some of these correspond to proteins that were split. Using all the available information from HMMSTR, we create a tabular data set with a row per pair of amino acids and with multiple features (columns) comprised of the amino acid profiles, HMM state probabilities, other structural context information, and a class (contact or noncontact).

Because association rules work only for categorical attributes, we need to convert the continuous state probabilities into discrete values. To do this we take the ratio of each of the 282 HMMSTR state probabilities for a_i against the background or prior probability of an amino acid being in that state; if the ratio is more than some threshold, we include the state in the context of a_i , else we ignore it. We repeat the same process for a_j . Using a similar thresholding method one can incorporate the amino acid profiles for positions i and j . With all this context information for both a_i and a_j we obtain a new database to be used to find the frequent closed item sets characterizing the contacts and noncontacts. In summary, the database has the following columns for pairs of amino acids over all proteins:

Protein and position information: ProteinID PairID i j $|i-j|$
 Amino acids and context: a_i a_j d_i d_j r_i r_j c_i c_j
 Profile: p_{i1} p_{i2} ... p_{j1} p_{j2} ...
 HMMSTR: q_{i1} q_{i2} ... q_{j1} q_{j2} ...
 Class: C or NC

Note that the number of columns can be variable for different pairs depending on the profile and HMMSTR state probabilities. p_{i1} , p_{i2} , and so forth show the other amino acids that can appear in position i (provided the probability is more than some threshold), and finally q_{i1} , q_{i2} , and so forth show HMMSTR states with probabilities more than some factor of the prior probability of those states. For additional details see Zaki, Jin, and Bystroff (2000).

Mining Predictive Rules

To mine the rules predictive of contacts, consider the augmented database of examples obtained by adding in the amino acid profile, structural contextual information, and the Markov states for each residue. Then mine the frequent patterns that distinguish contacts from noncontacts as described earlier.

Figure 24.4 shows the prediction results using this augmented database. If we look at the cross-over point we get almost 19% accuracy and coverage, whereas the model is 5.2 times better than random. For 18% accuracy we can get coverage of 25% (still 5.1 times better than random).

If we look at proteins of various lengths in Fig. 24.4, we find that for $N < 100$, we get 26% accuracy and 63% coverage at the extreme point (4 times over random). For $100 \leq N < 170$, we get 21.5% accuracy and 10% coverage toward the end (6 times over random); for $170 \leq N < 300$, we get 13% accuracy and around 7.5% coverage (6.5 times over random); and for longer proteins we get 9.7% accuracy and 7.5% coverage (7.8 times over random).

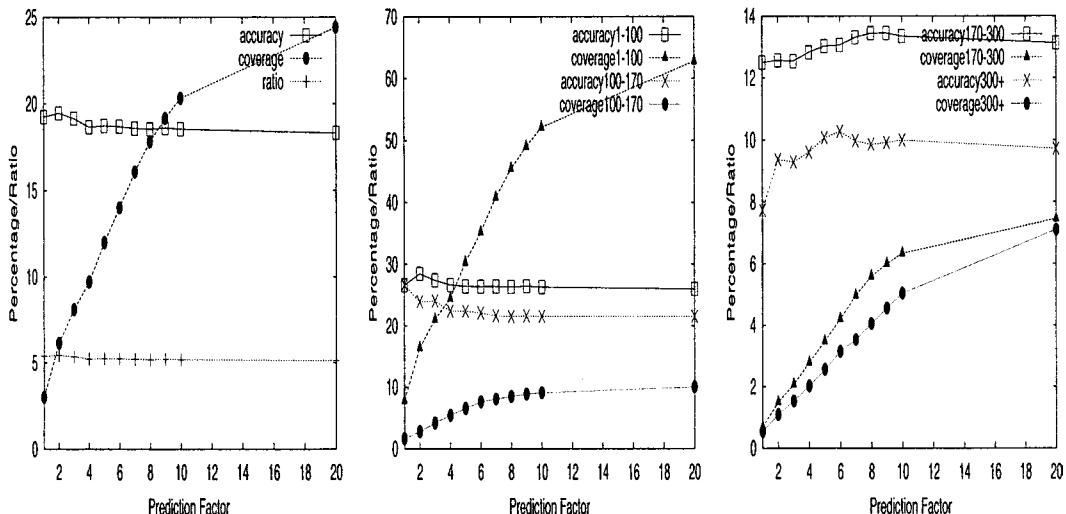


FIG. 24.4. Predicting contacts using HMMSTR states and amino acids. Coverage, Accuracy and Improvement over random predictions for, (a) all proteins, (b) proteins with length 1–100 and 100–170, and (c) proteins with length 170–300 and 300+.

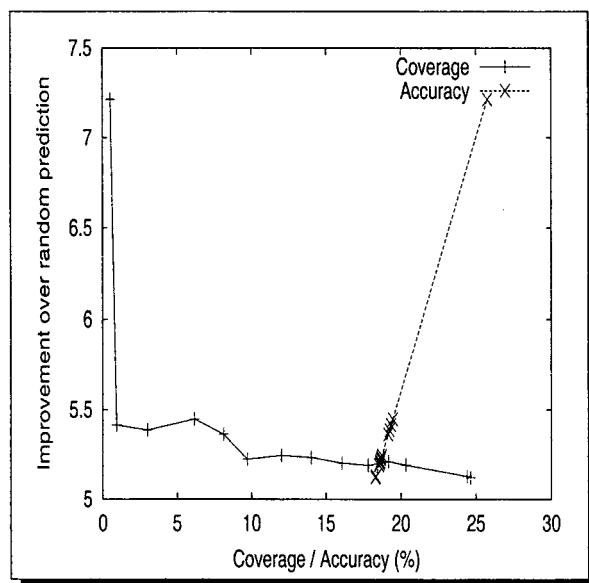


FIG. 24.5. Improvement over random prediction.

Figure 24.5 shows the results in a slightly different format. It plots the improvement in coverage/accuracy over a random prediction. For example, at around 1% coverage we have a model that is 7.25 better than the random predictor, whereas at 25% coverage the model is about 5.1 better than random prediction. For the accuracy plot, at 18% accuracy the model is 5.1 better than random, whereas at 25% accuracy it is 7.25 better.

These results are comparable to or better than the results recently reported in Zhao and Kim (2000), in which the authors examined pairwise amino acid interactions in the context of secondary structural environment (helix, strand, and coil) and used the environment dependent contact energies for contact prediction experiments. For about 25% coverage our model does more than 5 times better than the random predictor, as compared with the 4 times improvement reported in Zhao and Kim (2000).

I believe these results are the best of, or at least comparable with, those reported so far in the literature on contact map prediction (Fariselli & Casadio, 1999; Olema & Valencia, 1997). For example, Fariselli and Casadio (1999), used a neural network based approach over pairs database, with other contextual information such as sequence context windows, amino acid profiles, and hydrophobicity values. They reported an 14.4% accuracy over all proteins, with 5.4 times improvement over random. They also got 18% accuracy for short proteins with 3.1 times improvement over random. Olmea and Valencia (1997), on the other hand, used correlated mutations in multiple sequence alignments for contact map prediction. They added other information such as sequence conservation, alignment stability, contact occupancy, and so forth to improve the accuracy. They reported 26% accuracy for short proteins, but they did not report the result for all proteins. Although I believe that the hybrid approach described in this chapter does better, I should say that direct comparison is not possible, because previous works used a different (and smaller) PDB_select database for training and testing. One drawback of these previous approaches is that they do not report any coverage values, so it is not clear what percentage of contacts are correctly predicted. Another approach to contact map prediction was

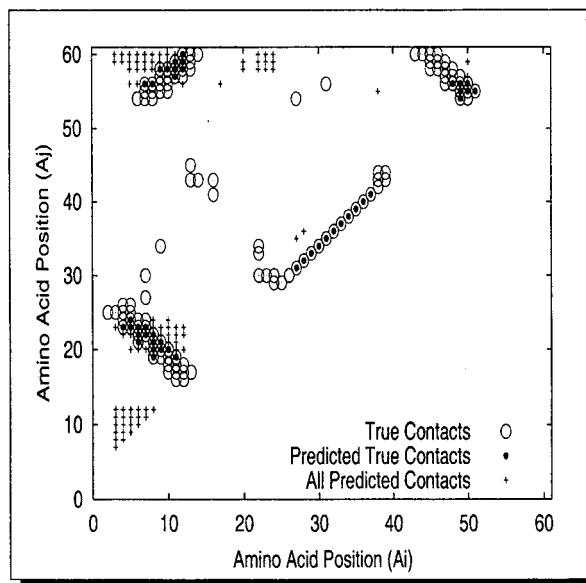


FIG. 24.6. Predicted contact map (PDB file 2igd).

presented in Thomas, Casari, and Sander (1996) which was based on correlated mutations. They obtained an accuracy of 13% or 5 times better than random.

Predicted Contact Map

Figure 24.6 shows the predicted contact map for the protein *2igd* from Fig. 24.1. There was 35% accuracy and 37% coverage for this protein. The figure shows the true contacts, the contacts correctly predicted, and all the contacts predicted (correctly or incorrectly). The prediction was able to capture true contacts representing portions of all the major interactions. For example, true contacts were found for the alpha helix, the two antiparallel beta sheets, and the parallel beta sheet. However, some spurious clusters of contacts also were discovered, such as the triangle in the lower left corner. In the next section I show how one can eliminate such false contacts by recognizing the fact that they never occur in real proteins, because such a cluster is physically impossible.

CHARACTERIZING PHYSICAL, PROTEIN-LIKE CONTACT MAPS

Proteins are self-avoiding globular chains. A contact map, if it truly represents a self-avoiding and compact chain, can be readily translated back to the 3-D structure from which it came. But, in general, only a small subset of all symmetric matrices of ones and zeros have this property. The task is to output a contact map that both satisfies the geometrical constraints and is likely to represent a low-energy structure. Interactions between different subsequences of a protein are constrained by a variety of factors. The interactions may be initiated at several short peptides (initiation sites) and propagate into higher-order intra- or intermolecular interactions.

The properties of such interactions depend on (a) the amino acid sequence corresponding to the interactions, (b) the physical geometry of all interacting groups in three dimensions, and (c) the immediate contexts (linear, and secondary components for tertiary structural motifs) within which such interactions occur.

Following sections describe in detail the method used for mining frequent dense patterns or structural motifs in contact maps. These motifs represent the typical nonlocal structures that appear in physical protein-like contact maps and that can be used to improve the quality of contact map prediction by eliminating impossible contacts (those that never occur in real proteins). Briefly, there are three major stages for the approach:

1. data generation, which involves creating a large set of protein-like contact maps,
2. mining, which involves computation of all the frequent dense patterns, and
3. pruning mined frequent patterns and integration of these patterns with biological data.

Generating a Database of Protein-Like Structures

Using the following procedure, 12,524 protein-like structures can be generated. We first generate 60-residue random amino acid sequences using HMMSTR (Bystroff et al., 2000) model based on the I-sites library of sequence-structure motifs. Here it is used as a stochastic emitter of Markov states. A Markov state sequence, length 60, is generated by starting in a random state and selecting the next state stochastically, using the state-state transition probabilities. The resulting state sequence is converted to an amino acid sequence by selecting each amino acid stochastically from the Markov state “profile” (amino acid probability distribution for the given state). The resulting random amino acid sequences are (locally) protein-like. We then use the ROSETTA server (Simons, Kooperberg, Huang, & Baker, 1997) to generate the structures for those random sequences. The algorithm underlying the ROSETTA server is the so-called Monte Carlo fragment insertion. It picks fragments from the I-sites library based on sequence homology, calculates the energy of the new structure, then either discards or keeps the fragments based on Monte Carlo. This is repeated until convergence.

To describe the procedure in greater detail, first a multiple sequence alignment is generated by PSI-BLAST (Altschul et al., 1997). The alignment is converted into a profile, which is used to generate the move-set (from which the inserted fragment is chosen). We use a sliding window of size three or nine to move along the sequence and match each of three or nine residue fragments from the target sequence with the motifs of the I-sites library. The 25 highest-scoring matches to the motifs are selected and put into the move-set. For a single move a fragment is chosen randomly from the move-set and inserted as a part of the structure. The energy of the current structure is evaluated with an energy function. If the new energy passes the Metropolis criterion (i.e., an exponentially distributed random variable), then the change is kept, otherwise the inserted fragment is discarded and the original structure is restored. Twelve thousand cycles of simulated annealing are applied to broaden the search for the global energy.

The five structures with the lowest energies are selected as the candidates of the resulting structure. To exclude the structures that are not as compact as natural proteins, we examine the radius of gyration (rg) of each candidate:

$$rg = \frac{\sum_{i=1}^N \sum_{j=i+1}^N dist(i, j)}{N^2 - N},$$

where $dist(i, j)$ is the distance between i and j . The higher the rg , the less compact the structure.

Of the five low-energy candidates we select only those having $rg < 11.0$ as our results. The final 12,524 structures convert into contact maps for further data mining analysis, that is, for each structure we note the pairwise residue distances and create the contact map with 7 Å cutoff threshold.

Mining Dense Patterns in Contact Maps

To enumerate all the frequent two-dimensional (2-D) dense patterns, the database of 12,524 contact maps was scanned with a 2-D sliding window of a user specified size. For all structures, any submatrix under the window that had a minimum “density” (the number of #1’s or contacts) was captured. For an $N \times N$ contact map using a 2-D $W \times W$ window, there are $(N - W) \times (N - W)/2$ possible submatrices. We have to tabulate those that are dense, using different window sizes. Window sizes from 5 to 10 capture denser contacts close to the diagonal (i.e., short-range interactions) as well as the sparser contacts far from the diagonal (i.e., long-range interactions).

Due to the intrinsic constraints in protein secondary and tertiary structures, the density of the contacts naturally decreases with chain separation distance (in the contact map, the distance from the main diagonal). To also capture these less dense but possibly significant patterns, the minimum density cutoff is scaled as a function of the chain separation distance. The density weighting function we used is as follows:

$$\text{min_d} = \text{minDensity} * (1 - (|i - j|)/N)$$

where *minDensity* is the user specified density threshold, *i* and *j* are the starting indices of a window in the 2-D contact map (here it represents the top left position of a submatrix).

Counting Dense Patterns. As we slide the $W \times W$ window, the submatrix under the window will be added to a dense pattern list if its density exceeds the *min_d* threshold. However, we are interested in those dense patterns that are frequent, that is, when adding a new pattern to the list of dense patterns we need to check if it already exists in the list. If yes, we increase the frequency of the pattern by one, and if no, we add it to the list initialized with a count of one.

The main complexity of the method stems from the fact that there can be a huge number of candidate windows. For instance, with a window size of $W = 5$, and for $N = 60$, we have 1,485 windows per contact map. This translates to 18,598,140 possible windows for the 12,524 contact maps. At the other extreme, for $W = 10$, there are 15,341,900 windows. Of these windows only relatively few will be dense, because the number of contacts is a lot less than the number of noncontacts. Still we need an efficient way of testing whether two submatrices are identical or not. Assume that P is the number of current dense patterns of size $W \times W$. The naive method to add a new pattern is to check for equality against all P patterns, where each check takes $O(W^2)$ time, giving a total time of $O(W^2 P)$ per equality check. A better approach is to use a hash table of dense patterns instead of a list. This can cut down the time to $O(W^2)$ per equality check if a suitable hash function is found. Following sections describe how we can further improve the time to just $O(W)$ per check.

Counting Dense Patterns via Hashing. For fast hashing and equality checking we encode each submatrix in the following way: Each row of the {0, 1} sequence will be converted into a number corresponding to the binary value represented by the sequence, and

all the numbers computed this way will be concatenated into a string. For example, the 5×5 submatrix below is encoded as the string: 0.12.8.8.0.

Submatrix	Binary value of row
00000	0
01100	12
01000	8
01000	8
00000	0

```
stringId (concatenate row values) = 0.12.8.8.0
```

According to our submatrix encoding scheme, each dense $W \times W$ window M is encoded as the string $stringId(M) = v_1.v_2.\dots.v_W$, where v_i is the value of the row treated as a binary string. For fast counting we employ a two-level hashing scheme. For the first level we use the sum of all the row values as the hash function:

$$h_1(M) = \sum_{i=1}^W v_i$$

The second level hashing uses the $stringId$ as the hash key and therefore is an exact hashing; that is, $h_2(M) = stringId(M)$. The use of this two-level hashing scheme allows us to avoid many unnecessary checks. The first level hashing (h_1) narrows the potential matching submatrices to a very small number. Then the second level hashing (h_2) computes the exact matches. Computing h_1 and h_2 both take $O(W)$ time; thus, equality check of a submatrix takes $O(W)$ time.

After all dense areas are hashed into the second level slot, the support counts for each unique $stringId$ of the dense patterns are collected, and those patterns that have support counts more than a user specified $minSupport$ will be considered frequent dense patterns and will be output for analysis.

Pruning and Integration

After obtaining mined patterns that are frequent and are relatively dense, we prune them using a number of heuristics to extract biologically meaningful structural motifs. All the potential structure motifs fall into two categories: that of secondary structures, which primarily consist of alpha helices or beta sheets (parallel or antiparallel); and that of tertiary structures, which involve interactions between secondary structure components. For example, alpha helices, beta sheets, and beta turn regions can have multiple contacts between them, such that components that are farther away in linear sequence could be brought together to form functional groups. These tertiary structures are particularly important for biological processes such as high-specificity binding of ligands and receptors.

Due to the intrinsic characteristics and constraints of the secondary structures, alpha helices form contact patterns that lie along the main diagonal of the contact matrix, whereas beta sheets form contact patterns that are either perpendicular (antiparallel beta sheets) or parallel to the main diagonal. The positions at which these patterns could occur are also constrained. In contrast, the contact patterns that belong to a tertiary structure (interactions between two secondary structural components) are more likely to be less dense and distant from the main diagonal. Furthermore, they do not have definitive contact shapes compared to the well-defined

secondary structure groups. Thus, it is difficult to extract these and isolate them from other patterns. We take several approaches to attack the difficulty: First, as described in the previous section, we weigh the minimum density according to the distance of each submatrix to the main diagonal, such that distal regions have smaller density threshold than proximal regions; second, by varying window-size until an appropriate size is reached, we can differentiate the tertiary interactions from the rest by measuring the density.

Once dense patterns are found, the final step is to incorporate the sequence information with them. That is, for each dense pattern and its occurrences in the different contact maps (i.e., in different 60 residue protein segments in PDB format), we note the protein ID, the start positions of the window (given as [X, Y] coordinates), the amino acid subsequences associated with the X and Y dimensions of the window, and the type of interaction. This information is then used to visualize the mined patterns. Following is an example dense pattern with associated information (only a few occurrences are shown, even though the pattern occurs 170 times):

```
StringId:0.12.8.8.0, Support = 170
00000
01100
01000
01000
00000
```

Occurrences:

pdb-name	(X,Y)	X_sequence	Y_sequence	Interaction
1070.0	52, 30	ILLKN	TFVRI	alpha::beta
1145.0	51, 13	VFALH	GFHIA	alpha::strand
1251.2	42, 6	EVCLR	GSKFG	alpha::strand
1312.0	54, 11	HGYDE	ATFAK	alpha::beta
1732.0	49, 6	HRFAK	KELAG	alpha::beta
2895.0	49, 7	SRCLD	DTIYY	alpha::beta
...				

By applying the preceding methods, two major groups of patterns can be isolated, as described in the next section: major secondary structure components such as alpha and beta sheets, and tertiary structural motifs involving two secondary structural components.

Experimental Results

We discovered frequent submatrix patterns whose supports reach 1 to 2% when using a sliding window of size 8 and a minimum density of 0.125. These patterns turned out to correspond to beta sheets secondary structures (parallel and antiparallel 1's in the binary contact map). Examples of the most frequent dense patterns are given in Table 24.1.

The second class of patterns involves interactions between different secondary structures. The kinds of the interactions revealed with the frequent dense patterns differ in terms of the involved secondary structure components, multiplicity of interacting atoms, and the contexts (linear and secondary) surrounding such interactions. Roughly, based on the type of secondary components that are involved, we observe that the mined tertiary structure motifs can be categorized into the following classes:

TABLE 24.1
Frequent Dense Submatrices

Submatrix	0 0 0 0 0 0 0	0 1 1 1 0 0 0 0	1 0 0 0 0 0 0 0
	0 0 0 0 0 0 0	1 1 1 0 0 0 0	0 1 0 0 0 0 0 0
	0 0 0 0 0 0 0	1 1 0 0 0 0 0	0 0 1 0 0 0 0 0
	0 0 0 0 0 0 0	1 0 0 0 0 0 0	0 0 0 1 0 0 0 0
	0 0 0 0 0 0 1	0 0 0 0 0 0 0	0 0 0 0 1 0 0 0
	0 0 0 0 0 1 1	0 0 0 0 0 0 0	0 0 0 0 0 1 0 0
	0 0 0 0 1 1 1	0 0 0 0 0 0 0	0 0 0 0 0 0 1 0
	0 0 0 1 1 1 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 1
Frequency	2.0%	2.2%	1.9%
Physical phenomena	Antiparallel beta	Antiparallel beta	Parallel beta

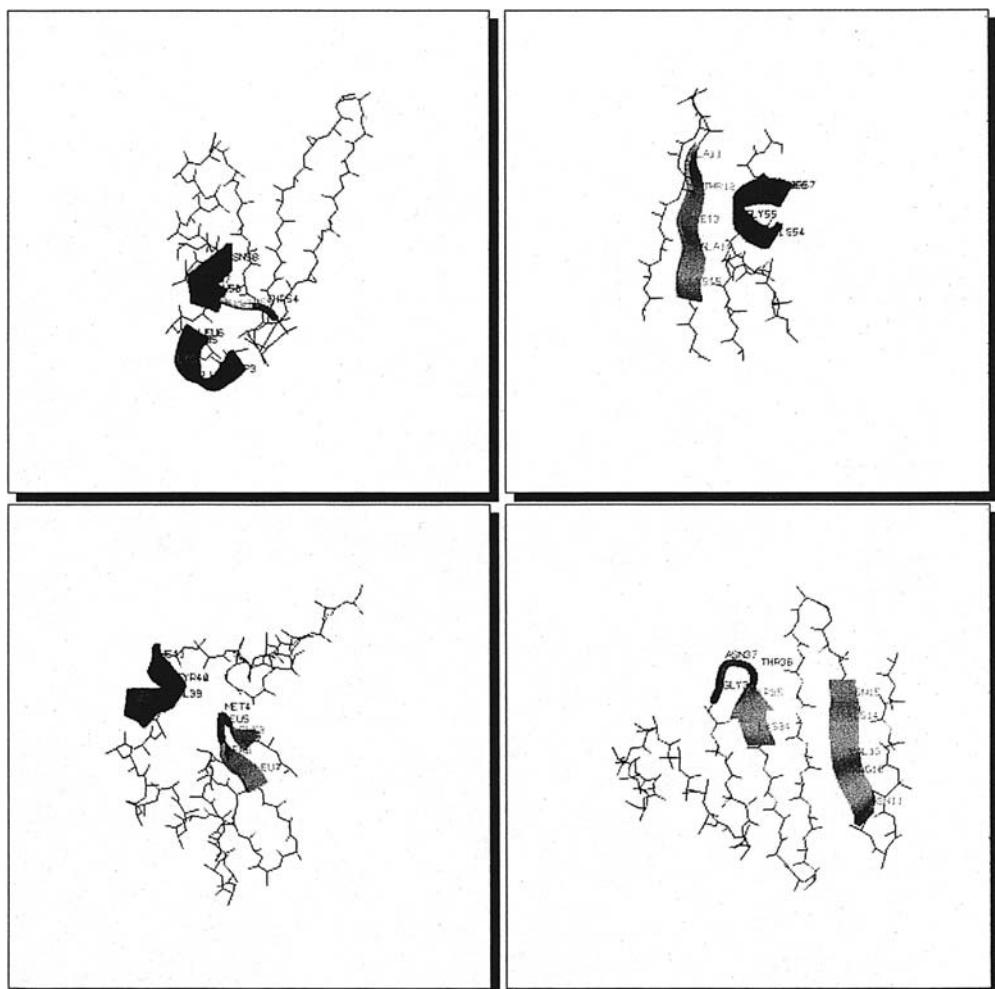


FIG. 24.7. Frequent patterns between secondary structures: (a) alpha helix-alpha helix; (b) alpha helix-beta sheet; (c) alpha helix-beta turn; (d) beta sheet-beta turn.

1. Alpha helix $a_1 :: \text{alpha helix } a_2$
2. Alpha helix $a :: \text{beta sheet } b$
3. Alpha helix $a :: \text{beta turn } bt$
4. Beta sheet $b :: \text{beta turn } bt$

Figure 24.7 shows an example of each of these four types of interactions. These four classes can be further divided into subclasses according to the number of contacts involved in each component, multiplicity of interacting atoms (one to one, one to many, or many to many), sequence specificities, and the linear/secondary structural contexts of the interaction. A library of all possible nonlocal interactions in “real” contact maps is in development.

FUTURE DIRECTIONS FOR CONTACT MAP MINING

Besides predicting and characterizing contact potentials between residues, one can formulate additional important mining tasks to extract interesting knowledge from contact maps. Here I highlight two such tasks: how to mine heuristic rules of contacts and how to mine folding pathways.

Heuristic Rules for “Physicality”

Simple geometric considerations may be encoded into heuristics that recognize physically possible and protein-like patterns within contact maps, C . For example, we may consider the following to be rules that are never broken in true protein structures:

If $C(i, j) = 1$ and $C(i + 2, j + 2) = 1$, then $C(i, j + 2) = 0$, and $C(i + 2, j) = 0$.
If $C(i + 2, j) = 1$ and $C(i, j + 2) = 1$, then $C(i, j) = 0$, and $C(i + 2, j + 2) = 0$.

These rules encode the observation that a beta sheet (contacts in a diagonal row) is either parallel or antiparallel, but not both. Another example may be drawn from contacts with alpha helices.

If $C(i, i + 4) = 1$ and $C(i, j) = 1$ and $C(i + 4, j) = 1$, then $C(i + 2, j) = 0$.

This follows from the fact that $i + 2$ lies on the opposite side of the helix from i and $i + 4$, and therefore cannot share contacts with nonlocal residue j . Local structure may be used in the definition of the heuristics. For example, if an unbroken set of $C(i, i + 4) = 1$ exists, the local structure is a helix, and therefore for all $|j - i| > 4$ in that segment, $C(i, j) = 0$. The question is whether one can mine these rules automatically.

Finding Heuristics by Data Mining

Consider the contact map for the parallel beta sheet shown in Table 24.1. We can discover “positional” rules, that is, the heuristic geometric rules by considering an appropriate neighborhood around each contact $C(i, j)$ and noting the relative coordinates of the other contacts and noncontacts in the neighborhood, conditional on the local structure type(s). Consider a lower one-layer (denoted LL1) neighborhood for a given point, $C(i, j)$. This includes all the coordinates within $i + 1$ and $j + 1$, i.e., each point has three other points in its LL1 neighborhood, namely $C(i, j + 1)$, $C(i + 1, j)$, and $C(i + 1, j + 1)$. If we repeat this process for each point, we obtain a database that can be mined for frequent combinations.

For instance, looking at the parallel beta sheet submatrix in Table 24.1, we would get the set $C(i, j) = 1$, $C(i + 1, j) = 0$, $C(i, j + 1) = 0$, $C(i + 1, j + 1) = 1$ for each contact. If one were to do this for many other proteins one would find the pattern “If $(C_i, j) = 1$ and $C(i + 1, j + 1) = 1$, then $C(i, j + 1) = 0$ and $C(i + 1, j) = 0$,” among several others. This is the same rule deduced by hand previously.

Other patterns can be found by defining an appropriate neighborhood, which can be t layers thick (where t is the maximum coordinate difference between points) and can encompass all points within t or some subset of that region. From each of these we can construct examples that can be mined for frequent patterns to obtain heuristic contact rules. We can also incorporate sequence information to mine more complicated patterns. Techniques are currently being developed to mine such heuristic rules of contact automatically.

Rules for Pathways in Contact Map Space

Currently, there is no strong evidence that specific nonnative contacts (i.e., those not present in the final 3-D structure) are required for the folding of any protein. Many simplified models for folding, such as lattice simulations, tacitly assume that nonnative contacts are “off pathway” and are not essential to the folding process. Therefore, I choose to encode the assumption of a “native pathway” into the algorithmic approaches. This simplifying assumption allows definition of potential folding pathways based on a known 3D structure. We may further assume that native contacts are formed only once in any given pathway.

The formation of a contact between two amino acids in the chain, an elemental part of a folding pathway, forms a cycle and incurs the loss of configuration entropy in that piece of chain. Loss of configuration entropy (i.e., the ordering of the backbone) is the main opposing force to protein folding. Entropy loss must be balanced by favorable energetic interactions, such as hydrophobic contacts or hydrogen bonds. We may impose a limit onto our pathway model by assuming that any new contact must occur within S_{max} residues of a contact that is already formed. In other words, we assume that $U(i, j) \leq S_{max}$, where $U(i, j)$ is the number of “unfolded” residues between i and j . Intervening residues are “folded” when a contact forms. This will be called the “condensation rule.” In computational experiments using lattice models of folding, the pathways that follow this rule are dominant.

A pathway in contact map space consists of a time-ordered series of contacts. The pathway is initiated by high-confidence I-sites, and thereafter it follows a tree-search format (Fig. 24.8). Each level of the tree is the addition of a contact that satisfies the condensation rule. A maximum of M branches can be selected based on the energy. In addition, contacts that are not physically possible can be rejected using heuristics or mined rules for physicality. Identical branches (same set of contacts, different order), of course, can be merged.

Note that the energy criterion includes an entropic penalty proportional to $U(i, j)$, the number of unfolded residues between i and j . If $U(i, j) = 0$, there is no entropic penalty, because it implies that a previous contact has already cyclized that segment. We believe that the rules for folding in contact map space are consistent with the accepted biophysical theories of folding, while confining the search to a greatly simplified and reduced space. Methods to discover the folding pathways in the contact map space are currently being developed. It is worth observing that although the structure prediction problem has attracted a lot of attention, the pathway prediction problem has received almost no attention. However, the solution of either task would greatly enhance the solution of the other; hence, it is natural to try to solve both of these problems within a unifying framework. The current work is a step toward this unified approach.

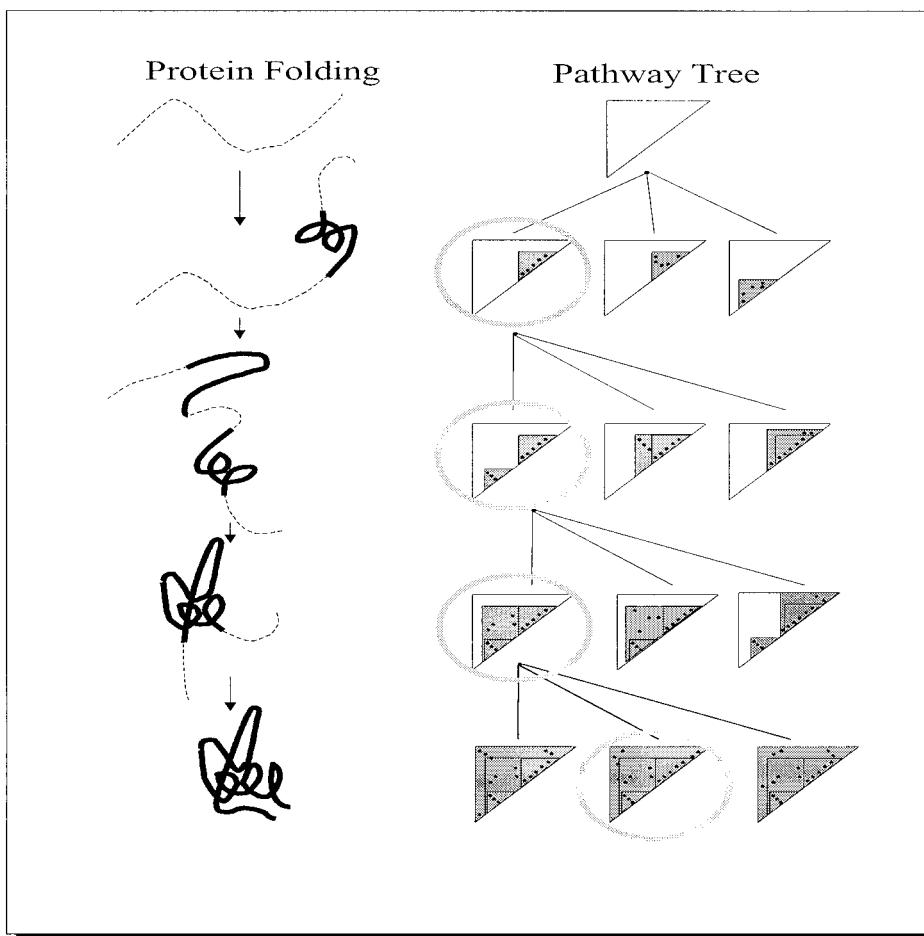


FIG. 24.8. Folding pathways in contact map: Large triangles represent the contact map, initially empty. Each branch defines a region of the contact map by setting pairs of amino acids to be in contact (dots) or not (space). Each node in the tree corresponds uniquely to a 3-D structure (left), where the dotted lines represent segments of the chain that have undefined structure.

SUMMARY

In this chapter I illustrated two main applications of data mining in protein structure prediction, namely, classifying contacts versus noncontacts and recognizing common 2-D patterns in protein-like contact maps. For the former I described data mining techniques to predict 3-D contact potentials among protein residues using a hybrid approach. We first used HMMs to extract folding initiation sites and then applied frequent closed itemset mining to discover contact potentials. The new hybrid approach achieved accuracy better than any reported previously.

For the latter problem, I described a novel string encoding and hashing technique to extract all the dense submatrices by sliding a 2-D window across the contact map. We discovered common nonlocal patterns using a dynamic density threshold and several pruning techniques.

Using this approach we were able to extract some typical interactions that occur in “physical” protein-like contact maps. These patterns can be used to refine the contact map predictions to remove nonphysical predictions. Currently, a library of such nonlocal interactions between different secondary structures is being compiled. This library would be analogous to the I-sites library, but whereas the I-sites library records the common motifs for short contiguous segments (3–19 residues), the new library will record interactions between noncontiguous segments.

I would like to close by saying that, in general, data mining approaches seem ideally suited for bioinformatics, because it is data-rich but lacks a comprehensive theory of life’s organization at the molecular level. The extensive databases of biological information create both challenges and opportunities for developing novel Knowledge Discovery and Data Mining (KDD) methods. I hope that this chapter has been successful in highlighting some aspects of how mining can be applied in the protein prediction domain.

REFERENCES

- Altschul, S., Madden, T., Schaffer, A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. (1997). Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25, 3389–3402.
- Anfinsen, C., & Scheraga, H. (1975). Experimental and theoretical aspects of protein folding. *Advances in Protein Chemistry*, 29, 205–300.
- Bystroff, C., & Baker, D. (1998). Prediction of local structure in proteins using a library of sequence-structure motifs. *Journal of Molecular Biology*, 281, 565–577.
- Bystroff, C., Thorsson, V., & Baker, D. (2000). HMMSTR: A hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*, 301, 173–190.
- Fariselli, P., & Casadio, R. (1999). A neural network based predictor of residue contacts in proteins. *Protein Engineering*, 12, 15–21.
- Hobohm, U., & Sander, C. (1994). Enlarged representative set of protein structures. *Protein Science*, 3, 522–524.
- Levinthal, C. (1968). Are there pathways for protein folding? *Journal of Chemistry and Physics*, 65, 44–45.
- Moult, J., Pedersen, J., Judson, R., & Fidelis, K. (1995). A large-scale experiment to assess protein structure prediction methods. *Proteins*, 23, ii–v.
- Noltning, B., Golbik, R., Neira, J., Soler-Gonzalez, A., Schreiber, G., & Fersht, A. (1997). The folding pathway of a protein at high resolution from microseconds to seconds. *Proceedings of the National Academy of Sciences USA*, 94, 826–830.
- Olmea, O., & Valencia, A. (1997). Improving contact predictions by the combination of correlated mutations and other sources of sequence information. *Folding and Design*, 2, S25–S32.
- Simons, K., Kooperberg, C., Huang, E., & Baker, D. (1997). Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *Journal of Molecular Biology*, 268, 209–225.
- Thomas, D., Casari, G., & Sander, C. (1996). The prediction of protein contacts from multiple sequence alignments. *Protein Engineering*, 9, 941–948.
- Vendruscolo, M., Kussell, E., & Domany, E. (1997). Recovery of protein structure from contact maps. *Folding and Design*, 2, 295–306.
- Wolf, Y. I., Grishin, N. V., & Koonin, E. V. (2000). Estimating the number of protein folds and families from complete genome data. *Journal of Molecular Biology*, 299, 897–905.
- Zaki, M. J., & Hsiao, C.-J. (2002). CHARM: An efficient algorithm for closed item set mining. In *Second SIAM International Conference on Data Mining* (pp. 457–473), SIAM Press, Philadelphia, PA.
- Zaki, M. J., Jin, S., & Bystroff, C. (2000). Mining residue contacts in proteins using local structure predictions. In *IEEE International Symposium on Bioinformatics and Biomedical Engineering* (pp. 168–175), IEEE computer press, Los Alamitos, CA.
- Zhao, C., & Kim, S.-H. (2000). Environment-dependent residue contact energies for proteins. *Proceedings of the National Academy of Sciences*, 97, 2550–2555.

25

Mining Customer Relationship Management (CRM) Data

Robert Cooley
KXEN Inc.

Introduction	597
Data Sources	599
Data Types	599
E-Commerce Data	601
Data Preparation	604
Data Aggregation	605
Feature Preparation	607
Pattern Discovery	608
Pattern Analysis and Deployment	610
Robustness	610
Interestingness	611
Deployment	611
Sample Business Problems	612
Strategic Questions	612
Operational Questions	613
Summary	615
References	616

INTRODUCTION

Retail enterprises collect data about their customers and their interactions with customers from a number of different sources. These data are a natural target for data mining, and indeed one of the first applications of “modern” data mining was *market-basket analysis* (Agrawal

& Srikant, 1994)—discovering which products are typically purchased together. The data comes from a variety of sources and in a myriad of formats, but ultimately it can be broken down to two basic types—*static* data and *event* data. Static data are attributes about customers such as demographics, financial information, or interest profiles. Events are interactions with customers, such as e-mails, phone calls, transactions, or Web clicks. When it comes to data mining these sources, the goal is to optimize the ratio between profitable and unprofitable events for a given set of customers. Certain events, such as product purchases, generate revenue and hopefully profit, whereas others do not, such as customer service phone calls. Ideally, data mining and other forms of analysis show a company how to increase revenues and/or decrease expenses. The most straightforward example of this is response modeling for a direct mail campaign. By predicting who will respond to a campaign, a company can decrease the cost of a campaign while hopefully maintaining a similar overall response. Even a task such as maximizing customer satisfaction can be viewed in terms of event ratios. Satisfied customers are less likely to generate unprofitable events such as complaints to a call center and more likely to reward the company with future profitable events. This chapter explores how data mining of various customer data sources can be used to maximize the profitable event ratio, with a particular emphasis on E-commerce (click-stream) data.

Although the rise and fall of E-commerce has been faster and larger than other new technologies, it has essentially followed the classic hype curve, in which the initial expectations far exceed what the technology can deliver, followed by a precipitous drop when the prevailing opinion is pessimistic, and finally a convergence of reality and hype into realistic expectations. The promise of E-commerce quickly outstripped reality, and for a period in the late 1990s “traditional” sales channels such as physical stores or printed catalogs were deemed to be obsolete. On the tail end of the Internet bubble, many consider E-commerce to have been a fad, improperly equating all of E-commerce with a few of the more visible commercial failures. Although the business model of many of the so-called dot-coms was indeed faulty, the value of E-commerce as a customer touchpoint is indisputable. More important, from a data mining perspective events collected from E-commerce sites are unique with respect to the types of customer behaviors that are recorded. *Channels* or *touchpoints* such as retail stores and catalog sales only record purchasing behavior—what is actually bought. E-commerce allows an organization to record a much broader range of behaviors, such as browsing or selecting items for purchase, regardless of whether the items are eventually bought. No other touchpoint provides such a complete record of both positive and negative customer behavior. Does this make E-commerce data inherently superior to data collected from other touchpoints? Not necessarily, because the additional information comes at a price. A common misperception is that data collected from an E-commerce site is “free,” and that any data mining enterprise automatically will be more cost-effective than an equivalent undertaking with traditional data sources. In reality, due to the sheer volume of data and the level of noise, E-commerce mining can be far more costly and complex. Whether the results justify the increased costs depends completely on the context of the business questions being asked and how the results are applied.

Web usage mining is the general application of data mining techniques to Web clickstream data to extract usage patterns (Cooley, 2000). E-commerce mining is a subset of Web usage mining with the goal of maximizing the profit from customer interactions. For enterprises with several customer touchpoints, E-commerce mining can be just one facet of an overall customer relationship management (CRM) mining system. Figure 25.1 shows the high-level steps involved in knowledge discovery for CRM mining. The raw data is gathered from a number of sources and imported into a CRM data warehouse through a set of extraction, translation, and loading (ETL) processes. *Data preparation* pulls data from the warehouse and gets it into a format suitable for data mining. Then *pattern discovery* uses data mining

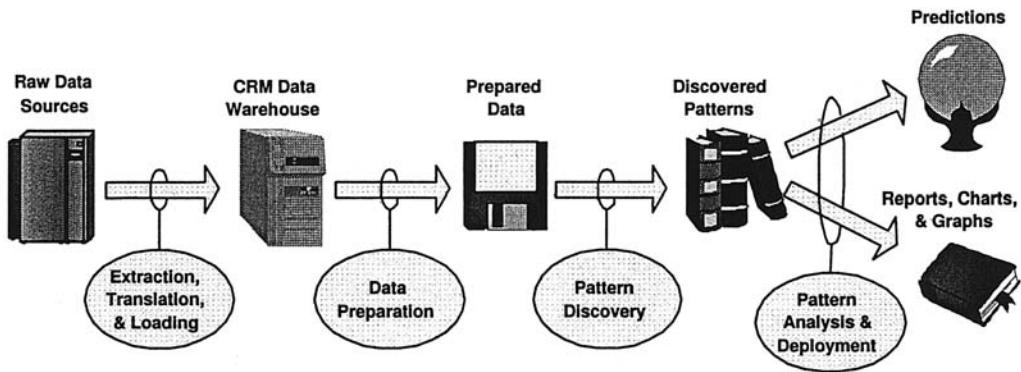


FIG. 25.1. CRM data mining process.

or machine learning algorithms to build models or discover relationships in the data, and *pattern analysis* transforms the patterns into usable knowledge. Finally, patterns that have been deemed to be actionable are deployed as models in an operational system. It is this final phase of deployment that directly affects the event ratio and creates a return on investment.

