Data Preparation for Data Mining

Dorian Pyle

Senior Editor: Diane D. Cerra
Director of Production & Manufacturing: Yonie Overton
Production Editor: Edward Wade
Editorial Assistant: Belinda Breyer
Cover Design: Wall-To-Wall Studios
Cover Photograph: © 1999 PhotoDisc, Inc.
Text Design & Composition: Rebecca Evans & Associates
Technical Illustration: Dartmouth Publishing, Inc.
Copyeditor: Gary Morris
Proofreader: Ken DellaPenta
Indexer: Steve Rath
Printer: Courier Corp.

Morgan Kaufmann Publishers, Inc.
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA
Telephone 415-392-2665
Facsimile 415-982-2665
Email *mkp@mkp.com*
WWW *http://www.mkp.com*

Order toll free 800-745-7323

Dedication

To my dearly beloved Pat, without whose love, encouragement, and support, this book, and very much more, would never have come to be

Table of Contents

Preface

What This Book Is About

This book is about what to do with data to get the most out of it. There is a lot more to that statement than first meets the eye.

Much information is available today about data warehouses, data mining, KDD, OLTP, OLAP, and a whole alphabet soup of other acronyms that describe techniques and methods of storing, accessing, visualizing, and using data. There are books and magazines about building models for making predictions of all types—fraud, marketing, new customers, consumer demand, economic statistics, stock movement, option prices, weather, sociological behavior, traffic demand, resource needs, and many more.

In order to use the techniques, or make the predictions, industry professionals almost universally agree that one of the most important parts of any such project, and one of the most time-consuming and difficult, is data preparation. Unfortunately, data preparation has been much like the weather—as the old aphorism has it, “Everyone talks about it, but no one does anything about it.” This book takes a detailed look at the problems in preparing data, the solutions, and how to use the solutions to get the most out of the data—whatever you want to use it for. This book tells you what can be done about it, exactly how it can be done, and what it achieves, and puts a powerful kit of tools directly in your hands that allows you to do it.

How important is adequate data preparation? After finding the right problem to solve, data preparation is often *the* key to solving the problem. It can easily be the difference between success and failure, between useable insights and incomprehensible murk, between worthwhile predictions and useless guesses.

For instance, in one case data carefully prepared for warehousing proved useless for modeling. The preparation for warehousing had destroyed the useable information content for the needed mining project. Preparing the data for mining, rather than warehousing, produced a 550% improvement in model accuracy. In another case, a commercial baker achieved a bottom-line improvement approaching $1 million by using data prepared with the techniques described in this book instead of previous approaches.

Who This Book Is For

This book is written primarily for the computer savvy analyst or modeler who works with data on a daily basis and who wants to use data mining to get the most out of data. The type of data the analyst works with is not important. It may be financial, marketing, business, stock trading, telecommunications, healthcare, medical, epidemiological, genomic, chemical, process, meteorological, marine, aviation, physical, credit, insurance, retail, or any type of data requiring analysis. What is important is that the analyst needs to get the most information out of the data.

At a second level, this book is also intended for anyone who needs to understand the issues in data preparation, even if they are not directly involved in preparing or working with data. Reading this book will give anyone who uses analyses provided from an analyst’s work a much better understanding of the results and limitations that the analyst works with, and a far deeper insight into what the analyses mean, where they can be used, and what can be reasonably expected from any analysis.

Why I Wrote It

There are many good books available today that discuss how to collect data, particularly in government and business. Simply look for titles about databases and data warehousing. There are many equally good books about data mining that discuss tools and algorithms. But few, if any books, address what to do with the “dirty data” after it is collected and before exploring it with a data mining tool. Yet this part of the process is critical.

I wrote this book to address that gap in the process between identifying data and building models. It will take you from the point where data has been identified in some form or other, if not assembled. It will walk you through the process of identifying an appropriate problem, relating the data back to the world from which it was collected, assembling the data into mineable form, discovering problems with the data, fixing the problems, and discovering what is in the data—that is, whether continuing with mining will deliver what you need. It walks you through the whole process, starting with data discovery, and deposits you on the very doorstep of building a data-mined model.

This is not an easy journey, but it is one that I have trodden many times in many projects. There is a “beaten path,” and my express purpose in writing this book is to show exactly where the path leads, why it goes where it does, and to provide tools and a map so that you can tread it again on your own when you need to.

Special Features

A CD-ROM accompanies the book. Preparing data requires manipulating it and looking at it in various ways. All of the actual data manipulation techniques that are conceptually described in the book, mainly in Chapters 5 through 8 and 10, are illustrated by C programs. For ease of understanding, each technique is illustrated, so far as possible, in a separate, well-commented C source file. If compiled as an integrated whole, these provide an automated data preparation tool.

The CD-ROM also includes demonstration versions of other tools mentioned, and useful for preparing data, including WizWhy and WizRule from WizSoft, KnowledgeSEEKER from Angoss, and Statistica from StatSoft.

Throughout the book, several data sets illustrate the topics covered. They are included on the CD-ROM for reader investigation.

Acknowledgments

I am indebted beyond measure to my dearly beloved wife, Pat Thompson, for her devoted help, support, and encouragement while this book was in progress. Her reading and rereading of the manuscript helped me to clarify many difficult points. There are many points that would without doubt be far less clear but for her help. I am also indebted to my friend Dr. Ralphe Wiggins who read the manuscript and helped me clarify a number of points and improve the overall organization of chapters.

My publisher helped me greatly by having the book reviewed by several anonymous reviewers, all of whom helped improve the final book. To those I can only generally express my thanks. However, one reviewer, Karen Watterson, was extremely helpful, and at times challenging, for which I was then, and remain, most grateful.

My gratitude also goes to Irene Sered of WizSoft, Ken Ono of Angoss, and Robert Eames of StatSoft, all of whom supported the project as it went forward, providing the demonstration software.

Last, but certainly not least, my gratitude goes to Diane Cerra, my editor at Morgan Kaufmann, to Edward Wade the production editor, to the copyeditor and proofreader who so carefully read the manuscript and made improvements, to the illustrators who improved my attempts at the figures throughout, and to all of the staff at Morgan Kaufmann who helped bring this project to completion.

In spite of all the help, support, encouragement, and constructive criticism offered by these and other people, I alone, of course, remain responsible for the book’s shortcomings, faults, and failings.

Introduction

Ever since the Sumerian and Elam peoples living in the Tigris and Euphrates River basin some 5500 years ago invented data collection using dried mud tablets marked with tax records, people have been trying to understand the meaning of, and get use from, collected data. More directly, they have been trying to determine how to use the information in that data to improve their lives and achieve their objectives.

These are the same objectives addressed by the latest technology to wring use and meaning out of data—the group of technologies that today have come to be called data mining. Often, something important gets lost in the rush to apply these powerful technologies to "find something in this data." The technologies themselves are not an answer. They are tools to help find an answer. It is no use looking for an answer unless there is a question. But equally important, given a question, both the data and the miner need to be readied to find the best answer to the question asked.

This book has two objectives: 1) to present a proven approach to preparing the data, and the miner, to get the most out of computer-stored data, and 2) to help analysts and business managers make cost-effective and informed decisions based on the data, their expertise, and business needs and constraints. This book is intended for everyone who works with or uses data and who needs to understand the nature, limitations, application, and use of the results they get.

In *The Wizard of Oz*, while the wizard hid behind the curtain and manipulated the controls, the results were both amazing and magical. When the curtain was pulled back, and the wizard could be seen manipulating the controls, the results were still amazing—the cowardly lion did find courage, the tin man his heart, the scarecrow his brain. The power remained; only the mystery evaporated. This book "pulls back the curtain" about the reason, application, applicability, use, and results of data preparation.

Knowledge, Power, Data, and the World

Francis Bacon said, "Knowledge is power." But is it? And if it is, *where* is the power in knowledge?

Power is the ability to control, or at least influence, events. Control implies taking an action that produces a known result. So the power in knowledge is in knowing what to do to get what you want—knowing which actions produce which results, and how and when to take them. Knowledge, then, is having a collection of actions that work reliably. But where does this knowledge come from?

Our knowledge of the world is a map of how things affect each other. This comes from observation—watching what happens. Watching implies making a record of happenings, either mental or in some other form. These records, when in nonmental form, are *data*, which is simply a collection of observations of things that happen, and what other things happen when the first things happen. And how consistently.