The rest of this chapter is organized as follows. The second section introduces the potential sources of data, the third section presents the major data preparation tasks that are necessary for CRM mining, the fourth section briefly discusses various pattern discovery techniques that can be used to discover knowledge, the fifth section explores methods for analyzing the discovered patterns, and the sixth section gives some CRM motivated examples. The final section provides a summary.

DATA SOURCES

The launch point for CRM mining is a data warehouse. A CRM data warehouse is populated from a number of sources, including point-of-sale (POS) databases, Web sites, call centers, e-mail responses, product databases, customer databases, and third-party databases. Obviously, the ETL effort to combine all of these data sources into a single clean and consistent schema is immense. For the purposes of this chapter, it will be assumed that these ETL challenges have been overcome, and the task at hand is now purely one of data mining. The rest of this section discusses the types of data that are typically encountered in a CRM data warehouse. The one exception is the discussion of E-commerce data, which starts from the raw logs themselves. The reason for this is that there are specific challenges involved with the ETL process for E-commerce data that often are overlooked.

Data Types

The two basic types of data associated with CRM mining have already been introduced—static and event data. In a data warehousing context these types often are referred to as fact and dimension tables, respectively. The division between these data types is not always a clean one, and a given attribute may act as static data in one context and event data in another. For example, an organization may choose to keep track of its customers' stated preference for a customer service channel. Although a given customer can have only one preferred channel, this channel may change over time, and a preference history can be recorded. When building

TABLE 25.1
Common CRM Data Sources

<i>Static</i>	<i>Event</i>
Demographic	Purchase
Financial	Web click
Profile	Phone call
	E-mail
	Store visit
	Physical mailing

a campaign response model, the current preference may be used as a static input to the data mining process, but for predicting customer churn a history of preference change events might be warranted. Table 25.1 lists some of the more common examples of static and event data found in a CRM data mining setting.

An important distinction between the two types of data is that static data describes a customer, and event data describes a customer's behavior. Ultimately, it is the behavior that is of interest, because, as stated earlier, the goal of CRM mining is to modify the ratio of positive and negative behaviors. Most static information, such as age or postal code, are merely surrogates for the actual information of importance—customer behavior. In studies that compare behavior with static information (Ehrenberg, 1988), past customer behavior was consistently shown to be superior for predicting future behavior. Why, then, is static information such as demographics often seen as the data of most value? One of the main reasons is the relative volume of data that is necessary to perform reliable data mining. A single attribute such as postal code can be a reliable predictor of many different customer behaviors. The number of events necessary to get the same prediction rate is often orders of magnitude higher. However, as more events are added to a data set, prediction error will generally decrease to the point at which it is better than with the static data. Also, for many types of analysis, event data must be aggregated before it can be used. The common types of event aggregation used for CRM mining will be addressed in a later section and a full discussion can be found in chapter 14. The use of event data, therefore, is more expensive due to the quantity and preprocessing requirements and also results in a higher-dimensional problem that must be solved. Finally, event data is not necessarily compatible across different domains. If a company wants to run a television advertising campaign, a description of its potential customers based on their buying behaviors is not useful for choosing which stations should run the ad. Different television stations and programs typically describe audiences in terms of demographic attributes. The company would need a demographic description of its customers to determine the right advertising strategy. Based on concerns about resources, scalability, and high dimensionality, static data has become the “default” for many CRM mining systems. However, as hardware costs decrease, processing power grows, and new methods for handling high-dimensional problems are created, the value of mining patterns based on true behaviors is increasing to the point at which it is preferable to static data for many situations.

To illustrate many of the points made in this chapter, a running example of a retail enterprise is used. This example enterprise, “MyWidgets, Inc.”, has a physical retail store, Web site (www.mywidgets.com), and a call center that handles both sales and support. Figure 25.2 show the data warehouse schema for MyWidgets. The notation for the relationships between tables uses combinations of lines, circles, and “crows feet” to indicate cardinality. A line and circle indicate zero or one rows, a double line indicates one row, a crow foot and circle indicate zero to many rows, and a crow foot and line indicate one to many rows in the relationship. For example, every customer has one or more purchases (otherwise he or she would not be

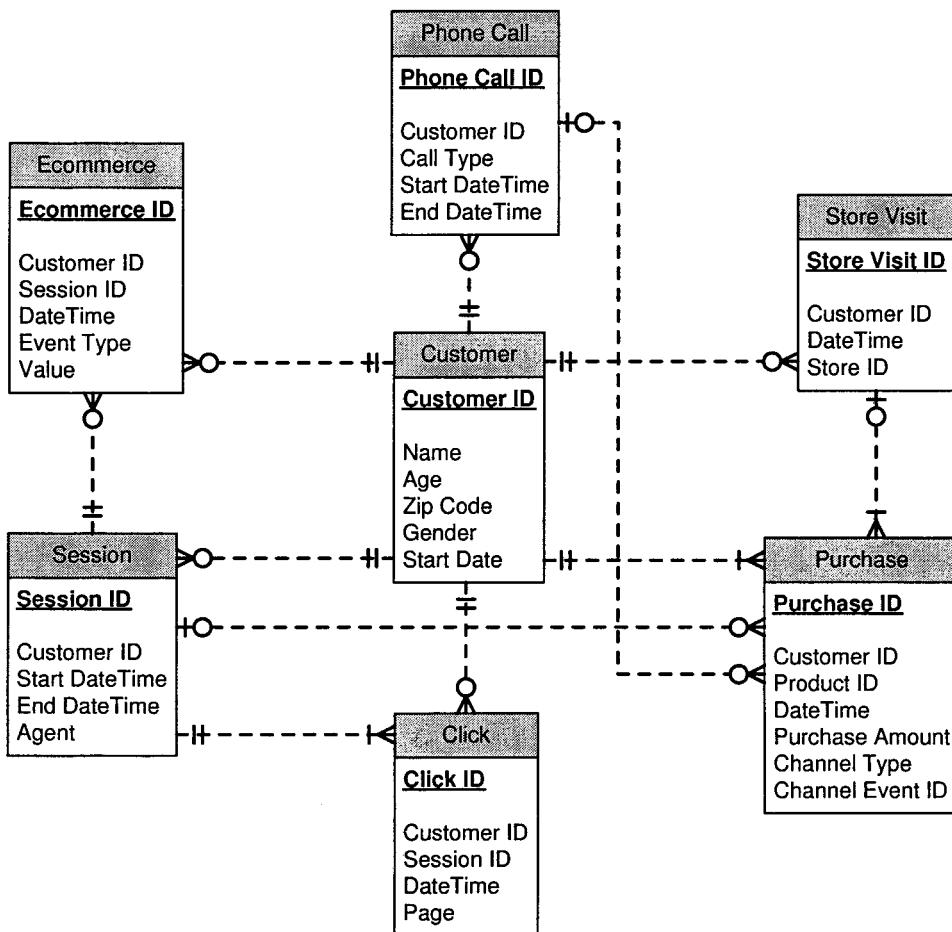


FIG. 25.2. MyWidgets data warehouse schema.

a customer), but a purchase must be associated with one and only one customer. The central table is the “Customer” table and there are six associated event tables. Each customer visit to the MyWidgets.com Web site is captured in the “Session,” “E-commerce Event,” and “Click” tables. Any calls to the phone center, either for customer service or sales, are captured in the “Phone Call” table. Purchases from the Web site, phone center, or a physical store are captured in the “Purchase” table. The “Store Visit” table has a record of every visit to a physical store that resulted in a sale. Again, unlike the Web site and call center, there is no automated method of tracking visits to a store that do not result in a purchase.

E-Commerce Data

The data collected from E-commerce systems is predominantly clickstream, or Web usage data. A clickstream is defined as a sequence of page views that are accessed by a user. A *page view* consists of all of the files that contribute to the browser presentation seen as the result of a single mouse “click” or action of a user. In addition to clickstream data, a Web site generates Web content and Web structure data. Web content data is the actual information that is contained in the “pages” that make up a Web site. The content data is usually made up of text and graphics,

but can also include multimedia content. Web structure data comes from the hypertext links between the various pieces of content. As it turns out, both Web content and Web structure data are critical for E-commerce ETL, as discussed later. Although profile information about customers can be collected through a Web site, the ETL requirements for this type of data are not fundamentally different from the same data collected through a different channel such as a call center.

E-Commerce Events. Although a clickstream gives an accounting of each action taken by potential customers on a Web site, this level of events is not necessarily the most useful for CRM mining. Tracking Web site usage in terms of interactions with *products* can be far more beneficial. These types of events are sometimes referred to as *E-commerce events* to distinguish them from clickstream events. There are two important differences between E-commerce and clickstream events:

- **Cardinality.** There is not necessarily a one-to-one relationship between clickstream and E-commerce events. One clickstream event may be responsible for any number of E-commerce events. A single click can lead to several products being displayed on a page, resulting in more than one “product-view” event. By the same token, a click can lead to a page without any product information, yielding no E-commerce events.
- **Standardized definitions.** The definition of a clickstream event and the associated terminology is fairly standardized. A single user action results in a single event. However, the definition of an E-commerce event is set by each individual enterprise. What it means to view a product and whether it is an event that should be tracked at all is something that must be decided by the organization responsible for a given E-commerce Web site.

Examples of common E-commerce events are *product views*, *product click-throughs*, *product adds*, and *product purchases*. A product view is typically considered to be an image or text description displayed on a Web page. A user may or may not have explicitly requested a product view. For instance, products displayed on a home page or in a side frame are usually chosen by the application server, not the user. As will be seen in a later section, personalizing the displays of these “nonrequested” products to increase up-sells and cross-sells is a natural target for CRM mining. A product click-through is an explicit request for more information. An image of a product or text link often leads to a Web page with detailed information about a product. A product click-through can be seen as a sign of interest in a particular product by a customer. A product add is an indication of an intent to purchase by placing a product into a virtual shopping basket. If an E-commerce site is analogous to a physical store, product views are all of the products visible in the aisles a customer walks down, and click-throughs are any product a customer picks up but does not necessarily put in a shopping cart. In addition to the E-commerce events mentioned above, a Web site may define any number of other events, such as bids for auction sites or orders for brokerage sites. There also can be multiple levels of each type of event. For example, a Web site may have a page that gives a quick summary of a product and a second page with the full product details, leading to two types of product click-through events. As another example, purchasing a product may require three separate “checkout” pages. Ideally, a CRM data warehouse will contain both E-commerce and clickstream events, but if forced to choose between the two, E-commerce events are far more valuable for CRM mining than a full clickstream.

E-Commerce ETL. Figure 25.3 depicts a simplified view of the Internet as it pertains to the World Wide Web. As shown, clickstream data can come from a client-level log, proxy-level log, or server-level log. The most common source for clickstream data are Web

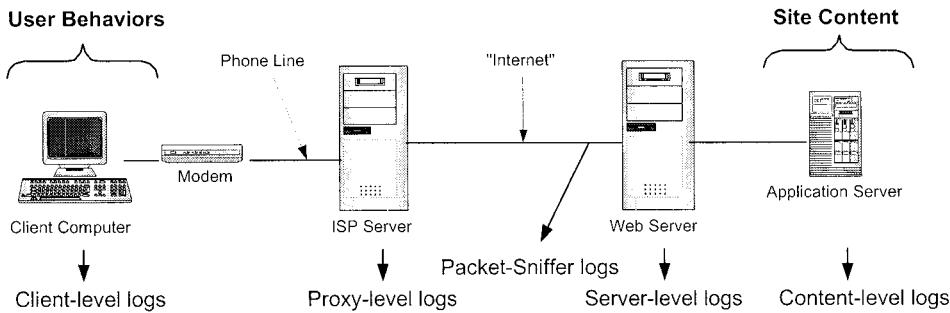


FIG. 25.3. Simplified World Wide Web.

server logs, which are collected at the server level. However, Web server logs were originally created to assist Web site administrators in debugging a site, and typically contain a line for every HyperText Transfer Protocol (HTTP) request that comes to a site. This means there is a line for every graphic, HyperText Markup Language (HTML) page, Common Gateway Interface (CGI) request, and so forth. When the Web started to gain popularity in the mid 1990s Web sites were relatively simple, and a server log was a reasonable picture of user behavior. Tools were created in both the research and commercial environments to analyze or mine these logs for usage patterns (Buchner & Mulvenna, 1998; Cooley, Tan, & Srivastava, 1999; Spiliopoulou & Faulstich, 1998; Wu, Yu, & Ballman, 1998). However, as the sites grew in complexity, and *application servers* were created to create *dynamic content* on the fly, the mapping between Web server logs and true customer behaviors became more and more tenuous. User behaviors occur at the browser or *client level*, and the site content is created by the application server. A Web server is no longer the end of the pipeline and is often unaware of important information maintained by the application server. An application server can maintain information about users, sessions, and the content being requested that is invisible to the Web server. For example, if an application server chooses to use the same file name for every dynamic page, from the point of view of a Web server the same content is being served over and over. Although information contained in CGI variables and HTTP header information can be used to disambiguate some of these problems, methods such as those described in (Cooley, 2000) are not always reliable.

In essence, most E-commerce log analysis systems are currently solving a much harder problem than necessary. An application server is able to accurately describe all events for which it is responsible. A clean and consistent event log can be recorded by an application server that is far superior to the information from a Web server log. More important, a true E-commerce event log can be created only by an application server. The information available to other logging methods is not guaranteed to be sufficient for E-commerce events. However, one drawback of collecting data from an application server is a potentially incomplete record of events. Browsers and proxy servers tend to keep a cache of files that have been requested recently. This means that pages can be viewed without recording an event in the application or Web server log. The caching of pages has become less of a problem with dynamic content, but an E-commerce ETL process must still take the possibility of an incomplete log into account. The major advantages and disadvantages of the various E-commerce data collection options are listed in Table 25.2.

Click-Stream Preprocessing. The practical difficulties in performing preprocessing are a moving target. As the technology used to deliver content over the Web changes, so do the preprocessing challenges. Although each of the basic preprocessing steps remains

TABLE 25.2
Web Data Collection Options

<i>Collection Point</i>	<i>Advantages</i>	<i>Disadvantages</i>
Client	Complete record of events Potentially multisite	Privacy concerns Can be disabled Not automatically created Can't track E-commerce events
Proxy	Multiuser & multisite	Large data volume Potentially incomplete Can't track E-commerce events
Packet-Sniffer	Can process HTTP headers	Large data volume Potentially incomplete Can't track E-commerce events
Web server	Widely available	Potentially incomplete Can't track E-commerce events
Application server	Can track E-commerce events Access to content information	Not automatically created Potentially incomplete

constant, the difficulty in completing certain steps has changed dramatically as Web sites have moved from static HTML served directly by a Web server to dynamic scripts created from sophisticated application servers and personalization tools. Both client-side tools (e.g., browsers) and server-side tools (e.g., application servers) have undergone several generations of improvements since the inception of the Web. *Data cleaning* is a site-specific step that involves mundane tasks such as merging logs from multiple servers and parsing the log into data fields. Typically, graphics file requests are stripped out at this stage. Next, user and *session identification* is performed through one of several methods, the most common being the use of cookies, user registration, or embedded session identifications in the uniform resource locators (URLs). *Page view identification* determines which page file requests are part of the same page view and what content was served. When frames are used, a single page view can consist of several HTML files. This step is highly dependent on knowledge of the Web site structure and content. Finally, *path completion* fills in page references that are missing due to local browser caching. This step differs from the others in that information is being added to the log. Each of these tasks is performed to create a server session file that is stored in the CRM data warehouse. In the case of MyWidgets, meta data for the Web sessions is stored in the “Session” table, as well as the clickstream in the “Click” table. Note that the server session file is usually only an estimate of what actually occurred due to techniques that obscure the data collection such as proxy servers and caching that are common in today’s browsing environment. An overview of heuristics and algorithms that can be used to handle these problems are contained in (Cooley, Mobasher, & Srivastava, 1999). A detailed discussion on setting up and populating a clickstream data warehouse can be found in (Kimball & Merz, 2000).

DATA PREPARATION

By this point, all of the appropriate static and event data has been collected, extracted, translated, and loaded into a clean and consistent RDBMS schema. However, pattern discovery cannot be performed directly on the data in the warehouse. Different pattern discovery methods and algorithms require input data to be in different formats. Although there is no universally

accepted term for getting data ready for input into a specific data mining algorithm, this chapter refers to this process as *data preparation*, to distinguish it from the preprocessing tasks associated with ETL. Many algorithms need all of the information associated with a customer to be in a single “line” of data. Other algorithms need all of the strings converted to numbers or all of the numbers within a certain range.

Data Aggregation

Data aggregation is normally something that is associated with ETL, reporting, or online analytical processing (OLAP). In fact, data mining is often discussed in terms of working with *unaggregated* data. Although it is true that the static information is rarely aggregated, it is not uncommon to aggregate events. Methods such as classification, regression, and clustering typically require a single line or a fixed number of lines of data for each training example. There is usually more than one event associated with each customer for CRM mining. In the example schema, a customer can have several purchase events, E-commerce events, click events, session events, store events, or phone events. There are two major dimensions for aggregating CRM events—*time* and *value*.

An example of time-based event aggregation is shown in Fig. 25.4. In this case both the count and amount of purchases are summed up for three quarters, starting in Q3 of 1999. This type of aggregation can easily be performed and stored in the data warehouse, similar to what is done for some OLAP data cubes. However, there is a different form of time-based aggregation that is potentially more useful but difficult to precompute. This is what is referred to as *relative aggregation*. Aggregation based on a fixed set of dates, such as quarters in a calendar year, is not necessarily meaningful with respect to customers. New customers are added at different times and have different event histories. This is seen in Fig. 25.4, where none of the purchases for the two customers overlap. A relative aggregation works off of a variable date, such as the date a customer first purchased a product, or the date an offer went out by e-mail.

<u>Customer ID</u>	<u>Zip Code</u>	<u>Gender</u>	<u>Start Date</u>	<u>Customer ID</u>	<u>Product</u>	<u>Date</u>	<u>Total</u>
1001	94116	Male	1/1/00	1001	Widget A	1/1/00	\$95.79
1002	94103	Female	7/1/99	1001	Widget B	1/5/00	\$48.87
-----	-----	-----	---	1001	Widget C	1/5/00	\$503.49
-----	-----	-----	---	1002	Widget A	7/1/99	\$95.79
-----	-----	-----	---	1002	Widget D	10/5/99	\$219.98
-----	-----	-----	---	1002	Widget A	12/3/99	\$95.79
-----	-----	-----	---	-----	-----	-----	-----
-----	-----	-----	---	-----	-----	-----	-----

Cust. ID	Zip Code	Gender	# Events Q3 19 99	\$ Total Q3 19 99	# Events Q4 19 99	\$ Total Q4 1999	# Events Q1 2000	\$ Total Q1 2000
1001	94116	Male	0	\$0.00	0	\$0.00	3	\$648.15
1002	94103	Female	1	\$95.79	1	\$315.77	0	\$0.00
---	---	---	---	---	---	---	---	---

FIG. 25.4. Fixed time-based event aggregation.

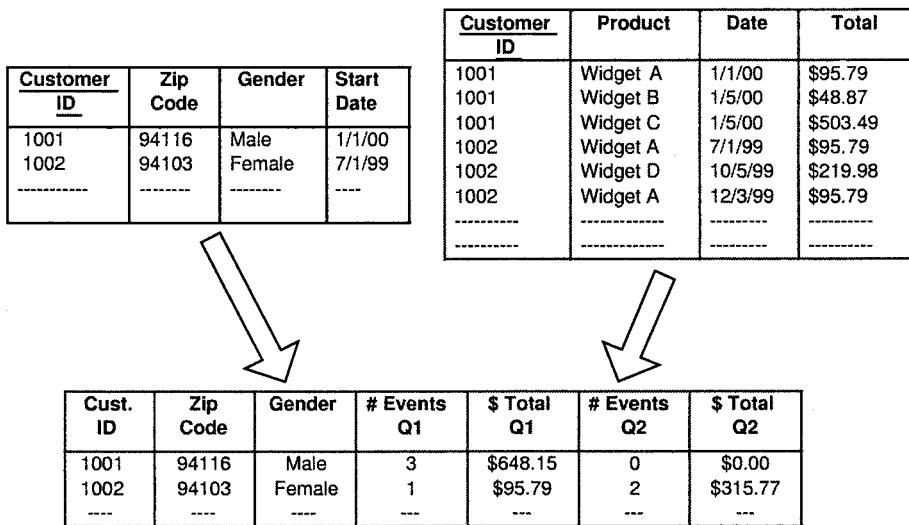


FIG. 25.5. Relative time-based event aggregation.

An example of relative aggregation is shown in Fig. 25.5. In this case the aggregation is based on the “start date” column in the customer table. The first two quarters of customer purchase history are shown. This type of aggregation is much harder to precompute and, more important, is very dependant on the business question being asked. The appropriate relative date and level of aggregation is likely to be different for various situations. As with data cubes, it is impossible to precompute every possible combination of aggregation levels and dates. As shown in both of the examples, in addition to a count of rows that fall into a given time period, a function can be applied to any continuous column. The “Total” column contains the sum of the purchase amount. Any number of functions could be applied to any of the continuous columns in the event table. Typically, functions that compute the sum, average, minimum, or maximum are applied.

An example of aggregation based on the second major dimension, *value*, is shown in Fig. 25.6. Here the values of a particular column or columns in an event log are used for the aggregation. In this case the count of each distinct value in the “Product” column becomes a separate column in the aggregated table. Any column that contains nominal or ordinal values is a candidate for value-based aggregation. Like time-based aggregation, the function is not limited to a simple count or sum.

In all of the examples discussed, one thing that becomes immediately clear is how quickly the number of dimensions for CRM mining can grow. Using value-based aggregation with a product catalog of 10,000 products would add 10,000 columns to the data mining process, assuming all 10,000 products actually appeared in the data being analyzed. Therefore, either a significant amount of feature space reduction must occur prior to the pattern discovery phase, or algorithms that can handle high-dimensional problems must be used. Feature space reduction techniques, such as information gain (Cover & Thomas, 1991) or principal component analysis (PCA) (Jolliffe, 1986), are an entire subject unto themselves and are discussed in chapter 15 and in Mitchell (1996). As mentioned earlier, this potential need for feature space reduction explains why event data often is overlooked when CRM mining is performed. Why spend time adding more dimensions with aggregation techniques if there is already a need to reduce the existing feature space?

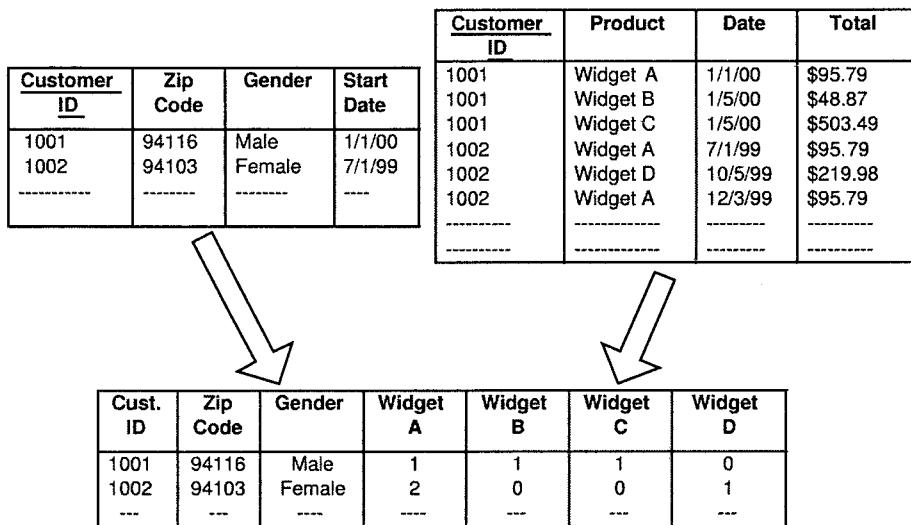


FIG. 25.6. Value-based event aggregation.

Feature Preparation

The final step before actually performing pattern discovery is *feature preparation*. Most data mining algorithms are very particular about the format of the input data. This means that missing, out of range, and outlying values must be taken care of, as well as the proper encoding of strings, nominal features, and ordinal features. Missing values usually need to be filled in with some sort of default value. Out-of-range values, such as -10 for an age, should be removed or grouped into a special category. Outliers, values that are not necessarily wrong but so far astray that they are suspicious, typically need to be removed or grouped together as well. When an algorithm requires the input variables to be numeric, an encoding strategy must be used. When a numeric target variable is available, this is commonly used to guide the encoding. Nominal variables (strings or numeric) can be replaced by the mean or median value of the target for each distinct value. Another common strategy is to use the rank order of the target mean for each value. Examples of these encoding strategies are shown in Table 25.3. The third encoding strategy shown in the table is referred to as *disjunctive Boolean*. This is a common

TABLE 25.3
Example Encoding Schemes

<i>Original Column</i>	<i>Target Column</i>	<i>Target Mean</i>	<i>Target Mean Rank</i>	<i>Disjunctive Boolean</i>		
				<i>Red</i>	<i>Green</i>	<i>Blue</i>
Red	5	6	1	1	0	0
Blue	10	11	2	0	0	1
Green	20	19	3	0	1	0
Blue	12	11	2	0	0	1
Blue	11	11	2	0	0	1
Green	18	19	3	0	1	0
Red	7	6	1	1	0	0
Red	6	6	1	1	0	0
Blue	11	11	2	0	0	1

scheme when a numeric target does not exist. Each distinct value in the column is broken out into a separate column with a zero or one indicating which value was assigned to a specific row. Like the aggregation schemes previously discussed, disjunctive boolean encoding can greatly increase the dimensionality of a data set.

Also commonly included in feature preparation is the notion of variable compression or *binning*. Instead of passing every distinct value of a continuous variable to a data mining algorithm, labels can be assigned to different ranges of values, converting the column into a nominal feature with far fewer distinct values than the original data. The same process can be applied to nominal or ordinal features. Because many algorithms scale with the number of distinct values, or *cardinality* of a feature, fewer distinct values can greatly reduce the processing time. In addition, by removing some of the noise from a data set, variable binning can increase the generalization performance of an algorithm. As with feature space reduction, each of the potential feature preparation steps is a topic unto itself. Chapters 14, 15, and Pyle (1999) provide in-depth discussions of many of the topics discussed in this section.

PATTERN DISCOVERY

Once the appropriate data has been pulled out of the data warehouse and properly prepared, any number of data mining methods can be applied to answer just about any business question that comes to mind. This section briefly discusses seven of the more popular methods—classification, clustering, regression, sequential patterns, association rules, time series, and collaborative filtering. Detailed discussions of the methods and algorithms can be found in Part I of this book.

When discussing pattern discovery, a lot of confusion exists about terminology. As shown in Fig. 25.7, there are four major layers involved in pattern discovery, the top layer being the actual *business question*. The base layer contains *algorithms*. There are thousands of data mining, machine learning, artificial intelligence, and numeric algorithms that can be applied to pattern discovery. Some classes of algorithms, such as neural networks or Markov models, are broad enough to merit separate discussions (such as Chapters 3 and 11). However, most algorithms are discussed in terms of general *methods*, which make up the second layer of

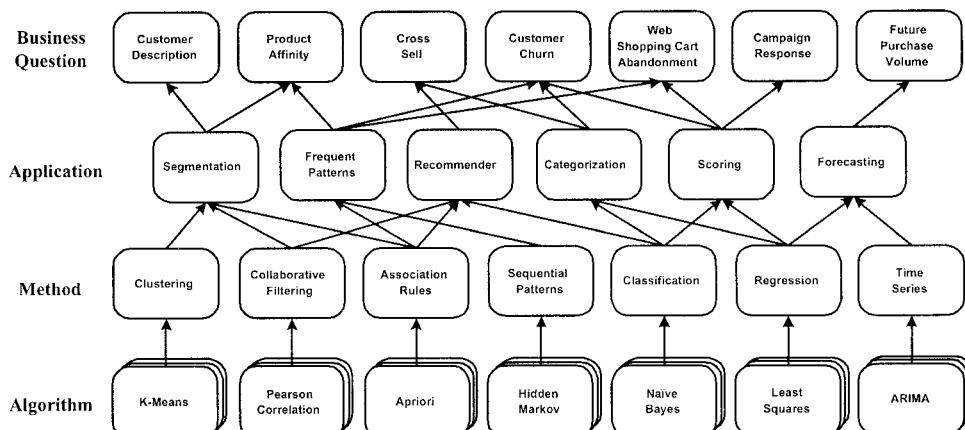


FIG. 25.7. Pattern discovery layers.

Fig. 25.7. A method is defined by the type of output obtained as a result of processing input data. The details of how the data is processed are not relevant when discussing methods. As shown, any number of algorithms can be used for a given method. For example, clustering could be performed using the *k*-means (Forgy, 1965), CLARANS (Ng & Han, 1994), or BIRCH (Zhang, Ramakrishnan, & Livny, 1996) algorithms, just to name a few. The third layer is the *application* layer. Applications wrap a method (or several methods) in business logic to solve a particular type of business question. For example, a categorization application answers questions about to what category a particular example belongs. The obvious choice of method for this application would be a classification method, but there is no reason a regression method could not be used instead. The methods simply output a score, probability, or yes/no for each possible category. A categorization application must handle a threshold for continuous outputs, the possibility of multiple categories, hierarchical categories, and so forth. As shown, a given business question can be answered by a number of different applications, which in turn can be solved by a range of methods. The connections and values in each layer of Fig. 25.7 are not meant to be an exhaustive list, but simply provide examples.

Classification methods (Friedman, 1994; Mitchell, 1996; Quinlan, 1993) determine how to assign examples to a set of classes. A training data set with the class labels already present is required for a classification method to learn. Clustering methods (Kaufman & Rousseeuw, 1990; Ng & Han, 1994) are used when a description of the various groups in a data set is required, and there is not necessarily a predefined set of class labels. Once a clustering is performed, the clusters often become the labels for a subsequent classification. Regression methods (Breiman, Friedman, Olshen, & Stone, 1984; Friedman, 1991) try to predict a continuous output by building a function that is an estimate of the “true” underlying pattern in the data. As previously mentioned, regression methods can be used for a binary classification application and are also commonly used for scoring and forecasting. Like regression, time-series methods predict continuous values, but do so over time, typically modeling both trends and cyclic behaviors in addition to the actual values. Both association rules (Agrawal & Srikant, 1994) and sequential patterns (Mannila, Toivonen, & Verkamo, 1995; Srikant & Agrawal, 1996) describe events that occur frequently in a data set. Association rules find correlations between items in a data set without taking order or time into consideration. For sequential patterns the order of the occurrence of the events is taken into account, as well as the actual values of the events.

A variant of clustering that is very useful for CRM mining is *targeted clustering*, sometimes referred to as supervised clustering. Because clustering is considered to be an *unsupervised* method, this chapter will use the term targeted clustering to avoid confusion. Targeted clustering groups the data with respect to a target variable. With an untargeted clustering method the best groups are found according to some generic distance metric that considers each dimension to be equally important. Although these groups can be differentiated from each other based on at least one of the features, such as age, gender, or behaviors, the groups may not be different in a meaningful business context. If all of the groups have the same average customer lifetime value or all tend to buy the same set of products, this is not necessarily useful. Targeted clustering attempts to find groups that are different with respect to the value of a target variable. This can be done by increasing the weight of the target variable in the distance metric or encoding the variables with respect to the target. In the extreme case, when the target is the only variable included in the distance metric, targeted clustering closely resembles classification. The important distinction between targeted clustering and classification is that there are no predefined categories for targeted clustering.

Collaborative filtering methods (Resnik, Iacovou, Suchak, Bergstrom, & Riedl, 1994; Shardanand & Maes, 1995) analyze events to find sets that are similar. In a CRM context,

events are usually grouped by customer. For each customer in the data set collaborative filtering methods find a group of other customers that are similar with respect to the occurrences in the event logs. Once the group is formed, events can be ranked by how popular they are across the entire group. The output of a collaborative filtering method can be both the customers that are similar and the event rankings. For example, if the events are product purchases, the event rankings can be used for product recommendations. Any product that is highly ranked and has not already been purchased by a customer can be recommended. The idea is that if other customers with similar purchase histories have bought a particular product, it is likely the first customer will buy that product as well. Collaborative filtering is similar to clustering in that it discovers natural groups in a data set. The important difference in collaborative filtering is that there is a distinct group formed around each item (customer) in the data set. In effect each customer is the center of a unique group, and group memberships are not necessarily transitive. Customer A may appear in Customer B's group, but not vice versa. This makes collaborative filtering methods ideal for personalized recommender system applications.

PATTERN ANALYSIS AND DEPLOYMENT

As shown in Fig. 25.1, the discovery of patterns is not the final step for CRM mining. Although the patterns themselves may be the desired outcome for some questions, in most cases the patterns must be applied to new data. Pattern analysis includes simple reporting, graphical displays, and even further computations on the patterns. For patterns that will potentially be deployed, one of the most important analysis techniques is evaluating the *robustness*. A measure of robustness indicates how well a pattern or model will perform on new data. For patterns that are merely descriptive and will not be used on new data, a measure of interestingness is often useful as well as robustness.

Robustness

Whether the discovered pattern is a classification, clustering, or some other application, one of the most important metrics for CRM mining is how well the pattern will perform on new data. A classification model that is 90% accurate on existing data but only 50% accurate on new incoming data is not necessarily useful, especially if 70% accuracy is needed to realize a profit for a particular scenario. A model that is 80% accurate on existing data and 80% accurate on new data is probably preferable. The measure of the relationship between training error and deployment or test error is referred to as robustness. The more consistent the performance, the more robust the model. Even for patterns that are mainly used to describe a data set, such as clustering or association rules, it is important to determine whether the discovered patterns will hold in general or are specific to a particular data set. In statistics this often is referred to as finding the balance between *underfitting* and *overfitting*. As with many of the steps discussed in this chapter, measuring or guaranteeing robustness is a topic unto itself. One of the most common techniques is *cross-validation*. Here several models are built using different subsets of the existing data. The models are either compared or combined to try to verify that the model is a true pattern inherent in the data, instead of random noise. When looking for patterns that occur in only a small portion of the data, like association rules, it is easy to mistake random correlations for true patterns. Another way of ensuring robustness is to use a framework such as Vapnik's statistical learning theory (Vapnik, 1995) when discovering patterns.

Interestingness

When the discovered patterns are being used to get a description of the customers, a common problem is a lack of “interesting” patterns. The most frequent or highly correlated patterns tend to highlight relationships that are obvious or already known. Although confirmation of expected patterns is not completely useless, the real goal of CRM mining is to discover patterns that were previously unknown or unexpected that can then be leveraged to increase the ratio of profitable events. To ensure that all patterns of potential interest are discovered, the thresholds of many pattern discovery algorithms must be set low enough such that hundreds or thousands of patterns are output. Although the number of items to analyze has been reduced by several orders of magnitude, a list of several thousand patterns is only marginally more useful than the millions of lines of raw data. To make matters worse, the notion of what is interesting is very dependent on the goals of the analysis. This means that interestingness for CRM mining is not necessarily an objective measure, but is more subjective. The concept of *subjective interestingness* for data mining has been studied extensively in Cooley (2000), Liu, Hsu, and Chen (1997), and Padmanabhan and Tuzhilin (1998). Usually, the idea is to define what is known or expected. The interesting patterns are then the ones that do not meet the definition of an expected pattern.

Deployment

Finally, a predictive pattern or model that meets the acceptable robustness criteria must be deployed. Figure 25.8 depicts a typical deployment scheme for a new data set. The first two steps, ETL and data preparation, are identical to the standard CRM mining process. The only difference is that the target values are unknown. Once the new data is prepared, an existing model is used to generate predictions. Depending on the application, the predictions may be categories, clusters, scores, probabilities, or other continuous values.

In some situations, such as a call center, a single data point or datum must be scored, instead of entire set of data. This is shown in Fig. 25.9. A single datum does not need to go through a full ETL process. The need to calculate a prediction for a single data point is usually associated with a real-time situation. For example, a customer calls to purchase some items. Any cross-sell offers or a discount to prevent churn must be made while the customer is still on the phone. This requires a streamlined data collection and preparation process followed immediately by a prediction.

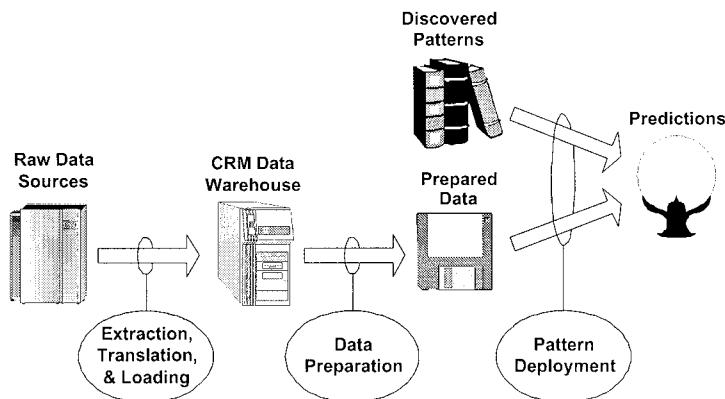


FIG. 25.8. Pattern deployment.

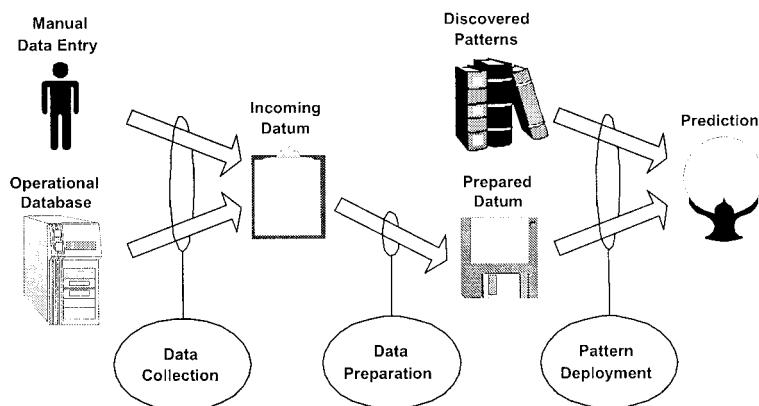


FIG. 25.9. Real-time deployment.

SAMPLE BUSINESS PROBLEMS

CRM business questions can be broken down into two major types—*strategic* and *operational*. Strategic questions are focused more on the descriptive power of CRM mining, whereas operational questions are aligned more closely with the predictive capabilities. This section uses the MyWidgets example for a number of common strategic and operational CRM mining questions.

Strategic Questions

A strategic question is one for which the answer is used to guide or influence a decision. When using CRM mining to answer a strategic question, it is the descriptive qualities of the discovered patterns that are the most important. An example would be an analysis to determine the key drivers of customer satisfaction. Actions can be taken based on a report summarizing the results of the analysis, but there is no model to be directly “deployed.” The descriptive properties of the analysis itself is the end goal of CRM mining for strategic questions. MyWidgets is planning its next advertising and promotional campaign. The goals for the campaign are to bring in new “high-value” customers and increase the effectiveness of the Web channel. The three top strategic questions have been identified as “What is the description of the high value customers?”, “What products are frequently purchased together?”, and “Why are people abandoning Web shopping carts?” The answers will help determine where to place advertisements and what special offers should be made.

MyWidgets does not have a standing definition of “high value” customers. It is generally difficult to define a hard cutoff for concepts such as high- and low-value customers. This makes the question more suited to a targeted segmentation application. By clustering with respect to the total number of purchases each customer has made, the definition of “high value” will be made by the algorithm. In addition, the analysis will describe medium- and low-value customers as well. Figure 25.10 shows sample data for the analysis after data aggregation has been performed on the MyWidgets data warehouse from Fig. 25.2. The data includes information from the Web session, phone call, store visit, and purchase event tables. A relative time-based aggregation has been used, with the “start date” column from the “customer” table. The results of the analysis could be used to determine whether one channel provides higher

Cust. ID	Age	Zip Code	Gender	Web Sessions Year 1	Web Sessions Year 2	Purchase Calls Year 1	Purchase Calls Year 2	Support Calls Year 1	Support Calls Year 2	Store Visits Year 1	Store Visits Year 2	Total Purchase
1001	35	94116	Male	3	6	1	1	3	2	0	2	\$648.15
1002	26	94103	Female	0	0	0	2	0	1	1	2	\$85.79
---	--	---	---	---	---	---	---	---	---	---	---	---

FIG. 25.10. Sample customer value clustering data.

value customers, or whether high-value customers have different demographic attributes. This will assist MyWidgets in choosing where to place the ads.

A description of what products are purchased together will assist MyWidgets in putting together promotional offers, such as “Get 15% off Widget A when Widget C is purchased.” If products are not normally purchased together, a campaign trying to increase their cross-selling may be ineffective. The only data necessary for the analysis is the purchase data, as shown in Fig. 25.11. Association rules are the natural choice for answering this business question. However, as discussed previously, a subjective interestingness filter will probably need to be applied to reduce the number of discovered rules to a manageable number.

Shopping cart abandonment (SCA) occurs on a Web site when a potential buyer performs one or more *product add* E-commerce events with no subsequent *product purchase* event. A description of things that tend to cause SCA is an important first step in trying to reduce abandonment. For the MyWidgets campaign, the results of the analysis may suggest what offers to put on certain Web pages. A well-placed offer can entice a customer to proceed with a purchase instead of abandoning a cart. Alternatively, it may turn out that the checkout process itself is confusing. A significant number of abandonments could be customers who intended to purchase but were unable to navigate the checkout process. A sequential pattern analysis of the E-commerce events should be used for this question. If possible, the click events could be included as well. Figure 25.12 shows a sample pulled from the E-commerce table.

Operational Questions

An operational question is one for which the results can be directly applied to increasing the ratio of profitable events. For example, in direct mail campaign response modeling the model scores themselves are the most important output. Although it is desirable to have an explainable pattern or model, the model itself is the goal of the analysis. The “answers” to operational questions must be deployed to bring a return on investment. In conjunction with the new promotional campaign, MyWidgets wants to recommend products for cross-selling

Transaction ID	Product ID
541	A
541	B
541	C
542	A
542	D
543	B
---	---

FIG. 25.11. Sample purchase affinity data.

Session ID	Datetime	Event Type	Event Value
10321	1/5/2001 10:43:32	View	A
10321	1/5/2001 10:43:54	View	B
10321	1/5/2001 10:44:05	Click-through	B
10321	1/5/2001 10:44:46	Add	B
10321	1/5/2001 10:45:29	Checkout 1	
10321	1/5/2001 10:47:01	Checkout 2	
10321	1/5/2001 10:47:56	Purchase	B
10566	1/7/2001 8:07:01	View	C
10566	1/7/2001 8:07:34	Click-through	C
10566	1/7/2001 8:08:18	Add	C
10566	1/7/2001 8:08:39	View	A
10566	1/7/2001 8:09:14	Click-through	A
---	---	---	---

FIG. 25.12. Sample shopping cart abandonment data.

and up-selling for customers who visit the Web site or order products on the phone. In addition, MyWidgets wants to make a special promotional offer to any customer who is in danger of churning.