The world forms a comprehensive interlocking system, called by philosophers “the great system of the world.” Essentially, when any particular thing happens in the world, other things happen too. We call this *causality* and want to know what causes what. Everything affects everything else. As the colloquial expression has it, “You can’t do just one thing.” This system of connected happenings, or events, is reflected in the data collected.

Data, Fishing, and Decision Making

We are today awash in data, primarily collected by governments and businesses. Automation produces an ever-growing flood of data, now feeding such a vast ocean that we can only watch the swelling tide, amazed. Dazed by our apparent inability to come to grips with the knowledge swimming in the vast ocean before us, we know there must be a vast harvest to be had in this ocean, if only we could find the means.

Fishing in data has traditionally been the realm of statistical analysis. But statistical analysis has been as a boy fishing with a pole from a riverbank. Today’s business managers need more powerful and effective means to reap the harvest—ways to explore and identify the denizens of the ocean, and to bring the harvest home. Today there are three such tools for harvesting: *data modeling* reveals each “fish,” *data surveying* looks at the shape of the ocean and is the “fish finder,” and *data preparation* clears the water and removes the murk so that the “fish” are clearly seen and easily attracted.

So much for metaphor. In truth, corporations have huge data “lakes” that range from comprehensive data stores to data warehouses, data marts, and even data “garbage dumps.” Some of these are more useful than others, but in every case they were created, and data collected, because of the underlying assumption that collected data has value, corporate value—that is it can be turned into money.

All corporations have to make decisions about which actions are best to achieve the corporate interest. Informed decisions—those made with knowledge of current circumstances and likely outcome—are more effective than uninformed decisions. The core business of any corporate entity is making appropriate decisions, and enterprise decision support is the core strategic process, fed by knowledge and expertise—and by the best available information. Much of the needed information is simply waiting to be discovered, submerged in collected data.

Mining Data for Information

The most recently developed tools for exploring data, today known as data mining tools, only begin the process of automating the search. To date, most modern data mining tools have focused almost exclusively on building models—identifying the “fish.” Yet enormous dividends come from applying the modeling tools to correctly prepared data. But preparing data for modeling has been an extremely time-consuming process, traditionally carried out by hand and very hard to automate.

This book describes automated techniques of data preparation, both methods and business benefits. These proven automated techniques can cut the preparation time by up to 90%, depending on the quality of the original data, so the modeler produces better models in less time. As powerful and effective as these techniques are, the key benefit is that, properly applied, the data preparation process prepares both the data *and* the modeler. When data is properly prepared, the miner unavoidably gains understanding and insight into the content, range of applicability, and limits to use of the data. When data is correctly prepared and surveyed, the quality of the models produced will depend mostly on the content of the data, not so much on the ability of the modeler.

But often today, instead of adequate data preparation and accurate data survey, time-consuming models are built and rebuilt in an effort to understand data. Modeling and remodeling are not the most cost-efficient or the most effective way to discover what is enfolded in a data set. If a model is needed, the data survey shows exactly which model (or models if several best fit the need) is appropriate, how to build it, how well it will work, where it can be applied, and how reliable it will be and its limits to performance. All this can be done before any model is built, and in a small fraction of the time it takes to explore data by modeling.

Preparing the Data, Preparing the Miner

Correct data preparation prepares both the miner and the data. Preparing the data means the model is built right. Preparing the miner means the right model is built. Data preparation and the data survey lead to an understanding of the data that allows the right model to be built, and built right the first time. But it may well be that in any case, the preparation and survey lead the miner to an understanding of the information enfolded in the data, and perhaps that is all that is wanted. But who is the miner?

Exploring data has traditionally been a specialist activity. But it is business managers who need the results, insights, and intuitions embedded in stored data. As recently as 20 years ago, spreadsheets were regarded as specialized tools used by accountants and were considered to have little applicability to general business management. Today the vast majority of business managers regard the spreadsheet as an indispensable tool. As with the spreadsheet, so too the time is fast approaching when business managers will directly access and use *data exploration* tools in their daily business decision making. Many important business processes will be run by automated systems, with business managers and analysts monitoring, guiding, and driving the processes from “control panels.” Such structures are already beginning to be deployed. Skilled data modelers and explorers will be needed to construct and maintain these systems and deploy them into production.