Because MyWidgets tracks E-commerce events from its Web site, product recommendations can be made based on what customers looked at as well as what was purchased in the past. MyWidgets has decided that a product view is an indication of first level interest, a product click-through is second level, product adds are third level, and product purchases are fourth level. Product purchases can come from any of the MyWidgets channels, but the first three levels of interest are available only from the Web site. Sample data for this question is shown in Fig. 25.13. The data has been pulled from the E-commerce, phone, and purchase tables. Only the highest level of interest shown for each product was recorded when preparing the data. A collaborative filtering method is ideally suited for this question. For a given customer each product gets a score on how likely the customer is to be interested in that product. In the sample data, customers 1002 and 1004 have very similar histories. They both viewed Widget D, clicked-through Widget A for more information, and purchased Widget E. Because customer 1004 also purchased Widget C, collaborative filtering would rank this highly for recommendation to customer 1002.

For the churn question, MyWidgets has decided that any customer who has not had any events in the past 3 months should be considered to have left, or “churned.” The data sources

Cust. ID	Widget A	Widget B	Widget C	Widget D	Widget E
1001	4	0	1	3	0
1002	2	0	0	1	4
1003	1	0	0	1	4
1004	2	0	4	1	4
---	---	---	---	---	---

FIG. 25.13. Sample product recommendation data.

Cust. ID	Age	Zip Code	Gender	Web Sessions Month -2	Web Sessions Month -1	Purchase Calls Month -2	Purchase Calls Month -1	Support Calls Month -2	Support Calls Month -1	Store Visits Month -2	Store Visits Month -1	Churn
1001	35	94116	Male	3	0	1	1	5	10	0	2	True
1002	26	94103	Female	1	2	0	2	0	1	1	2	False
-----	---	-----	---	---	---	---	---	---	---	---	---	-----

FIG. 25.14. Sample churn data.

that will be included in the analysis are the Web sessions, phone calls, and purchase events. For aggregation purposes the date of the last event will be used as a relative aggregation point. Hopefully, there will be a significant pattern in the behavior of the customers who churn, prior to the last event. Figure 25.14 shows a sample of the training data after data aggregation has been performed. The target is binary, “true” for customers who have churned and “false” for those who are still active. This makes the churn question a binary classification application. Like the high-value customer description question, an algorithm that handles high-dimensional data will need to be used for predicting customer churn.

SUMMARY

This chapter has given an overview of the steps required for CRM mining. The process starts with a data warehouse and continues through data preparation, pattern discovery, pattern analysis, and in some cases pattern deployment. The bulk of the work involved with CRM mining is in the data preparation and pattern analysis. This is because the situation for every enterprise is unique, with different data sources and business questions to be answered. Although pattern discovery makes use of some very complex data mining and machine learning algorithms, there is no additional work that has to be done to apply them to CRM data once the data has been properly prepared.

The ideal data sources for CRM mining are customer behavior data. Because the ultimate goal of CRM mining is to increase ratio of profitable to unprofitable customer behaviors, data that directly measure those behaviors are superior to data that are merely surrogates. E-commerce data is especially valuable because it is the only data source that logs every interaction a customer has with a touchpoint.

As the examples in the sixth section show, a properly designed CRM data warehouse can support data mining for a wide variety of business questions. However, with data from a real CRM warehouse the number of dimensions in the prepared data can quickly grow into the hundreds or thousands. This means that CRM mining needs to take advantage of advanced feature space reduction techniques or algorithms that scale and perform well in high-dimensional spaces.

To be cost effective, a CRM mining system must be used to answer a variety of business questions. If the fixed costs associated with data preparation, data analysis, and pattern deployment are not amortized over several projects, it is unlikely that the CRM mining system will be profitable. Not every analysis leads to a million-dollar saving that will cover the costs of the data mining project. It is more likely that each analysis will lead to a more modest return on investment, and some may not result in any savings at all. A successful CRM mining system must be set up to answer questions in a timely manner, for example, days not weeks. Only then will the increased profits from maximizing the customer event ratio outweigh the costs of CRM mining.

REFERENCES

- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th VLDB Conference* (pp. 487–499). San Francisco: Morgan Kaufmann.
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Belmont, CA: Wadsworth.
- Buchner, A., & Mulvenna, M. D. (1998). Discovering Internet marketing intelligence through online analytical Web usage mining. *SIGMOD Record*, 27, 54–61. New York: ACM Press.
- Cooley, R. (2000). *Web usage mining: Discovery and application of interesting patterns from Web data*. Unpublished doctoral dissertation, University of Minnesota.
- Cooley, R., Mobasher, B., & Srivastava, J. (1999). Data preparation for mining World Wide Web browsing patterns. *Knowledge and Information Systems*, 1, 5–32.
- Cooley, R., Tan, P.-N., & Srivastava, J. (1999). Websift: The Web site information filter system. In *International WEBKDD Workshop*. New York: ACM Press.
- Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory*. New York: John Wiley & Sons.
- Ehrenberg, A. S. C. (1988). *Repeat-Buying: Facts, theory and applications*. London: Charles Griffin.
- Forgy, E. W. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21, 768–769.
- Friedman, J. H. (1991). Multivariate adaptive regression splines. *Annals of Statistics*, 19, 1–141.
- Friedman, J. H. (1994). *Flexible nearest neighbor classification* (Tech. Rep.). Stanford, CA: Stanford University.
- Jolliffe, I. T. (1986). Principal component analysis. In *Principal component analysis*. New York: Springer-Verlag.
- Kaufman, L., & Rousseeuw, P. (1990). *Finding groups in data: An introduction to cluster analysis*. New York: John Wiley & Sons.
- Kimball, R., & Merz, R. (2000). *The data webhouse toolkit*. New York: Wiley.
- Liu, B., Hsu, W., & Chen, S. (1997). Using general impressions to analyze discovered classification rules. In *Third International Conference on Knowledge Discovery and Data Mining* (pp. 31–36). Menlo Park, CA: AAAI Press.
- Mannila, H., Toivonen, H., & Verkamo, A. I. (1995). Discovering frequent episodes in sequences. In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining* (pp. 210–215). Menlo Park, CA: AAAI Press.
- Mitchell, T. (1996). *Machine learning*. New York: McGraw-Hill.
- Ng, R., & Han, J. (1994). Efficient and effective clustering method for spatial data mining. In *Proceedings of the 20th VLDB Conference* (pp. 144–155). Santiago, Chile.
- Padmanabhan, B., & Tuzhilin, A. (1998). A belief-driven method for discovering unexpected patterns. In *Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 94–100). Menlo Park, CA: AAAI Press.
- Pyle, D. (1999). *Data preparation for data mining*. San Mateo, CA: Morgan Kaufmann.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.
- Resnik, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. In *Conference on Computer Supported Cooperative Work* (pp. 175–181). New York: ACM Press.
- Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating “word of mouth.” In *ACM Conference on Human Factors in Computing Systems (CHI)* (pp. 210–217). New York: ACM Press.
- Spiliopoulou, M., & Faulstich, L. C. (1998). WUM: A Web utilization miner. In *EDBT Workshop WebDB98* (pp. 109–115). Berlin: Springer-Verlag.
- Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In *Fifth International Conference on Extending Database Technology* (pp. 3–17). Berlin: Springer-Verlag.
- Vapnik, V. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.
- Wu, K.-l., Yu, P. S., & Ballman, A. (1998). Speedtracer: A Web usage mining and analysis tool. *IBM Systems Journal*, 37, 89–105.
- Zhang, T., Ramakrishnan, R., & Livny, M. (1996). Birch: An efficient data clustering method for very large databases. In *ACM-SIGMOD International Conference on Management of Data* (pp. 103–114). New York: ACM Press.

26

Mining Computer and Network Security Data

Nong Ye

Arizona State University

Introduction	618
Intrusive Activities and System Activity Data	618
Phases of Intrusions	619
Data of System Activities	620
Extraction and Representation of Activity Features for Intrusion Detection	623
Features of System Activities	624
Feature Representation	625
Existing Intrusion Detection Techniques	628
Application of Statistical Anomaly Detection Techniques to Intrusion Detection	629
Hotelling's T^2 Test and Chi-Square Distance Test	629
Data Source and Representation	631
Application of Hotelling's T^2 Test and Chi-Square Distance Test	633
Testing Performance	633
Summary	634
References	635

This chapter is based on the work performed with the funding support from the Air Force Office of Scientific Research (AFOSR) under grant number F49620-99-1-001, the Air Force Research Laboratory (AFRL), Rome, under agreement number F30602-98-2-0005, and the DARPA/AFRL-Rome under grant number F30602-99-1-0506. The U.S. government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of AFOSR, AFRL-Rome, DARPA, or the U.S. Government.

INTRODUCTION

As computer and network systems have been increasingly used to support critical operations in many fields, including defense, banking, telecommunication, transportation, E-commerce, education, and so on, intrusions into computer and network systems have increased. Intrusions compromise the security (i.e., integrity, confidentiality, and availability) of computer and network systems and have become significant threats to the quality of services from computer and network systems with severe consequences. Layers of defense can be set up against intrusions through prevention, detection, reaction, and so on.

Research on offline intrusion prevention focuses mainly on methodologies of secure software engineering (Viega & McGaw, 2002). Although secure software methodologies will continue to improve, bugs and vulnerabilities in computer and network systems are inevitable due to the difficulty in managing the complexity of such large-scale systems during their specification, design, implementation, and installation. Attackers explore bugs and vulnerabilities in systems to attack them. Online intrusion prevention techniques (Fisch & White, 2000; Pfleeger, 1997; Ye, 2003), such as firewalls for network packet filtering, cryptography for data encryption and decryption, and authorization/authentication/identification for access control, prevent intrusions by limiting access to computer and network assets (data and programs). Firewalls control access to a host machine or a network. Authentication/identification/authorization controls access to a local or remote service from a program. Cryptography controls access to data. By limiting access to various kinds of assets in computer and network systems, the online intrusion prevention techniques raise the difficulty of launching successful intrusions. However, intrusion prevention techniques cannot completely prevent intrusions, especially from determined attackers with resources and sophisticated skills. As long as access to system assets exists, vulnerabilities exist for that access to assets and can be discovered if they are unknown at present. Hence, it is expected that some intrusions can pass through online intrusion prevention mechanisms and act on computer and network systems to carry out intrusive activities. Moreover, intrusion prevention techniques can hardly prevent insider attacks.

The goal of intrusion detection is to detect intrusive activities while they are taking place on computer and network systems. System activities usually are monitored by collecting data of system activities and analyzing the data to detect intrusive activities. The detection of an intrusion then triggers intrusion reaction to assess the intrusion and damage and take actions for system recovery and further intrusion prevention.

This chapter presents applications of data mining techniques to intrusion detection, especially regarding which data are collected from computer and network systems for monitoring and detecting intrusive activities and how data mining techniques are applied to intrusion detection. The second section describes intrusive activities and data that capture system activities, including intrusive activities. The third section discusses the features extracted from system activity data and the representation of the features for intrusion detection. The fourth section reviews existing intrusion detection techniques. The fifth section presents an application of statistical anomaly detection techniques to intrusion detection. The final section summarizes the chapter.

INTRUSIVE ACTIVITIES AND SYSTEM ACTIVITY DATA

An intrusion usually includes a series of activities. Activities on computer and network systems, including intrusive activities, can be recorded for intrusion detection. Existing work on intrusion detection primarily has used data of system activities to detect intrusions, although other kinds

of data (e.g., system state and performance data) in computer and network systems also may be helpful for intrusion detection. In this chapter data of system activities for intrusion detection are described. Readers are referred to Ye (2003) for the description of other data that also may be useful for intrusion detection.

To understand the data of system activities and the detection of intrusive activities recorded in system activity data, intrusive activities are described by showing various phases of intrusions. Then two types of system activity data and how intrusive activities may manifest in these data are presented and discussed.

Phases of Intrusions

Successful intrusions often go through several phases. Different phases have different goals and attack different vulnerabilities of different system assets using different methods. There are mainly six phases of intrusions: reconnaissance, scanning, gaining access, maintaining access, launching further attacks, and covering tracks. Not every phase is used in every intrusion case. These phases are briefly described in following paragraphs. More details can be found in Skoudis (2002) and Ye (2003).

In the reconnaissance phase an attacker investigates a target computer and network system using publicly available information. Reconnaissance techniques include low-technology reconnaissance (e.g., social engineering to trick people into releasing information and physical break-in to obtain information), search of offline sources (e.g., phone books) and online sources, and so on. Examples of online sources to gain useful information about a target organization are general search engines (e.g., Google), whois databases, the domain name system (DNS), and the organization's Web site. Information obtained may include names of computer and network domains, IP (Internet protocol) addresses of computer resources within the organization, e-mail addresses of employees, and so on. Such information can be used in later phases of intrusions, e.g., the use of an IP address to construct an IP address spoofing attack for gaining access to a host machine.

In the scanning phase an attacker attempts to map the topology of a target computer and network system, find potential entry points into the system, and discover vulnerabilities that can be exploited in later intrusion phases, using information such as IP addresses and domain names obtained from the reconnaissance phase. Network mapping uses network tools such as a *traceroute* to find out the network topology of the system and active host machines. Active host machines are then accessed to find out available remote and local services such as modem dial-in and network services (e.g., Web). System responses to requests for available services on a host machine may reveal the type of the operating system running on the host machine. Then vulnerabilities of that particular operating system can be exploited in later phases of intrusions.

In the gaining-access phase an attacker exploits vulnerabilities on a target host machine to gain access to the host machine or gain higher privileges such as root user privileges. After gaining access to the host machine, the attacker may use assets (e.g., sensitive files and data) on the host machine in later phases of intrusions such as maintaining access, covering tracks, and launching further attacks. As there are a variety of system vulnerabilities, there are a variety of ways to gain access, including password guessing, buffer overflow attacks, IP spoofing attacks, Web attacks, viruses, and so on.

In the maintaining-access phase, an attacker attempts to establish an easy, safe access to a target host machine so that the attacker can come back later without going through the initial, complicated, and risky process of gaining access. Methods of maintaining access include creating a new user account, using a Trojan program to steal an existing user's password, creating a back door, and so forth.

After an attacker gains and maintains access to a target host machine, the attacker may launch further attacks on a larger scale and with a wider impact, for example, attacks on other host machines within or beyond the network domain of the compromised host machine. Typical examples of such further attacks are distributed Denial of Service (DoS) attacks that use compromised victim hosts simultaneously to attack a target computer and network system by remotely flooding the target system with network requests or network traffic. Note that there are some attacks on a target system, such as many forms of DoS attacks, that do not require access rights to the system, and therefore can be launched without going through the gaining-access and maintaining-access phases.

Host machines usually are equipped with auditing/logging facilities to record activities, including intrusive activities. If an attacker does not want to be caught, the attacker may want to alter audit/log files to remove the attacker's activity trails in these files.

Data of System Activities

Because a computer and network system has two kinds of components, host machines and network links, data can be collected to capture activities on both. Hence, there are mainly two types of system activity data: network traffic data capturing network activities and computer audit/log data capturing host machine activities.

Network Traffic Data. Because network activities involve mainly the transmission of data, network activity data are a collection of data packets being transmitted over network links. Because data packets are the traffic on a network, network activity data are also called network traffic data. A data packet consists of the following two parts (Northcutt, Cooper, Fearnaw, & Frederick, 2001; Stevens, 1994; Ye, 2003):

- Data payload. The data payload is generated by a network application program, for example, an FTP (file transfer protocol) application for transferring files, an SMTF (simple mail transfer protocol) application for sending and receiving emails, and an HTTP (hypertext transfer protocol) application for Web browsing.
- Header. The header information is added by various layers of network communication protocols such as TCP (transmission control protocol) and IP for enabling and controlling the transmission of the data payload from the source IP address to the destination IP address on the Internet.

A data packet travels over network links in binary form. Special software programs often are used to capture and interpret the binary information in a data packet. These software programs are called sniffers. Tcpdump is a common sniffer. Versions of tcpdump running in Solaris and Linux environments can be downloaded at www.tcpdump.org. WinDUMP is the Windows version of tcpdump and can be downloaded at netgroup-serv.polito.it/windump. The Mac OS X operating system comes with tcpdump. The following is an example of a data line produced by tcpdump for an intercepted data packet:

```
15:11:32.812372 122.111.1.30.43590 > 122.111.0.40.www: S  
768602:768629(27) ack 888916 win 4096 <mss 1024> (ttl 60, id 33916)  
XXXX XXXX .....
```

where

“15:11:32.812372” is the time of receiving the data packet, in which “812372” is the data packet’s unique time-stamp because multiple data packets may arrive at any given second,
“122.111.1.30” is the source IP address of the data packet,
“43590” is the source port,
“>” indicates the traffic direction,
“122.111.0.40” is the destination IP address,
“www” is the destination port recognized by tcpdump from the TCP port number of 80,
“S” is a control bit used by TCP indicating the request for establishing a network session,
“768602” and “768629” are the sequence numbers indicating the range of the data payload with the length of 27 bytes,
“ack 888916” indicates the acknowledge number,
“win 4096” indicates the window size for controlling the data flow,
“<mss 1024>” shows the maximum segment size (mss) of 1024 bytes,
“ttl” indicates 60 as the time to live (TTL) value,
“id 33916” is the session identification (ID), and
“XXXX XXXX” is the data payload in the binary form, where “X” denotes a hexadecimal digit.

Network traffic data can be useful in detecting intrusive activities in the reconnaissance and scanning phases, for example, access to online sources (including whois databases, DNS servers, and Web sites) for reconnaissance, and access to network services (including HTTP, FTP, and SMTP applications) for scanning active hosts and open ports for network services such as HTTP. Those intrusive activities are captured in network traffic data and can be identified using information in the header and data payload of a data packet. For example, the destination port in the header of a data packet indicates the requested network service. If the destination port is associated with a network service with known vulnerabilities, it indicates a possible attempt to exploit those known vulnerabilities for an intrusion. For another example, the time of access and the destination port from the header of each data packet can tell us the intensity of network traffic for a particular network service, and thus can be used to detect DoS attacks through the traffic flooding of that network service.

Computer Audit/Log Data. A variety of data can be collected from a host machine to capture activities on the host machine. These data include mainly computer audit data, system log data, and application log data (Ye, 2003).

Computer audit data capture host activities managed by the kernel of the operating system, that is, actions or events taking place in the kernel of the operating system. Auditable events are those actions that may have security implications. The following are some examples of auditable events:

- Actions involved in authentication/identification/authorization
- Addition and deletion of objects in a user’s address space
- Actions of adding or deleting user accounts by system administrators
- Use of printer, network interface card, and other I/O (input/output) devices

Auditable information recorded for each auditable event may reveal, for example:

- Time of the event
- Type of the event
- User generating the event
- Process requesting the event
- Object accessed in the event
- Source location of the request leading to the event
- Return status of the event

Hence, from computer audit data we can obtain information about access to files, users, and processes (e.g., privileged processes for authentication/identification/authorization, user account creation, and access to I/O devices).

For instance, the Sun Solaris operating system comes with the SunSHIELD Basic Security Module (BSM) that is an auditing facility in Solaris for recording auditable events. In BSM, audit events are generated by either kernel-level system calls from the kernel of the operating system or user-level system calls from processes or applications outside the kernel of the operating system. We can select from the following classes of audit events for the system to audit:

- File read events
- File write events
- File attribute-related events such as changes of file attribute (e.g., chown, clock)
- Nonattribute related events
- File modification events
- File creation events
- File deletion events
- File close events
- Process events, such as exec, fork, exit
- Network events
- IPC events
- Administrative events
- Login/logout events
- Application events
- ioctl (IO control) events
- Program execution events
- Miscellaneous events

In Solaris 2.5.1 there are 284 different types of auditable events. A complete list of audit event types in Solaris can be found at www.sun.com/docs.

In BSM each auditable event produces an audit record. Audit records are kept in audit files. Each audit record includes a number of audit tokens. Each audit token describes an attribute of the audit event. The following is an example of a BSM audit record from the DARPA (Defense Advanced Research Projects Agency) Intrusion Detection Evaluation 2000 Data generated by the MIT Lincoln Laboratory (<http://ideval.ll.mit.edu>). Token IDs in these audit records are underlined to highlight separate tokens.

```
header,128,2,sendto(2),,Tue 07 Mar 2000 10:50:19 AM EST, +834856051 msec,  
argument,1, 0 × 4,fd,argument,3, 0 × 9,len,argument,4, 0 × 0,flags,argument,6, 0 × 1
```

0,tolen,socket,0 × 0002,0 × 246d,255.255.255.255,subject,root,root,root,root,root,2
808,2806,0 0 ppp5-213.iawhk.com.return,success,9,trailer,128

This audit record starts with a header token, where

- “header” is the token ID indicating the token type,
- “128” indicates the length of the audit record in bytes,
- “2” is the Event ID identifying the event type,
- “sendto(2)” is the Event ID monitor giving the descriptive information about the event type, and
- “Tue 07 Mar 2000 10:50:19 AM EST, + 834856051 msec” shows the date and time when the audit event is recorded.

An argument token describes an argument of the system call generating the audit event. A socket token gives information about the socket for communication. A subject token describes the process that generates the system call for the audit event, including user IDs, group IDs, process ID, session ID, and so on. A return token describes the return status of the system call for the audit event. A trailer token at the end of an audit record allows the backward search of the audit record in an audit file.

BSM audit records are stored in the binary form. The *praudit* command allows the translation of audit data in the binary form into a user-readable format. The *auditreduce* command allows the audit events to be selected based on the time, user, and event.

System log data and application log data capture activities of given system programs and application programs, respectively (Ye, 2003). System log data may record user login and logout activities, access to privileged programs and network services, and so on. Application log data generated by a Web application may record information such as the host name of the user accessing a Web site, the user ID if a user ID is used for authentication, the date and time of the Web request, the name of the page requested and the protocol used for the request, the number of bytes returned for the request, and so on.

Computer audit/log data can be useful in detecting intrusive activities in the gaining-access, maintaining-access, launching-further-attack, and covering-track phases, for example, gaining root user privileges, creating a user account, installing a DoS attack program, and modifying audit/log files. Those intrusive activities are captured in computer audit/log data. For example, intrusive activities of gaining root user privileges, creating a user account and modifying audit/log files are not usually performed by regular users. Hence, those intrusive activities produce different types of audit events from the types of audit events from regular user activities. The header token of each BSM audit event reveals the type of the audit event that can be useful in detecting intrusions.

EXTRACTION AND REPRESENTATION OF ACTIVITY FEATURES FOR INTRUSION DETECTION

Network sniffing and computer auditing/logging have performance impact on the disk space and CPU (central processing unit) time of the host machine running the sniffing, auditing, or logging facility. Not all information included in a data packet or an audit/log record of an event is useful for intrusion detection. It is important to know what features of system activities, and thus

what information from large amounts of system activity data, are useful to intrusion detection or are effective to distinguish intrusive activities from normal activities. Using and keeping only the information about those features of system activities can enhance the performance and reduce the overhead of intrusion detection. The features of system activities that have been extracted from system activity data for intrusion detection in existing work are described in the next section.

Features of System Activities

In existing work on intrusion detection the following features of system activities have been extracted from network traffic data or computer audit/log data for intrusion detection (Ye, Li, Chen, Emran, & Xu, 2001).

1. Occurrence of single attributes. A single attribute can be an individual data field of network traffic data and computer audit/log data such as the destination port for a network application, the type of audit event, the error message, or the source IP address, or can be an aggregate attribute such as the duration of a connection derived from all the data packets in that connection session and the CPU time spent by a process derived from all the audit events in that process.
2. Frequency of single attributes, for example, count of consecutive password failures.
3. Occurrence of multiple attributes, for example, the source port of 60000 together with the destination port of 2140 that are used by a well-known intrusion.
4. Frequency histogram (distribution) of multiple attributes or multiple attribute values, for example, frequency distribution of audit event types.
5. Sequence or transition of attributes, for example, event transition and event sequence.
6. Intensity of events, for example, number of data packets per time unit and number of audit events per time unit.

The single occurrence of an attribute, for instance, the “su root” command in UNIX environments, may indicate a possible violation of normal network or host operations. Considering the relatively low frequency of an attribute in a normal condition, for example, login failure, a high frequency of the attribute may indicate a possible deviation from normal network or host operations. The frequency distribution of attributes or attribute values, such as commands invoked by a user may be used to reveal a compromised user account when the distribution in the recent past is different from the long-term historical profile. The event intensity, for example, the number of data packets to a host machine per second, may indicate a possible DoS attack at the host machine through traffic flooding.

Features 1–6 can be grouped into three general categories: frequency feature, order feature, and intensity feature. Frequency features include features 1–4 about the frequency of single attributes or the frequency distribution of multiple attributes or multiple attribute values. The single occurrence of attributes in features 1 and 3 can be considered as a frequency feature that has only two possible values, 0 and 1. An intrusion typically consists of a series of events. A frequency feature extracted from a series of events can provide information about the collective activity level of these events. For a given series of events, a frequency feature of a single attribute value from a single event, for example, the type of an audit event, carries less information but at lower cost than the frequency distribution of multiple attributes or multiple attribute values from the same series of multiple events, for instance, the frequency distribution of all audit event types. One study (Ye, Li, et al., 2001) reveals that the frequency feature of

a single attribute value from a single event is not sufficient for intrusion detection. Northcutt et al. (2001) also points out that any detection of an intrusion based on a single data packet is very difficult to validate.

The order feature includes feature 5 about the sequential order of attributes. The order feature contains more information but at higher cost than the frequency feature for the same series of events, because the frequency feature does not take into account the sequential order of events. Ye, Li, et al. (2001) reveals that the order feature of a given event sequence provides an additional advantage to the frequency feature of the same event sequence for intrusion detection.

The intensity feature includes feature 6 about the intensity of attributes. This feature is important in detecting DoS attacks. The intensity feature is different from the frequency feature in that the intensity feature is measured based on a time unit, whereas the frequency feature is not necessarily measured based on a time unit.

Feature Representation

For a frequency feature we can use a vector, (X_1, \dots, X_K) , to represent the frequency of K attributes or attribute values, where K is an integer greater than or equal to 1, and X_k can take a binary value or an integer value for $k = 1, \dots, K$. For the order feature we can use (X_1, \dots, X_T) to represent a time series data for a sequence of events, where X_t denotes an event occurring at time t , for $t = 1, \dots, T$. Note that the vector representation of a frequency feature for a given event sequence, (X_1, \dots, X_K) , can be derived from the times-series representation of the order feature for the same event sequence, (X_1, \dots, X_T) . For the intensity feature, we can use $X(t)$ to represent the event intensity at time t .

The frequency feature, the order feature, or the intensity feature is often extracted from a sequence of events in the recent past of the current event to capture a short-term behavior. The short-term value of that feature is then compared with the long-term profile of that feature to determine the difference that is in turn used to detect intrusions. Three methods mainly have been used to obtain the measurement of a feature in the recent past of the current event: time unit, moving/sliding window, and decaying factor (Ye, 2003).

The time-unit method measures a given feature from all the events within a time unit, for example, second, minute, hour, day, and so on. The recent past can be defined as the last time unit, for instance, the last second. If the recent past can also be defined as the last T time units, the value of a given feature in the recent past can be obtained by averaging the values of the feature from those last T time units. For example, the intensity feature is often measured in terms of the number of events (e.g., data packets and audit events) per second.

The moving window method uses a window of a fixed or variable size to view a sequence of events up to the present event n . A measurement of a given feature is taken from the sequence of events viewed through the window. For the following sequence of events up to the present event n ,

$$E_1, E_2, \dots, E_{n-6}, E_{n-5}, E_{n-4}, E_{n-3}, E_{n-2}, E_{n-1}, E_n$$

the window of size 7 at the present event n allows us to view $E_{n-6}, E_{n-5}, E_{n-4}, E_{n-3}, E_{n-2}, E_{n-1}, E_n$ as the event sequence in the recent past from which the measurement of a given feature is taken. Then the event is advanced to $n + 1$. The window is also moved to $n + 1$, and the event sequence of $E_{n-5}, E_{n-4}, E_{n-3}, E_{n-2}, E_{n-1}, E_n, E_{n+1}$ is viewed in the window. In general, E_{n-L+1}, \dots, E_n is the event sequence in the recent past of the present event n for the window size of L .

Unlike the method of time unit, the method of moving window uses the number of events rather than the number of time units to define the recent past. Event intervals in time often

TABLE 26.1
An Example of Measuring the Frequency Distribution of Event Types in the Recent Past Using the Moving Window Method

Event	Event Sequence in the Moving Window of the Present Event	$(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$
n	$E_{n-6}=2, E_{n-5}=4, E_{n-4}=8, E_{n-3}=3,$ $E_{n-2}=4, E_{n-1}=7, E_n=1$	$(1, 1, 1, 2, 0, 0, 1, 1, 0, 0)$
$n + 1$	$E_{n-5}=4, E_{n-4}=8, E_{n-3}=3, E_{n-2}=4,$ $E_{n-1}=7, E_n=1, E_{n+1}=10$	$(1, 0, 1, 2, 0, 0, 1, 1, 0, 1)$
$n + 2$	$E_{n-4}=8, E_{n-3}=3, E_{n-2}=4, E_{n-1}=7,$ $E_n=1, E_{n+1}=10, E_{n+2}=6$	$(1, 0, 1, 1, 0, 1, 1, 1, 0, 1)$

vary with different times of a day, for example, daytime and nighttime. In a given time unit there may be few or no events, making it difficult to take the measurement of a given feature for that time unit. A large variation of event intervals in time may result in a large variance of values for a given feature, which may not be desirable. Using the number of events rather than the number of time units helps overcome this problem and produces stable measurements for a given feature. Moreover, when we are interested in the frequency distribution of event types in the recent past, we focus on the mixture of different event types in an event sequence of the same length rather than in an event sequence within a time unit. In such a case, using the method of moving window is more reasonable than using the method of time unit to define the recent past.

Table 26.1 shows an example of measuring the frequency distribution of event types in the recent past defined by the window size of 7 from the following sequence of audit events:

$$E_{n-6}=2, E_{n-5}=4, E_{n-4}=8, E_{n-3}=3, E_{n-2}=4, E_{n-1}=7, E_n=1, E_{n+1}=10, E_{n+2}=6.$$

Because we are interested in only the information of the event type, only the event type of each event is given in the above sequence of events. Suppose that there are 10 different event types. We use (X_1, \dots, X_{10}) to represent the frequency distribution of the 10 event types in the recent past.

Like the method of moving window, the method of decaying factor defines the recent past based on events rather than time units. However, in the method of moving window the events in the window are treated equally. In the above example $E_{n-5}=4$ and $E_{n-2}=4$ in the window for the present event n give us the frequency count of 2, although $E_{n-2}=4$ is more recent than $E_{n-5}=4$. In the method of decaying factor a weight is assigned to each event when computing the value for a given feature. The weight value assigned to an event depends on how recent the event is. A more recent event gets more weight in the computation of the value for a given feature. Hence, the weight reflects the decaying of an event, and the measurement of a given feature takes into account the decaying of events.

In recent studies on intrusion detection (Emran & Ye, 2002; Ye & Chen, 2001; Ye, Li, et al., 2001), the authors used the exponentially weighted moving average (EWMA) method to take into account the decaying of events. Let $X(n)$ be the measurement of a given feature at event n . The following formula is used to compute $X(n)$:

$$X(n) = \lambda \times x + (1 - \lambda) \times X(n - 1) \quad (1)$$

where x is a value from event n , and λ is a constant in the range of $[0, 1]$ that determines the decaying rate. Using this formula event n receives the weight of λ , event $n - 1$ receives the weight of $\lambda(1 - \lambda)$, and event $n - k$ receives the weight of $\lambda(1 - \lambda)^k$. If we are interested in

TABLE 26.2
An Example of Computing the Frequency Distribution of Event Types
in the Recent Past Using the Method of Decaying Factor

Event	$(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$
$n - 6$	$X_1 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_2 = 0.3 \times 1 + (1 - 0.3) \times 0 = 0.3$
	$X_3 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_4 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_5 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_6 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_7 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_8 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_9 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_{10} = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
$n - 5$	$X_1 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_2 = 0.3 \times 0 + (1 - 0.3) \times 0.3 = 0.21$
	$X_3 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_4 = 0.3 \times 1 + (1 - 0.3) \times 0 = 0.3$
	$X_5 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_6 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_7 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_8 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_9 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_{10} = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
$n - 4$	$X_1 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_2 = 0.3 \times 0 + (1 - 0.3) \times 0.21 = 0.147$
	$X_3 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_4 = 0.3 \times 0 + (1 - 0.3) \times 0.3 = 0.21$
	$X_5 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_6 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_7 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_8 = 0.3 \times 1 + (1 - 0.3) \times 0 = 0.3$
	$X_9 = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$
	$X_{10} = 0.3 \times 0 + (1 - 0.3) \times 0 = 0.0$

the long-term measurement of a given feature in the past, we should use a small value of λ , for example, 0.0001. If we are interested in the short-term measurement of a given feature in the recent past, we should use a large value of λ , perhaps, 0.3. In general, if we wish to have an exponential weighting method that is equivalent to an L -event moving window method, λ is set as follows (Montgomery & Mastrangelo, 1991):

$$\lambda = \frac{2}{L + 1} \quad (2)$$

Table 26.2 gives an example of computing the frequency distribution of event types in the recent past, $(X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9, X_{10})$, for the same sequence of events used in Table 26.1:

$$E_{n-6}=2, E_{n-5}=4, E_{n-4}=8, E_{n-3}=3, E_{n-2}=4, E_{n-1}=7, E_n=1, E_{n+1}=10, E_{n+2}=6$$

using the decaying factor method based on EWMA. $X_i(n)$ is computed as follows for $i = 1, \dots, 10$:

$$X_i(n) = \lambda \times x_i + (1 - \lambda) \times X_i(n - 1) \quad (3)$$

where $x_i = 1$ if event n falls in the i th event type, or $= 0$ otherwise, provided that $(n - 6) = 1$ is the beginning of the event sequence, the value of $X_i(0)$ is initialized to zero for $i = 1, \dots, 10$, and λ is set to 0.3.

EXISTING INTRUSION DETECTION TECHNIQUES

There are two general approaches to intrusion detection: signature recognition and anomaly detection (Debar, Dacier, & Wespi, 1999; Lippmann et al., 2000; Ye, Giordano, & Feldman, 2001; Ye, Li, et al., 2001). Signature recognition techniques, also referred to as “misuse detection” in some literature, compare activities in a computer and network system with signatures of known intrusions, and signal intrusions when there is a match. For a subject (user, file, privileged program, host, network, etc.) of interest, anomaly detection techniques establish a profile of the subject’s long-term normal behavior (norm profile), compare the observed behavior of the subject with its norm profile, and signal intrusions when the subject’s observed behavior deviates significantly from its norm profile. Signature recognition techniques and anomaly detection techniques often are used together to complement each other for intrusion detection.

Signature recognition techniques utilize intrusion signatures, profiles of intrusion characteristics, and consider the presence of an intrusion signature as evidence of an intrusion. Anomaly detection techniques use only data of normal activities in a computer system for training and building a norm profile. Signature recognition techniques rely on data of both normal and intrusive activities for learning intrusion signatures, either manually or automatically, through data mining.

Most commercial intrusion detection systems are based on signature recognition techniques (Escamilla, 1998; Ye, 2003). Intrusion signatures have been characterized as strings (e.g., command names), event frequency distributions, event sequences, activity graphs, and intrusion scenarios with event sequences, event preconditions, and target compromised states. Intrusion signatures have been represented using finite state machines (Vigna, Eckmann, & Kemmerer, 2000), colored Petri nets (Kumar, 1995), association rules (Lee, Stolfo, & Mok, 1999), decision trees (Ye, Li, et al., 2001), clusters (Li & Ye, in press) and production rules of expert systems (Anderson, Frivold, & Valdes, 1995) to store and recognize intrusion signatures. Intrusion signatures are either manually encoded or automatically learned through data mining. However, signature recognition techniques have a limitation in that they cannot detect novel intrusions the signatures of which are unknown.

Anomaly detection techniques capture both known intrusions and unknown intrusions if the intrusions demonstrate a significant deviation from a norm profile. Existing anomaly detection techniques differ mainly in the representation of a norm profile and the inference of intrusions using the norm profile.

Forrest, Hofmeyr, and Somayaji (1997); Ghosh, Schwatzbard, and Shatz (1999); and Warrender, Forrest, and Pearlmutter (1999) collect short sequences of system calls or audit events that appear in normal activities, represent those short sequences as strings, store those strings as a norm profile, and employ either negative selection or positive selection to determine whether an observed string deviates from the string-based norm profile. Strings-based anomaly detection techniques model the order feature of normal activities, that is, event sequences. Ko, Fink, and Levitt (1997) uses predicates in formal logic to specify normal activities for a norm profile and employ logical reasoning to infer the consistency of observed activities using that norm profile.

A number of studies, (Denning, 1987; Emran & Ye, 2002; Javitz & Valdes, 1991, 1994; Jou et al., 2000; Li & Ye, 2002; Ye, Borror, & Zhang, in press; Ye & Chen, 2001; Ye, Emran,

Chen, & Vilbert, 2002; Ye, Li, et al., 2001) use statistical distributions to model the frequency feature and the intensity feature of normal activities for a norm profile and employ statistical tests to determine whether observed activities deviate significantly from the norm profile. An advantage of statistical-based anomaly detection techniques is their capability of explicitly representing and handling variations and noises in normal activities. Anomaly detection techniques based on formal logic lack such a capability of noise handling and variance representation. Strings-based anomaly detection techniques must rely on a large, costly repository of short sequences of normal events to capture variations of normal activities.

Both artificial neural networks and stochastic models (e.g., Markov chain model and hidden Markov model) have been used to model the order feature of normal activities (e.g., event transitions or event sequences) for a norm profile and detect intrusions based on the deviation of the observed events from the expected event or based on the probabilistic support of the norm profile to the observed events (Debar, Becker, & Siboni, 1992; DuMouchel, 1999; Eskin, Lee, & Stolfo, 2001; Ghosh, Schwatzbard, & Shatz, 1999; Schonlau et al., 1999; Scott, 2002; Warrender et al., 1999; Ye, Ehiabor, & Zhang, 2002; Ye, Zhang, & Borror, in press). Ye et al. (2002) reveals the better performance of the Markov chain model compared with artificial neural networks for intrusion detection. Despite that, the Markov chain technique for intrusion detection is not robust to the degree in which normal activities (noises) and intrusive activities (signals to detect) are mixed in system activity data (Ye et al., in press). Noise cancellation techniques have been developed to remove effects of normal activities and thus make intrusive activities more distinct for detection (Ye, Chen, & Mou, 2002).

As can be seen from the above review of existing intrusion detection techniques, many data mining techniques have been applied to intrusion detection, including decision trees (Ye, Li, et al., 2001), association rules (Lee, Stolfo, & Mok, 1999), clustering algorithms (Li & Ye, 2002), artificial neural networks (Debar et al., 1992; Ghosh et al., 1999), Markov models (DuMouchel, 1999; Ju & Vardi, 1999; Schonlau et al., 1999; Scott, 2002; Warrender et al., 1999; Ye et al., 2002, in press), and statistical anomaly detection (Emran & Ye, 2002; Li & Ye, 2002; Ye, Borror, & Zhang, in press; Ye & Chen, 2001; Ye, Emran, et al., 2002; Ye, Li, et al., 2001). In the next section a multivariate statistical anomaly detection technique to intrusion detection is described in detail.

APPLICATION OF STATISTICAL ANOMALY DETECTION TECHNIQUES TO INTRUSION DETECTION

As an example, this section applies Hotelling's T^2 test and the chi-square distance test to intrusion detection through anomaly detection. Materials in this section are taken from Ye, Emran, Chen, & Vilbert (2002) with the permission of IEEE Press. In this section Hotelling's T^2 test and the chi-square distance test are first described. Then the data set used to test the performance of Hotelling's T^2 test and the chi-square distance test are presented. The testing performance of these two multivariate statistical anomaly detection techniques is also provided.

Hotelling's T^2 Test and Chi-Square Distance Test

Let $X_i = (X_{i1}, X_{i2}, \dots, X_{ip})$ denote the i th observation of p measures on a process or system. Assume that when the process is operating normally (in control), the population of X follows a multivariate normal distribution with the mean vector μ and the covariance matrix Σ . Using

a data sample of size n , the sample mean vector \bar{X} and the sample covariance matrix S are usually used to estimate μ and Σ (Chou, Mason, & Young, 1999), where

$$\bar{X} = (\bar{X}_1, \bar{X}_2, \dots, \bar{X}_p) \quad (4)$$

$$S = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})'. \quad (5)$$

Hotelling's T^2 statistic for an observation X_i is determined by the following:

$$T^2 = (X_i - \bar{X})' S^{-1}(X_i - \bar{X}). \quad (6)$$

A large value of T^2 indicates a large deviation of the observation X_i from the in-control population. We can obtain a transformed value of the T^2 statistic, $\frac{n(n-p)}{p(p+1)(n-1)} T^2$ which follows an F distribution with p and $n - p$ degrees of freedom, by multiplying T^2 by the constant

$$\frac{n(n-p)}{p(n+1)(n-1)}.$$

If the transformed value of the T^2 statistic is greater than the tabulated F value for a given level of significance, α , then we reject the null hypothesis that the process is in control (normal) and thus signal that the process is out of control (anomalous).

If X does not follow a multivariate normal distribution, the transformed value of the T^2 statistic may not follow an F distribution. As a result, we cannot use the tabulated F value as a signal threshold to determine whether a transformed value of the T^2 statistic is large enough for an out-of-control signal. However, if the p variables are independent and p is large (usually greater than 30), T^2 follows approximately a normal distribution according to the central limit theorem, regardless of what distribution each of the p variables follows. Using an in-control sample of T^2 values, the mean and standard deviation of the T^2 population can be estimated from the sample mean \bar{T}^2 and the sample standard deviation S_{T^2} . The in-control limits to detect out-of-control anomalies are usually set to 3-sigma control limits as determined by $[\bar{T}^2 - 3S_{T^2}, \bar{T}^2 + 3S_{T^2}]$. Because we are interested in detecting significantly large T^2 values for intrusion detection, we need to set only the upper control limit to $\bar{T}^2 + 3S_{T^2}$ as a signal threshold. That is, if the T^2 for an observation is greater than $\bar{T}^2 + 3S_{T^2}$, we signal an anomaly.