So who is the miner? Anyone who needs to understand and use what is in corporate data sets. This includes, but is not limited to, business managers, business analysts, consultants, data analysts, marketing managers, finance managers, personnel managers, corporate executives, and statisticians. The *miner* in this book refers to anyone who needs to directly understand data and wants to apply the techniques to get the best understanding out of the data as effectively as possible. (The miner may or may not be a specialist who implements these techniques for preparation. It is at least someone who needs to use them to understand what is going on and why.) The *modeler* refers to someone versed in the special techniques and methodologies of constructing models.

Is This Book for You?

I have been involved, one way or another, in the world of using automated techniques to extract "meaning" from data for over a quarter of a century. Recently, the term "data mining" has become fashionable. It is an old term that has changed slightly in meaning and gained a newfound respectability. It used to be used with the connotation that if you mess around in data long enough, you are sure to find something that seems useful, but is probably just an exercise in self-deception. (And there is a warning to be had there, because self-deception is very easy!)

This "mining" of data used to be the specialist province of trained analysts and statisticians. The techniques were mainly manual, data quantities small, and the techniques complex. The miracle of the modern computer (not said tongue in cheek) has changed the entire nature of data exploration. The rate of generation and collection of raw data has grown so rapid that it is absolutely beyond the means of human endeavor to keep up. And yet there is not only meaning, but huge value to be had from understanding what is in the data collections. Some of this meaning is for business—where to find new customers, stop fraud, improve production, reduce costs. But other data contains meaning that is important to understand, for our lives depend on knowing some of it! Is global warming real or not? Will massive storms continue to wreak more and more havoc with our technological civilization? Is a new ice age almost upon us? Is a depression imminent? Will we run out of resources? How can the developing world be best helped? Can we prevent the spread of AIDS? What is the meaning of the human genome?

This book will not answer any of those questions, but they, along with a host of other questions large and small, will be explored, and explored almost certainly by automated means—that is, those techniques today called data mining. But the explorers will not be exclusively drawn from a few, highly trained professionals. Professional skill will be sorely needed, but the bulk of the exploration to come will be done by the people who face the problems, and they may well not have access to skilled explorers. What they will have is access to high-powered, almost fully automated exploration tools. They will need to know the appropriate use and limits of the tools—and how to best prepare their data.

If you are looking at this book, and if you have read this far through the introduction, almost certainly this book is for you! It is *you* who are the "they" who will be doing the exploring, and this book will help you.

Organization

Data preparation is both a broad and a narrow topic. Business managers want an overview of where data preparation fits and what it delivers. Data miners and modelers need to know which tools and techniques can be applied to data, and how to apply them to bring the benefits promised. Business and data analysts want to know how to use the techniques and their limits to usefulness. All of these agendas can be met, although each agenda may require a different path through the book.

Chapters 1 through 3 lay the ground work by describing the data exploration process in which data preparation takes place. Chapters 4 through 10 outline each of the problems that have to be addressed in best exposing the information content enfolded in data, and provide conceptual explanations of how to deal with each problem. Chapters 11 and 12 look at what can be discovered from prepared data, and how both miner and modeling performance are improved by using the techniques described.

Chapter 1 places data preparation in perspective as part of a decision-making process. It discusses how to find appropriate problems and how to define what a solution looks like. Without a clear idea of the business problem, the proposed business objectives, and enough knowledge of the data to determine if it's an appropriate place to look for at least part of the answer, preparing data is for naught. While Chapter 1 provides a top-down perspective, Chapter 2 tackles the process from the bottom up, tying data to the real world, and explaining the inherent limitations and problems in trying to capture data about the world. Since data is the primary foundation, the chapter looks at what data is as it exists in database structures. Chapter 3 describes the data exploration process and the interrelationship between its components—data preparation, data survey, and data modeling. The focus in this chapter is on how the pieces link together and interact with each other.