An out-of-control signal from the T^2 test can be caused by a shift from the in-control mean vector (mean shift), a departure from the in-control covariance structure or variable relationships (counterrelationship), or combinations of the two situations. In a mean shift situation one or more of the p variables is out of control. In a counterrelationship situation a relationship between two or more of the p variables changes from the variable relationship established in the covariance matrix. Although the T^2 test detects both mean shifts and counterrelationships, the T^2 test is more sensitive to counterrelationships than mean shifts, because the T^2 test relies largely on the correlated structure of variables (covariance matrix) for anomaly detection (Chou et al., 1999).

The computation of the covariance matrix and its inverse in Equation 6 is computationally costly for a large data set with many variables. Hence, a scalable multivariate statistical anomaly detection technique is developed based on the chi-square distance measure. If we have p variables to measure and X_j denotes an observation of the j th ($1 \leq j \leq p$) variable, the χ^2 distance measure is given by the following:

$$\chi^2 = \sum_{j=1}^p \frac{(X_j - \bar{X}_j)^2}{\bar{X}_j} \quad (7)$$

This test statistic measures the distance of a data point from the center of a data population. Hence, we refer to this test as the chi-square distance test. When the p variables are independent and p is large (usually greater than 30), the X^2 statistic follows approximately a normal distribution according to the central limit theorem, regardless of what distribution each of the p variables follows. Using an in-control sample of χ^2 values, the mean and standard deviation of the χ^2 population can be estimated from the sample mean $\bar{\chi}^2$ and the sample standard deviation S_{χ^2} . The in-control limits to detect out-of-control anomalies are usually set to 3-sigma control limits as determined by $[\bar{\chi}^2 - 3S_{\chi^2}, \bar{\chi}^2 + 3S_{\chi^2}]$. Because we are interested in detecting significantly large χ^2 values for intrusion detection, we need to set only the upper control limit to $\bar{\chi}^2 + 3S_{\chi^2}$ as a signal threshold. That is, if the χ^2 for an observation is greater than $\bar{\chi}^2 + 3S_{\chi^2}$, we signal an anomaly.

Both the T^2 test statistic and the χ^2 test statistic measure the distance of an observation from the multivariate mean vector of a population. The T^2 test statistic uses the statistical distance that incorporates the multivariate variance-covariance matrix, whereas the χ^2 test statistic uses the chi-square distance. The chi-square distance is similar to a Euclidean distance but uses the average value on each variable to scale the Euclidean distance on that variable or dimension.

In general, anomalies involving multiple variables can be caused by shifts from the means of these variables (mean shifts), departures from variable relationships (counterrelationships), or combinations of mean shifts and counterrelationships. In contrast to the T^2 statistic, the χ^2 statistic does not account for the correlated structure of the p variables. Only \bar{X} is estimated to establish the norm profile according to Equation 4. Hence, the T^2 test detects both mean shifts and counterrelationships, whereas the χ^2 test detects only the mean shift on one or more of the p variables.

Data Source and Representation

BSM audit data from the Sun SPARC 10 workstation with the Solaris 2.5.1 operating system is used. There are 284 different types of events in BSM audit data from Solaris 2.5.1. Each audit event is characterized by the type of audit event. That is, only the event type information from each audit record is used for intrusion detection. Many studies have shown the effectiveness of the event type information for intrusion detection (Ye, Li, et al., 2001).

Nine weeks of audit data obtained from the DARPA Intrusion Detection Evaluation 1998 data generated by the MIT Lincoln Laboratory (<http://ideval.ll.mit.edu>) contain 7 weeks of labeled training data and 2 weeks of unlabeled testing data. Because the testing data are not labeled, it is difficult to evaluate the performance of the techniques used. Hence, the training data set was used for both training and testing in a study. During training, audit data of activities with the label of normal event was used to build the norm profile. During testing, the label of audit data was removed and the techniques previously discussed here were used to generate a label (normal or intrusive). The labels of activities from testing were then compared with the given labels for evaluating the performance of the techniques.

There are in total 35 days of data in the 7 weeks of training data. Four days of data are chosen as a representative of the entire training data set. Two days are chosen with relatively few intrusions and 2 days with comparatively more intrusions, varied in length. Week 1, Monday data was designated as day 1 data; Week 4, Tuesday data as day 2 data; week 4, Friday data as day 3 data; and week 6, Thursday data as day 4 data. Table 26.3 summarizes the statistics about these 4 days of data.

As we can see from Table 26.3, the average session length was comparatively smaller in day 2 and day 3 than that in day 1 and day 4. Around 3% of audit events on day 2 and day 4 were due to intrusive activities, whereas less than 0.80% of audit events in day 1 and day 3

TABLE 26.3
Statistics About the Data Set for Intrusion Detection

	Day 1	Day 2	Day 3	Day 4
Event information				
Number of events	744,085	1,320,478	2,249,505	925,079
Number of intrusive events	3,090	36,575	16,524	31,476
Percentage of intrusive events	0.42%	2.77%	0.73%	3.40%
Session information				
Number of sessions	298	503	339	447
Number of intrusive sessions	2	131	29	14
Percentage of intrusive sessions	0.67%	26.04%	8.55%	3.13%
Number of events per normal session				
Average	2,503	3,451	7,203	2,064
Minimum	1	69	74	96
Maximum	253,827	462,287	1,019,443	214,705
Number of events per intrusive session				
Average	1,545	279	570	2,248
Minimum	1,101	142	166	1,107
Maximum	1,989	1,737	4,986	2,841

Note: © 2002 IEEE. Reprinted, with permission, from Ye, Emran, Chen, & Vilbert (2002).

data were from normal activities. In terms of sessions, almost one fourth of the sessions in day 2 and one twelfth of the sessions in day 3 were intrusion sessions. Day 1 contained mostly normal sessions, and day 4 also had few intrusion sessions. A total of 176 instances of 9 types of intrusions are present in these 4 days of data. Because the objective is to detect any ongoing intrusions rather than any particular type of intrusion, the concern is about how many of these intrusion sessions can be detected.

Only the normal events of the first 2 days of data were used for training. Day 1 and day 2 contain 740,995 and 1,283,903 audit events arising from 296 and 372 normal sessions, respectively. Day 3 and day 4, which contain 2,232,981 and 893,603 audit events for normal activities, respectively, were used for testing. Numbers of audit events for intrusive activities in these 2 days were 16,524 and 31,476 arising from 29 and 14 intrusion sessions, respectively. Day 3 contained 339 sessions and day 4 contained 447 sessions in total comprising both normal and intrusive sessions. In this data set an intrusion occurred in one session only.

Activities on a host machine are captured through a sequence of audit events, each of which is characterized by the event type. For intrusion detection, we want to build a long-term profile of normal activities, and to compare the activities in the recent past of the present event to the long-term norm profile for detecting a significant deviation of the present event. We define audit events in the recent past of the present event n by a vector of $(X_1, X_2, \dots, X_{284})$ for the frequency distribution of the 284 event types, respectively, using the decaying factor method described previously. An observation for the n th event, $(X_1, X_2, \dots, X_{284})$, is obtained as follows.

$$X_i(n) = \lambda * 1 + (1 - \lambda) * X_i(n - 1) \quad (8)$$

if the n th audit event belongs to the i th event type

$$X_i(t) = \lambda * 0 + (1 - \lambda) * X_i(t - 1)$$

if the type of the n th audit event is different from the i th event type, where $X_i(n)$ is the observed value of the i th variable in the vector of an observation for the n th audit event, λ is a

smoothing constant that determines the decay rate, and $i = 1, \dots, 284$. We initialize $X_i(0)$ to 0 for $i = 1, \dots, 284$, and set λ to 0.3.

Application of Hotelling's T^2 Test and Chi-Square Distance Test

For the T^2 test, the long-term profile of normal activities is modeled by the sample mean vector \bar{X} and the sample covariance matrix S . The audit events for normal activities in the training data give us a sample of (X_1, \dots, X_{284}) 's to obtain the sample mean vector \bar{X} and the sample covariance matrix S . For each of the audit events in the training data and the corresponding observation of (X_1, \dots, X_{284}) , we compute the T^2 statistic according to Equation 6. We then determine the upper limit of T^2 in terms of $\bar{T}^2 + 3S_{T^2}$ as the signal threshold. For each of the audit events in the testing data and the corresponding observation of (X_1, \dots, X_{284}) , we compute the T^2 statistic according to Equation 6. The T^2 value is small if the observation is close to the norm profile. The upper limit, $\bar{T}^2 + 3S_{T^2}$, is used as the signal threshold to determine whether we should signal an event as intrusive based on the T^2 value of this event. If the T^2 value exceeds the signal threshold, we signal the audit event as intrusive. Because an intrusion session corresponds to one intrusion, we compute the session signal ratio for each session in the testing data by counting how many of the audit events in that session are signaled and dividing the signal count by the total number of audit events in that session. If the session is an intrusion session, we expect a high session signal ratio. If it is a normal session, we expect the session signal ratio to be low.

For the χ^2 distance test, the long-term profile of normal activities is modeled by the sample mean vector \bar{X} only. The audit events for normal activities in the training data give us a sample of (X_1, \dots, X_{284}) 's to obtain the sample mean vector \bar{X} . For each of the audit events in the training data and the corresponding observation of (X_1, \dots, X_{284}) , we compute the χ^2 statistic according to Equation 7. We then determine the upper limit of χ^2 in terms of $\bar{\chi}^2 + 3S_{\chi^2}$ as the signal threshold. For each of the audit events in the testing data and the corresponding observation of (X_1, \dots, X_{284}) , we compute the χ^2 statistic according to Equation 7. The χ^2 value is small if the observation is close to the norm profile. The upper limits, $\bar{\chi}^2 + 3S_{\chi^2}$, is used as the signal threshold to determine whether we should signal an event as intrusive based on the χ^2 value of this event. If the χ^2 value exceeds the signal threshold, we signal the audit event as intrusive. For each session in the testing data, we compute the session signal ratio.

Testing Performance

Based on the session signal ratio for each session in the testing data from a given test (the T^2 test or the χ^2 distance test), we can set a session-wise signal threshold on the session signal ratio. If the session signal ratio for a given session exceeds the session-wise signal threshold, the session is signaled as intrusive; otherwise the session is considered normal. We have a hit if a truly intrusive session is signaled by a given test as intrusive. We have a false alarm if a truly normal session is signaled by a given test as intrusive. We can then compute the false alarm rate as the ratio of the number of false alarms on the normal sessions to the total number of normal sessions in the testing data and compute the hit rate as the ratio of the number of hits on the intrusive sessions to the total number of intrusive sessions in the testing data. We thus obtain a pair of false alarm rate and hit rate using a given session-wide signal threshold. For different values of session-wise signal threshold, we can obtain different pairs of false

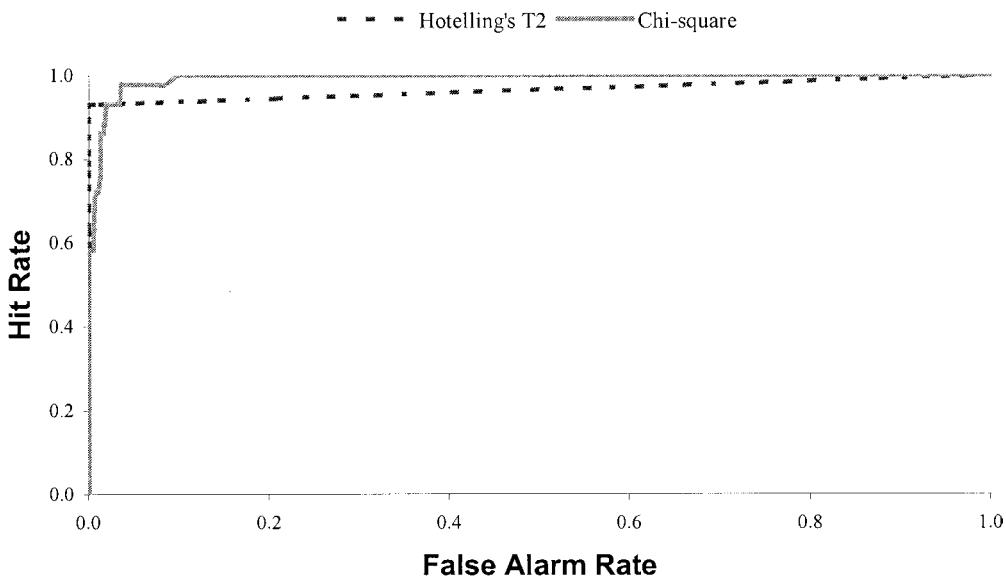


FIG. 26.1. The ROC Curves of the T^2 test and the χ^2 test based on session signal ratios. (© 2002 IEEE. Reprinted, with permission, from Ye, Emran, Chen, & Vilbert [2002]).

alarm rate and hit rate. A receiver operating characteristic (ROC) curve from signal detection theory (Kantowitz & Sorkin, 1983) can be used to plot pairs of false alarm rate and hit rate when various signal thresholds are used. The closer the ROC curve is to the top-left corner (representing the 100% hit rate and the 0% false alarm rate) of the ROC chart, the better the detection performance. Hence, the ROC curve shows the overall detection performance of a given test.

Figure 26.1 shows the ROC curves of the χ^2 and T^2 tests, respectively, based on the session signal ratios from the tests. It appears that both the χ^2 test and the T^2 test perform very well in distinguishing intrusion sessions from normal sessions. The T^2 test achieves the 95% hit rate at the 0% false alarm rate. The χ^2 test achieves the 60% hit rate at the 0% false alarm rate. However, after the 5% false alarm rate the χ^2 test performs slightly better than the T^2 test. The χ^2 test is much more scalable in processing large amounts of audit data than the T^2 test. Because large amounts of computer audit/log data must be processed in real time for intrusion detection, the scalable χ^2 test provides a more viable solution for intrusion detection than the T^2 test.

SUMMARY

This chapter showed examples of computer audit/log data and network traffic data that can be collected for intrusion detection. It presented the features of activities in computer and network systems that have been extracted from computer audit/log data and network traffic data for intrusion detection. It also reviewed existing intrusion detection studies and the use of data mining techniques in those studies. The chapters illustrated the application of data mining techniques based on multivariate statistical anomaly detection to intrusion detection.

REFERENCES

- Anderson, D., Frivold, T., & Valdes, A. (1995). *Next-generation intrusion detection expert system (NIDES): A summary* (Tech. Rep. SRI-CSL-97-07). Menlo Park, CA: SRI International.
- Chou, Y.-M., Mason, R. L., & Young, J. C. (1999). Power comparisons for a Hotelling's T^2 statistic. *Communications in Statistics—Simulation and Computation*, 28, 1031–1050.
- Debar, H., Becker, M., & Siboni, D. (1992). A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy* (pp. 240–250). Los Alamitos, CA: IEEE Computer Society Press.
- Debar, H., Dacier, M., & Wespi, A. (1999). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, 31, 805–822.
- Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering*, SE-13, 222–232.
- DuMouchel, W. (1999). *Computer intrusion detection based on Bayes factors for comparing command transition probabilities* (National Institute of Statistical Sciences Tech. Rep. No. 91). Retrieved August, 2002 from <http://www.niss.org/downloadabletechreports.html>
- Emran, S. M., & Ye, N. (2002). Robustness of chi-square and Canberra techniques in detecting intrusions into information systems. *Quality and Reliability Engineering International*, 18, 19–28.
- Escamilla, T. (1998). *Intrusion detection: Network security beyond the firewall*. New York: Wiley.
- Eskin, E., Lee, W., & Stolfo, S. J. (2001). Modeling system calls for intrusion detection with dynamic window sizes. In *Proceedings of the Second DARPA Information Survivability Conference and Exposition (DISCEX II)* (pp. 165–175). Los Alamitos, CA: IEEE Computer Society Press.
- Fisch, E. A., & White, G. B. (2000). *Secure computers and networks: Analysis, design and implementation*. Boca Raton: CRC Press.
- Forrest, S., Hofmeyr, S. A., & Somayaji, A. (1997). Computer immunology. *Communications of the ACM*, 40(10), 88–96.
- Ghosh, K., Schwatzbard, A., & Shatz, M. (1999). Learning program behavior profiles for intrusion detection. In *Proceedings of the First USENIX Workshop on Intrusion Detection and Network Monitoring*.
- Javitz, H. S., & Valdes, A. (1991). The SRI statistical anomaly detector. In *Proceedings of the 1991 IEEE Symposium on Research in Security and Privacy*. Los Alamitos, CA: IEEE Computer Society Press.
- Javitz, H. S., & Valdes, A. (1994, March). *The NIDES statistical component description and justification* (Tech. Rep. A010). Menlo Park, CA: SRI International.
- Jou, Y., Gong, F., Sargor, C., Wu, X., Wu, S., Chang, H., & Wang, F. (2000). Design and implementation of a scalable intrusion detection system for the protection of network infrastructure. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (pp. 69–83). Los Alamitos, CA: IEEE Computer Society.
- Ju, W. H., & Vardi, Y. (1999). *A hybrid high-order Markov chain model for computer intrusion detection*. (National Institute of Statistical Sciences, Tech. Rep. No. 92). Retrieved August, 2002 from <http://www.niss.org/downloadabletechreports.html>
- Kantowitz, B. H., & Sorkin, R. D. (1983). *Human factors: Understanding people-system relationships*. New York: Wiley.
- Ko, C., Fink, G., & Levitt, K. (1997). Execution monitoring of security-critical programs in distributed systems: A specification-based approach. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy* (pp. 134–144). Los Alamitos, CA: IEEE Computer Society Press.
- Kumar, S. (1995). *Classification and detection of computer intrusions*. Unpublished doctoral dissertation, Purdue University, West Lafayette, Indiana.
- Lee, W., Stolfo, S. J., & Mok, K. (1999). Mining in a data-flow environment: Experience in network intrusion detection. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*. New York: ACM Press.
- Li, X., & Ye, N. (2002). Grid- and dummy-cluster-based learning of normal and intrusive clusters for computer intrusion detection. *Quality and Reliability Engineering International*, 18, 231–242.
- Lippmann, R., Fried, D., Graf, I., Haines, J., Kendall, K., McClung, D., Weber, D., Webster, S., Wyschogrod, D., Cunningham, R., & Zissman, M. (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (pp. 12–26). Los Alamitos, CA: IEEE Computer Society.
- Montgomery, D. C., & Mastrangelo, C. M. (1991). Some statistical process control methods for autocorrelated data. *Journal of Quality Technology*, 23, 179–193.
- Northcutt, S., Cooper, M., Fearnaw, M., & Frederick, K. (2001). *Intrusion signatures and analysis*. Indianapolis, IN: New Riders.

- Pfleeger, C. P. (1997). *Security in computing*. Upper Saddle River, NJ: Prentice Hall PTR.
- Schonlau, M., DuMouchel, W., Ju, W. H., Karr, A. F., Theus, M., & Vardi, Y. (1999). *Computer intrusion: Detecting masquerades* (National Institute of Statistical Sciences Tech. Rep. No. 95). Retrieved August, 2002 from <http://www.niss.org/downloadtechreports.html>.
- Scott, S. L. (2002). *Detecting network intrusion using a Markov modulated nonhomogeneous Poisson process*. Retrieved August, 2002 from <http://www.rcf.usc.edu/~sls/fraud.ps>
- Skoudis, E. (2002). *Counter hack*. Upper Saddle River, NJ: Prentice Hall PTR.
- Stevens, W. R. (1994). *TCP/IP illustrated* (Vol. 1). Boston: Addison-Wesley.
- Viega, J., & McGaw, G. (2002). *Building secure software*. Boston: Addison-Wesley.
- Vigna, G., Eckmann, S., & Kemmerer, R. (2000). The STAT tool suite. In *Proceedings of the DARPA Information Survivability Conference and Exposition* (pp. 46–55). Los Alamitos, CA: IEEE Computer Society.
- Warrender, C., Forrest, S., & Pearlmuter, B. (1999). Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy* (pp. 133–145). Los Alamitos, CA: IEEE Computer Society Press.
- Ye, N. (2003). *Modeling and Analysis of Cyber-Security Data*. London: Springer-Verlag.
- Ye, N., Borror, C., & Zhang, Y. (in press). EWMA techniques for computer intrusion detection through anomalous changes in event intensity. *Quality and Reliability Engineering International*.
- Ye, N., & Chen, Q. (2001). An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. *Quality and Reliability Engineering International*, 17, 105–112.
- Ye, N., Chen, Q., & Mou, J. I. (2002). Univariate and multivariate noise cancellation for computer intrusion detection. In *Proceedings of the Third IEEE SMC Information Assurance Workshop* (pp. 284–291). New York: IEEE Press.
- Ye, N., Ehiabor, T., & Zhang, Y. (2002). First-order versus high-order stochastic models for computer intrusion detection. *Quality and Reliability Engineering International*, 18, 243–250.
- Ye, N., Emran, S. M., Chen, Q., & Vilbert, S. (2002). Multivariate statistical analysis of audit trails for host-based intrusion detection. *IEEE Transactions on Computers*, 51, 810–820.
- Ye, N., Giordano, J., & Feldman, J. (2001). A process control approach to cyber attack detection. *Communications of the ACM*, 44(8), 76–82.
- Ye, N., Li, X., Chen, Q., Emran, S. M., & Xu, M. (2001). Probabilistic techniques for intrusion detection based on computer audit data. *IEEE Transactions on Systems, Man, and Cybernetics*, 31, 266–274.
- Ye, N., Zhang, Y., & Borror, C. M. (in press). Robustness of the Markov chain model for cyber attack detection. *IEEE Transactions on Reliability*.

27

Mining Image Data

Chabane Djeraba

IRIN, Nantes University, France

Gregory Fernandez

Wayne State University

Introduction	637
Related Works	639
Method	641
How to Discover the Number of Clusters: k	641
K-Automatic Discovery Algorithm	644
Clustering Algorithm	646
Experimental Results	646
Data Sets	647
Data Item Representation	648
Evaluation Method	649
Results and Analysis	650
Summary	654
References	655

INTRODUCTION

A key issue in data analysis methods either in a model design step or as part of the model projection in real-time operations is the grouping of data items into clusters. First, the clustering analysis is essentially an unsupervised process. Its objective is to group a given repository of unclassified data items into meaningful clusters. In a sense clusters are data driven and obtained solely from the data. Second, clustering may support an supervised process on the basis of classes of preclassified data items. The objective is to classify newly encountered yet unclassified data items. Typically, the given clusters (obtained by the unsupervised process

training) of data items are used to learn the descriptions of classes, which in turn are used to classify new data items. As a result, data items within a cluster are more similar than data items belonging to different clusters.

Any clustering method belongs to either partition or hierarchical clustering classes. A partition clustering method produces a single partition of the data. A partition method is better suited to data mining than a hierarchical clustering method because it is better at handling large data sets for which the construction of a hierarchy is computationally expensive, and it is insensitive to the order of input data items.

In spite of important efforts in partition clustering developments (Jain, Murty, & Flynn, 1999), a problem remains, and few partition clustering approaches deal with it seriously. It is the automatic discovery of the number of clusters. What is the desired number, known by k ? Which is the best one? Is it often possible to have an idea of the desired number? When considering clustering in large repositories of data items such as images, is it realistic for the user, even if an expert, to specify the number of the desired clusters in an accurate way?

The goal of this chapter is to contribute answers to these questions by presenting a method that automatically discovers the number of clusters (k). Therefore, the method computes k automatically, rather than manually. We experimented with the method in image repositories. Two keys are necessary and sufficient to open the door of the k -discovery solution. The first one is the partition clustering method that supports multiple iterations according to multiple values of k . The second is the clustering confidence measure. It combines together an intercluster measure (global variance criterion ratio normalized; GVRCN) that considers the confidence of the whole clusters, and an intrACLusters measure (local variance criterion ratio; LVCR) that considers the confidence of individual clusters. We will show that the combination of two levels of measures presents an interesting result. Furthermore, the confidence of results depend not only on the clustering algorithm and cluster confidence measures, but also on data item descriptors. As data item descriptors, we tried Wavelet CDF (2,2)-Cohen-Daubechies-Feauveau (2,2), and CDF (1,1)-Cohen-Daubechies-Feauveau (1, 1) (Cohen, Daubechies, & Feauveau, 1992), known, too, by Haar. The former presents better confidence of clustering results compared with the latter, because it supports less noise. The choice of wavelet descriptors, as in Sheikholeslami and Chatterjee Surojit (2000), is justified by the fact that wavelet descriptors contribute to the detection of clusters of arbitrary frontiers; they are insensitive to the noise; and using the multiresolution property of wavelet descriptors contributes to effectively identifying frontier clusters arbitrarily at different degrees of detail.

We will not present in this chapter the different aspects of indexing and retrieval linked to clustering or how to do semantic clustering. Our goal in this chapter is an experimental study of relationships between content descriptions based on wavelets, k -automatic discovery, and confidence measures of clusters.

We can summarize the scope of the chapter in the following points:

- An overview clustering of large repositories of data items
- Data items, to be clustered, are images
- The number of clusters, known by k , is automatically discovered
- The description of images is based on wavelet coefficients

Confidence measures of the clusters are composed of GVRCN and LVRC.

The chapter is composed of the following sections. The second section presents the related works. The third section highlights our method; more particularly, it presents the k -discovery and clustering methods. The fourth section presents the result experiments of the proposed method.

RELATED WORKS

During these last 30 years many techniques, including partition methods, have been proposed to represent data, to measure proximity (similarity), and to group data items. The most popular partition clustering methods are k -means (McQueen, 1967), k -medoids (Vinod, 1969), and their variants, such as dynamic clustering (Diday, 1973). K -means methods create a random clustering, compute the center of gravity of each cluster, and then assign each data item of the data set to the cluster with the nearest center of gravity. This process is active until obtaining a stable state of clusters. K -medoids are very similar to k -means, except that the centers of gravity are data items of the data set. This characteristic avoids empty clusters. The k -medoids process follows, generally, two steps. In the first step the process selects k data items randomly. In the second step each data item of the data set is assigned, on the basis of similarity, to a selected cluster, called medoid or gravity center. The two steps are repeated until the best clustering is reached.

The different variants of k -medoids methods have been applied in several domains concerned with data item exploration, decision-making, document retrieval, image segmentation, and data item classification. We focus this presentation on three recent variants of k -medoids methods (Kaufman & Rousseeuw, 1990), which may be considered as a representative sample of partition clustering methods.

The first variant of k -medoids, is the partition around medoids (PAM) method. This method tries to find the best medoids from the data set. This is done by evaluation of result swapping, when a medoid is swapped with a data item. The method is robust and returns the best clustering results. However, it is inefficient: Compute TC_{ih} requires $(\tilde{n}k)$ operations (one per nonmedoid item). There are $k(\tilde{n}k)$ pairs, k selected items—medoids—and $(\tilde{n}k)$ unselected items; for each pair of data item we compute one TC_{ih} . Therefore, the complexity for one iteration is $O(k(\tilde{n}k)^2)$. Moreover, we cannot estimate the number of iterations. Here, n = the number of items; k = the number of clusters; x_i = the i th item; C_j = the j th cluster; C_{xi} = the cluster of x_i ; C_{ijh} = the cost of swapping roles of x_i (a medoid) and x_h (a normal item) for x_j . The total cost for swapping role of x_i and x_h : $TC_{ih} = \sum_j C_{ijh}$.

The second variant of k -medoids, is called clustering large applications (CLARA). The method is more suitable for large repositories of data items than the PAM method because it is computationally quicker, the cardinality of the sample is less voluminous than the data set, and its complexity is about $O(k(40k)^2 + k(\tilde{n}k))$ for each iteration. The method is applied to a sample of a data repository rather than an entire data repository. Therefore, it is quicker than PAM. However, the disadvantage of the method is the random selection of the sample, so the confidence of clustering is not as good as for the PAM method.

The third variant of k -medoids, is a clustering large application based on randomized search (CLARANS). The method presents better performance than the previous ones. The complexity of the method is about $O(\tilde{n}k)$ by iteration. Experimental results (Raymond & Jiawei, 1985) showed that for the same confidence of clustering, the method outperforms PAM, and for the same run time, the method provides better clusters than CLARA. On the basis of these efficiency and effectiveness characteristics, we considered this partition method as a basic level of our solution to support multiple run executions discover the best values of k .

The best variants of k -medoids methods, such as CLARANS, suffer of two shortcomings. The first is the automatic discovery of k . The second, which is shared by all clustering approaches, is the degeneration of the method performances when the data descriptors are high dimensional. The second disadvantage is not within the scope of the chapter; however, on basis of the strong presence and importance of the high dimensionality in multimedia descriptors, it is interesting to mention it. The high dimensionality of data descriptors does not simplify at all

the complexity of the clustering problem. The performance of lots of clustering algorithms decrease quickly with increasing dimension. The problem of clustering of high-dimensional data has been well investigated very recently, and clustering approaches such as X-tree (Berchtold, Keim, & Kriegel, 1996), Birch (Zhang, Ramakrishnan, & Linvy, 1996), Sting (Wand, Yang, & Muntz, 1997) and OptiGrid (Hinneburg & Keim, 1999) have been proposed to improve the efficiency and the effectiveness of the clustering. For example, OptiGrid (Hinneburg & Keim, 1999) finds clusters in high-dimension spaces with noise by projecting the data onto each axis and then partitioning the data using cutting planes at low-density points. The method shows how projections on subspaces of the input space improve the effectiveness of the clustering process. The method uses statistical measures to produce good projections and, hence, good clustering results.

In this chapter, we highlight an experimental solution to the second shortcoming: automatic discovery of k . In the state of the art, there are very few implemented and proven solutions. Furthermore, to our knowledge there are no solutions “ k -automatic discovery” that have been tested in image data sets. Therefore, we limit our presentation to one of the least tested approaches of the state of the art. The approach discovers k automatically in the context of spatial data mining (Raymond & Jiawei, 1985). The first difference between the approach and our method is the domain of application. Our domain of application is the content-based indexing of large image databases; however, the domain of application of the state of the art approach is spatial data mining. This difference means that there are different objectives, different data item descriptors, and different confidence measures. We particularly emphasize the confidence measure. Our method is to propose a solution that considers confidence measures that are more accurate for content-based indexing of large image databases than the approach proposed for spatial data mining. Our solution is inspired, partly, by metrics discussed in Milligan and Cooper (1985) for hierarchical clustering, extended and tested for image databases.

The state of the art approach, called spatial data mining based on clustering algorithms (Raymond & Jiawei, 1994) tries to find k , where k is the most suitable number of clusters for the input data sets. The approach adopts the heuristics of computing the silhouette coefficients developed in Kaufman and Rousseeuw (1990) and Dubes (1987). The silhouette of an object O_j is a value varying between -1 and 1 , which indicates how much O_j belongs to the cluster in which O_j is classified. The closer the value is to 1 , the higher the degree O_j belongs to its cluster. The silhouette value of a cluster is the average silhouette of all data items in the cluster. Based on experiments, Kaufman and Rousseeuw (1990) proposes the following interpretation of the silhouette: $71\% \leq \text{cluster silhouette} \leq 1$ means it is a strong cluster; $51\% \leq \text{cluster silhouette} \leq 70\%$ means it is a reasonable cluster, $26\% \leq \text{cluster silhouette} \leq 50\%$ means it is a weak or artificial cluster; less than 25% means no cluster was found. The silhouette value for k is the mean silhouette values of the k clusters. If the value of k is too small, the reason may be that the clusters are grouped together incorrectly. If the k value is too large, then some clusters may be artificially split. The most suitable k is the one with the highest silhouette value. The experiments of Kaufman and Rousseeuw showed that in spatial data mining, using the highest silhouette coefficient might not lead to intuitive results. For example, some clusters may not have reasonable information (e.g., silhouette value $<50\%$). Therefore, they introduce the following heuristics:

1. Find the value k with the highest silhouette value.
2. When all the k clusters have silhouette values $>51\%$, than *final k = k*, and stop. Otherwise, remove the data items in those clusters with silhouette values below 50% , if the total number of data items removed so far is less than 25% .

3. The data items removed are considered noise. Go back to step 1 for a new data set without noise.
4. When in step 3, the amount of noise to be removed exceeds the threshold, then setting final k value is set to 1, indicating in effect that no clustering is reasonable.

The usefulness of these heuristics has been highlighted in specific applications of spatial data mining. However, it is not realistic to generalize the applications of heuristics to any application domain. In our experiments on image data sets, considering 50% of the silhouette value is not realistic. Many images are close together with silhouette values less than 20%. That is why the heuristics used in this approach are not applied with the same ratios. A more fundamental difference concerns silhouette computing itself. The approach presented in the state of the art considers the silhouette of the clustering equal to the average of cluster silhouettes. A cluster silhouette is the average of the data item silhouettes of the considered cluster. This manner of deducing the silhouette of clusters is similar to LVRC, as used in our approach. However, our experiments showed that the highest values of the silhouette do not mean the best clustering. What is missing is the silhouette variance between clusters. The lowest silhouette variance between clusters has been considered in our method; we called it the global variance ratio criterion (GVRC). Therefore, the best value of k corresponds to the biggest silhouette with the lowest silhouette variance. In the following section we present an example in which the highest value of the silhouette (for $k = 8$) does not mean the best clustering; however, the lowest value of the GVRC with the highest silhouette corresponds the best clustering ($k = 13$). The GVRC also has been tested and compared with several measures in Milligan and Cooper (1985). Our experiments may be considered as a form of validation of the Milligan and Cooper (1985) experiments in image repositories.

METHOD

How to Discover the Number of Clusters: k

A critical question in the partition clustering method is the number of clusters (k). In the majority of real-world experimental methods, the number of cluster k is manually estimated. The estimation needs a minimum knowledge of both data repositories and applications, and this requires the study of data. Bad values of k can lead to very bad clustering. Therefore, automatic computation of k is one of the most difficult problems in cluster analysis.

To deal with this problem, we consider a method based on the confidence measures of the clustering. To compute the confidence of the clustering, we consider together two levels of measures: the global level and the local level. GVRC, inspired of Calinski and Harabasz (1974), underlines the global level, and LVRC, inspired by Kaufman and Rousseeuw (1990). GVRC has never been used in partition clustering. It computes the confidence of the whole cluster, and LVRC computes the confidence of individual clusters, known also as the silhouette. Therefore, we combine the two levels of measures to improve the accuracy the computation of the clustering confidence.

Our choices of GVRC and LVRC confidence measures are based on the following arguments. Previous experiments in Milligan and Cooper (1985) in the context of hierarchical clustering, examined 30 measures to find the best number of clusters. They applied all of the measures to test data set and computed how many times an index gave the right k . GVRC presented the best results in all cases, and GVRCN normalized the GVRC measure (GVRCN value is between 0 and 1). A legitimate question may be: How is this

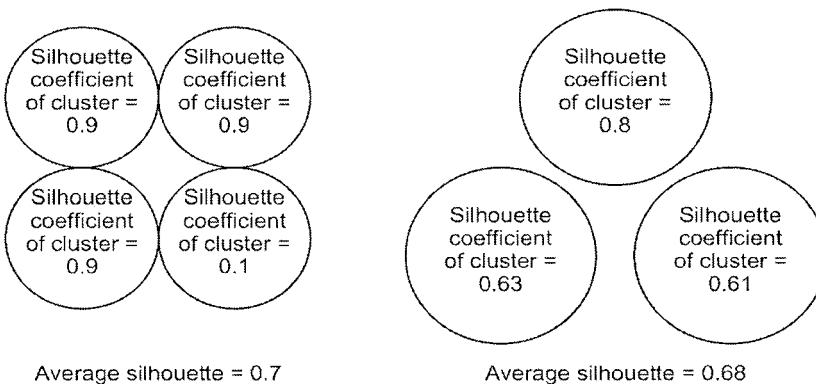


FIG. 27.1. Two clustering with $k = 4$ and $k = 3$.

measure (GVRC) good for partition methods? If GVRC is a good measure for hierarchical methods, is GVRC also good for partition methods? Are there any direct dependencies between data sets (in our case, image descriptors) and the confidence of clustering measures? What are the relationships between the initial medoids and the measures of clustering confidence?

It is too hard to answer all these questions at once without theory improvements and exhaustive real-world experiments. Therefore, our objective approach is not to answer all these important questions accurately, but to contribute to these answers by presenting the results of a real-world experiment. We extend a measure of clustering confidence by considering the best one presented in the literature (Milligan & Cooper, 1985). Then, we examined this measure in our repository data sets.

We combine the two levels of measures to make an accurate computation of clustering confidence. The combination of two levels of measures is an interesting facet of our method. It avoids a clustering state in which the max value of average the confidence of clusters does not mean the best confidence of clustering. For example, in Fig. 27.1, even if $k = 4$ the average confidence (0.7) of clustering is greater than the average confidence for $k = 3$ (0.61); however, for $k = 3$ the local confidence of clusters is better. All local confidence of clusters, when $k = 3$, are greater than 60%. And, for $k = 4$, three clusters have local confidence greater than 90% and one cluster less than 20%. If we consider $k = 4$, then we should ignore the fourth cluster, for which the local confidence is equal to 10%. It is considered as a noisy class. The data items of the fourth cluster are considered noise and then ignored, as in Raymond and Jiawei (1985). That is why we consider $k = 3$, because the variance is lowest, and the average confidence is among the highest, the local qualities are greater than 60%, and there are no noisy classes. On the basis of global and local measures, we consider $k = 3$ as the best number of clusters.

The method is improved by identifying the best cluster qualities with the lowest variance. In certain approaches, such as Raymond and Jiawei (1985), for different values of k , if the confidence of individual clusters is less than 50% and the cardinal of these clusters is less than 25%, then the data items are considered noise. So they are removed, and the approach recalculates the global and local qualities for the remaining data items. These data items are noise. Our experiments showed that fixing such a ratio (50 and 25%, etc.) is not suitable for evaluating cluster confidence. We often obtain good clustering, very similar to image references, with confidence values less than 20%.

GVRC is described by the following formula.

$$VRC(k) = \frac{\frac{\text{trace}(B)}{k-1}}{\frac{\text{trace}(W)}{n-k}}$$

where n is the cardinality of the data set, k is the number of clusters, $\text{trace}(B)/(k-1)$ is the dispersion (variance) between clusters, and $\text{trace}(W)/n - k$ is the dispersion within clusters. The expanded formula is:

$$GVRC(k) = \frac{n-k}{k-1} \frac{\left(\sum_{i=1}^n \|x_i - \bar{x}\|^2 \right) - \left(\sum_{l=1}^k \left(\sum_{x_j \in C_l} \|x_j - \bar{x}_l\|^2 \right) \right)}{\sum_{l=1}^k \left(\sum_{x_j \in C_l} \|x_j - \bar{x}_l\|^2 \right)}$$

The best clustering coincides with the maximum of GVRC. To normalize the GVRC value ($0 \leq GVRCN \leq 1$), we compute GVRCN, where $GVRC_max = GVRC(k')$, and $\forall k, 0 < k \leq k_max, k \neq k'$, $GVRC(k) < GVRC(k')$. K_max is the maximum of the clusters considered. $K_max \leq$ the cardinal of the data items.

$$GVRCN(k) = \frac{GVRC(k)}{GVRC_max}$$

LVRC measures the confidence of cluster j (C_j). The formula is:

$$LVRC(c_j) = \frac{\sum_{i=1}^{\text{cardinality}(c_j)} lvrc_{x_i}}{\text{cardinality}(c_j)}, \quad \text{with } lvrc_{x_i} = \frac{b_{x_i} - a_{x_i}}{\max(a_{x_i}, b_{x_i})}$$

LVRC measures the probability of x_i to belong to the cluster C_{xi} , where a_{xi} is the average dissimilarity of object x_i to all other objects of the cluster C_{xi} , and b_{xi} is the average dissimilarity of object x_i to all objects of the closest cluster C'_{xi} (neighbor of object x_i). Note that the neighbor cluster is rather a second-best choice for object x_i . When cluster C_{xi} contains only one object x_i , the s_{xi} is set to zero ($LVRC_{xi} = 0$).

In our experiments we considered descriptor size: 2048 float, wavelet type: CDF (2, 2). To speed up the clustering process, we specified that $4 \leq k \leq 15$, because we know in advance that the best value of k turns around 10. After activating the clustering method that discovers the best value of k , we obtain the following results:

- $k = 8$, LVRC = 16%, 1 – GVRCN = 55%, Medoids = {flag05, dawndusk00, belfry04, waterfall05, waterfall08, animals08, boat03, usa07}
- $k = 10$, LVRC = 3%, 1 – GVRCN = 60%, Medoids = {flag07, dawndusk09, belfry04, waterfall05, waterfall08, animals08, montagne08, boat03, flower08, animals07}
- $k = 13$, LVRC = 10%, 1 – GVRCN = 90%, Medoids = {flag05, dawndusk00, belfry04, animals07, waterfall05, animals06, flower08, boat01, montagne08, waterfall07, waterfall08, usa06, avion07}
- $k = 14$, LVRC = 9%, GVRCN = 85%, Medoids = {flag05, dawndusk09, belfry04, animals06, waterfall05, animals08, flower08, boat03, belfry02, waterfall09, waterfall08, usa07, belfry03, flower05}

If we consider $k = 13$, we notice that the LVCR of any cluster is greater than 8% (e.g., Table 27.1). However, for $k = 8$, certain LVCRs of clusters are greater than 10% and others

TABLE 27.1
 $K = 2$ Clusters: (LVRC: 0.228815, GVRCN: 0.15)

	<i>LVRC</i>	<i>Medoid</i>	<i>Data Items</i>									
1	0.126265	usa05	usa01	usa02	usa04	Usa06	usa07	Usa08	belfry00	belfry01	belfry02	belfry03
2	0.331365	flag05	flag00	flag01	flag02	Flag03	flag04	Flag06	flag07	flag08	flag09	

are less than 5%, so the variance of cluster qualities for $k = 8$ is greater than the variance of cluster qualities for $k = 13$. So $\text{GVRCN}(k = 13) < \text{GVRCN}(k = 8)$. For $k = 10$, clustering confidence is weak (3%). So the best value of k discovered automatically does not mean the best value of k manually referenced. However, considering $k = 13$ with $\text{LVRC} = 9\%$, we obtain the best value of recall and accuracy. That is why in our approach we consider the k values with the greatest values of LVCR and with the lowest values of the variances. Final confidence = $(\text{LVRC} + (1 - \text{GVRCN}))/2$. In our example, $\text{confidence}(k = 13) = (10\% + 90\%)/2 = 50\%$. Confidence ($k = 8$) = $(16\% + 55\%)/2 = 35\%$. When we look carefully at the clustering result for $k = 13$, we notice that there are two medoids “animals” (animals06 and animals08), two medoids “waterfall” (waterfall09, waterfall08), and two medoids “belfry” (belfry02, belfry03). So the clusters obtained automatically are more detailed than image references (clusters obtained manually). The resolutions considered to discriminate images are too high. From the user’s point of view there are no contradictions. If we used LVRC exclusively, as in the state of the art approaches (Raymond & Jiawei, 1985), we would consider $k = 8$ as the best confidence value, which is not the best clustering result. In our example, “flowers” and “mountains” have not been discriminated, because there are no medoids that represent flowers and mountains.

K-Automatic Discovery Algorithm

The algorithm is run several times with different k values, and the best configuration of k obtained from all runs is used as the output clustering.

We consider the sequence variable *sorted_clustering* initialized to an empty sequence ($<>$). *sorted_clustering* contains a sorted, on the basis of *confidence_i*, sequence of elements in the form of $<\text{confidence}_{i-1}, k_{i-1}>$, where k_i is the cluster number at i iteration, and *confidence_i* is the GVRC and all local variance ratio criterion associated to k_i clusters. *sorted_clustering* = $<, \dots, <\text{confidence}_{i-1}, k_{i-1}>, <\text{confidence}_i, k_i>, >$, where $\text{confidence}_{i-1} < \text{GVRC}(k), <\text{LVRC}(c_1), \text{LVRC}(c_2), \dots, \text{LVRC}(c_{k_i})>$.

The algorithm follows five steps:

- The first step initializes the maximum number of k authorized (*max_k*), by default $\text{max_k} = n/2$. n is the length of the data item repositories.
- The second step applies the clustering algorithm at each iteration k_i (for $k = 1$ to *max_k*), computes $\text{GVRC}(k)$, and for each k value computes $\text{LVRC}(c_j)$, for $j = 1, \dots, k$. $\text{confidence}_i = <\text{GVRC}(k), <\text{LVRC}(c_1), \text{LVRC}(c_2), \dots, \text{LVRC}(c_k)>$
- The third step normalizes the GVRC by computing the GVRCN. $\text{GVRC_max} = \text{max}_\text{GVRC}(k)$ where GVRC_max corresponds to the best clustering
- The fourth step considers only k clustering for which $\text{GVRCN}(k) = \text{GVRC}(k)/\text{GVRC_max}$ is the minimum value and $\text{LVRC}(k)$ is the maximum. So,

$$\frac{\sum_{i=1}^k \text{LVRC}(C_i)}{k} + (1 - \text{GVRCN}(k))$$

is the maximum, and $\forall j, j \in [1, k], LRC(c_k) \geq 1\%$. The results are sorted in *correct_sorted_clustering*. If the final *correct_sorted_clustering* is not empty, therefore, we have at least one solution and the algorithm is stopped. If not, the current step is followed by the fifth step.

- The fifth step looks for false or weak clusters ($LVRC < 1\%$). All false or weak data items of these false or weak clusters are moved to a specific cluster called “noisy cluster.” If the cardinality of the noisy cluster is less than 10%, then we compute the k -discovery without considering the false or weak data items. However, if the cardinal of the noisy cluster is greater than 10%, then we consider that there is too much noise and the data item features of the initial repositories should be reconsidered before again applying the algorithm. We deduce that the data item descriptors are not suited to clustering analysis.

```

K-discovery()
{
Set max_k //By default max_k = n/2
sorted_clustering <- <> /* empty sequence */
sorted_clustering contains a sequence of sorted qualities of
clustering and associated k.
sorted_clustering = <..., <confidence_{i-1}, k_{i-1}>, <confidence_i, k_i>>,
where confidence_{i-1} < confidence_i. <confidence_i, k_i> = <GVRC(k_i),
k_i>
// The clustering confidence is often the best when the value
is high.
for k=1 to max_k do
    Apply chosen clustering algorithm.
    current <- GVRC(k), <LVRC(c_1), LVRC(c_2), ..., LVRC(c_k)>
    /* GVRC(k) = confidence of actual clustering */
    sorted_clustering <- insert <k, current> in best_clustering, by
    considering sorted sequence of sorted_clustering.
    end for
    GVRC_max <- max_confidence (sorted_clustering) /*GVRC_max =
    best_clustering */
    correct_sorted_clustering <- <>
    for k=1 to max_k do
        if VRCN(k) =  $\frac{VRC(k)}{VRC\_max} \geq 1\% \text{ and } \forall j, j \in [1, k], LRC(c_k) \geq 1$ 
            then /* C_k is a correct cluster */
                correct_sorted_clustering <- insert <k, confidence_k> in
                correct_sorted_clustering>
            end if
        end for
        if correct_sorted_clustering = <>
        then
        {
        for i=1 to cardinal(correct_sorted_clustering) do
        {
        moving false or weak data items of C_{ij} for which LRC(c_{ikj}) < 1% to
        Noisy_cluster
        if cardinal(Noisy_cluster) < 10% then k-discovery() without

```

```

    considering the noisy data items in Noisy Class
}
Return correct_sorted_clustering.
}

```

Clustering Algorithm

The clustering algorithm is a variant of k -medoids, inspired by Ray and Jiawei (1994). The particularity of the algorithm is the replacement of sampling by heuristics. Sampling consists of finding better clustering by changing one medoid. But finding the best pair (medoid, item) to swap is very costly ($O(k(\bar{n}k)^2)$). That is why heuristics have been introduced in [Ray 94] to improve the confidence of the swap (medoid, data item). To speed up the choice of a pair (medoid, data item), the algorithm sets a maximum number of pairs to test (*num_pairs*), then choose randomly a pair and compares the dissimilarity (the comparison is done by evaluating TC_{th}). If this dissimilarity is greater than the actual dissimilarity, the algorithm continues choosing pairs until the number of pairs chosen reaches the fixed maximum. The medoids found are very dependant on the k first medoids selected. So the approach selects k other item and restarts *num_tries* times (*num_tries* is fixed by user). The best clustering is kept after the *num_tries* tries.

```

Clustering()
{
Initialize num_tries and num_pairs
min_cost ← big_number
for k=1 to num_tries do
    current ← k randomly selected items in the entire data set.
    l ← 1
    repeat
        xi ← a randomly selected item in current
        xh ← a randomly selected item in {entire data set current}.
        if  $TC_{ih} < 0$  then
            current ← currentxi+xh
        else
            j ← j+1
            end if
        until j ≤ num_pairs
        if min_cost < cost(current) then
            best ← current.
        end if
    end for
    Return best.
}

```

EXPERIMENTAL RESULTS

We conducted experiments in large image data sets to analyze the performance of the k -automatic discovery method. However, before presenting the performance analysis, we outline the data sets and the metrics used to evaluate the performance.

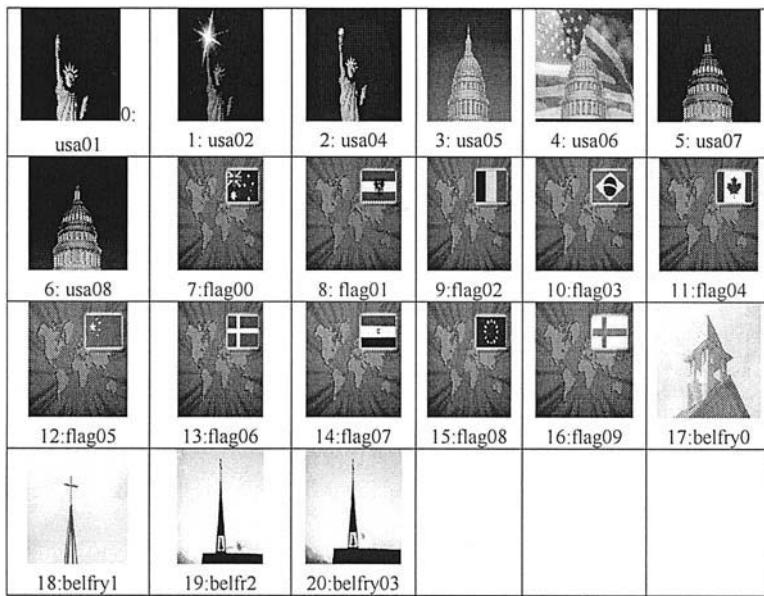


FIG. 27.2. Sample of a data set composed of 21 images.

Data Sets

We conducted 40 experiments on large image data sets covering a range of categories including panorama, scenery, flowers, and so on. The data sets came from collections compiled by IMSI Soft Company, which specializes in image libraries. These image data sets vary from 21 and 100,000 images of different resolutions. For example, the data set composed of 21 images is presented in Fig. 27.2. All images are cataloged into broad categories and each image carries an associated description. In this case the manual partition of image data sets into relevant clusters was feasible. The preclassification of images in semantic categories such as panorama and scenery helped our manual partition process. For all images we predetermined by hand a set of relevant clusters that constitutes a partition of image data sets. However, we may obtain several manual partitions that depend on our interpretation of image content. For example, in the sample of images in Fig. 27.3, manual clustering returns two “semantically correct” partitions. The first one (Table 27.2) is composed of two clusters (image references). It contains, respectively, U.S. symbols and flags. The second one (Table 27.3) is composed of five clusters (image references). For example, Cluster 1 (line 1 of Table 27.3) is composed of flag images = {flag00, flag01, flag02, Flag03, Flag04, flag05, flag06, flag07, Flag08, flag09}. It contains, respectively, flags, Statues of Liberty, Whites Houses, big belfries, and thin belfries. In Table 27.3 we consider essentially the visual proximity between images that is, cluster 1 = flag set, cluster 2 = Statue of Liberty set, cluster 3 = White House set, cluster 4 = big belfry set, and cluster 5 = thin belfry set. Of course, we can group all the belfries together, but because a person must cluster this data set without considering the semantic content of the data, when we consider just the visual proximity of data, the algorithm is not sure to group all belfries together. We can see that Belfry02 and belfry03 are very similar, and near belfry01. The flags usa01 usa02 and usa4 are also very similar. That is why they belong to the same clusters.

To summarize our approach, we can say that two clusterings are highlighted: manual and automatic clustering. Manual clustering generates several “correct” partitions. The clusters

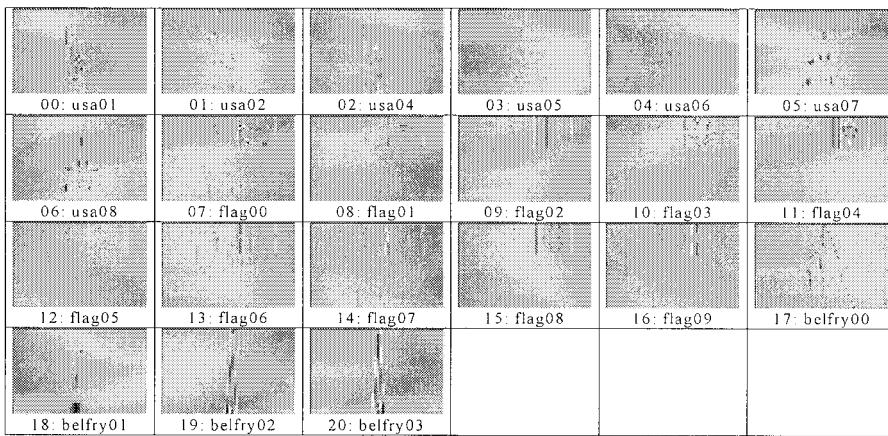


FIG. 27.3. Wavelet coefficients of experiment 4.

TABLE 27.2
First Manual Clustering of the Data Set Composed of 21 Images

Cluster 1	usa01	usa02	usa04	usa05	usa06	usa07	usa08	belfry00	belfry01	belfry02	belfry03
Cluster 2	flag00	flag01	flag02	flag03	flag04	flag05	flag06	flag07	flag08	flag09	

TABLE 27.3
Second Manual Clustering of the Data Set Composed of 21 Images

Cluster 1	flag00	flag01	flag02	flag03	flag04	flag05	flag06	flag07	flag08	flag09
Cluster 2	usa01	usa02	usa04							
Cluster 3	usa05	usa06	usa07	usa08						
Cluster 4	belfry00									
Cluster 5	belfry01	belfry02	belfry03							

obtained manually (image references) are compared with clusters obtained automatically on the basis of the following evaluation method.

Data Item Representation

Data items are represented by wavelet descriptors, and their proximity measures are based on Euclidean distance (for simplification). Image descriptors for indexing may be based on color histograms, anglogram, Fourier coefficients (frequency representation), and so on. The advantages of wavelet descriptors, compared with the previous descriptors are the representation of images with different resolutions, keeping in touch with the spatial and frequency information of the images.

The legitimate question is: how is the content of images represented by wavelet descriptors? Images are naturally represented by a two-dimensional spatial signal: abscissa (x) and ordinate (y). Although wavelet analysis could be generalized into a two-dimensional spatial signal, the computation involved would be time-consuming. Therefore, we consider the image signal as a one-dimensional linear signal. To maximize information details in the descriptors, we use four series of detail coefficients by scanning images in four different directions.

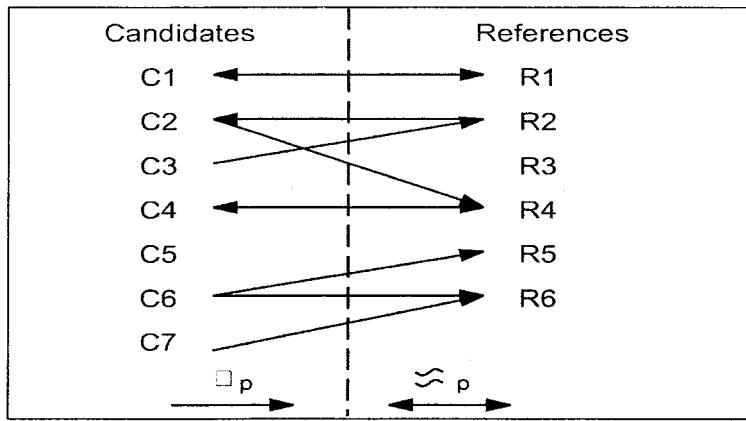


FIG. 27.4. Different cases of cluster matching.

With these directions we can extract horizontal, vertical, and diagonal remarkable frequency changes (i.e., horizontal, vertical, and diagonal contours). In addition, these different directions make the description of the image more accurate.

To speed up the clustering, we reduce the size of the descriptor vector. We choose a vector length equal to 2^n , with $n = 9, 10$. To reduce the feature vector to a smaller size, we apply the wavelet transform many times. In addition, we consider two reductions.

In the first reduction we loop on the lifting scheme; the input signal is the output signal of the previous iteration. When the signal is small enough (512, 1024, 2048), we store the wavelet coefficients as a feature vector.

In the second reduction the detail coefficients not saved in Fig. 27.4 contain information. How can we evaluate the lost information? In fact, in this approach we calculate all the detail coefficients and compute the average of each vector. Then, we reduce the vector with the higher average value. The reduction is the same as for the first method.

The higher the prediction degree of a wavelet, the more significant are the details of descriptors. Therefore, we experimented with Wavelet CDF (1,1): Haar and Wavelet CDF (2,2): Cohen-Daubechies-Feauveau (2,2). For Wavelet CDF (1,1): Haar, detail coefficients (or wavelet coefficients): $d_{j,1,l} = s_{j,2l+1} - s_{j,2l}$; coarser signal: $s_{j,1,l} = (s_{j,2l} + s_{j,2l+1})/2$. For Wavelet CDF (2,2): Cohen-Daubechies-Feauveau (2,2), detail coefficients (or wavelet coefficients): $d_{j,1,l} = s_{j,2l+1} - (s_{j,2l} + s_{j,2l+2})/2$, coarser signal: $s_{j,1,l} = s_{j,2l} + (d_{j,1,l1} + d_{j,1,l})/4$. We choose to reduce the size of the feature vector to 512, 1024, and 2048.

Evaluation Method

The clustering is evaluated by comparing automatic and manual clustering (image references). Manual clustering is expected. The evaluation method, inspired by Koschke and Eisenbarth (2000), considers the following input parameters: initial data set: I ; reference clusters R_i , $\cup R_i = I$, R_i is obtained manually; candidate clusters C_i that are obtained automatically by the method presented in the previous section, $\cup C_i = I$. It also considers the following output measures: $overlap(C, R) = [(|C \cap R|)/(|C \cup R|)]$; affinity relationship \approx_p , $X \approx_p Y$ if and only if $overlap(X, Y) \geq p$; partial subset relationship \subseteq_p , $X \subseteq_p Y$ if and only if $[(|X \cap Y|)/(|X|)] \geq p$. P measures the degree of overlapping between two sets. A significant value of P means significant overlapping between two sets. And conversely a weak value of p means a weak overlapping between two sets.

The evaluation method compares the reference clusters with candidate clusters. So, for each couple of clusters (R_i, C_j) , we obtain good matches: $R_i \approx_p C_j$; this matching is denoted $1 \sim 1$. Or, acceptable matches: $R_i \subseteq_p C_j$ or $C_j \subseteq_p R_i$ and not $R_i \approx_p C_j$. If $R_i \subseteq_p C_j$ then the candidate cluster C_j is too detailed. This case is denoted $n \sim 1$. Conversely, if $C_j \subseteq_p R_i$, then the candidate cluster has too few details. This case is denoted $1 \sim n$. There may also be false positive matches. Candidate clusters that neither match a reference nor are matched by any reference are called *false positives*. Inversely, they are *true negatives* when they are not even partially detected.

In Fig. 27.1 we have:

- Good matches = $\{(C_1, R_1), (C_4, R_4)\}, -\{n \sim 1\} = \{(R_2, C_2)\}, -\{1 \sim n\} = \{(C_3, R_2), (C_2, R_4), (C_6, R_5), (C_6, R_6), (C_7, R_6)\}$
- False positive = $\{C_5\}$
- True negative = $\{R_3\}$

The ideal evaluation scheme is composed exclusively of good matches. In this case, \forall a cluster C , \exists a reference R for which $C \approx_p R$, and \forall a reference R , \exists a cluster C for which $C \approx_p R$, with $p = 1.0$. However, a more realistic evaluation scheme considers a vector composed of the number of false positives, true negatives, the average accuracies of $1 \approx_p 1$, $1 \approx_p n$, $n \approx_p 1$ matches, and *overall recall rate*, with $p = 70\%$.

For two clusters, the accuracy between A and B , denoted $accuracy(A, B) = overlap(A, B)$, and for two sets of clusters: $accuracy(\{A_1, A_2, \dots, A_a\}, \{B_1, B_2, \dots, B_b\}) = overlap(\cup_{i=1}^a A_i, \cup_{i=1}^b B_i)$. So, for a class M of matches (M considers clusters in which we have: $1 \approx_p 1$, $1 \approx_p n$ or $n \approx_p 1$):

$$accuracy(M) = \frac{\sum_{(a,b) \in M} accuracy(a, b)}{card(M)}$$

Overall recall rate:

$$recall = \frac{\sum_{(a,b) \in GOOD} accuracy(a, b) + \sum_{(a,b) \in OK} accuracy(a, b)}{card(GOOD) + card(OK) + card(true\ negative)}$$

where *card* is the cardinality of a set. In other words, it corresponds to the number of data items in the set.

Results and Analysis

We considered 40 experiments with wavelet CDF (1, 1), wavelet CDF (2, 2), monochrome, RGB colors with descriptor size equal to 512 and 2048 elements, and data sets of respectively 21, 100, 1,000, 10,000 and 100,000 images. We obtained the results shown in Table 27.1.

In the column color, RGB means “red, green, blue,” so these are color images. N/B means “black and white” images. To illustrate the meaning of an experiment, we detail experiment 4 (see Table 27.4).

The descriptor size is 2048, wavelet type is CDF (2, 2) (Cohen et al., 1992), color information is RGB, and the cardinality of the data set is equal to 21 images. In Fig. 27.2 we present a visual representation of wavelet image descriptors of the data set presented in the Fig. 27.1. We can see using these visual images how multiresolution representation based on wavelets

TABLE 27.4
Experiments

Experiment	Size of Descriptor	Wavelet Type	Color	Data Sets
1	2048	CDF(1,1)	N/B	21 images
2	2048	CDF(1,1)	RGB	21 images
3	2048	CDF(2,2)	N/B	21 images
4	2048	CDF(2,2)	RGB	21 images
5	512	CDF(1,1)	N/B	21 images
6	512	CDF(1,1)	RGB	21 images
7	512	CDF(2,2)	N/B	21 images
8	512	CDF(2,2)	RGB	21 images
9	2048	CDF(1,1)	N/B	100 images
10	2048	CDF(1,1)	RGB	100 images
11	2048	CDF(2,2)	N/B	100 images
12	2048	CDF(2,2)	RGB	100 images
13	512	CDF(1,1)	N/B	100 images
14	512	CDF(1,1)	RGB	100 images
15	512	CDF(2,2)	N/B	100 images
16	512	CDF(2,2)	RGB	100 images
17	2048	CDF(1,1)	N/B	1000 images
18	2048	CDF(1,1)	RGB	1000 images
19	2048	CDF(2,2)	N/B	1000 images
20	2048	CDF(2,2)	RGB	1000 images
21	512	CDF(1,1)	N/B	1000 images
22	512	CDF(1,1)	RGB	1000 images
23	512	CDF(2,2)	N/B	1000 images
24	512	CDF(2,2)	RGB	1000 images
25	2048	CDF(1,1)	N/B	10000 images
26	2048	CDF(1,1)	RGB	10000 images
27	2048	CDF(2,2)	N/B	10000 images
28	2048	CDF(2,2)	RGB	10000 images
29	512	CDF(1,1)	N/B	10000 images
30	512	CDF(1,1)	RGB	10000 images
31	512	CDF(2,2)	N/B	10000 images
32	512	CDF(2,2)	RGB	10000 images
33	2048	CDF(1,1)	N/B	100000 images
34	2048	CDF(1,1)	RGB	100000 images
35	2048	CDF(2,2)	N/B	100000 images
36	2048	CDF(2,2)	RGB	100000 images
37	512	CDF(1,1)	N/B	100000 images
38	512	CDF(1,1)	RGB	100000 images
39	512	CDF(2,2)	N/B	100000 images
40	512	CDF(2,2)	RGB	100000 images

discriminates the outline of the image. The wavelet descriptor is a suitable signature of images, particularly when the images support variations of textures and colors.

As mentioned previously, the data set may be manually classified in two or five clusters. So we may have two or five image references. Then we activate the clustering method, which is the focus of this chapter, to find the best values of k . To limit the computing time, we introduce the born [2, 8]. This born means that $8 \geq k \geq 2$. So, k have to be between 2 and 8. Tables 27.5–27.10 show the LVRC and GLVRC values for each cluster.

The best value of k is 2 with LVRC = 22.88%, GVRCN = 15% and with 2 (good+ acceptable) matches with image references. More generally, we notice that for all experiments the best value of k is 2 (see Tables 27.11 and 27.12).

TABLE 27.5

TABLE 27.6

<i>Silhouette Coefficient</i>	<i>Medoid</i>	<i>Data Items</i>									
1	0.162024	usa05	usa01	usa02	usa04	Usa06	usa07	usa08	belfry00	belfry01	belfry02
2	0.209901	flag05	flag00	flag02	flag03	flag07					
3	0.000000	belfry03									
4	0.030651	flag08	flag01	flag04	flag06	flag09					

TABLE 27.7

TABLE 27.8
Six Clusters: (LVRC: 0.079843, GVRCN = 0.43)

<i>Silhouette Coefficient</i>	<i>Medoid</i>	<i>Data Items</i>						
1	0.194658	usa05	usa01	usa02	usa04	Usa06	usa07	usa08
2	0.209901	flag05	flag00	flag02	flag03	flag07		
3	0.000000	belfry03						
4	0.030651	flag08	flag01	flag04	flag06	flag09		
5	0.000000	Belfry02						
6	0.000000	Belfry00						

TABLE 27.9

	Silhouette Coefficient	Medoid	Data Items				
			usa05	usa01	usa02	usa04	Usa06
1	-0.008364	usa05	usa01	usa02	usa04	Usa06	belfry01
2	0.209901	flag05	flag00	flag02	flag03	flag07	
3	0.000000	belfry03					
4	0.030651	flag08	flag01	flag04	flag06	flag09	
5	0.000000	belfry02					
6	0.000000	belfry00					
7	0.552677	usa8	usa7				

TABLE 27.10
Eight Clusters: (LVRC: 0.099835, GVRCN = 0.7)

	<i>Silhouette Coefficient</i>	<i>Medoid</i>	<i>Data Items</i>			
1	0.021699	usa05	usa01	usa02	usa04	usa06
2	0.209901	flag05	flag00	flag02	flag03	flag07
3	0.000000	belfry03				
4	0.030651	flag08	flag01	flag04	flag06	flag09
5	0.000000	belfry02				
6	0.000000	belfry00				
7	0.536432	usa8	usa7			
8	0.000000	belfry01				

TABLE 27.11
Matches with Two References

2	1,000000	0	1	0.181818	0	0	0.727273
1	1,000000	0	1	0.55	0	0	0.7
2	1,000000	0	1	0.181818	0	0	0.727273
2	1,000000	0	0		0	0	1.0
2	1,000000	0	1	0.181818	0	0	0.727273
2	0.908333	0	1	0.181818	0	0	0.666162
2	1.0	0	0		0	0	1.0
2	1.0	0	0		0	0	1.0

TABLE 27.12
Matches with Five References

1	1	1	0.888889	1	0.666667	0	0	0.851852
1	1	1	0.571429	3	0.422222	0	0	0.567619
1	1	1	0.777778	1	0.5	0	0	0.759259
1	1	1	1	0		0	0	1
1	1	1	0.888889	1	0.666667	0	0	0.851852
1	0.9	1	0.8	1	0.666667	0	0	0.788889
1	1	1	1	0	0		0	1
1	1	1	1	0	0		0	1

We compare here the experiments 1–8 on the basis of overall recall rate with two image references and five image references respectively. So experiments 4, 7, and 8 return cluster results very close to those of manual clustering. The overall rate is near 1. The first conclusion is that the CDF (2,2) wavelet returns better results than CDF (1,1). This is due to the fact that CDF (2,2) attenuates the noise of images. Only real and hard changes in the color of the pictures are noticed. For example, a CDF (2,2) wavelet is not perturbed by a sky color gradation. When comparing experiments on the basis of good matches and acceptable matches to see which automatic clustering best matches our manual clustering, we obtain the results presented in the Fig. 27.5. All experiments return two (acceptable+good) matches. In this case there is no $1 \sim n$ matche (because there is no clustering less detailed than a clustering with two clusters).

Experiments 4, 7, and 8 all give the same results: two clusters. But if they return results comparable with our five-clusters clustering, that is because the manual clusters are totally included in another cluster. So our manual clustering is just more precise. The evaluation method looks for misplaced objects. That is why experiments 4, 7, and 8 return good results.

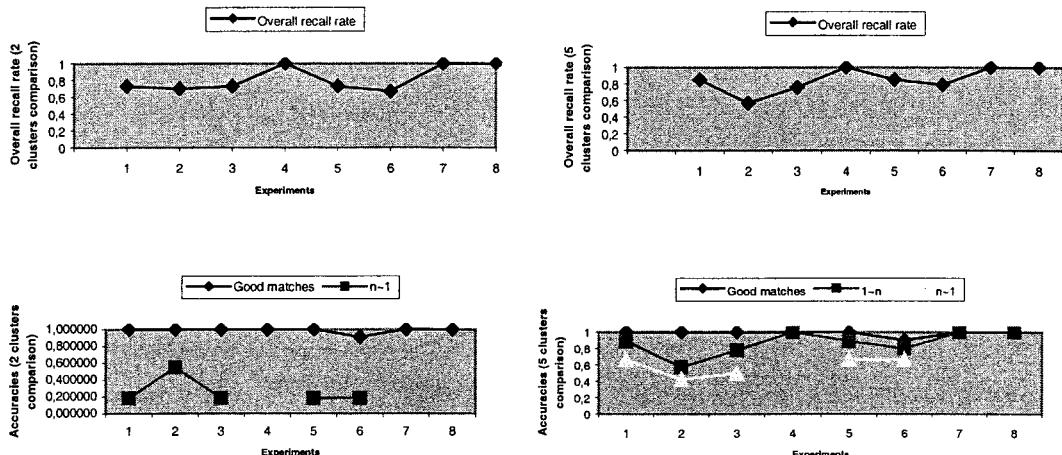


FIG. 27.5. Experiments 1–8.

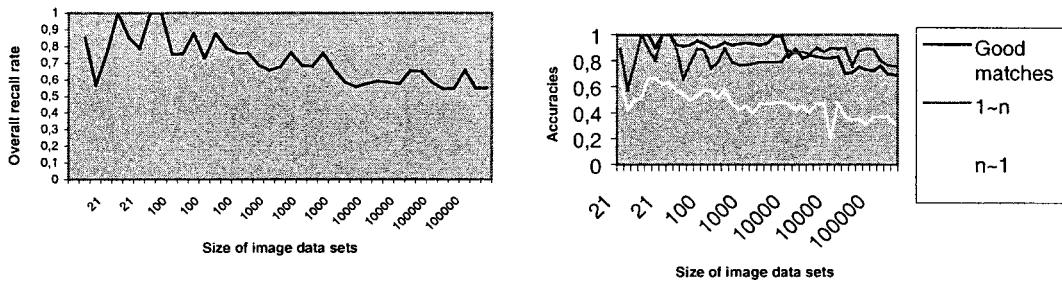


FIG. 27.6. Experiments 1–40.

Figure 27.6 illustrates the evolution of accuracies and overall recall rate for the 40 experiments. We notice the light degradation of accuracies and overall recall rate for very large image data sets. However, the measure $1 \sim n$ presents stable performance. This may be explained by the fact that the clusters obtained automatically are included in image references.

SUMMARY

This chapter focuses on k -discovery in the context of a partition method in voluminous data items. Partitions organize data item repositories (usually represented by vectors of values) into clusters based on similarity. More particularly, this chapter presents a method that automatically discovers the number of clusters (k). We tested the method on image repositories. Our strategy is based on the confidence measures composed of intercluster confidence (GVRCN) that considers the confidence of the whole cluster, and intracluster confidences (LVRC) that considers the confidence of individual cluster. We showed that the two levels of confidences present interesting results. Results of experiments showed that the value of k obtained automatically is generally the better one, if we consider the confidence measure computed on the basis of GVRCN and LVRC. So the value of k corresponds to clustering results that have in the

first step, the best GVRCN, and in the second step the best LVRC. Many current approaches consider only LVRC known by silhouette. In these cases the best values of k did not mean the best clusters, when compared with image references (manual clusters).

Many efforts should be focused to speed up the k -discovery process for very large data items. Our experiments showed that when considering very large data items, the algorithm of k -discovery remains time-consuming.

The experiments showed, too, that the type of wavelet coefficients considered—wavelet CFD (2,2)—have more influence on the final results than the levels of resolution (512 or 2048 wavelet coefficients) or color/gray parameters. So the clustering process in which image descriptors are based (wavelet CFD [2,2]) returns results that are better than wavelet CFD (1,1), independently of level or resolutions or color/gray parameters.

The overall experiment results showed that the confidence measures of clusters are close to image references (clusters obtained manually). In these cases is it sufficient to ensure that wavelet descriptors and the k -discovery method are well suitable to cluster semantically images? This question is not easy to answer. However, we can say that wavelet descriptors are suitable to discriminate images by considering levels of resolutions; however, it is not clear at all whether there is a causal link between levels of resolutions and semantic content of clusters.

REFERENCES

- Berchtold, S., Keim, D. A., & Kriegel, H.-P. (1996). The X-tree: An index structure for high dimensional data, *Proceedings of the 22nd International Conference on Very Large Data Bases* (pp. 28–39.) San Francisco: Morgan Kaufmann.
- Calinski, T., & Harabasz, J. (1974). A dendrite method for cluster analysis. *Communications in Statistics*, 3, 1–27.
- Cohen, A., Daubechies, I., & Feauveau, J. (1992). Bi-orthogonal bases of compactly supported wavelets. *Communications on Pure Applied Mathematics*, 45, 485–560.
- Diday, E. (1973). The dynamic cluster method on non-hierarchical clustering. *Journal of Information Science*, 2, 61–88.
- Dubes, R. C. (1987). How many clusters are best?—An experiment. *Pattern Recognition Journal*, 20, 645–663.
- Hinneburg, A., & Keim, D. (1999). Optimal grid-clustering: Towards breaking the curse of dimensionality. In *Proceedings of the 25th International Conference on Very Large Data Bases* (pp. 506–517). San Francisco: Morgan Kaufmann.
- Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31, 264–323.
- Kaufman, L., & Rousseeuw, P. J. (Eds.). (1990). *Finding groups in data: An introduction to cluster analysis*. New York: J. Wiley.
- Koschke, R., & Eisenbarth, T. (2000). A framework for experimental evaluation of clustering techniques. In *International Workshop on Program Comprehension (IWPC '2000)*.
- McQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297). Berkeley, CA: University of California.
- Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50, 159–179.
- Raymond, T. N., & Jiawei, H. (1994). *Efficient and effective clustering methods for spatial data mining* (Tech. Rep. TR-94-13). University of British Columbia, Vancouver, B.C., Canada.
- Sheikholeslami, G., & Chatterjee Surojit, Z. A. (2000). WaveCluster: A wavelet-based clustering approach for spatial data in very large databases. *VLDB Journal*, 8, 289–304.
- Vinod, H. D. (1969). Integer programming and the theory of grouping. *Journal of the American Statistical Association*, 64, 506–519.
- Wand, W., Yang, J., & Muntz, R. (1997). STING: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd Int. Conf. on Very Large Data Bases*, Santa Clara, CA: Morgan Kaufmann.
- Zhang, T., Ramakrishnan, R., & Linvy, M. (1996). BIRCH: An efficient data clustering method for very large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 103–114). New York: ACM Press.

28

Mining Manufacturing Quality Data

Murat C. Testik and George C. Runger
Arizona State University

Introduction	657
Multivariate Control Charts	658
Hotelling's T^2 Control Charts	658
MEWMA Charts	660
Nonparametric Properties of the MEWMA Control Charts	663
Summary	667
References	668

INTRODUCTION

In many modern plants massive process data sets are available from the recent developments in data acquisition and processing equipment. Although interest in quality control methods has increased substantially, far more improvements in quality and profitability can be achieved through mining of abundant and valuable process data. As a result of the developments in multivariate statistical methods and increased computer computation capabilities, quality practitioners are more interested in multivariate (multiple variable) quality methods.

The common practice of simultaneously monitoring multiple quality characteristics by maintaining a separate chart for each variable increases the probability that at least one chart will falsely signal an out-of-control situation. Furthermore, multivariate statistical techniques can gain advantage from the relationships among the variables to detect assignable causes that are not evident from separate control charts for each variable. The latter point is clearly illustrated in our example.

MULTIVARIATE CONTROL CHARTS

Many situations arise in which several related quality characteristics are to be monitored simultaneously. In this section, two commonly used multivariate control charts, namely Hotelling's T^2 and multivariate exponentially weighted moving average (MEWMA) charts, will be discussed. For further information on multivariate control charts the reader is referred to Jackson (1985); Lowry, Woodall, Champ, and Rigdon (1992); Lowry and Montgomery (1995); Pignatiello and Runger (1990); Prabhu and Runger (1997); Woodall and Montgomery (1999); and Montgomery (2001).

Hotelling's T^2 Control Charts

The pioneering work in multivariate quality control area by Hotelling (1947) has gained a great deal of attention and application over the years. This work established the multivariate analog of the univariate Shewhart control charts.

Let X_t be a $p \times 1$ vector of observations from a multivariate normal random process sampled sequentially at each time t . The vectors are assumed to be independent over time, but it is expected that there are substantial correlations between variables. Furthermore, let the process mean vector be μ_0 and the covariance matrix be Σ , such that $X_t \approx N_p(\mu_0, \Sigma)$ when the process is in control.

The χ^2 control statistic for an observation vector is the squared statistical distance from the in-control mean,

$$\chi^2 = (X_t - \mu_0)' \Sigma^{-1} (X_t - \mu_0).$$

Hence, a χ^2 control chart may be constructed by plotting the χ^2 statistic at each time an observation vector is available. An alarm indicating a statistically significant shift in the mean vector is triggered as soon as the χ^2 statistic exceeds a predetermined upper control limit,

$$h_1 = \chi_{\alpha, p}^2$$

where $\chi_{\alpha, p}^2$ is the upper α percentage point of the chi-square distribution with p degrees of freedom.

When the covariance matrix Σ is unknown and must be estimated, the T^2 control statistic may be used. Let S denote the sample covariance matrix calculated from m observation vectors representing the normal operating conditions of a process. This matrix is commonly used as an estimate of the in-control process covariance matrix, Σ . Then the Hotelling T^2 control chart is a plot of the T^2 statistic,

$$T^2 = (X_t - \mu_0)' S^{-1} (X_t - \mu_0)$$

along with the upper control limit,

$$h_1 = \frac{(m^2 - 1)p}{m(m - p)} F_{\alpha, p, m-p}$$

where $F_{\alpha, p, m-p}$ is the upper percentage point of the F distribution with p and $m - p$ numerator and denominator degrees of freedom, respectively. Note that, when m is large, it is customary to approximate the upper control limit by $\chi_{\alpha, p}^2$. Lowry and Montgomery (1995) provided the

minimum value of m , necessary to maintain a relative error of less than 0.1 when the chi-square approximation is used as the upper control limit.

As in the univariate Shewhart control charts, Hotelling T^2 charts perform well in rapid detection of moderate to large shifts in the mean. However, their performance degrades quickly in detecting moderate to small shifts.

Example 1. In semiconductor manufacturing a thin layer of dielectric is deposited on each wafer in a high-vacuum plasma operation. Standardized thickness measurements of the dielectric layer from two fixed locations are presented in Table 28.1 for 40 runs.

TABLE 28.1
Thickness Measurements with χ^2 and MEWMA Control Chart Statistics

Run No.	Location 1	Location 2	Hotelling T^2	MEWMA T_t^2
1	-1.56	-1.71	3.38	0.00
2	2.08	1.23	4.31	1.55
3	-0.02	1.06	1.79	1.60
4	0.89	0.55	0.80	2.17
5	0.14	-0.49	0.53	1.30
6	1.69	1.75	3.70	4.09
7	-0.28	0.08	0.17	2.35
8	-0.04	2.38	9.02	6.67
9	-0.16	-0.53	0.32	3.00
10	0.58	-0.44	1.30	1.34
11	-0.89	-2.11	4.67	0.49
12	0.26	-0.48	0.70	1.13
13	-0.88	0.08	1.35	0.19
14	-0.40	-0.52	0.28	0.44
15	-0.15	-1.62	3.68	2.56
16	0.41	-0.23	0.52	2.53
17	0.28	1.28	2.01	0.19
18	1.45	1.36	2.47	0.87
19	-0.58	0.06	0.61	0.26
20	-0.81	0.66	2.70	1.28
21	-0.16	0.79	1.25	2.49
22	-0.70	1.06	3.94	5.94
23	0.10	0.14	0.02	3.98
24	-0.89	-0.86	0.96	2.49
25	-2.31	-0.91	5.70	6.34
26	0.85	0.57	0.73	2.73
27	0.55	0.62	0.43	1.69
28	2.44	2.81	8.77	4.71
29	-0.04	0.69	0.80	4.93
30	2.26	2.44	6.92	10.01
31	0.83	-0.70	2.95	5.47
32	1.08	0.10	1.64	5.54
33	1.99	-0.60	8.97	10.20
34	-0.95	-0.56	0.90	4.08
35	-0.14	1.48	3.83	2.27
36	1.27	1.40	2.25	4.39
37	0.46	0.33	0.22	3.79
38	1.93	-1.31	13.23	8.19
39	0.65	1.27	1.64	6.14
40	-0.01	1.12	1.98	4.19

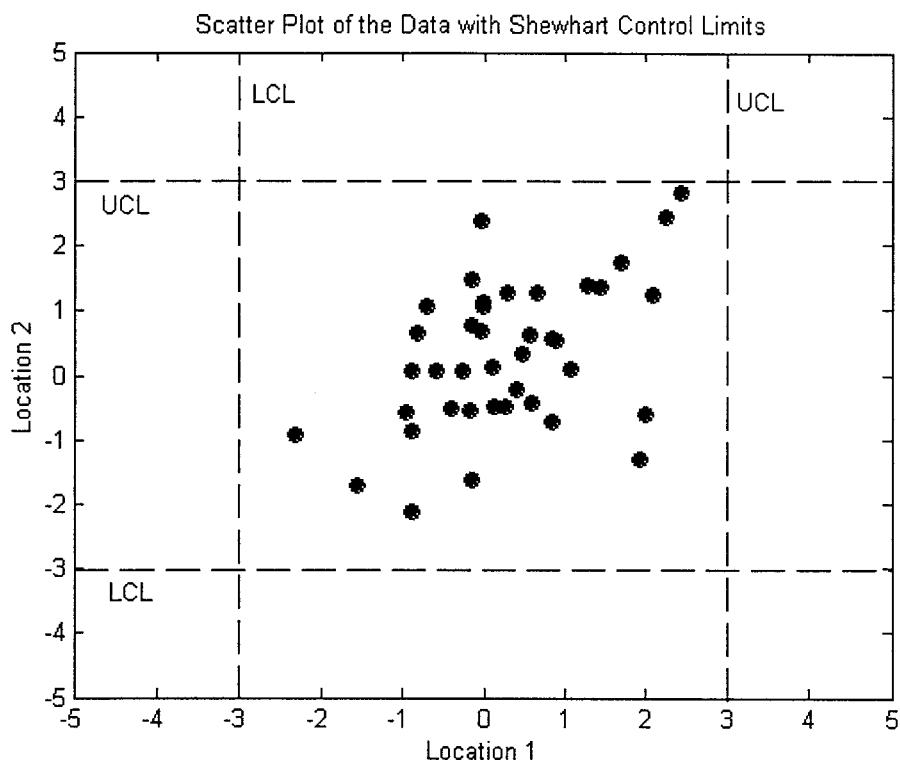


FIG. 28.1. Scatter plot of the thickness measurements with Shewhart control limits.

Note that the data are standardized to mean zero and unit variance for each location. It is also known that the covariance of the thickness measurements between the two locations is 0.6.

Using the two variables of interest, one can design two separate control charts. In Fig. 28.1, a scatter plot of the thickness data from two locations is presented. Although the data are not in time order, two upper (UCL) and lower (LCL) control limit pairs are included to indicate that these two charts do not trigger an out-of-control alarm. However, it is known that a sustaining assignable cause has occurred at time 25.

On the other hand, a multivariate control chart, which will take advantage of the correlation between the measurements, may be useful. A Hotelling T^2 control chart for the thickness data is given in Fig. 28.2. The T^2 values can be found from Table 28.1. A control limit (CL) of $h_1 = 10.6$ is used. Notice that the nonnormal event at time 25 is detected at time 38, because the Hotelling T^2 statistic 13.23 exceeds CL.

MEWMA Charts

As an alternative to the Shewhart control charts, Roberts (1959) introduced the exponentially weighted moving average (EWMA) control charts. These EWMA control charts are based on the intuitive idea of accumulating information over the time by an exponentially discounted weighting of the observations. Consequently, their ability to rapidly detect small to moderate changes in the mean is better than that of the Shewhart control charts.

Let x_t be a univariate normal distributed observation at time t and denote the in-control mean and variance by μ_0 and σ^2 , respectively. Hence, $x_t \approx N(\mu_0, \sigma^2)$ under the normal operating

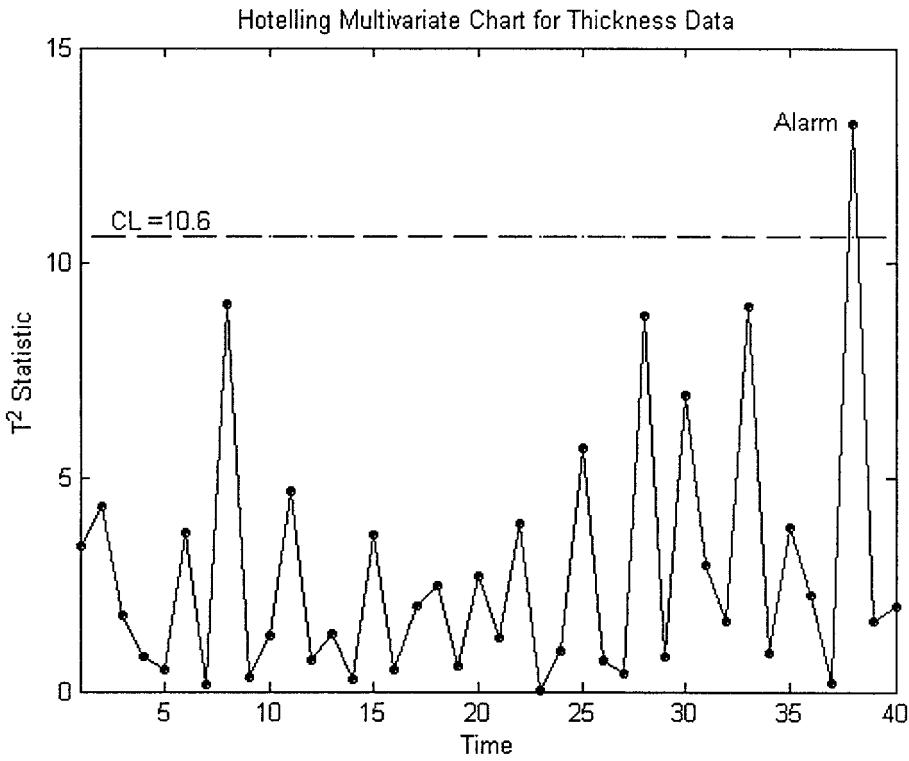


FIG. 28.2. Hotelling T^2 chart for the wafer thickness measurements.

conditions. The EWMA statistic z_t is computed as

$$z_t = rx_t + (1 - r)z_{t-1}$$

where $z_0 = \mu_0$ and $0 \leq r \leq 1$.

To detect a change in the mean, that is, $\mu \neq \mu_0$, the following asymptotic UCL is used for the increasing mean case,

$$h_u = \mu_0 + k\sigma \sqrt{\frac{r}{(2-r)}}.$$

As soon as $z_t \geq h_u$, presence of an assignable cause is considered. Similarly, to detect a decrease in the mean, the following asymptotic LCL is used,

$$h_l = \mu_0 - k\sigma \sqrt{\frac{r}{(2-r)}}.$$

Similarly, the presence of an assignable cause is considered as soon as $z_t \leq h_l$.

A straightforward multivariate extension of the univariate EWMA control chart is presented in Lowry et al. (1992). In the multivariate-quality-monitoring case, the $p \times 1$ MEWMA vector Z_t is

$$Z_t = RX_t + (I - R)Z_{t-1}$$

where R is a diagonal $p \times p$ matrix with the smoothing constants $0 \leq r_i \leq 1$, $i = 1, \dots, p$.

Lowry et al. suggested using $r_i = r$, $i = 1, \dots, p$ if there is no reason to weight past observations differently. In this case, MEWMA control charts are directionally invariant (Prabhu & Runger, 1997) and their performance depends only on the magnitude of the shift through the square root of the noncentrality parameter,

$$\lambda = [(\mu_1 - \mu_0)' \Sigma^{-1} (\mu_1 - \mu_0)]^{1/2}$$

where μ_1 is the out-of-control mean vector.

If it is desired to apply different smoothing constants for the variables in the observation vector, then the MEWMA control chart turns out to be direction-specific, and it loses its simplicity. From now on we consider the case where $r_i = r$, $i = 1, \dots, p$.

The MEWMA control chart is the plot of

$$T_t^2 = Z_t' \Sigma_Z^{-1} Z_t$$

where $\Sigma_Z = \{r/(2-r)\} \Sigma$ is the asymptotic covariance matrix of the MEWMA statistic. Because it is more common that a process starts in-control and then shifts to an out-of-control state, it is suggested to use the asymptotic covariance matrix. However, if it is believed that there may be some initial process upsets, one can use the exact covariance matrix.

A statistically significant change in the mean vector is detected the first time the MEWMA statistic exceeds the control limit h_2 , i.e. $T_t^2 \geq h_2$. Note that in the case $r_i = 1$, $i = 1, \dots, p$, the MEWMA control chart is equivalent to Hotelling's χ^2 control chart.

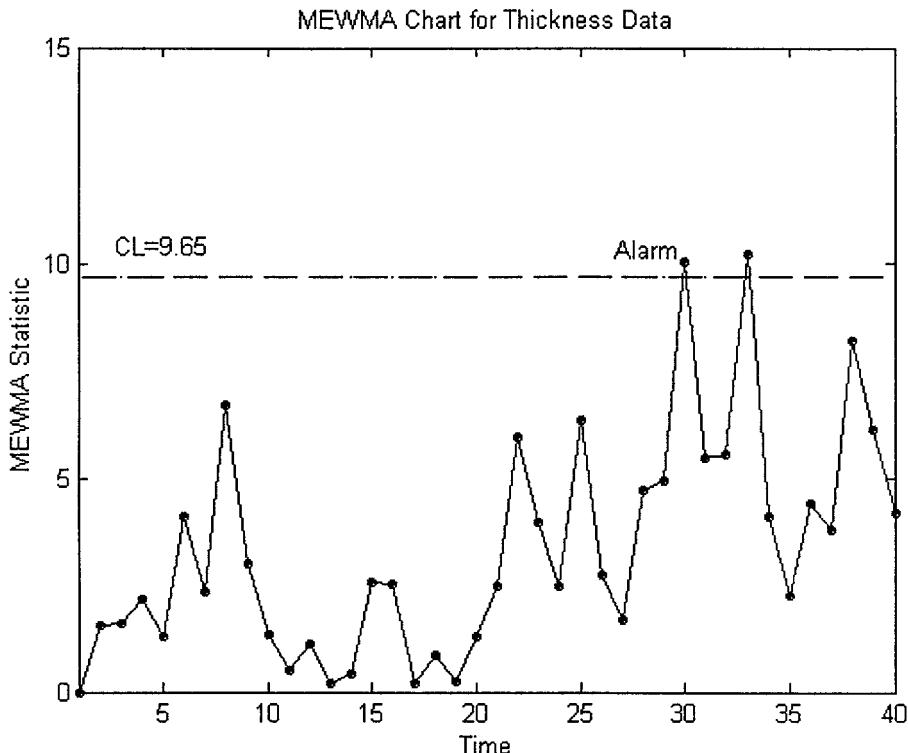


FIG. 28.3. MEWMA chart for the wafer thickness measurements.

Example 2. Consider again the thickness measurements data used in example 1. A MEWMA control chart with a CL of $h_2 = 9.65$ is presented in Fig. 28.3. Also, the MEWMA statistic values are given in Table 28.1.

Note that this chart triggers an alarm at time 30, when $T_{30}^2 = 10.01$ exceeds the CL the first time. Detection is faster than for the Hotelling chart. This faster response is due to the additional information gathered from the previous observations.

For the performance evaluation and design issues in MEWMA charts, the interested reader is referred to Runger and Prabhu (1996) and Prabhu and Runger (1997), respectively.

NONPARAMETRIC PROPERTIES OF THE MEWMA CONTROL CHARTS

One restrictive assumption in many multivariate control charts is the multivariate normality of the observations. The use of individual observation vectors does not assure the validity of this assumption through the central limit theorem. Consequently, quality practitioners should check the validity of the normality assumption as well as the performance sensitivity of multivariate control charts to departures from normality.

In a recent paper Borror, Montgomery, and Runger (1999) investigated the robustness of EWMA control charts to nonnormal data. Observations from various gamma distributions representing skewed data and from various t distributions representing symmetric, heavy-tailed data were simulated to test the performances of these charts. The in-control and out-of-control performance demonstrated that EWMA charts maintain their desired performance levels. Finally, it is concluded that EWMA control charts are extremely insensitive to nonnormal data and in this sense, they can be considered approximately nonparametric control charts.

In this section we extend the work of Borror et al. (1999) to MEWMA control charts. Through Monte Carlo simulations we investigate the performance of the widely used χ^2 control charts and the MEWMA control charts. These charts are designed under the assumption of multivariate normality, but the data is generated from a symmetric and heavy-tailed multivariate t distribution and a skewed multivariate gamma. We show that MEWMA control charts are almost nonparametric. For generating multivariate random numbers, the interested reader is referred to Ronning (1977) and Dagpunar (1988).

To evaluate control chart performance, run-length (RL) measurements are used. Here, RL can be defined as the number of time units until a monitoring chart triggers an alarm. The average in-control RL is denoted as ARL_0 , and it measures the mean number of plotted points until a control chart falsely signals an alarm. The average out-of-control RL, ARL_1 is the mean number of plotted points until a control chart signals an alarm when there is a change in the process mean.

The multivariate control charts considered are the χ^2 and MEWMA with $r = 0.05$ and 0.1 . We use different degrees of freedom for the multivariate t distribution and different scale parameters for the multivariate gamma distribution. The dimensions of the observation vectors are $p = 2, 4$, and 20 . To compare the results, performance of the above-mentioned control charts are also given for a multivariate normal distributed process.

Simulations are performed using Matlab 6.1. Each case is replicated for 5,000 runs. Steady-state control limits (asymptotic) are used. The out-of-control states are generated by a step change in the mean vector of the observations, which are initiated at the beginning of the simulations. Hence, MEWMA control charts are started at the out-of-control process mean vector. One can obtain slightly different results if the MEWMA control charts warm up (Prabhu & Runger, 1997) with a start at the in-control mean vector and a shift at a later time. The

TABLE 28.2
 In-Control ARLs for the MEWMA Control Charts for Various
 Multivariate t Distributions and Dimensions

	MEWMA			χ^2
	$R = 0.05I_p$	$R = 0.1I_p$	$R = 1I_p$	
$p = 2$	$h_2 = 7.40$	$h_2 = 8.66$	$h_1 = 10.60$	
MV Normal ($\mathbf{0}, I_p$)	201.66 (2.64)	198.67 (2.75)	200.34	
MV t (10)	112.32 (1.45)	93.27 (1.26)	37.05 (0.51)	
MV t (30)	167.50 (2.14)	151.68 (2.05)	94.97 (1.30)	
MV t (60)	185.41 (2.47)	174.79 (2.42)	133.44 (1.87)	
MV t (100)	190.43 (2.52)	188.75 (2.50)	152.07 (2.12)	
$p = 4$	$R = 0.05I_p$	$R = 0.1I_p$	$R = 1I_p$	
	$h_2 = 11.20$	$h_2 = 12.73$	$h_1 = 14.86$	
MV Normal ($\mathbf{0}, I_p$)	198.40 (2.67)	199.84 (2.71)	199.98	
MV t (20)	136.69 (1.72)	118.65 (1.57)	48.34 (0.68)	
MV t (60)	178.72 (2.31)	163.34 (2.17)	111.91 (1.58)	
MV t (120)	185.83 (2.47)	181.51 (2.41)	140.71 (1.97)	
MV t (200)	189.38 (2.49)	192.58 (2.61)	167.51 (2.38)	
$p = 20$	$R = 0.05I_p$	$R = 0.1I_p$	$R = 1I_p$	
	$h_2 = 34.50$	$h_2 = 37.01$	$h_1 = 40.00$	
MV Normal ($\mathbf{0}, I_p$)	198.77 (2.48)	204.78 (2.73)	200.18	
MV t (100)	171.41 (2.10)	159.99 (2.14)	77.15 (1.08)	
MV t (300)	188.31 (2.32)	183.76 (2.46)	136.68 (1.91)	
MV t (600)	196.79 (2.47)	192.71 (2.51)	164.24 (2.31)	
MV t (1000)	197.50 (2.49)	196.91 (2.55)	175.65 (2.43)	

TABLE 28.3
 In-Control ARLs for the MEWMA Control Charts for Various Multivariate
 Gamma Distributions and Dimensions

	MEWMA			χ^2
	$R = 0.05I_p$	$R = 0.1I_p$	$R = 1I_p$	
$p = 2$	$h_2 = 7.40$	$h_2 = 8.66$	$h_1 = 10.60$	
MV Normal ($\mathbf{0}, I_p$)	201.66 (2.64)	198.67 (2.75)	200.34	
MV Gam ($1I_p, 1I_p$)	200.24 (2.75)	153.69 (2.13)	31.12 (0.44)	
MV Gam ($2I_p, 1I_p$)	200.19 (2.67)	169.39 (2.35)	41.75 (0.58)	
MV Gam ($3I_p, 1I_p$)	203.56 (2.68)	184.64 (2.59)	48.70 (0.70)	
MV Gam ($4I_p, 1I_p$)	210.33 (2.80)	186.97 (2.67)	56.44 (0.79)	
$p = 4$	$R = 0.05I_p$	$R = 0.1I_p$	$R = 1I_p$	
	$h_2 = 11.20$	$h_2 = 12.73$	$h_1 = 14.86$	
MV Normal ($\mathbf{0}, I_p$)	198.40 (2.67)	199.84 (2.71)	199.98	
MV Gam ($1I_p, 1I_p$)	174.31 (2.35)	125.54 (1.71)	23.38 (0.33)	
MV Gam ($2I_p, 1I_p$)	184.48 (2.43)	152.00 (2.00)	31.76 (0.44)	
MV Gam ($3I_p, 1I_p$)	193.05 (2.53)	163.47 (2.20)	38.49 (0.54)	
MV Gam ($4I_p, 1I_p$)	195.74 (2.57)	172.36 (2.38)	45.56 (0.65)	
$p = 20$	$R = 0.05I_p$	$R = 0.1I_p$	$R = 1I_p$	
	$h_2 = 34.50$	$h_2 = 37.01$	$h_1 = 40.00$	
MV Normal ($\mathbf{0}, I_p$)	198.77 (2.48)	204.78 (2.73)	200.18	
MV Gam ($1I_p, 1I_p$)	153.60 (1.88)	104.38 (1.36)	14.49 (0.20)	
MV Gam ($2I_p, 1I_p$)	173.00 (2.24)	135.39 (1.84)	21.95 (0.31)	
MV Gam ($3I_p, 1I_p$)	179.04 (2.27)	146.23 (1.89)	28.91 (0.39)	
MV Gam ($4I_p, 1I_p$)	186.35 (2.39)	158.41 (2.07)	35.15 (0.49)	

TABLE 28.4
 Out-of-Control ARLs for MEWMA Control Charts with $p = 2$ for Various
 Multivariate t Distributions

		Noncentrality Parameter λ				
		$p = 2$	0.5	1.0	1.5	2.0
MEWMA	MV Normal	26.45 (0.21)	11.27 (0.06)	7.13 (0.03)	5.30 (0.02)	3.57 (0.01)
	MV t (10)	25.20 (0.22)	11.05 (0.06)	7.17 (0.03)	5.29 (0.02)	3.57 (0.01)
	$R = 0.05I_p$	25.87 (0.21)	11.28 (0.06)	7.14 (0.03)	5.30 (0.02)	3.57 (0.01)
	$h_2 = 7.40$	26.16 (0.21)	11.21 (0.06)	7.16 (0.03)	5.32 (0.02)	3.56 (0.01)
MEWMA	MV t (100)	26.58 (0.22)	11.17 (0.06)	7.15 (0.03)	5.28 (0.02)	3.56 (0.01)
	MV Normal	28.42 (0.28)	10.29 (0.07)	6.05 (0.03)	4.43 (0.02)	2.94 (0.01)
	MV t (10)	24.58 (0.25)	9.75 (0.07)	6.10 (0.03)	4.40 (0.02)	2.94 (0.01)
	$R = 1I_p$	26.67 (0.26)	10.15 (0.07)	6.03 (0.03)	4.42 (0.02)	2.93 (0.01)
$h_2 = 8.66$	MV t (30)	27.22 (0.28)	10.10 (0.07)	6.12 (0.03)	4.39 (0.02)	2.94 (0.01)
	MV t (60)	28.07 (0.28)	10.16 (0.07)	6.08 (0.03)	4.45 (0.02)	2.92 (0.01)
	MV t (100)	28.07 (0.28)	10.16 (0.07)	6.08 (0.03)	4.45 (0.02)	2.92 (0.01)
	MV Normal	115.71	41.97	15.79	6.88	2.16
χ^2	MV t (10)	30.94 (0.43)	19.73 (0.28)	10.61 (0.14)	5.82 (0.08)	2.11 (0.02)
	$R = 1I_p$	66.38 (0.95)	31.31 (0.45)	13.92 (0.20)	6.54 (0.09)	2.14 (0.02)
	$h_1 = 10.60$	86.25 (1.21)	35.66 (0.50)	14.67 (0.20)	6.70 (0.09)	2.18 (0.02)
	MV t (100)	95.05 (1.35)	38.82 (0.54)	15.01 (0.21)	6.81 (0.09)	2.14 (0.02)

computer program is validated using the published results in Lowry et al. (1992) and the univariate results of Borror et al. (1999).

The simulated ARL results are given in Tables 28.2 through 28.9 with the estimated standard errors of the ARL in parentheses. Table 28.2 and Table 28.3 are the in-control ARL performances of the χ^2 and the MEWMA control charts for multivariate normal (MV Normal), multivariate t (MV t), and multivariate gamma (MV Gam) distributions with various parameters and dimensions.

TABLE 28.5
 Out-of-Control ARLs for MEWMA Control Charts with $p = 4$ for Various
 Multivariate t Distributions

		Noncentrality Parameter λ				
		$p = 4$	0.5	1.0	1.5	2.0
MEWMA	MV Normal	32.20 (0.26)	13.51 (0.07)	8.55 (0.03)	6.27 (0.02)	4.22 (0.01)
	MV t (20)	30.57 (0.25)	13.22 (0.07)	8.43 (0.03)	6.23 (0.02)	4.20 (0.01)
	$R = 0.1I_p$	31.94 (0.26)	13.34 (0.07)	8.46 (0.03)	6.29 (0.02)	4.19 (0.01)
	$h_2 = 11.20$	32.24 (0.26)	13.37 (0.07)	8.49 (0.03)	6.29 (0.02)	4.22 (0.01)
MEWMA	MV t (120)	32.45 (0.26)	13.42 (0.07)	8.52 (0.03)	6.28 (0.02)	4.22 (0.01)
	MV t (200)	34.92 (0.35)	12.16 (0.08)	7.22 (0.03)	5.17 (0.02)	3.40 (0.01)
	MV t (20)	30.88 (0.31)	11.84 (0.08)	7.08 (0.04)	5.17 (0.02)	3.39 (0.01)
	$R = 0.1I_p$	33.74 (0.33)	12.01 (0.08)	7.16 (0.03)	5.14 (0.02)	3.40 (0.01)
$h_2 = 12.73$	MV t (120)	34.10 (0.35)	12.16 (0.08)	7.18 (0.03)	5.17 (0.02)	3.41 (0.01)
	MV t (200)	34.92 (0.35)	12.12 (0.08)	7.21 (0.03)	5.21 (0.02)	3.41 (0.01)
	MV Normal	138.13	60.95	24.62	10.63	2.93
	MV t (20)	42.26 (0.59)	26.96 (0.37)	15.77 (0.21)	8.34 (0.11)	2.82 (0.03)
χ^2	$R = 1I_p$	84.51 (1.21)	44.61 (0.62)	20.74 (0.62)	9.56 (0.13)	2.91 (0.03)
	$h_1 = 14.86$	08.65 (1.52)	50.34 (0.70)	23.18 (0.32)	10.34 (0.14)	2.95 (0.03)
	MV t (200)	15.92 (1.60)	55.02 (0.79)	23.69 (0.33)	10.30 (0.14)	2.92 (0.03)

TABLE 28.6
Out-of-Control ARLs for MEWMA Control Charts with $p = 20$ for Various
Multivariate t Distributions

		Noncentrality Parameter λ				
		$p = 20$	0.5	1.0	1.5	2.0
MEWMA $R = 0.05I_p$ $h_2 = 34.50$	MV Normal	52.89 (0.45)	21.52 (0.10)	13.58 (0.05)	10.01 (0.03)	6.72 (0.01)
	MV t (100)	50.82 (0.42)	21.13 (0.10)	13.51 (0.05)	10.02 (0.03)	6.65 (0.01)
	MV t (300)	52.65 (0.44)	21.28 (0.10)	13.54 (0.05)	10.04 (0.03)	6.69 (0.01)
	MV t (600)	52.50 (0.44)	21.38 (0.10)	13.56 (0.05)	9.99 (0.03)	6.68 (0.01)
	MV t (1000)	52.83 (0.44)	21.52 (0.10)	13.53 (0.05)	10.04 (0.03)	6.72 (0.01)
MEWMA $R = 0.1I_p$ $h_2 = 37.01$	MV Normal	63.70 (0.71)	20.09 (0.14)	11.33 (0.05)	8.03 (0.03)	5.18 (0.01)
	MV t (100)	57.27 (0.64)	19.68 (0.14)	11.20 (0.05)	7.98 (0.03)	5.19 (0.01)
	MV t (300)	60.32 (0.65)	19.81 (0.13)	11.31 (0.05)	7.96 (0.03)	5.18 (0.01)
	MV t (600)	61.73 (0.69)	19.96 (0.14)	11.26 (0.05)	7.98 (0.03)	5.18 (0.01)
	MV t (1000)	61.33 (0.68)	20.21 (0.14)	11.33 (0.05)	8.01 (0.03)	5.21 (0.01)
χ^2 $R = 1I_p$ $h_1 = 40.00$	MV Normal	173.60	117.01	66.24	34.28	9.11
	MV t (100)	68.89 (0.97)	53.87 (0.78)	35.84 (0.50)	22.84 (0.32)	7.76 (0.10)
	MV t (300)	121.36 (1.71)	85.22 (1.23)	51.94 (0.71)	29.25 (0.41)	8.77 (0.12)
	MV t (600)	45.07 (1.99)	99.25 (1.40)	58.86 (0.83)	31.74 (0.45)	8.78 (0.12)
	MV t (1000)	155.04 (2.20)	107.15 (1.45)	60.57 (0.87)	32.79 (0.46)	8.94 (0.12)

Similarly, the ARL_1 performances of MEWMA charts with $r = 0.05$ stay approximately unchanged for different out-of-control states. The out-of-control performances are given in Tables 28.4 through 28.9.

Because smaller values for r correspond to greater smoothing, it is expected that smaller values will provide more robustness to nonnormality. In practice, values of r from 0.05 to 0.2 are common. The results show an excellent robustness for $r = 0.05$ and still quite good performance for $r = 0.1$. Based on our simulation results, MEWMA control charts with r less than or equal to 0.1 will be insensitive to the normality assumption when monitoring individual multivariate vectors in a wide range of applications.

TABLE 28.7
Out-of-Control ARLs for MEWMA Control Charts with $p = 2$ for Various Multivariate
Gamma Distributions

		Noncentrality Parameter λ				
		$p = 2$	0.5	1.0	1.5	2.0
MEWMA $R = 0.05I_p$ $h_2 = 7.40$	MV Normal	26.45 (0.21)	11.27 (0.06)	7.13 (0.03)	5.30 (0.02)	3.57 (0.01)
	MV $Gam(1I_p, 1I_p)$	26.12 (0.24)	11.69 (0.08)	7.56 (0.05)	5.53 (0.03)	3.71 (0.02)
	MV $Gam(2I_p, 1I_p)$	26.47 (0.24)	11.57 (0.07)	7.40 (0.04)	5.51 (0.03)	3.71 (0.02)
	MV $Gam(3I_p, 1I_p)$	26.60 (0.24)	11.61 (0.07)	7.39 (0.04)	5.44 (0.03)	3.65 (0.02)
	MV $Gam(4I_p, 1I_p)$	26.52 (0.23)	11.43 (0.07)	7.29 (0.04)	5.44 (0.03)	3.67 (0.02)
MEWMA $R = 0.1I_p$ $h_2 = 8.66$	MV Normal	28.42 (0.28)	10.29 (0.07)	6.05 (0.03)	4.43 (0.02)	2.94 (0.01)
	MV $Gam(1I_p, 1I_p)$	25.20 (0.28)	10.57 (0.09)	6.51 (0.05)	4.72 (0.03)	3.15 (0.02)
	MV $Gam(2I_p, 1I_p)$	26.27 (0.29)	10.42 (0.08)	6.41 (0.04)	4.63 (0.03)	3.05 (0.02)
	MV $Gam(3I_p, 1I_p)$	26.35 (0.28)	10.19 (0.08)	6.33 (0.04)	4.56 (0.03)	3.03 (0.01)
	MV $Gam(4I_p, 1I_p)$	26.46 (0.28)	10.31 (0.08)	6.24 (0.04)	4.54 (0.02)	3.05 (0.01)
χ^2 $R = 1I_p$ $h_1 = 10.60$	MV Normal	115.71	41.97	15.79	6.88	2.16
	MV $Gam(1I_p, 1I_p)$	17.76 (0.25)	9.96 (0.13)	6.22 (0.08)	4.35 (0.05)	2.43 (0.03)
	MV $Gam(2I_p, 1I_p)$	22.29 (0.31)	12.16 (0.17)	6.93 (0.09)	4.49 (0.06)	2.33 (0.02)
	MV $Gam(3I_p, 1I_p)$	26.14 (0.35)	13.62 (0.18)	7.44 (0.10)	4.61 (0.06)	2.29 (0.02)
	MV $Gam(4I_p, 1I_p)$	28.34 (0.38)	14.31 (0.20)	7.97 (0.11)	4.78 (0.06)	2.28 (0.02)

TABLE 28.8

Out-of-Control ARLs for MEWMA Control Charts with $p = 4$ for Various Multivariate Gamma Distributions

		Noncentrality Parameter λ				
		0.5	1.0	1.5	2.0	3.0
$p = 4$						
MEWMA	MV Normal	32.20 (0.26)	13.51 (0.07)	8.55 (0.03)	6.27 (0.02)	4.22 (0.01)
	MV Gam ($1I_p, 1I_p$)	31.31 (0.29)	13.70 (0.09)	8.87 (0.05)	6.61 (0.03)	4.39 (0.02)
	$R = 0.05I_p$	30.85 (0.27)	13.73 (0.09)	8.71 (0.05)	6.48 (0.03)	4.34 (0.02)
	$h_2 = 11.20$	31.67 (0.27)	13.64 (0.08)	8.70 (0.04)	6.46 (0.03)	4.30 (0.02)
	MV Gam ($4I_p, 1I_p$)	32.21 (0.27)	13.54 (0.08)	8.71 (0.04)	6.38 (0.03)	4.33 (0.02)
MEWMA	MV Normal	34.92 (0.35)	12.16 (0.08)	7.22 (0.03)	5.17 (0.02)	3.40 (0.01)
	MV Gam ($1I_p, 1I_p$)	29.02 (0.32)	12.15 (0.01)	7.47 (0.05)	5.41 (0.03)	3.60 (0.02)
	$R = 0.1I_p$	30.70 (0.32)	12.12 (0.10)	7.28 (0.05)	5.32 (0.03)	3.57 (0.02)
	$h_2 = 12.73$	31.53 (0.33)	12.07 (0.09)	7.43 (0.05)	5.33 (0.03)	3.52 (0.02)
	MV Gam ($4I_p, 1I_p$)	31.78 (0.34)	12.14 (0.09)	7.28 (0.04)	5.34 (0.03)	3.50 (0.02)
χ^2	MV Normal	138.13	60.95	24.62	10.63	2.93
	MV Gam ($1I_p, 1I_p$)	16.25 (0.23)	10.70 (0.14)	7.02 (0.09)	4.87 (0.06)	2.72 (0.03)
	$R = 1I_p$	20.90 (0.29)	13.11 (0.18)	8.12 (0.11)	5.21 (0.07)	2.77 (0.03)
	$h_1 = 14.86$	25.65 (0.34)	15.19 (0.21)	9.10 (0.12)	5.69 (0.07)	2.75 (0.03)
	MV Gam ($4I_p, 1I_p$)	28.61 (0.40)	16.98 (0.24)	9.67 (0.13)	5.86 (0.07)	2.77 (0.03)

TABLE 28.9

Out-of-Control ARLs for MEWMA Control Charts with $p = 20$ for Various Multivariate Gamma Distributions

		Noncentrality Parameter λ				
		0.5	1.0	1.5	2.0	3.0
$p = 20$						
MEWMA	MV Normal	52.89 (0.45)	21.52 (0.10)	13.58 (0.05)	10.01 (0.03)	6.72 (0.01)
	MV Gam ($1I_p, 1I_p$)	47.54 (0.42)	21.28 (0.13)	13.79 (0.07)	10.25 (0.04)	6.89 (0.03)
	$R = 0.05I_p$	50.07 (0.44)	21.60 (0.12)	13.56 (0.06)	10.19 (0.04)	6.83 (0.02)
	$h_2 = 34.50$	50.29 (0.44)	21.47 (0.12)	13.66 (0.06)	10.15 (0.04)	6.83 (0.02)
	MV Gam ($4I_p, 1I_p$)	50.71 (0.43)	21.19 (0.11)	13.70 (0.06)	10.08 (0.04)	6.77 (0.02)
MEWMA	MV Normal	63.70 (0.71)	20.09 (0.14)	11.33 (0.05)	8.03 (0.03)	5.18 (0.01)
	MV Gam ($1I_p, 1I_p$)	44.02 (0.49)	18.82 (0.15)	11.44 (0.07)	8.22 (0.04)	5.39 (0.02)
	$R = 0.1I_p$	49.00 (0.55)	19.42 (0.15)	11.33 (0.07)	8.09 (0.04)	5.33 (0.02)
	$h_2 = 37.01$	51.24 (0.57)	19.35 (0.15)	11.37 (0.06)	8.08 (0.04)	5.27 (0.02)
	MV Gam ($4I_p, 1I_p$)	53.07 (0.60)	19.57 (0.14)	11.29 (0.06)	8.04 (0.04)	5.28 (0.02)
χ^2	MV Normal	173.60	117.01	66.24	34.28	9.11
	MV Gam ($1I_p, 1I_p$)	12.99 (0.18)	10.42 (0.14)	8.41 (0.11)	6.45 (0.08)	4.04 (0.05)
	$R = 1I_p$	19.48 (0.27)	15.54 (0.22)	11.51 (0.16)	8.17 (0.11)	4.42 (0.05)
	$h_1 = 40.00$	25.07 (0.35)	19.36 (0.27)	13.52 (0.19)	9.46 (0.12)	4.69 (0.06)
	MV Gam ($4I_p, 1I_p$)	29.82 (0.42)	22.45 (0.31)	16.09 (0.22)	10.75 (0.15)	5.04 (0.06)

SUMMARY

Mining process data with multivariate quality control techniques has received tremendous interest recently. Many techniques with different features are available for multivariable process monitoring. In this chapter two widely used multivariate quality control charts, Hotelling T^2 and MEWMA charts, were presented with illustrative examples.

Deviations from the multivariate normality impose a restriction on the performances of many multivariate control charts. Hence, performances of MEWMA control charts are

investigated under nonnormal distributions. It is shown that a MEWMA control chart is almost a nonparametric technique.

REFERENCES

- Borror, C. M., Montgomery, D. C., & Runger, G. C. (1999). Robustness of the EWMA control charts to non-normality. *Journal of Quality Technology*, 31, 309–316.
- Dagpunar, J. (1988). *Principles of random variate generation*. New York: Oxford University Press.
- Hotelling, H. (1947). Multivariate quality control Illustrated by the air testing of sample bombsights. In C. Eisenhart, M. W. Hastay, & W. A. Wallis (Eds.), *Techniques of statistical analysis* (pp. 111–184). New York: McGraw-Hill.
- Jackson, J. (1985). Multivariate quality control. *Communications in Statistics—Theory and Methods*, 34, 2657–2688.
- Lowry, C. A., & Montgomery, D. C. (1995). A review of multivariate control charts. *IIE Transactions*, 27, 800–810.
- Lowry, C. A., Woodall, W. H., Champ, C. W., & Rigdon, E. (1992). A multivariate exponentially weighted moving average control chart. *Technometrics*, 34, 46–53.
- Montgomery, D. C. (2001). *Introduction to statistical quality control* (4th ed.). New York: Wiley.
- Pignatiello, J. J., & Runger, G. C. (1990). Comparisons of multivariate CUSUM charts. *Journal of Quality Technology*, 22, 173–186.
- Prabhu, S. S., & Runger, G. C. (1997). Designing a multivariate EWMA control chart. *Journal of Quality Technology*, 29, 8–15.
- Roberts, S. W. (1959). Control chart tests based on geometric moving averages. *Technometrics*, 1, 239–250.
- Ronning, G. (1977). A simple scheme for generating multivariate gamma distributions with non-negative covariance matrix. *Technometrics*, 19, 179–183.
- Runger, G. C., & Prabhu, S. S. (1996). A Markov chain model for the multivariate exponentially weighted moving averages control chart. *Journal of the American Statistical Association*, 91, 1701–1706.
- Woodall, W. H., & Montgomery, D. C. (1999). Research issues and ideas in statistical process control. *Journal of Quality Technology*, 31, 376–386.

Author Index

Numbers in *italics* indicate pages with complete bibliographic information.

- A**
- Abarbanel, H. D. I., 319, 339
Achlioptas, D., 296, 302
Adam, N. R., 443, 451
Adams, B. M., 89, 90, 102
Adams, N. M., 130
Adhikary, J., 520, 546
Adler, G. L., 472, 477
Adler, P. S., 464, 477
Aggarwal, C. C., 445, 451
Agha, G., 344, 358
Agrawal, D., 444, 445, 451
Agrawal, P., 291, 302
Agrawal, R., 8, 10, 15, 16, 23, 24,
 27, 28, 29, 35, 36, 37, 38, 39,
 265, 275, 285, 294, 302, 353,
 358, 451, 501, 518, 526, 527,
 530, 545, 597, 609, 616
Agresti, A., 4, 23
Ahalt, S. C., 266, 275
Aitkin, N., 242, 244
Akaike, H., 231, 244, 471, 477
Albert, J. H., 134, 136, 138, 155
Albert, P., 520, 545
Alencar, A., 526, 546
Alsabti, K., 10, 23, 344, 358
Altschul, S., 588, 596
Ambroise, C., 543, 545
Amir, A., 482, 517
Anastassiou, D., 43, 66
Anderson, D., 628, 635
Anderson, E. B., 242, 244
Anderson, J. R., 474, 476, 477
Andrich, D., 242, 244
Andrieu, C., 134, 155
- Anfinsen, C., 576, 596
Ankori, K., 508, 517
Ansari, A., 242, 244
Anselin, L., 534, 545
Apley, D. W., 203, 204, 205, 206,
 209, 211, 212, 213
Appelt, D. E., 506, 507, 517, 518
Applet, D., 482, 518
Arfken, G. B., 338
Argote, L., 464, 465, 467, 468, 472,
 474, 477, 478
Argyris, C., 464, 477
Aronis, J., 348, 358
Aseltine, J., 497, 518
Asher, H., 472, 477
Asker, L., 552, 561, 569
Atallah, M., 450, 451
Attias, H., 242, 244
Aubele, J., 561, 569
Auerbach, D., 314, 338
Augusteijn, M. F., 533, 547
Aumann, A., 36, 37, 38
Aumann, Y., 482, 508, 509, 517
Axsater, S., 472, 477
Ayyagari, R., 350, 358, 360
Azimi-Sadjadi, M. R., 56, 66
- B**
- Badiru, A. B., 467, 472, 478
Baeza-Yates, R., 258, 275
Bagozzi, R. P., 241, 244
Bailey, C. D., 474, 478
Bailey, S., 356, 357, 359, 458, 459
Bailey, S. M., 359
- Baker, D., 582, 583, 588, 596
Baker, F., 242, 244
Balasubramanian, S. K., 242, 244
Ballman, A., 603, 616
Ballou, D. P., 474, 478
Bandeen-Roche, K., 230, 244
Banerjee, A., 261, 264, 266, 275
Baraniuk, R. G., 134, 155, 157
Barbieri, M. M., 113, 129
Barnette, V., 521, 545
Barron, A. R., 182, 183, 190
Barry, R., 535, 546
Bartholomew, D. J., 242, 244
Barton, R. R., 211, 213
Basford, K. E., 231, 241, 245
Bastide, Y., 33, 37, 38
Batistakis, Y., 540, 545
Battiti, R., 46, 47, 65
Bauer, E., 346, 358
Baum, L. E., 134, 138, 155
Bayardo, R. J., 33, 35, 37, 38
Bayarri, M. J., 127, 129
Bayes, T., 104, 129
Beale, R., 476, 478
Bear, J., 482, 517, 518
Becker, M., 629, 635
Becker, R., 553, 554, 572
Beckman, R. H., 569
Beckman, S. L., 468, 472, 474, 477
Beckmann, N., 291, 302
Bedrosian, E., 331, 338
Belew, R., 344, 360
Belkaoui, A., 467, 478
Ben Yehuda, Y., 508, 509, 517
Benettin, G., 319, 338
Bennett, W. R., 469, 479

- Bentler, P. M., 242, 246
 Bentley, J. L., 281, 291, 302
 Berchtold, S., 640, 655
 Berger, J., 127, 129, 163, 190
 Berger, J. O., 113, 129
 Bergstrom, P., 609, 616
 Bernardo, J. M., 104, 113, 129
 Berndt, D. J., 283, 302
 Berson, A., 403, 407
 Bertino, E., 450, 451
 Besag, J., 139, 155, 535, 545
 Besancon, R., 482, 518
 Best, N. G., 106, 121, 131, 154, 157
 Bhanu, B., 567, 569
 Bibby, J., 568, 571
 Bijmolt, H. A., 242, 246
 Bingham, E., 296, 302
 Bishop, C., 4, 23
 Bishop, C. E., 242, 246
 Biswas, G., 267, 276
 Bjork, R. A., 467, 478
 Blake, A., 567, 569
 Blanton, R., 557, 565, 572
 Blei, D., 129
 Bloedorn, E., 422
 Bloomfield, P., 190
 Blott, S., 291, 304
 Blum, A., 414, 422
 Boashash, B., 329, 331, 338
 Bock, D., 243, 245
 Bock, R. D., 242, 244
 Böckenholz, I., 241, 244
 Böckenholz, U., 241, 242, 244, 245, 246
 Boley, D., 272, 275
 Bollobas, B., 286, 302
 Bonnet, P., 357, 358
 Bonney, M., 472, 474, 478
 Borowiak, D. S., 471, 478
 Borror, C., 95, 102, 628, 636
 Borror, C. M., 663, 665, 668
 Borth, M., 357, 361
 Botstein, D., 259, 275
 Boykov, Y., 535, 545
 Boyle, R., 567, 571
 Bradley, P. S., 262, 263, 275
 Bradlow, E., 242, 244
 Brand, M., 261, 275
 Brecht, B., 476, 478
 Breiman, L., 4, 7, 8, 10, 15, 18, 19, 20, 23, 126, 128, 129, 190, 346, 358, 609, 616
 Brence, J., 557, 569
 Brier, G. W., 164, 190
 Brill, E., 517
 Brin, S., 31, 38
 Brislawns, C., 558, 569
 Brodley, C., 561, 569
 Brodley, C. E., 415, 422
 Brooks, V., 526, 546
 Brown, C., 535, 545
 Brown, D., 557, 569
 Brown, L. G., 564, 569
 Brown, P. O., 259, 275
 Brown, R., 319, 339
 Bryant, P., 319, 339
 Buchanan, B., 348, 358, 361
 Buchner, A., 603, 616
 Buck, J. R., 472, 478
 Buescher, K., 351, 359
 Buja, A., 172, 190
 Burges, C. J. C., 184, 190
 Burl, M., 552, 561, 569, 570
 Byrne, C., 344, 358
 Bystroff, C., 582, 583, 585, 588, 596
- C
- Cadez, I., 127, 129, 238, 241, 244
 Cadez, I. V., 127, 129, 267, 275
 Calinski, T., 641, 655
 Callan, J. P., 485, 518
 Camm, J., 464, 479
 Camm, J. D., 472, 479
 Campbell, C., 128, 130
 Campbell, W., 556, 571
 Candela, G., 558, 570
 Candler, G. V., 558, 571
 Cantú-Paz, E., 554, 570
 Carbone, P. L., 401, 407
 Cardie, C., 497, 518
 Cardoso, J. F., 205, 206, 208, 213
 Carlin, B. P., 113, 114, 126, 127, 129, 130, 131
 Carlin, J. B., 106, 111, 115, 118, 125, 126, 130, 145, 156
 Carlson, J. G., 472, 474, 478
 Carmel, D., 344, 358
 Carmone, f. J., 242, 245
 Carr, G. W., 472, 478
 Casadio, R., 580, 586, 596
 Casari, G., 587, 596
 Casdagli, M., 308, 340
 Casella, G., 139, 155
 Celeux, G., 138, 155, 156
 Chakrabarti, K., 264, 275, 288, 296, 302, 303
 Chaloner, K., 111, 129
 Champ, C., 98, 101, 102
 Champ, C. W., 658, 661, 668
 Chan, K., 295, 302
 Chan, P., 346, 358
 Chan, Y. T., 323, 324, 339
 Chang, H., 628, 635
 Chang, K., 251, 259, 275
 Chao, V., 358
 Chapman, P., 458, 459
 Chatterjee Surojit, Z. A., 638, 655
 Chattratichat, J., 354, 358
 Chaudhuri, S., 13, 23
 Chawla, S., 520, 537, 546
 Chedid, R., 558, 570
 Cheeseman, P., 4, 127, 23, 129
 Chellappa, R., 558, 570
 Chen, F.-L., 56, 65
 Chen, G., 306, 339
 Chen, H., 56, 66
 Chen, J. J. G., 472, 479
 Chen, M.-H., 126, 130
 Chen, M.-S., 33, 38
 Chen, P., 266, 275, 347, 355, 361
 Chen, Q., 624, 625, 626, 628, 629, 631, 636
 Chen, R., 352, 358
 Chen, S., 31, 38, 611, 616
 Cheng, S. W. J., 472, 478
 Chervinsky, H., 407
 Chervonenkis, A., 450, 452
 Cheung, D., 358
 Chib, S., 114, 129, 134, 136, 138, 139, 155, 156
 Chickering, D. M., 107, 130
 Chien, S., 520, 547
 Chipman, H. A., 126, 129
 Chirstensen, R., 4, 23
 Cho, V., 348, 353, 361
 Choi, H., 134, 155, 157
 Chopin, N., 129
 Chou, P., 535, 545
 Chou, Y.-M., 630, 635
 Chu, S., 287, 303
 Chu, W., 301, 303, 304
 Churchill, G. A., 134, 156
 Chute, C. G., 487, 518
 Cihlar, J., 526, 546
 Ciliberto, S., 318, 319, 339
 Clifford, J., 283, 302
 Clifton, C., 351, 353, 359, 361, 447, 449, 450, 451, 452
 Clinton, J., 458, 459
 Cochrane, M., 526, 546
 Codd, E. F., 398, 407
 Cohen, A., 650, 655
 Cohen, E., 35, 37, 38
 Cohen, M. D., 467, 478
 Cohen, W., 485, 488, 517
 Comon, P., 207, 208, 213, 242, 244
 Congdon, P., 124, 129
 Conover, H., 556, 571
 Consonni, G., 111, 129
 Cooley, R., 598, 603, 604, 611, 616
 Coomans, D., 420, 422

- Cooper, M., 620, 625, 635
 Cooper, M. C., 640, 641, 642, 655
 Copper, G. F., 107, 129
 Copper, P., 535, 545
 Corkill, D., 344, 359
 Cormen, T. H., 284, 302
 Corter, J. E., 257, 276
 Cover, T. M., 258, 275, 606, 616
 Cowell, R. G., 156
 Cowie, J., 482, 517
 Craven, M., 272, 275
 Crawford, M. M., 261, 276
 Creel, E., 459
 Crites, R., 344, 361
 Croft, W. B., 518
 Cromp, R., 556, 571
 Crosier, R., 98, 102
 Crowder, S., 93, 102
 Crumpler, L., 552, 561, 569
 Csiszar, I., 241, 244
 Cunningham, R., 628, 635
 Cutting, D. R., 264, 275
 Cvitanovic, P., 314, 338
 Cybenko, G., 42, 65
- D**
- Dacier, M., 628, 635
 Dagan, I., 482, 517
 Dagpunar, J., 663, 668
 Daille, B., 517
 Dang, V., 543, 545
 Dar-El, E. M., 467, 472, 478
 Darlington, J., 354, 358
 Das, G., 281, 286, 302, 303
 Dasgupta, S., 265, 275
 Dash, M., 414, 421, 422
 Datar, M., 35, 37, 38
 Daubechies, I., 650, 655
 Davies, R., 344, 359
 Davis, R. G., 56, 65
 Davison, A. C., 436, 439
 Dawid, A. P., 107, 129, 130, 156
 Dawson, S., 339
 Dayton, C. M., 241, 244
 Dean, C., 571
 Debar, H., 628, 629, 635
 De Boeck, P., 242, 245
 de Finetti, B., 110, 129
 de freitas, N., 244
 De Jong, K., 418, 421, 422
 De Jong, W., 243, 246
 Dempster, A. P., 136, 138, 156,
 241, 254, 244, 275
 Denison, D., 126, 130
 Denison, D. G. T., 130
 Denning, D. E., 443, 451, 628, 635
- Dennis, J. E., 43, 65
 Dennis, J. E. Jr., 469, 478
 Derin, H., 535, 545
 DeSarbo, W., 241, 245
 DeSarbo, W. S., 241, 242, 244,
 245, 246
 de Silva, V., 298, 304
 Devadas, R., 465, 468, 478
 de Vel, O., 420, 422
 Devore, J., 102
 Dey, D. K., 114, 130
 Dhamala, M., 315, 316, 322,
 323, 339
 Dhillon, I., 258, 262, 275, 351, 359
 Dhillon, I. S., 253, 275
 Dickson, E. R., 187, 190
 Diday, E., 639, 655
 Diebolt, J., 138, 156
 Dietterich, T. G., 346, 359
 Ding, M., 311, 339
 DiPasquo, D., 272, 275
 Dobkins, D., 443, 451
 Domany, E., 577, 582, 596
 Donoghue, J. R., 242, 246
 Donoho, D., 566, 570
 Donoho, S., 420, 422
 Doucet, A., 134, 155, 244
 Douzono, H., 56, 65
 Dowe, D. L., 127, 131
 Draper, D., 111, 114, 122, 130
 Dressens, J., 241, 245
 Druckman, D., 467, 478
 Dube, L., 242, 244
 Dubes, R., 411, 423
 Dubes, R. C., 248, 276, 640, 655
 Duda, R. O., 127, 130, 248, 249,
 264, 275, 293, 302
 Dumais, S. T., 485, 488, 517
 DuMouchel, W., 111, 130, 629,
 635, 636
 Dunmur, A. P., 241, 244
 Dunning, T., 517
 Durfee, E., 344, 359
 Dutton, J. M., 464, 478
 Dy, J. G., 415, 422
- E**
- Ebbinghaus, H., 472, 474, 476, 478
 Eckmann, J.-P., 314, 318, 319, 321,
 338
 Eckmann, J. P., 339
 Eckmann, S., 628, 636
 Edwards, P., 344, 358
 Efron, B., 14, 23, 436, 439, 440
 Ehrenberg, A. S. C., 616
 Eisen, M. B., 259, 275
- Ehiabor, T., 629, 636
 Eisenbarth, T., 649, 655
 Elmagarmid, A., 407, 450, 451
 Elmaghraby, S. E., 472, 474, 478
 Elkan, C., 263, 275
 Elliott, H., 535, 545
 Ellis, S., 474, 478
 Elmaghraby, S. E., 477
 Emran, S. M., 624, 625, 626, 628,
 629, 631, 635, 636
 Epple, D., 465, 468, 472, 474, 478
 Epple, E., 477
 Erickson, J., 291, 302
 Escamilla, T., 628, 635
 Eskin, E., 629, 635
- F**
- Fahlmann, S. E., 50, 65
 Faloutsos, C., 258, 265, 275, 288,
 291, 293, 294, 296, 297, 301,
 302, 303, 304
 Fan, D., 346, 347, 355, 361
 Fan, W., 359
 Fariselli, P., 580, 586, 596
 Farmer, J. D., 308, 312, 339
 Farnstrom, F., 263, 275
 Farr, M. J., 474, 478
 Farrara, J., 520, 547
 Faulstich, L. C., 603, 616
 Fayyad, U., 13, 23, 412, 422, 552,
 561, 569, 570
 Fayyad, U. M., 4, 23, 262, 263, 275
 Fearnow, M., 620, 625, 635
 Feauveau, J., 650, 655
 Fekete, G., 556, 571
 Feldman, J., 628, 636
 Feldman, R., 37, 38, 482, 502, 508,
 509, 517
 Feng, F., 517
 Fersht, A., 576, 596
 Fetterer, A., 520, 546
 Fidelis, K., 577, 596
 Fiez, T., 353, 360
 Finin, T., 344, 359
 Fink, G., 628, 635
 Finkelstein-Landau, M., 517
 Finlay, J., 476, 478
 Fisch, E. A., 618, 635
 Fisher, D., 497, 517, 518
 Fisher, G., 242, 245
 Fisher, L. D., 190
 Fisher, N. I., 36, 37, 38
 Fisher, R. A., 172, 187, 190
 Fisk, J. C., 474, 478
 Fitzpatrick, J., 565, 571
 Fitzpatrick, J. M., 571

- Flannery, B. P., 340
 Fleming, T. R., 187, 190
 Flynn, P. J., 248, 250, 262, 272,
 276, 638, 655
 Fodor, I., 554, 565, 566, 570
 Fodor, I. K., 570
 Forgy, E. W., 609, 616
 Forman, G., 351, 359, 362
 Formann, A. K., 241, 242, 245
 Forrest, S., 628, 635, 636
 Forsyth, R., 48, 65
 Frakes, W., 273, 275
 Fraley, C., 127, 130
 Frank, E., 423
 Frantzi, T. K., 506, 517
 Frawley, W. J., 517, 518
 Frederick, K., 620, 625, 635
 Fredkin, D. R., 134, 156
 Freedman, L. S., 111, 131
 Freitag, D., 272, 275
 Freitas, A. A., 31, 38, 344, 359
 Frenking, M. E., 326, 339
 Fried, D., 628, 635
 Friedman, J., 126, 129, 130, 177,
 182, 185, 186, 190, 191, 235,
 242, 251, 245, 609, 616
 Friedman, J. H., 4, 7, 8, 10, 15, 18,
 19, 20, 36, 37, 23, 38, 275,
 568, 570, 616
 Frischholz, R., 558, 570
 Frivold, T., 635
 Fu, Aa., 358
 Fu, M., 66
 Fu, W., 295, 302
 Fu, Y., 358
 Fujiwara, S., 35, 37, 38
 Fukuda, T., 7, 23
 Funahashi, K., 42, 66
- G**
- Gaffney, S., 242, 245, 267, 275
 Galen, R., 431, 440
 Galgani, L., 319, 338
 Gambino, S., 431, 440
 Gan, F. F., 89, 102
 Ganti, V., 8, 10, 13, 23, 265, 276
 Gao, L., 301, 303
 Garavaglia, S. B., 56, 66
 Gaussier, E., 517
 Gavrin, J., 111, 131
 Ge, X., 288, 303
 Gehrke, J., 8, 10, 11, 13, 23, 265,
 275, 276, 357, 358
 Geiger, D., 107, 130
 Geisser, S., 114, 130
 Geist, A., 351, 361
- Gelfand, A. E., 114, 130, 139, 156
 Gelman, A., 106, 115, 118, 125,
 126, 130, 145, 156
 Geman, D., 116, 139, 130, 535, 545
 Geman, S., 116, 139, 155, 130,
 156, 535, 545
 Gentleman, R., 154, 156
 George, E. I., 126, 129, 139, 155
 Ghahramani, Z., 241, 242, 243, 246
 Ghosh, J., 251, 254, 259, 260, 261,
 264, 266, 270, 271, 272, 275,
 276, 277, 349, 352, 361
 Ghosh, K., 628, 629, 635
 Gilks, W. R., 116, 130, 139, 154,
 156, 157
 Gini, M., 272, 275
 Gionis, A., 35, 37, 38
 Giordano, J., 628, 636
 Giorgilli, A., 319, 338
 Glass, S., 194, 213
 Globerson, S., 472, 474, 478
 Glover, J. H., 478
 Gluck, M. A., 257, 276
 Glymour, C., 470, 478
 Goebel, M., 4, 23
 Golombok, R., 576, 596
 Goldberg, D. E., 4, 23
 Goldin, D. Q., 282, 303
 Goldman, C., 344, 360
 Golub, G. H., 209, 213
 Gong, F., 628, 635
 Gonzalez-Bareto, D. R., 211, 213
 Goodman, L. A., 241, 245
 Gopal, S., 556, 570
 Gordon, L., 178, 190
 Gordon, N., 243, 244
 Gorodetski, V., 348, 359
 Gorsuch, R. L., 242, 245
 Gouda, K., 35, 38
 Govaert, G., 543, 545
 Graefe, G., 13, 23
 Graepel, T., 128, 130
 Graf, I., 628, 635
 Grambsch, P. M., 187, 190
 Grassberger, P., 309, 312, 339
 Graven, M., 275
 Graves, S., 556, 571
 Graves, S. J., 556, 571
 Grebogi, C., 306, 311, 312, 314,
 339
 Greenberg, E., 139, 156
 Green, P., 139, 155
 Green, P. E., 242, 245
 Green, P. J., 127, 155, 130, 131, 156
 Greene, W. H., 173, 190
 Greenman, G., 520, 545
 Greer, K., 351, 360
 Gregg, M., 554, 572
- Gribki, L., 520, 546
 Grinstein, G., 412, 422
 Grishin, N. V., 576, 596
 Grishman, R., 507, 517
 Gross, R., 272, 275
 Grossman, R., 356, 357, 359, 458,
 459, 552, 570
 Grossman, R. L., 353, 356, 361
 Gruenwald, L., 4, 23
 Gujarati, D., 470, 478
 Gunaratne, G. H., 314, 338
 Gunopoulos, D., 35, 37, 38, 265,
 275, 281, 286, 301, 302,
 303, 304
 Gunther, O., 525, 545
 Guntzer, U., 525, 546
 Guo, Y., 347, 354, 358, 359
 Gupta, G. K., 260, 276
 Guting, R., 520, 545
 Gutti, S., 459
 Guttman, A., 281, 291, 303
 Guttorp, P., 134, 157
- H**
- H.-S. Wong, L. G., 567, 571
 Hagen, M. T., 43, 46, 47, 66
 Hahn, S. L., 326, 339
 Hahn, U., 517
 Haines, J., 628, 635
 Haining, R., 520, 545
 Halkidi, M., 540, 545
 Hallstrom, P., 357, 359
 Hambleton, R., 242, 246
 Hamilton, J., 280, 303
 Hamilton, J. D., 134, 156
 Hammel, S. M., 339
 Hamzaoglu, I., 351, 359
 Han, C., 113, 130
 Han, E., 272, 275, 344, 359
 Han, E.-H., 253, 254, 255, 276
 Han, J., 33, 38, 251, 276, 410, 422,
 520, 527, 546, 609, 616
 Hanagandi, V., 351, 359
 Hancock, P., 520, 546
 Hancock, W. M., 472, 476, 478
 Hand, D., 4, 23, 422
 Hand, D. J., 127, 130, 171, 190
 Hara, S., 56, 65
 Harabasz, J., 641, 655
 Haritsa, J., 353, 361
 Haritsa, J. R., 445, 452
 Harman, H. H., 242, 245
 Harrington, D. P., 187, 190
 Harrison, J., 126, 131, 136, 157
 Hart, D., 287, 303

- Hart, P. E., 127, 130, 248, 249, 275, 293, 302
 Hartigan, J. A., 248, 276
 Harville, D. L., 469, 479
 Hasan, H., 468, 478
 Hastie, R., 472, 476, 479
 Hastie, T., 126, 130, 172, 177, 182, 185, 190, 191, 213, 235, 242, 245, 568, 570
 Hastings, K., 272, 275
 Hastings, W. K., 116, 139, 264, 130, 156
 Hatonen, K., 56, 66
 Hawkins, C., 343, 360
 Hawkins, D., 89, 90, 98, 102, 521, 545
 Hayden, R., 310, 339
 Hayes, P., 485, 517
 Hayes, P. J., 485, 486, 517, 518
 Haykin, S., 42, 50, 66, 205, 213
 Heckerman, D., 107, 130, 485, 488, 517
 Hedvall, S., 354, 358
 Heijden, P. G. M., 241, 245
 Helfand, D., 553, 554, 569, 572
 Henderdon, D., 407
 Hénon, M., 310, 339
 Henry, N. W., 241, 245
 Herbrich, R., 128, 130
 Hershberger, D., 349, 350, 355, 359, 360
 Herskovits, E., 107, 129
 Hertz, J., 42, 43, 66
 Heuch, I., 108, 109, 130
 Higdon, D., 139, 155
 Hinkley, D. V., 436, 439
 Hinneberg, A., 640, 655
 Hinton, G., 241, 242, 246
 Hinton, G. E., 43, 66
 Hirsh, H., 482, 502, 517
 Hipp, J., 357, 361, 525, 546
 Hjaltason, G. R., 291, 303
 Hlavac, V., 567, 571
 Hoballah, I., 344, 359
 Hobbs, J., 482, 506, 507, 517
 Hobbs, J. R., 506, 517, 518
 Hobohm, U., 578, 596
 Hodges, J. S., 111, 130
 Hoeffding, W., 440
 Hoeting, J., 114, 127, 130, 131
 Hoffman, D. L., 242, 244
 Hoffman, T., 229, 231, 241, 245
 Hofmeyr, S. A., 628, 635
 Hogan, R., 397, 407
 Hoglund, A. J., 56, 66
 Hohn, M., 520, 546
 Holland, J. H., 344, 359
 Holmes, C. C., 130
 Holte, R. C., 556, 570
 Honavar, V., 355, 359
 Honkela, J., 56, 66
 Horling, B., 344, 360
 Hornick, M., 459
 Hornik, K., 42, 66
 Horowitz, S. L., 287, 303
 Hosken, M., 242, 245
 Hotelling, H., 98, 102, 658, 668
 Hsiao, C.-J., 579, 596
 Hsu, C., 301, 304
 Hsu, M., 351, 362
 Hsu, W., 31, 37, 38, 611, 616
 Huang, E., 588, 596
 Huang, F., 526, 546
 Huang, N. E., 306, 307, 323, 324, 328, 331, 333, 339
 Huang, W., 352, 360
 Huang, Y., 530, 546
 Huber, P., 163, 191
 Hull, D., 518
 Humphreys, R., 552, 571
 Hurn, M., 155, 156
 Hurvitz, E., 517
 Hyland, P. N., 468, 478
 Hyvarinen, A., 206, 213, 242, 245, 568, 570
- I**
- Iacobou, N., 609, 616
 Ibrahim, J. G., 126, 130
 Ibrahim, M., 450, 451
 Ihaka, R., 154, 156
 Imielinski, T., 27, 38
 Indyk, P., 35, 37, 38, 264, 276, 291, 296, 303
 Ingris, R., 506, 518
 Interrante, V., 558, 571
 Ip, E., 127, 129, 241, 242, 244, 245, 246
 Irgens, L. M., 108, 109, 130
 Isard, M., 567, 569
 Israel, D., 482, 517, 518
 Issaks, E. H., 520, 546
- J**
- Jaakola, T., 120, 121, 130
 Jaber, M. Y., 472, 474, 478
 Jabri, M. A., 66
 Jackson, J., 98, 102, 658, 668
 Jackson, J. E., 198, 199, 202, 213, 567, 570
 Jacobs, R. A., 241, 245
 Jagadish, H. V., 282, 291, 303, 304
- K**
- Jagpai, H. S., 242, 245
 Jain, A., 264, 276, 411, 423
 Jain, A. K., 248, 250, 262, 272, 276, 533, 546, 638, 655
 Jain, R., 291, 304
 James, M., 4, 23, 429, 440
 Janosi, I. M., 315, 339
 Jarvis, J., 552, 570
 Javitz, H. S., 628, 635
 Jedidi, K., 242, 244, 245
 Jenkner, H., 343, 360
 Jennings, N., 344, 354, 361
 Jennings, N. R., 361
 Jennrich, R. I., 242, 245
 Jensen, V. C., 351, 359
 Jhung, Y., 533, 546
 Jiawei, H., 639, 640, 642, 644, 646, 655
 Jin, S., 585, 596
 Joachims, T., 488, 518
 John, G., 416, 422
 Johnson, E., 349, 350, 352, 355, 359, 360
 Johnson, R. A., 199, 202, 204, 213
 Johnson, W. B., 303
 Jolliffe, I. T., 293, 303, 606, 616
 Jones, A. K., 443, 451
 Jones, M., 476, 478
 Jordan, M., 129, 241, 245
 Jordan, M. I., 120, 121, 130, 241, 245
 Joshi, A., 344, 359
 Joshi, M., 344, 359
 Jou, Y., 628, 635
 Ju, W. H., 629, 635, 636
 Juang, B. H., 134, 156
 Judson, R., 577, 596
 Justel, A., 127, 130
 Justeson, J. S., 518
- K**
- Kalman, R., 136, 156
 Kamakura, W. A., 242, 244
 Kamath, C., 552, 554, 565, 566, 570, 571, 572
 Kamath, K., 344, 359
 Kamber, M., 31, 38, 251, 276, 410, 422
 Kameyama, M., 517, 518
 Kamphorst, S. O., 318, 319, 339
 Kanapady, R., 558, 571
 Kanellakis, P. C., 282, 303
 Kantarcioglu, M., 353, 359, 447, 451
 Kantor, P. B., 464, 478
 Kantowitz, B. H., 634, 635

- Kantz, H., 306, 339
 Karger, D. R., 264, 275
 Kargupta, H., 349, 350, 351, 352,
 355, 357, 358, 359, 360
 Karhunen, J., 568, 570
 Karplus, W. J., 558, 572
 Karr, A. F., 629, 636
 Karsaev, O., 348, 359
 Karwan, K. R., 472, 478
 Karypis, G., 253, 254, 255, 256,
 257, 272, 275, 276, 277,
 344, 359
 Kasif, S., 356, 359
 Kaski, S., 56, 66
 Katz, S. M., 518
 Kaufman, L., 609, 616, 639, 640,
 641, 655
 Kegelmeyer, P., 552, 570
 Kehagias, A., 155, 156
 Keim, D. A., 640, 655
 Keiser, K., 556, 571
 Kelley, S., 291, 304
 Kemmerer, R., 628, 636
 Kendall, K., 628, 635
 Kent, J., 568, 571
 Keogh, E., 287, 288, 296, 301, 303
 Keogh, E. J., 288, 287, 296, 303
 Kerber, R., 458, 459
 Khabaze, T., 458, 459
 Kim, B., 233, 245
 Kim, S., 301, 303
 Kim, S.-H., 585, 596
 Kim, Y., 415, 422
 Kimball, R., 604, 616
 Klassner, F., 344, 360
 Klemettinen, M., 31, 38
 Klösgen, W., 482, 517, 518
 Knecht, G. R., 472, 479
 Knoblock, C. A., 505, 518
 Knoshevis, B., 474, 479
 Knott, M., 242, 244
 Ko, C., 628, 635
 Kohavi, R., 4, 23, 346, 358,
 416, 422
 Kohonen, T., 55, 56, 58, 66, 251,
 259, 276
 Koler, M., 354, 358
 Kollias, S., 43, 66
 Kollios, G., 301, 304
 Koonin, E. V., 576, 596
 Kooperberg, C., 588, 596
 Koperski, K., 520, 527, 546
 Koschke, R., 649, 655
 Kostelich, E. J., 315, 316, 322, 339
 Kowalewski, M. J., 476, 479
 Kriegel, H.-P., 291, 302, 640, 655
 Krishnamoorthy, S., 352, 358, 360
 Krishnamrthy, S., 352, 360
 Krishnamurthy, A. K., 266, 275
 Krishnan, T., 241, 245
 Krishnaswamy, S., 356, 360
 Krogh, A., 42, 43, 66
 Kronmal, R. A., 126, 131
 Krugman, P., 520, 546
 Kruizinga, P., 558, 570
 Kruskal, J., 297, 303
 Kruskal, J. B., 285, 303
 Kubat, M., 556, 570
 Kulikowski, C.A., 4, 24, 435, 440
 Kulluri, V., 348, 358
 Kumar, S., 254, 261, 272, 276,
 628, 635
 Kumar, V., 253, 254, 255, 256, 275,
 276, 344, 359, 558, 570, 571
 Künsch, H., 155, 156
 Kuo, K. S., 556, 571
 Kurths, J., 339
 Kushraj, D., 350, 357, 360
 Russell, E., 577, 582, 596
 Kutner, M. H., 479
 Kwek, S., 353, 360
- L
- Labrou, Y., 344, 359
 Lagus, K., 56, 66
 Lahav, O., 552, 571
 Lai, Y.-C., 310, 311, 314, 315, 316,
 322, 339
 Laird, N. M., 136, 138, 156, 241,
 244, 254, 275
 Lakhal, L., 33, 37, 38
 Lalwani, N., 241, 245
 Lam, W., 353, 360
 Lance, C. E., 469, 479
 Landau, D., 508, 509, 517
 Lander, S., 344, 360
 Lang, K., 272, 276
 Lange, J. M., 517
 Langeheine, R., 241, 242, 243, 246
 Langford, J. C., 298, 304
 Langley, P., 414, 422
 Langworthy, A., 187, 190
 Lappin, S., 506, 518
 Larkey, L. S., 518
 Larsen, B. N., 107, 130
 Lasker, B., 343, 360
 Lathrop, D. P., 315, 339
 Lattanzi, M., 343, 360
 Laughlin, J. E., 242, 246
 Lauritzen, S. L., 106, 107, 130,
 131, 136, 138, 156
 Lavington, S. H., 344, 359
 Lawrence, C. E., 134, 156
 Lazarsfeld, P. F., 241, 245
 Le Moigne, J., 565, 570
 Lebiere, C., 50, 65
 Lee, B., 526, 546
 Lee, D., 242, 246
 Lee, H. Y., 203, 206, 209, 211,
 212, 213
 Lee, S., 567, 569
 Lee, W., 347, 360, 628, 629, 635
 Lefebvre, P., 526, 546
 Lehnert, W., 482, 497, 517, 518
 Leighton, T., 256, 276
 Leimer, H-G., 107, 130
 Leiserson, C. E., 284, 302
 Lele, S., 520, 547
 LeMoigne, J., 556, 571
 Lenk, P., 241, 245
 Lent, B., 518
 Lerner, D., 310, 339
 Leroux, B. G., 134, 156
 LeSage, J., 535, 546
 LeSage, J. P., 546
 Lesser, V., 344, 359, 360
 Levenshtein, V., 266, 276
 Levin, N., 472, 474, 478
 Levinthal, C., 576, 596
 Levitt, K., 628, 635
 Levy, F. K., 472, 479
 Lewin, D. I., 557, 570
 Lewis, D. D., 127, 128, 130,
 485, 518
 Lewis, J., 263, 275
 Lewis, T., 521, 545
 Li, C., 267, 276, 468, 479
 Li, H., 242, 245
 Li, M., 361
 Li, S., 535, 546
 Li, W., 37, 39, 344, 360, 362
 Li, X., 624, 625, 626, 628, 629,
 631, 635, 636
 Li, Z., 526, 546
 Liberzon, Y., 482, 508, 517
 Lie, R. T., 108, 109, 130, 131
 Liebold, A., 520, 546
 Ligon, W., 556, 571
 Lim, T.-S., 4, 15, 23
 Lima, E., 526, 546
 Lin, D., 482, 518
 Lin, K., 258, 265, 275
 Lin, K.-I., 285, 291, 297, 302, 303
 Lin, S., 56, 59, 66
 Lindell, Y., 36, 37, 38, 447, 451
 Lindenstrauss, J., 303
 Linvy, M., 640, 655
 Lipman, D., 588, 596
 Lippmann, R., 628, 635
 Lipshtat, O., 508, 509, 517
 Lipton, R. J., 443, 451
 Liu, B., 31, 37, 38, 42, 66, 611, 616

- Liu, C., 136, 137, 156
Liu, D.-R., 525, 546
Liu, H., 410, 412, 413, 414, 417,
 419, 420, 421, 422
Liu, H. H., 306, 307, 323, 324,
 331, 339
Liu, J. S., 134, 139, 156
Liu, L., 350, 357, 360
Liu, S.-F., 56, 65
Liu, W., 556, 570
Liu, X., 487, 518, 520, 546
Livny, M., 263, 277, 609, 616
Loan, C. F. V., 209, 213
Loh, W.-Y., 4, 10, 13, 15, 18, 23
Loke, S., 356, 360
Long, S. R., 306, 307, 322, 324,
 331, 339
Longford, N. T., 242, 245
Lord, F. M., 242, 245
Lorenz, E. N., 339
Louis, T. A., 111, 113, 114, 126,
 127, 129
Low, B., 346, 361
Lowry, C., 98, 101, 102
Lowry, C. A., 658, 661, 668
Lu, C., 525, 546
Lu, C.-T., 520, 546
Lu, H., 33, 38, 410, 422
Luc, A., 521, 522, 546
Lucas, J., 89, 93, 98, 102
Luenberger, D. G., 219, 245
Lynch, K., 241, 246
Lynden, S., 361
- M
- Ma, Y., 31, 37, 38
MacDonald, I. L., 136, 141, 153,
 156
Maclin, R., 346, 360
Macready, G. B., 241, 244
Madden, T., 588, 596
Madigan, D., 108, 109, 111, 114,
 120, 122, 126, 127, 130, 131,
 186, 187, 191, 470, 478
Maes, P., 609, 616
Magidson, J., 246
Maintz, J., 565, 571
Makaeizadeh, G., 546
Makov, U. E., 136, 157, 241, 246
Malhi, B., 356, 357, 359
Malik, J., 566, 571
Malladi, R., 566, 571
Mallet, Y., 420, 422
Mallick, B., 126, 130
Mallows, C., 111, 130
Mandava, V., 565, 571
- Mannila, H., 31, 38, 127, 129, 238,
 241, 244, 267, 276, 286, 296,
 302, 422, 609, 616
Manolopoulos, Y., 301, 302
Mao, J., 264, 276
Mao, R., 38
Marchette, D. J., 558, 571
Mardia, K. V., 568, 571
Mark, D., 520, 546
Markovitch, S., 344, 358
Marks, D. G., 449, 451
Marshall, L. L., 242, 246
Martin, G., 355, 360
Martin, P., 506, 507, 518
Marusic, I., 558, 571
Mason, R., 98, 102
Mason, R. L., 630, 635
Masters, G. N., 234, 245
Mastrangelo, C. M., 635
Matheus, C. J., 517
Matsuzawa, H., 7, 23
Mattessich, R., 407
Matwin, S., 556, 570
Maurer, C. R., 565, 571
Mayfield, J., 344, 359
Mazur, J. E., 472, 476, 479
Mazzola, J. B., 472, 478
Mazzucco, M., 459
McCallum, A., 231, 245, 246,
 272, 275
McCarthy, J., 517
McClean, S., 351, 360
McClelland, J., 344, 361
McClung, D., 628, 635
McCord, M., 506, 518
McCullagh, P., 173, 191, 230, 245
McCulloch, R. E., 126, 129
McGaw, G., 618, 636
McLachlan, G., 241, 242, 245
McLachlan, G. J., 231, 241,
 242, 245
McLean, B., 343, 360
McQueen, J., 639, 655
McShane, L., 520, 545
Mechoso, R., 520, 547
Mega, M., 557, 565, 572
Mehrotra, S., 264, 275, 288, 296,
 302, 303
Mehta, M., 8, 10, 15, 16, 23, 24
Meila, M., 241, 245
Melechko, A., 351, 361
Melton, D. E., 266, 275
Menczer, F., 344, 360, 415, 422
Mendelzon, A. O., 282, 282, 294,
 303, 304
Mendoza, E., 526, 546
Meng, X.-L., 136, 137, 156, 157
Mengersen, K., 139, 155
- Menhaj, M. B., 43, 46, 47, 66
Meredith, J., 464, 479
Merrett, T., 525, 546
Merz, C. J., 346, 360
Merz, R., 604, 616
Mesrobian, E., 520, 547
Metropolis, N., 139, 156
Meyer, G., 459
Michalski, R., 417, 422, 423
Michie, D., 4, 23
Miglioretti, D., 230, 244
Miller, L., 355, 359
Miller, W., 588, 596
Milligan, G. W., 640, 641, 642, 655
Milo, T., 282, 303
Minsky, M., 344, 360
Minton, S., 505, 518
Mislevy, R., 242, 243, 245
Mitchell, T., 231, 246, 272, 275,
 606, 609, 616
Mitsch, W., 531, 546
Mladenic, D., 37, 38
Mobasher, B., 254, 272, 275, 276,
 604, 616
Modha, D., 258, 275, 351, 359
Modha, D. S., 262, 275
Mok, K., 347, 360, 628, 629, 635
Mon, D., 356, 359
Monro, S., 66
Montgomery, D., 68, 69, 71, 73, 77,
 89, 93, 95, 98, 102, 658, 663,
 665, 668
Montgomery, D. C., 635
Mooney, R., 270, 276
Moore, J., 272, 275
Moore, R. A., 443, 452
Mor, Y., 344, 360
Morales, J., 127, 129
Moreau, L., 526, 546
Morey, R. C., 472, 478
Morimoto, Y., 7, 23
Moss, A., 558, 571
Motoda, H., 410, 412, 413, 419,
 421, 422
Motwani, R., 31, 35, 37, 38, 291,
 296, 303
Mou, J. I., 636
Moukas, A., 344, 360
Moult, J., 577, 596
Moutinho, P., 526, 546
Mulvenna, M. D., 603, 616
Muntz, R., 520, 547, 640, 655
Murthy, S. K., 4, 5, 22, 23
Murty, M. N., 248, 250, 262, 272,
 276, 638, 655
Musick, R., 344, 359
Muthén, B., 241, 242, 245, 246
Múthen, L., 246

N

- Nachtsheim, C. J., 479
 Nagai, Y., 311, 314, 339
 Nagaralu, S., 356, 358, 361
 Nagin, D., 241, 246
 Nair, U. S., 556, 571
 Najjar, N., 558, 570
 Nakaeizadeh, G., 520, 525, 546
 Nakamura, H., 547
 Namburu, R., 552, 570
 Nanda, R., 472, 477
 Narendra, K. S., 42, 66
 Narr, K., 557, 565, 572
 Navathe, S., 33, 39
 Neal, R., 241, 246
 Neal, R. M., 126, 131
 Neira, J., 576, 596
 Nelder, J., 173, 191
 Nelder, J. A., 230, 245
 Nemrbhard, D. A., 468, 469, 471, 472, 474, 479
 Nepstad, D., 526, 546
 Netanyahu, N., 556, 571
 Neter, J., 479
 Neuwald, A. F., 134, 156
 Newell, A., 344, 360
 Ng, A., 129
 Ng, K., 420, 422, 520, 547
 Ng, R., 609, 616
 Ng, V., 358
 Nigam, K., 231, 245, 246, 272, 275
 Nii, P., 344, 360
 Nishil, S., 33, 38
 Nobre, C., 526, 546
 Noguchi, Y., 56, 65
 Nolting, B., 576, 596
 Northcutt, S., 620, 625, 635
 Novick, M. R., 242, 245

O

- Obradovic, D., 353, 360
 Obradovic, Z., 353, 360
 Odewahn, S., 552, 571
 Ogihara, M., 37, 39, 344, 351, 360, 362
 Oja, E., 57, 66, 206, 213, 242, 245, 568, 570
 O'Kane, J., 186, 191
 Okunev, Y., 325, 339
 Olema, O., 580, 586, 596
 Oliver, N., 261, 275
 Olshen, R., 609, 616
 Olshen, R. A., 4, 7, 8, 10, 15, 18, 19, 20, 23, 178, 190
 Omiecinski, E., 33, 39

P

- Opitz, D., 346, 360
 Ord, J. K., 213
 Orenstein, A., 525, 546
 Ortega, J., 410, 422
 Osothsilp, N., 471, 474, 479
 Ostrouchov, G., 351, 361
 Ott, E., 306, 310, 311, 312, 314, 339
 Ozesmi, S., 531, 532, 546
 Ozesmi, U., 531, 532, 546
 Ozsogoglu, Z. M., 285, 301, 304
- Paatero, V., 56, 66
 Pace, R., 535, 546
 Padmanabhan, B., 611, 616
 Palley, M. A., 452
 Palmer, R. G., 42, 43, 66
 Papadimitriou, C., 297, 303
 Papka, R., 485, 518
 Parisi, A. G., 469, 479
 Park, B., 349, 350, 352, 355, 357, 360
 Park, J., 42, 66
 Park, J.S., 33, 38
 Park, K., 233, 245
 Park, S., 301, 303, 304
 Parmar, M. K. B., 111, 131
 Parthasarathy, K., 42, 66
 Parthasarathy, S., 39, 344, 351, 354, 360, 362
 Pasquier, N., 33, 37, 38
 Pavlidis, T., 287, 296, 303, 304
 Pazzani, M., 287, 288, 296, 303, 422
 Pazzani, M. J., 287, 288, 296, 303, 346, 360
 Pearl, J., 106, 131
 Pearlmuter, B., 628, 636
 Pedersen, J., 577, 596
 Pedersen, J. O., 264, 275
 Peel, D., 241, 242, 245
 Pegels, C. C., 472, 479
 Pei, J., 33, 38
 Pena, D., 127, 130
 Pennington, R., 552, 571
 Pentland, A., 261, 275
 Perng, C.-S., 289, 304
 Perona, P., 552, 561, 566, 569, 570, 571
 Perry, S., 567, 571
 Petrie, T., 134, 138, 155
 Pfeleger, C. P., 618, 636
 Pfeleger, K., 416, 422
 Piatetsky-Shapiro, G., 4, 23, 31, 32, 39, 517, 518, 561, 570

Q

- Pickens, D., 565, 571
 Pignatiello, J., 98, 102
 Pignatiello, J. J., 658, 668
 Pinkas, B., 447, 451
 Pittie, S., 350, 357, 360
 Platt, J., 485, 488, 517
 Pokrajac, D., 353, 360
 Pole, A., 126, 131
 Popyack, L., 348, 359
 Potter, C., 526, 546
 Powers, E. J., 338
 Prabhu, S., 102
 Prabhu, S. S., 658, 662, 663, 668
 Pratsini, E., 472, 479
 Pregibon, D., 111, 130, 470, 478
 Press, W. H., 340
 Procaccia, I., 309, 312, 314, 338, 339
 Prodromidis, A., 347, 353, 355, 361
 Provost, F., 348, 358
 Provost, F. J., 361
 Pulleyn, I., 357, 359
 Pun, J., 361
 Puterman, M. L., 134, 156
 Pyle, D., 608, 616
- Qin, X., 357, 359
 Qu, Y., 304
 Quandt, R. E., 242, 246
 Quinlan, J. R., 4, 7, 15, 23, 48, 66, 357, 361, 609, 616

R

- Rabiner, L. R., 134, 156
 Rada, R., 48, 65
 Rafiei, D., 282, 294, 304
 Raftery, A. E., 111, 114, 126, 127, 130, 131, 187, 191, 471, 479
 Raghavan, P., 265, 275, 297, 303
 Raja, A., 344, 360
 Rajman, M., 482, 518
 Ramachandran, R., 556, 571
 Ramachandran, R. P., 571
 Ramakrishnan, R., 8, 10, 11, 13, 23, 263, 265, 276, 277, 609, 616, 640, 655
 Ramos, M. A., 472, 479
 Ramsey, J. B., 242, 246
 Ramsey, M., 56, 66
 Ramu, A., 356, 357, 359
 Rana, O., 361
 Ranganathan, M., 301, 302
 Ranka, S., 10, 23, 344, 358

- Rao, S., 256, 276
Rasmussen, J., 476, 479
Rastogi, R., 15, 16, 17, 24, 262, 276
Rathouz, P., 230, 244
Raturi, A. S., 472, 479
Rauch, H. E., 243, 246
Ravada, S., 520, 546
Raymond, T. N., 639, 640, 642,
 644, 646, 655
Redner, R. A., 241, 246
Reed, C. W., 209, 213
Regev, Y., 517
Reina, C., 262, 263, 275
Reinhertz, T., 458, 459
Rendell, L., 420, 422
Resnik, P., 609, 616
Ribeiro-Neto, B., 258, 275
Rice, J. A., 134, 156
Richardson, S., 116, 130, 131,
 139, 156
Richardson, T., 186, 191
Ridgeway, G., 120, 131, 172,
 186, 191
Riedl, J., 609, 616
Rigdon, E., 658, 661, 668
Rigdon, S., 98, 101, 102
Riloff, E., 482, 497, 518
Ringuette, M., 485, 518
Ripley, B., 4, 24
Rissanen, J., 10, 15, 16, 23, 24,
 471, 479
Rivest, R. L., 15, 23, 284, 302
Rizvi, S., 353, 361
Rizvi, S. J., 445, 452
Robbins, H., 66
Robert, C. P., 138, 155, 157
Roberts, J. S., 242, 246
Roberts, S., 93, 102
Roberts, S. W., 660, 668
Robinson, J. T., 291, 304
Rodrick, J.-F., 520, 546
Roeder, K., 241, 246
Romberg, J. K., 134, 155, 157
Ronakainen, P., 267, 276
Ronkainen, P., 31, 38
Ronning, G., 663, 668
Rosenblatt, M., 213
Rosenbluth, A. W., 156
Rosenbluth, M. N., 139, 156
Rosenfeld, B., 482, 508, 517
Rosenchein, J., 344, 360, 361
Rössler, O. E., 315, 338, 340
Rost J., 241, 246
Rousseau, P., 609, 616
Rousseau, P. J., 639, 640,
 641, 655
Roussopoulos, N., 291, 304
Rowe, R. G., 472, 474, 478
Roweis, S., 241, 243, 246, 298, 304
Rubin, D., 241, 242, 246
Rubin, D. B., 106, 115, 118, 125,
 126, 130, 136, 137, 138, 145,
 156, 244, 254, 275
Rubinovitz, J., 472, 478
Ruelle, D., 318, 319, 339
Rumelhart, D. E., 43, 66, 344, 361
Runger, G., 69, 95, 98, 102
Runger, G. C., 658, 662, 663,
 665, 668
Rushing, J. F., 556, 571
Rusinkiewicz, M., 407
Rydén, T., 138, 155, 157
- S
- Saarela, A., 56, 66
Saccucci, M., 93, 98, 102
Sahami, M., 485, 488, 517
Sahar, S., 31, 39
Salojarvi, J., 56, 66
Salton, G., 272, 276
Samatova, N., 351, 361
Samet, H., 291, 303
Sandberg, I. W., 42, 66
Sander, C., 578, 587, 596
Sandholm, T., 344, 361
Sandhu, S. S., 558, 571
Sankoff, D., 285, 303
Sano, M., 319, 340
Sanseverino, E., 360
Santos, J., 520, 547
Sapiro, G., 566, 571
Sarawagi, S., 356, 361
Sargor, C., 628, 635
Sarkar, K., 350, 357, 358, 360
Sauer, T. D., 308, 339, 340
Saul, L., 242, 246, 298, 304
Savasere, A., 33, 39
Sawada, Y., 319, 340
Sawhney, H. S., 285, 302
Sayal, M., 352, 361
Saygin, Y., 450, 452
Schaffer, A., 588, 596
Schapire, R. E., 185, 191, 485, 518
Schek, H.-J., 291, 304
Scheraga, H., 576, 596
Scheuermann, P., 352, 361
Schler, J., 482, 508, 509, 517
Schler, Y., 517
Schlesinger, P., 526, 546
Schnable, R. B., 43, 65, 469, 478
Schnattinger, K., 517
Schneider, R., 291, 302
Schneider, T., 231, 246
Schonlau, M., 629, 636
Schrater, P. R., 537, 546
Schreiber, G., 576, 596
Schreiber, T., 306, 339
Schuster, A., 361
Schwartz, A. B., 56, 59, 62, 64, 66
Schwarz, G., 148, 157, 231, 246
Schwartzbard, A., 628, 629, 635
Scotney, B., 351, 360
Scott, S., 242, 246
Scott, S. L., 134, 138, 139, 140,
 146, 155, 157, 629, 636
Sebastiani, F., 485, 518
Seeger, B., 291, 302, 304
Segre, A. M., 353, 360
Sekaran, M., 344, 361
Sellis, T. K., 304
Sen, S., 344, 361
Senge, P. M., 464, 479
Seshadri, P., 357, 358
Sethian, J., 566, 567, 571
Setiono, R., 412, 417, 419, 422
Shafer, J., 8, 24
Shafer, J. C., 358
Shafer, S. M., 468, 479
Shardanand, U., 609, 616
Shatz, M., 628, 629, 635
Shatkay, H., 296, 304
Shearer, C., 458, 459
Sheikholeslami, G., 638, 655
Sheinvald, J., 209, 213
Shek, E., 520, 525, 547
Shekhar, S., 520, 530, 537, 546
Shen, Z., 306, 307, 323, 324,
 331, 339
Shenoi, K., 340
Sheth, A., 403, 407
Shi, J., 203, 204, 205, 209, 211,
 212, 213
Shibata, R., 114, 131
Shih, H. H., 306, 307, 323, 324,
 331, 339
Shih, Y.-S., 4, 15, 18, 23, 24
Shim, K., 15, 16, 17, 24, 262, 276,
 285, 302
Shinghal, R., 31, 38
Short, N., 555, 556, 571
Shtub, A., 472, 474, 478
Shumway, R. H., 243, 246
Si, J., 42, 56, 59, 65, 66
Siboni, D., 629, 635
Siebes, A., 295, 304
Silberschatz, A., 31, 39
Silvestre, L., 360
Simon, H., 344, 360
Simonoff, J. S., 443, 452
Simons, K., 588, 596
Simoudis, E., 561, 571
Singer, J. E., 467, 478

- Singer, Y., 185, 191, 485, 488, 517
 Singh, V., 10, 23, 344, 358
 Sinha, D., 126, 130
 Sivakumar, H., 356, 359, 459
 Skormin, V., 348, 359
 Skoudis, E., 619, 636
 Slattery, S., 272, 275
 Smith, A. F. M., 104, 113, 126,
 129, 130, 136, 139, 156, 157,
 241, 246
 Smith, R., 344, 359, 361
 Smith, S. J., 403, 407
 Smunt, T. L., 472, 474, 479
 Smyth, P., 4, 23, 127, 129, 238,
 241, 242, 244, 245, 267, 275,
 276, 288, 303, 422, 470, 478,
 552, 561, 570
 Soderland, S., 497, 517, 518,
 569, 570
 Sodre, L., 552, 571
 Sofge, D. A., 42, 66
 Solberg, A. H., 533, 546
 Soler-Gonzalez, A., 576, 596
 Somayaji, A., 628, 635
 Sonka, M., 567, 571
 Soparker, N., 351, 359
 Sorkin, R. D., 634, 635
 Sorvair, A. S., 56, 66
 Soules, G., 134, 138, 155
 Souloumiac, A., 208, 209, 213
 Sowell, E., 557, 565, 572
 Spagna, A., 343, 360
 Spangler, W., 258, 275
 Spark-Jones, K., 430, 440
 Spearman, C., 271, 276
 Spellman, P. T., 259, 275
 Spiegelhalter, D. J., 4, 23, 106, 111,
 116, 121, 124, 130, 131, 139,
 255, 156, 157
 Spiliopoulos, M., 520, 546,
 603, 616
 Sproul, L. S., 467, 478
 Srikanth, R., 28, 29, 36, 37, 38, 39,
 353, 358, 501, 518, 526, 527,
 530, 545, 598, 609, 616
 Srivastava, J., 603, 604, 616
 Srivastava, R. M., 520, 546
 Stafford, B., 351, 359, 444,
 451, 518
 Stallard, D., 506, 518
 Steiger, W. L., 190
 Steinbach, M., 255, 276
 Stephens, M., 155, 157
 Stern, H. S., 106, 115, 118, 125,
 126, 130, 145, 156
 Stevens, W. R., 620, 636
 Stickel, M., 506, 507, 518
 Stigler, S. M., 104, 131
 Stinchcombe, M., 42, 66
 Stockwell, E., 552, 571
 Stoffer, D. S., 243, 246
 Stolfo, S., 346, 347, 353, 355, 358,
 359, 360, 361
 Stolfo, S. J., 628, 629, 635
 Stolorz, P., 520, 547, 571
 Stone, C., 609, 616
 Stone, C. J., 4, 7, 8, 10, 15, 18, 19,
 20, 23
 Stoppi, J., 482, 517
 Stork, D. G., 248, 249, 264, 275
 Storrie-Lombardi, L., 552, 571
 Storrie-Lombardi, M., 571
 Stott Parker, D., 289, 304
 Stout, W., 242, 246
 Street, W., 415, 422
 Strehl, A., 270, 271, 272, 276,
 352, 361
 Strelecyn, J. M., 319, 338
 Struzik, Z. R., 295, 304
 Stuart, A., 213
 Stuetzle, W., 213, 568, 570
 Stutz, J., 4, 127, 23, 129
 Subbareddy, P. K., 558, 571
 Subramonian, R., 354, 360
 Suchak, M., 609, 616
 Sule, D. R., 474, 479
 Sundheim, B., 518
 Sutiwaraphun, J., 347, 359
 Swain, M. J., 533, 535, 545
 Swain, P. H., 546
 Swami, A., 27, 38, 294, 302
 Swift, J. B., 319, 340
 Swinney, H. L., 319, 340
 Syed, J., 354, 358
 Szalay, A., 343, 361
- T
- Takens, F., 306, 340
 Talavera, L., 414, 422
 Tamaki, H., 297, 303
 Tamma, K. K., 558, 571
 Tan, P.-N., 603, 616
 Tang, N., 554, 570
 Tang, S., 33, 38
 Tanner, M. A., 157
 Taouil, R., 33, 37, 38
 Taxt, T., 533, 546
 Taylor, C. C., 4, 23
 Teachout, M. S., 469, 479
 Tejada, S., 505, 518
 Tél, T., 315, 339
 Teller, A. H., 139, 156
 Teller, E., 139, 156
 Tenenbaum, J. B., 298, 304
- U
- Teukolsky, S. A., 340
 Thayer, D., 242, 246
 Theiler, J., 308, 340
 Theus, M., 629, 636
 Thiele, T. N., 138, 157
 Thissen, D., 243, 246
 Thomas, A., 106, 121, 154, 131,
 157, 464, 478
 Thomas, D., 587, 596
 Thomas, J. A., 258, 275, 606, 616
 Thompson, P., 557, 565, 572
 Thomsen, J., 194, 213
 Thorsson, V., 582, 583, 588, 596
 Thrun, S., 231, 246
 Thuraisingham, B. M., 403, 407
 Thurstone, L. L., 226, 246
 Tibshirani, R., 172, 177, 182, 185,
 190, 191, 235, 242, 245
 Tibshirani, R.J., 14, 23, 436, 440
 Tibshirani, T., 126, 130
 Tilton, J., 556, 571
 Ting, K., 346, 361
 Tipping, M. E., 242, 246
 Titterington, D. M., 136, 138, 155,
 157, 241, 244, 246
 Toga, A., 557, 565, 572
 Toivonen, H., 31, 35, 38, 39,
 609, 616
 Tokuyama, T., 7, 23
 Torgerson, W., 249, 259, 276
 Tracy, N., 98, 102
 Traina, A. J. M., 291, 304
 Tselepis, S., 347, 355, 361
 Tsur, S., 31, 38
 Tukey, J. W., 264, 275
 Turner, K., 349, 361
 Tung, C. C., 306, 307, 323, 324,
 331, 339
 Turinsky, A. L., 353, 356, 361
 Turkinsky, A. L., 359
 Tusnady, G., 241, 244
 Tuzhilin, A., 31, 39, 611, 616
 Tye, J. B., 476, 479
 Tyson, J., 552, 570
 Tyson, M., 482, 517, 518
- U
- Uebersax, J. S., 241, 246
 Ullman, J. D., 31, 35, 37, 38
 Umbaugh, S., 565, 567, 572
 Unruh, A., 355, 360
 Urban, S., 355, 360
 Uthurusamy, R., 4, 23, 561, 570
 Uzumeri, M. V., 467, 468, 471,
 472, 474, 479

- V**
- Vafaie, H., 418, 421, 422
 Vaidya, J., 351, 361
 Vaidya, J. S., 447, 452
 Valdes, A., 628, 635
 Valencia, A., 580, 586, 596
 VanCott, H., 467, 478
 van de Pol, F., 243, 246
 van der Linde, A., 113, 131
 Van der Linden, W. J., 242, 246
 van Dyk, D. A., 136, 157
 Vapnik, V., 450, 452, 488, 518,
 610, 616
 Vardi, Y., 629, 635, 636
 Varshney, P., 344, 359, 361
 Vastano, J. A., 340
 Vatsavai, R. R., 537, 546
 Vazirgiannis, M., 540, 545
 Veksler, O., 535, 545
 Vempala, S., 297, 303
 Vemuri, V., 558, 572
 Vendruscolo, M., 577, 582, 596
 Venter, N. C., 561, 572
 Verissimo, A., 526, 546
 Verkamo, A., 31, 38
 Verkamo, A. I., 609, 616
 Vermunt, J. K., 242, 246
 Veronese, P., 111, 129
 Verykios, V., 450, 451
 Verykios, V. S., 452
 Vetterling, W. T., 340
 Viega, J., 618, 636
 Viergever, M., 565, 571
 Vigna, G., 628, 636
 Vilbert, S., 629, 636
 Vincent, R., 291, 304
 Vinod, H. D., 639, 655
 Viswanathan, R., 344, 361
 Viterbi, A. J., 134, 157
 Vlachos, M., 301, 304
 Volinsky, C. T., 114, 126, 130, 131,
 187, 191
 Von der Malsburg, C. J., 55, 66
 Von der Malsburg, C., 55, 66
- W**
- Wachspress, D. P., 242, 245
 Wagner, T., 344, 360
 Wainer, H., 242, 244
 Walker, D., 361
 Walker, H. F., 241, 246
 Wallace, C. S., 127, 131
 Wand, W., 640, 655
 Wang, F., 628, 635
 Wang, H., 289, 304
- Wang, S., 352, 360
 Wang, X., 242, 244, 301, 303, 304
 Wang, X. S., 468, 479
 Ward, M., 361
 Warrender, C. E., 533, 547, 628,
 629, 636
 Wasserman, W., 479
 Watanabe, S., 256, 276
 Wax, M., 209, 213
 Webb, G. I., 35, 36, 37, 39
 Weber, D., 628, 635
 Weber, H. J., 338
 Weber, R., 291, 304
 Webster, S., 628, 635
 Wedel, M., 242, 241, 243, 246
 Weeratunga, S. K., 566, 572
 Weiß, G., 344, 361
 Weickert, J., 566, 572
 Weinstein, S. P., 485, 486, 517
 Weiss, N., 134, 138, 155
 Weiss, S., 435, 440
 Weiss, S. M., 4, 24
 Welch, R. M., 556, 571
 Welles, M. L., 469, 479
 Werbos, P. J., 43, 66
 Wespi, A., 628, 635
 West, M., 126, 131, 136, 157
 White, D., 291, 304
 White, D. A., 42, 66
 White, G. B., 618, 635
 White, H., 42, 66
 White, R., 343, 360, 553, 554, 569
 White, R. L., 572
 Wichern, D. W., 199, 202, 204, 213
 Wichmann, K., 556, 571
 Wierse, A., 412, 422
 Willet, P., 430, 440
 Williams, C. K. I., 128, 131
 Williams, K., 339
 Williams, R., 459, 497, 518
 Williams, R. J., 43, 66
 Willshaw, D. J., 55, 66
 Wirth, R., 357, 361, 458, 459
 Wish, M., 297, 303
 Witten, I., 423
 Wixson, L., 535, 545
 Wnek, J., 417, 423
 Wolf, A., 319, 340
 Wolf, Y. I., 576, 596
 Wolfe, M., 354, 361
 Wolfe, P. M., 474, 479
 Wolff, R., 361
 Wolpert, D., 346, 361
 Wolpert, R., 163, 190
 Wong, J., 355, 359
 Wong, W. H., 157
 Woodall, W., 89, 101, 102
- Woodall, W. H., 98, 102, 658,
 661, 668
 Woodcock, C., 556, 570
 Wooldridge, M., 344, 354, 361
 Worboys, M., 520, 547
 Wortmann, J. C., 443, 451
 Wu, K.-I., 603, 616
 Wu, M. C., 306, 307, 323, 324,
 331, 339
 Wu, S., 628, 635
 Wu, W., 537, 546
 Wu, X., 628, 635
 Wüthrich, B., 348, 353, 358, 361
 Wyschogrod, D., 628, 635
 Wyse, N., 411, 423
- X**
- Xu, M., 624, 625, 626, 628, 629,
 631, 636
- Y**
- Yalcinkaya, T., 333, 334, 340
 Yamanishi, K., 242, 245, 353, 362
 Yanez-Suarez, O., 56, 66
 Yang, C., 35, 37, 38
 Yang, D., 33, 38
 Yang, J., 640, 655
 Yang, T., 546
 Yang, Y., 487, 518
 Yao, J., 410, 422
 Yao, K., 209, 213
 Yaroshevich, A., 517
 Yasui, Y., 520, 547
 Yates, J. F., 164, 191
 Yazdani, N., 285, 301, 304
 Ye, N., 618, 619, 620, 621, 623,
 624, 625, 626, 628, 629, 631,
 635, 636
 Yelle, L. E., 464, 467, 472, 479
 Yen, N.-C., 306, 307, 323, 324,
 331, 339
 Yi, B., 304
 Yi, B.-K., 288, 296, 301, 304
 Yi, J., 520, 547
 Yianilos, P. N., 281, 291, 304
 Yin, Y., 33, 38
 Yoda, K., 7, 23
 Yoon, J., 301, 304
 York, J., 108, 109, 131
 Yorke, J. A., 306, 308, 311, 312,
 314, 339, 340
 Young, J., 98, 102
 Young, J. C., 630, 635
 Yu, K., 127, 130, 171, 190

- Yu, P. S., 33, 38, 603, 616
Yuan, K. H., 242, 246
Yung, Y. F., 242, 246
- Z**
- Zabih, R., 535, 545
Zagelow, G., 403, 407
Zaki, M., 344, 360
Zaki, M. J., 35, 38, 39, 362, 579,
 585, 596
Zangwill, W. I., 464, 478
- Zaslavaky, A., 356, 360
Zdonik, S., 296, 304
Zeger, S., 230, 244
Zhang, B., 56, 66, 351, 359, 362
Zhang, F., 212, 213
Zhang, J., 346, 359, 361, 588, 596
Zhang, P., 525, 546
Zhang, S., 344, 360
Zhang, S. R., 289, 304
Zhang, T., 263, 277, 609, 616,
 640, 655
Zhang, Y., 628, 636
Zhang, Z., 596
- Zhao, C., 585, 596
Zhao, Y., 257, 277
Zheng, Q., 306, 307, 323, 324,
 331, 339
Zheng, Z., 423
Zhong, S., 261, 266, 277
Zhu, B., 66
Zilberstien, A., 482, 517
Zissman, M., 628, 635
Zoubir, A. M., 338
Zucchini, W., 134, 136, 141, 153,
 156, 157
Zumach, W., 552, 571

Subject Index

A

Accuracy, 425–426, 430, 532, 580, see also Error
 Spatial, 532
Adequacy of data, 372
Affinity, see Association
Agent-based model, 344, 354–355
 Mobile agents, 355
Aggregate, 525
Aggregation, 346–349, 351, 597, 604–606
Analytic signal, 305, 324, 326–327
Anomaly detection, 71, 617, 628–629, see also Control chart
APIs, 453–454, 456
Approximation
 Distribution, 349
 Function, 42
Artificial Neural Network, 41, 53, 55, 126, 159, 182,
 184, 367, 369, 370, 531, 629
Computer simulation, 46
Learning rate, 43–44, 49
Model
 Cascade network, 50–51
 Multilayer feedforward network, 41–42,
 412, 553
 Multilayer perceptrons, 42, 411
 Radial basis function networks, 42, 184
 Recurrent work, 42
Momentum, 43, 45
Software tools, 65
 Matlab, 65
Association, 27–28, 31–33
 Confidence, 27–28, 348
Frequency-based statistics, 27
Infrequent, 25, 35
 Rule space search, 35
Support, 27–28, 348
Association rule discovery, 25–31, 445–446, 450, 525,
 528, 584
Antecedent, 27
Apriori algorithm, 25, 28, 350
Consequent, 27
Distributed, 341, 350, 447
Attribute variables, 4–5, 7–8, 12–13, 15–22, 125,
 160, 261

Autocorrelation, 308, 311
 Spatial, 520, 532, 534, 543
Autocovariance, 209
Automated target recognition, 558
Autoregression, 534
 Spatial, 521
Average run length (ARL), 67, 96, 665, see also False positive and True positive
In-control, 664
Out-of-control, 665

B

Baum-Welch algorithm, see EM algorithm
Bayes' theorem, 104, 171–172, 487, 542
Bayesian data analysis, 103, 115
Bayes risk consistency, 175, 178, 182
Empirical approach, 111
Inference, 103–104
Learning, 105, 353
Model assessment, 103, 114
Model averaging, 103, 114
Model selection, 103, 113, 148, 149
 Bayesian information criterion, 149–150
 Schwarz criterion, 148
Model, 103, 125, 127
 Descriptive model, 103, 127
 Naïve Bayes model, 127
 Predictive model, 103, 125
Tree model, 126
Graphical network model, 127, 352
Software tools, 103, 121, 127
 BUGS, 121
Variational approximation, 120–121
Bias, 382–383
Binning
 Equal frequency, 378
 Supervised, 379
Bioinformatics, 573–574
Biometric identification, 558
 Fingerprint and retinal identification, 558
Blind source separation, 193, 205, 206
Fourth-Order Blind Separation, 193,
 206–210

- Bootstrapping, 14, 425, 436
 Boundary, 567
- C**
- Chaotic time series data, 312
 Attractors, 312
 Basin of attraction, 312
 Deterministic dynamical systems, 306
 Empirical mode decomposition, 331–335
 LK algorithm, 315, 316
 Lyapunov exponents, 305, 311, 312, 317, 321
 Morlet nanalysis, 323
 Natural measure, 312, 314
 Periodic orbits, 306, 314
 Unstable periodic orbits, 305, 311–315
 Phase space, 307–308, 310
 Reconstruction, 305, 307
 Recurrence time, 315–316
- Chi-square test of independence, 50
- Class label, see Target variable
- Classification, 3–4, 22, 26, 37, 41, 47, 52, 136, 159, 161, 169, 221, 411, 426, 429, 454, 533, 535, 536, 553, 576, 609, 639, see also Prediction
 Accuracy, 258
 Bayesian, 519, 535
 Binary, 169
 Distributed, 341
 Error, 425, 429
 Example-based, 487
 Meta-classifier, 347–348
 Probabilistic, 487
 Clustering, 55, 127, 136, 247–249, 351, 411, 488, 512–513, 519, 537, 539, 541, 609, 629, 637–638, 640, 646
 Balanced clustering, 247, 249, 260–261, 266, 274
 Cluster distance
 Average link, 255
 Complete link, 255
 Single link, 255
 Confidence measure, 638
 Dendrogram, 255, 539
 Global, 351
 Local, 351
 Distributed, 341, 346, 351
 Evaluation, 247, 256, 258, 425, 432
 F-measure, 258, 425, 430–431
 Mutual information-based quality measure, 258
 Recursive agglomeration of clustering hierarchies, 351
 Global variance ratio criterion, 641–643
 Indexing, 638
 Method
 Agglomerative, 250, 255
 Hierarchical agglomerative clustering, 147–248, 250, 255
 Density based, 248, 251, 539
 Discriminative, 247, 256
 Divisive, 250, 255
 Fuzzy, 254
 Fuzzy logic, 248
 Generative, 247, 251, 256
 Grid based, 253, 540
 Graph based, 253, 272
 Hierarchical, 539, 638
 k-means, 248, 250–251, 255, 262–263, 268, 272, 411, 539, 639
 k-median, 250–251
 k-medoid, 411, 519, 539, 540, 639
 LVRC, 643, 644
 Mode-seeking, 251
 Nested, 250
 Partitional, 263, 539, 638
 Spatial, 538
 Sequential building, 247, 261–263, 266
 Soft, 248, 254
 Supervised, 609
 Number of clusters, 637–638, 641
 Scalable, 247–248, 259, 262
 BIRCH, 263, 539–640
 DBSCAN, 251
 Software, 247, 268, 274
 CLIQUE, 540
 CURE, 539
 ROCK, 539
 SAS, 520
 SPSS, 520
 Collaborative filtering, 271, 299, 610
 Collection of data, 365, 370
 Collective data mining, 341, 349, see also Distributed data mining
 Collocation, 527
 Pattern, 526
 Rules, 526–527
 Spatial, 519, 525
 Complex data handling, 230–231, 261
 Compression of data, 287
 Computer underflow, 141
 Conditional independence, 127, 143, 224, 225, 227
 Spatial, 519, 533, 534
 Confidentiality, 448
 Contamination of data, 162
 Continuous data handling, see Numeric data handling
 Control chart, 68, 657
 Attributes control chart, 67, 81
 c control chart, 85
 Chi-square distance test, 617, 629, 633
 Control chart for nonconformities, 68, 81, 85
 Cumulative sum (CUSUM) control chart, 67, 89, 95
 Exponentially weighted moving average control chart, 67, 93, 95
 Fraction nonconforming control chart, 68, 81
 Hotelling T² control chart, 67, 98, 617, 629, 630, 633, 657–660
 Individuals control chart, 68, 76

- Multivariate control chart, 67, 98, 657–658
 Multivariate Exponentially Weighted Moving Average (MEWMA) chart, 67, 101, 657, 660, 662
 np control chart, 81, 84
 p control chart, 81, 82
 R control charts, 68, 69
 S control charts, 68, 73
 Shewhart control charts, 68, 89, 95
 u control chart, 87
 Univariate control charts, 67–68
 Variables control charts, 67
 \bar{x} control charts, 68, 69, 73, 98
 Parameters
 Centerline, 69, 74, 77, 82, 84, 86, 87
 Control limits, 69–70, 74, 77, 82, 84, 86, 87, 93, 99
 Decision interval, 90
 Three-sigma limits, 71
 Correlation, 194, 197
 Coefficient, 197
 Pearson correlation, 271
 Cost, 163, 356
 Estimation, 425, 433, 436
 Negative log-likelihood cost, 429
 Covariance, 49, 198–199, 202–204
 Cross validation, 425, 435–436, 555, 610
 Cryptography, 353, 447
 Cyclicity, 377
- D**
- Data
 Astronomical data, 552–553, 549–550
 Basket data, 26
 Biological, 573–577
 Genetic structure, 574–576, 588
 Computer audit/log data, 617–618, 621–623, 631
 Correlated multivariate data, 193, 194
 Coupled sequence data, 247, 261
 Customer-Relationship Management (CRM) Data, 597–598, 600
 Discrete valued time series, 134, 136
 Earth data, 549
 Ecological data, 526
 E-commerce data, 597–603
 Electrical data, 557
 Fetal lamb data, 134, 144, 147, 149
 Geospatial Data, 519–520, 530, see also Spatial data
 Human performance, 463
 Image data, 553–555, 557, 559, 567, 637, 640, 647
 Image, 638
 Log data, 260
 Market baskets, 247, 267
 Manufacturing data, 194, 657
 Mechanical data, 557
 Medical imaging data, 549
 Network traffic data, 620
 Normal versus abnormal data, 67
 Optical data, 552
 Pixel image data, 553–554
 Process data, 194
 Productivity, 468
 Radio frequency data, 552
 Remote sensing data, 550, 556
 Science and engineering data, 549
 Scientific Data, 549, 550, 551
 Sensor data, 559, see also Remote sensing data
 Sequence data, 261, 266
 Simulation data, 549, 558
 Spatial data, 520–522
 Surveillance data, 549, 558
 Survival data, 159, 166
 Text data, see Text data
 Time series, see Time series data
 Traffic data, 522, 525
 Transaction data, 27, 215–216, 236, 238, 247, 259, 267
 Web data, 247, 267, 270, 454
 Clickstream data, 247, 260, 598, 601–603
 Content data, 601
 Server log data, 603
 Structure data, 601
 Usage data, 601
 Database, 393, 397, 401
 DBMS (Database management system), 397, 405
 Distributed, 393, 404
 Hierarchical, 398
 Multidimensional, 402
 Object-Oriented, 393, 399
 Object Definition Language, 400
 Object-Oriented Query language, 401
 Relational, 345, 393, 398, 508
 Space partitioning, 247–248, 250, 263, 281
 B-tree, 263
 KD-tree, 281, 291
 KDB-tree, 263, 291
 Metric-tree, 291
 R-tree, 263, 281, 291
 SS-tree, 291
 TV-tree, 291
 VP-tree, 281, 291
 X-tree, 640
 SQL (Structural query language), 399, 453, 456
 Data type
 Binary, 248, 261
 Categorical, 4, 49, 216, 248, 261, 365, 367, 369, 379
 Nominal, 4, 50, 160, 216
 Numerical, 4, 25, 49, 68–69, 160, 173, 261, 365, 367, 369, 377–379
 Ordinal, 4, 50, 160, 164, 216, 261, 367
 Data warehouse, 342, 393, 403, 445, 599, 601
 Decision tree, 3–22, 41, 47, 49, 50, 53, 55, 159, 161, 177, 350, 367, 369, 444, 455, 553, 555, 629
 Binary tree, 5
 CART, 178, 182

- Decision tree (*Cont.*)
- CHAID (Chi-squared automatic interaction detection), 373
 - Classification tree, 4–7, 15, 21, 163
 - Construction, 3, 7, 13, 20
 - Data access, 3, 7–8, 10, 21
 - Database, 13
 - SQL query, 13
 - Scalable, 8
 - BOAT algorithm, 10, 13, 15
 - CLOUDS algorithm, 10
 - RainForest algorithm, 8, 11
 - SLIQ algorithm, 10
 - SPRINT algorithm, 8, 10, 22
 - Non-binary tree, 5
 - Pruning, 3, 15
 - Bottom-up pruning, 15
 - MDL pruning, 15
 - PUBLIC Algorithm, 15–16
 - Top-down pruning, 15
 - Regression Tree, 3–4, 20
 - Model, 20
 - Software tools, 22
 - C4.5, 22
 - C5.0, 520
 - CART, 22
 - Clementine, 520
 - IBM Intelligent Miner, 22
 - QUEST, 22
 - SAS Decision Trees, 22
 - SAS Enterprise Miner Suite, 22
 - SPSS Answer Tree, 22
 - Decryption, see Cryptography
 - Dense pattern, 573, 589
 - Dependent variable, see Response variable
 - Dimension, 305–307, 309, 312
 - Box-counting, 309
 - Correlation, 309, 310
 - Delay-coordinate embedding, 306–311
 - Fractal, 311
 - Dimensionality, 291, also see High-dimensional data
 - Dimensionality reduction, 264, 279, 280, 281, 292, 301, 410, 412, 469, 549, 567
 - Isomap, 298
 - Karhunen-Loëve transformation, 293
 - Linear partitioning, 281
 - Locally linear embedding, 298
 - Random projection techniques, 281, 296
 - Discriminant analysis, 41, 47–48
 - Linear, 49, 53, 55, 172, 553
 - Quadratic, 49, 53, 55
 - Distance measure
 - Chi-square, 630
 - Cosine, 271
 - Edit, 266, 284
 - Entropy, 220
 - Episode, 267
 - Euclidean, 58, 249, 270, 279, 281, 290, 351, 523, 529, 533, 648
 - Hamming, 267
 - Jaccard similarity, 267, 271
 - Kullback-Leibler measure, 220, 351, 429–430
 - Levenshtein distance, 266
 - Longest common subsequence distance, 267, 279–280, 284–286, 300, 301
 - L_p norms, 279
 - Mahalanobis distance, 48, 250, 271
 - Metric, 290
 - Minkowski distance, 270
 - Nonmetric, 299
 - Probabilistic, 279–280, 288
 - Time series similarity, 279, 280, 283
 - Distortion of data, 353
 - Distributed artificial intelligence, 344
 - Distributed data mining, 341–346, 349–350, 445, 447
 - Arbiter scheme, 347
 - Combiner scheme, 347
 - Data communication, 341, 351, 356
 - Components maintenance, 341
 - Environments, 344
 - Facilitator, 355
 - Heterogeneous data schemata, 341, 349, 351
 - Homogeneous data schemata, 341, 345–346, 348
 - Key association, 345
 - Knowledge networks, 355
 - Orthonormal basis, 349–350
 - Dynamic programming, 284
 - Dynamic time warping, 279–280, 283, 300, 301
 - Warping path, 283
 - Warping window, 283

E

- E-commerce, 344, 356
- Edge, 567
- EM algorithm, 133, 136–139, 146, 224, 219, 220–222, 226, 241, 254, 411 519, 540–544
- Encryption, see Cryptography
- Entropy, 258
- Error measure, 425, 427, 429, 610
 - Absolute prediction error, 162
 - Confusion table, 425, 431
 - Cost, 438
 - Estimation, 425
 - Resampling, 425, 435
 - Lift curves, 425, 432
 - Mean absolute deviation, 428
 - Mean square error, 428
 - Optimal Bayes, 429
 - Rate, 429
 - Recall, 425, 430–431
 - ROC (Receiving operating characteristic) curves, 425, 432, 634

- Spatial, 520
 Squared error, 162, 168
 Evaluation, 415, 425, 426, 507, 637, 649
 Expert, 426, 438
 Exponential weighted moving average (EWMA), 626–627
- F**
- Factor analysis, 193, 216, 226, 234–236, 242–243
 Software, 193, 211
 Matlab, 211, 212
 Minitab, 212
 SAS, 211, 212
 SPSS, 211, 212
 Factor loadings, 198, 227–228
 Factor Rotation, 193, 202–204, 212
 Varimax algorithm, 205, 211
 False alarm, see False positive
 False negative, 163, 425, 430
 False positive, 90, 95–96, 163, 425, 430, 633, 650
 Feature
 Construction, 409–410, 417–420
 Extraction, 409–411, 420
 Functional mapping, 411–412
 Linear transformation, 413
 Nonlinear transformation, 413
 Frequency, 624–625
 Intensity, 624–625
 Order, 624–625
 Selection, 409–410, 414
 LVF, 415–417
 Field testing, 426, 438
 Forecasting, see Prediction
 Forgetting, 463–464, 471, 474
 Fourier transform, 281, 294, 307, 323, 326–327, 350
 Discrete, 294
- G**
- Genetic algorithm, 418, 419
 Generalization, 42
 Generalized linear model (GLM), 159, 172, 230
 Gibbs sampler, 116, 118, 121, 139
- H**
- Hashing, 589–590
 Heterogeneous data, 236
 Hidden Markov models, 133–134, 582–584, 629
 Data augmentation, 138–139
 Empirical Bayes estimation, 147
 Full Bayes estimation, 147
 Label Switching, 144, 146, 151, 155
 Local computation, 133, 135, 140
- Forward-backward recursion, 133, 135, 139–142, 147
 Viterbi algorithm, 133, 135, 140, 142, 147
 Parameter estimation, 146
 Point estimation, 146
 Schwarz approximation, 149
 Software, 133, 154
 High-dimensional data handling, 236, 247, 259, 265, 264, 640
 Curse-of-dimensionality, 251, 260
 non-Gaussian data, 261
 Visualization, 258
 Hilbert transform, 305–307, 323, 324, 325, 326, 328
 Hit rate, see True positive
 Human face detection and recognition, 558
- I**
- Impact rules, 36
 Imperfection in measurement, 382
 Independent components analysis, 208
 Independent variable, see Attribute variable
 Instability, 368
 Instantaneous frequency, 324, 327–331
 Interface, see APIs
 Internet, 344
 Interpretability, 426, 438, 472
 Interestingness measure, 25, 31, 528, 597, 610–611
 Lift, 25, 31–33
 Leverage, 25, 32–33
 Intrinsic mode, 329, 331
 Intrusion detection, 558, 617–619, 628–629
 Signature recognition, 628
 Item set discovery, 25, 32
 Item set, 27
 Closed item set strategy, 25, 33
 Frequent closed item set, 33, 579
 Frequent item set, 25, 28–29, 33
 Long item set, 25, 35
 Maximal frequent item set, 35
- J**
- Java, 453, 456–457
 J2EE, 457
- K**
- Kalman filter, 235
 Kernel method, 159, 174
- L**
- Lack of data, 368
 Large data handling, see Massive data handling

- Latent variable modeling, 215–217
 Model
 Finite mixture model, 215–216, 221, 541
 Generative model, 218, 222, 224, 226
 Individualized mixture model, 215, 237
 Item response theory (IRT) model, 224, 233–234, 242–243
 Latent class model, 215–221, 229–231, 241, 243
 Latent trait model, 215–216, 224, 233
 Mixture model, 232, 236–237, 241, 541, 553
 Software tools, 215, 241
 AMOS, 243
 AUTOCLASS, 243
 BILOG, 243
 EMMIX, 243
 EQS, 243
 GENSTAT, 243
 GLMMIX, 243
 Latent GOLD, 243
 LISREL, 243
 MATLAB, 243
 Mplus, 243
 MULTILOG, 243
 MULTIMIX, 243
 PANMARK, 243
 SAS, 243
 SPLUS, 243
 SPSS, 243
 SVD, 243
 SYSTAT, 243
 WEKA, 243
 Learning, 219, 222, 226–227
 Batch, 41, 47
 Boosting, 159, 185
 Competitive, 266
 Distributed, 347
 Knowledge probing, 347–348
 Incremental, 41, 47
 Local, 346, 348
 Metalearning, 347–348
 Rule, 43–47, 57
 Back propagation algorithm, 43–45
 Cyclic coordinate descent, 219
 Gauss-Newton method, 45–46, 169
 Gradient ascent method, 185
 Gradient decent method, 43
 Hebbian learning rule, 57
 Levenberg-Marquardt algorithm, 43, 46
 Nonlinear least squares method, 41, 43, 45
 Quasi-Newton method, 43
 Supervised, 42, 46, 367
 Unsupervised learning, 217, 221, 248, 367
 Learning curves, 463–464, 467–468, 470
 Learning map, 475
 Lifecycle, 394
 Likelihood principle, 163
 Linear model, 148, 159, 167–168
 Local independence, 219
 Logistic regression, 120, 169, 226, 232, 531
 Logistic SAR model, 519, 534
 Loss functions, 159, 161, 162, 164, 185
 Brier score, 164, 166
 Calibration, 164–165
 Classification, 159, 163, 166–167
 Cox model, 159, 166
 Regression, 159, 162, 166
 AdaBoost, 166

 M
 Machine learning, 485
 MAP, 147
 Market Basket Analysis, 25–26, 597
 Aggregate market baskets, 36
 Markov model, 127
 Discrete-time, 288
 Markov chain, 116, 134, 629
 Markov random field (MRF) model, 519–520, 535
 Markov transition probabilities, 153
 Massive data handling, 33, 36, 103, 118, 119, 154, 230, 236, 247, 249, 259, 262, 638
 Sampling, 25, 35
 Measurement accuracy, 382–383
 Mesh, 559
 Cartesian mesh, 559
 Composite mesh, 559
 Metadata, 456, 458
 Metropolis-Hastings algorithm, see EM algorithm
 Misclassification rate, 5–6, 15, 163, 166
 Missing data handling, 3, 17–19, 118, 133, 135–136, 138, 180, 345, 365, 367, 373, 375–376
 Check model, 373
 Estimation, 18
 Surrogate splits, 18–19
 Misuse, 448
 Model quality, 384
 Monotonicity, 226
 Monte Carlo method, 115
 Fragment insertion, 588
 Markov Chain Monte Carlo (MCMC), 103, 116, 127, 133, 136, 138, 146–148, 150, 535
 Bayesian MCMC, 140
 Metropolis-Hastings algorithm, 116–117
 Sampling, 103, 115, 118, 119, 236
 Moving average, 282
 Moving range, 77–80
 Moving window, 625
 Multidimensional scaling, 232, 265, 281, 297
 Multi-nominal log-likelihood, 164
 Multisensor data fusion, 344
 Multivariate adaptive regression splines, 182

N

Naïve Bayes classifier, 171
 Nearest neighbor, 159, 174, 487, 553
 Neighborhood function, 58, 529
 Noise, 262, 368, 371, 470
 De-noising, 549, 565–566
 Non-linear models, 159, 468
 Additive models, 159, 179, 180–182
 B-splines, 180–181
 Natural spline, 179–180
 Piecewise polynomial, 179
 Polynomial, 179, 184
 Spline fitting, 332–334
 Thin-plate splines, 181
 Nonlinear time series data, 305–306, see also Chaotic time series data
 Nonparametric, 663
 Nonstationary time series, 306
 Normalization of data, 345, 365, 377–378
 Numeric data handling, 36, 216, 236
 Bumping hunting, 36
 Discretization, 36

O

Objective function, 433
 OLAP (On-Line Analytical Processing), 260, 393, 394, 402, 605
 Organizational Learning, 463, 464, 465, 467
 Outlier, 127, 162, 172, 260, 365, 368, 377–378, 519, 521, 525
 Spatial, 519, 522–523, 525
 Overfitting, 15, 427

P

Parallel computing, 344
 Performance measure, 411–412, 425
 Perturbation of data, 443, 443, 445
 Precision, 425, 430–431
 Predictor variable, see Attribute variable
 Prediction, 159–191, 280, 426, 427, 466, 530
 Accuracy, 410, 426
 Location, 519, 530, 532–533
 Model, 159, 161, 174
 Protein structure, 573–576, 580, 585
 Software, 159, 188
 Preparation of data, 341, 345, 365–368, 370, 374, 410, 597–598, 603–604, 607, 610
 Model driven, 370
 Cleaning, 604
 Pre-processing of data, see Preparation of data
 Protein contact, 573, 578

Non-contact, 573, 578
 Prediction, 573, 582
 Protein contact map, 573, 576–578, 587–588
 Prediction, 586
 Principal Components Analysis, 193, 198, 200, 202, 211, 227, 243, 352, 265, 412, 567–568
 Collective, 352
 Principles of data mining, 368, 371–379, 383
 Privacy, 341, 352–353, 441–442, 445, 447, 463, 477
 Probability distribution
 Bernoulli, 164, 169
 Posterior, 104, 105, 120, 144, 150, 153
 Prior, 111, 120, 144–145
 Probability model
 Conditional, 126
 Full, 104, 125
 Poisson distribution, 173
 Psychometric method, 215–216
 Purity, 258

Q

Quality of data, 365, 380, 382, 384, 410

R

Real AdaBoost algorithm, 185
 Regression, 160, 465, 534, 609, see also Linear model, Nonlinear model, and Prediction
 Error, 425–426, 428
 Linear, 465
 Multivariate, 350
 Nonlinear, 465
 Ordinal, 164
 Poisson, 189
 Proportional hazards, 166–167
 Resource management, 354
 Response variable, 4, 125
 Binary, 164, 173
 Continuous, 173
 Count, 173
 Risk, 425–426, 436
 Robustness, 162, 597, 610, 663

S

Sampling, 264
 Scalability, 162, 281
 Scaling, 283
 Decimal, 345
 Global, 286
 Local, 285
 Standard deviation, 345

- Scatter plots, 194, 522–524
 Moran, 523
- Security, 441, 549, 558, 617–618
- Self organizing map, 41, 55, 56, 57, 58, 251, 259, 265,
 272, 384
- Semantic Web, 453, 457
- Sensitivity, 307, 425, 431
- Sequential patterns, 608–609
 Variable length, 266
- Signal processing, 205
- Significance testing, 425, 433
- Similarity, 48, 247–249, 265–267, 270–271, see also
 Distance
- Smoothing, 159, 179
- Sparse data handling, 33
- Spatial ordering, 55–56
- Spatial distribution, 530
- Spatial structure, 520
- Spatiotemporal pattern, 526
- Specificity, 425, 431
- Split selection, 3–5, 7–8, 10, 12–13, 20–22
 Binary split, 13
 Criterion
 Coarse splitting, 14
 Correlation x^2 , 7
 Entropy, 7–8
 Gini-index, 7–8
 Impurity, 7, 13
 Impurity function, 7, 14, 21
 Cost, 16
- Spreadsheets, 393, 395
- Spurious measure, 162
- Stability, 471
- Standard, 453–454
 CRISP 1.0 Process Model, 458
 CRISP-DM, 458
 CWM DM, 457–458
 JDM, 457–458
 OLE (Object Linking and Embedding) DB, 453,
 457
 PMML, 454–458
 SQL/MM, 456, 457, 458
 UML (Unified Modeling Language), 456
 Web, 453
 XML, 453–454, 456–458, 483
- Standardization of data, 345
- Statistical process control, see Control chart
- Storage of data, 393, also see Database
 Software, 393, 406
 Text files, 393, 397
- Structural equation modeling, 243
- Support vector machines, 159, 163, 166, 177, 183, 184,
 185, 488
- Survival analysis, 166
- Survival model, 161, 189
- T
- Target variable, see Dependent variable
- Target class, see Target variable
- Text data, 37, 247, 270–272, 481, 488–489
 Anaphora resolution, 481, 506
 Categorization, 481, 485, 488
 CONSTRUE, 485
 Category Connection Map, 481, 509
 Coreference, 506
 Parsing, 507
 Propositional rules learning, 487
 Ripper, 488
 Relation Map, 481, 510
 Rule editor, 494
 Semantic tagging, 481, 489
 DIAL (Declarative Information Analysis
 Language), 481, 491, 494
- Soft matching, 481, 505
- Tagging
 Statistical, 481, 485
 Structural, 481, 500
- Taxonomy construction, 481, 501
- Temporal resolution, 481
- Term extraction, 481, 489
- Testing
 Non-destructive, 549, 557
- Time measurement, 365, 376, 425, 438
- Time-frequency analysis, 305–306, 323
- Time series, 279–280, 608
 Acceleration, 282
 Deceleration, 282
 General transformations, 279
 Global preserving, 292
 Indexing, 279–280, 289–290
 Multidimensional, 291
 Line segment approximation, 296
 Local preserving, 292
 Normalization transformations, 282
 Phase shifts, 282
 Piecewise aggregate approximation, 296
 Piecewise linear approximation, 279–280, 287
 Retrieval, 279–280, 289–301
 String matching, 299
 Seasonality, 280
 Shape preserving, 292
 Shift transformation, 283
 Singular value decomposition, 281, 293
 Smoothening, 283
 Subsequence matching, 289, 301
 Topology preserving, 292
- Topological mapping, see Spatial ordering
- Topological relationship, 529
- Training time, 425–426, 438
- Trend analysis, 280
- True negatives, 430, 650
- True positives, 430, 633

U

Uncertainty, 368
Underfitting, 427
Undirected graph, 56

Visualization, 56, 247, 365–366, 384
 Absolute, 366, 384
 Relative, 366, 384
Text, 481, 508
Trend graph, 481, 516

V

Variation, 68, 382, 428
 Chance causes, 68
Structure, 193, 197, 227

W

Wavelet transform, 281, 295, 307, 323, 350, 638,
 647–648
Thresholding, 565
World Wide Web, 344, 453, 457–458