Chapters 4 through 9 describe how to actually prepare data for survey and modeling. These chapters introduce the problems that need to be solved and provide conceptual descriptions of all of the techniques to deal with the problems. Chapter 4 discusses the data assay, the part of the process that looks at assembling data into a mineable form. There may be much more to this than simply using an extract from a warehouse! The assay also reveals much information about the form, structure, and utility of a data set. Chapters 5 through 8 discuss a range of problems that afflict data, their solutions, and also the concept of how to effectively expose information content. Among the topics these chapters address are discovering how much data is needed; appropriately numerating alpha values; removing variables and data; appropriately replacing missing values;

normalizing range and distribution; and assembling, enhancing, enriching, compressing, and reducing data and data sets. Some parts of these topics are inherently and unavoidably mathematical. In every case, the mathematics needed to understand the techniques is at the "forgotten high school math" level. Wherever possible, and where it is not required for a conceptual understanding of the issues, any mathematics is contained in a section titled Supplemental Material at the end of those particular chapters. Chapter 9 deals entirely with preparing series data, such as time series.

Chapter 10 looks at issues concerning the data set as a whole that remain after dealing with problems that exist with variables. These issues concern restructuring data and ensuring that the final data set actually meets the need of the business problem.

Chapter 11 takes a brief look at some of the techniques required for surveying data and examines a small part of the survey of the example data set included on the accompanying CD-ROM. This brief look illustrates where the survey fits and the high value it returns. Chapter 12 looks at using prepared data in modeling and demonstrates the impact that the techniques discussed in earlier chapters have on data.

All of the preparation techniques discussed here are illustrated in a suite of C routines on the accompanying CD-ROM. Taken together they demonstrate automated data preparation and compile to provide a demonstration data preparation program illustrating all of the points discussed. All of the code was written to make the principles at work as clear as possible, rather than optimizing for speed, computational efficiency, or any other metric. Example data sets for preparation and modeling are included. These are the data sets used to illustrate the discussed examples. They are based on, or extracted from, actually modeled data sets. The data in each set is assembled into a table, but is not otherwise prepared. Use the tools and techniques described in the book to explore this data. Many of the specific problems in these data sets are discussed, but by no means all. There are surprises lurking, some of which need active involvement by the miner or modeler, and which cannot all be automatically corrected.

Back to the Future

I have been involved in the field known today as data mining, including data preparation, data surveying, and data modeling, for more than 25 years. However, this is a fast-developing field, and automated data preparation is not a finished science by any means. New developments come only from addressing new problems or improving the techniques used in solving existing problems. The author welcomes contact from anyone who has an interest in the practical application of data exploration techniques in solving business problems.

The techniques in this book were developed over many years in response to data problems and modeling difficulties. But, of course, no problems are solved in a vacuum. I am indebted to colleagues who unstintingly gave of their time, advice, and insight in bringing this book to

fruition. I am equally indebted to the authors of many books who shared their knowledge and insight by writing their own books. Sir Isaac Newton expressed the thought that if he had seen further than others, it was because he stood on the shoulders of giants. The giants on whose shoulders I, and all data explorers stand, are those who thought deeply about the problems of data and its representations of the world, and who wrote and spoke of their conclusions.

Chapter 1: Data Exploration as a Process

Overview

Data exploration starts with data, right? Wrong! That is about as true as saying that making sales starts with products.

Making sales starts with identifying a need in the marketplace that you know how to meet profitably. The product must fit the need. If the product fits the need, is affordable to the end consumer, and the consumer is informed of your product’s availability (marketing), then, and only then, can sales be made. When making sales, meeting the needs of the marketplace is paramount.

Data exploration also starts with identifying a need in its “marketplace” that can be met profitably. Its marketplace is corporate decision making. If a company cannot make correct and appropriate decisions about marketing strategies, resource deployment, product distribution, and every other area of corporate behavior, it is ultimately doomed. Making correct, appropriate, and informed business decisions is the paramount business need. Data exploration can provide some of the basic source material for decision making—information. It is information alone that allows informed decision making.

So if the marketplace for data exploration is corporate decision making, what about profit? How can providing any information not be profitable to the company? To a degree, any information is profitable, but not all information is equally useful. It is more valuable to provide accurate, timely, and useful information addressing corporate strategic problems than about a small problem the company doesn’t care about and won’t deploy resources to fix anyway. So the value of the information is always proportional to the scale of the problem it addresses. And it always costs to discover information. Always. It takes time, money, personnel, effort, skills, and insight to discover appropriate information. If the cost of discovery is greater than the value gained, the effort is not profitable.

What, then, of marketing the discovered information? Surely it doesn’t need marketing. Corporate decision makers know what they need to know and will ask for it—won’t they? The short answer is no! Just as you wouldn’t even go to look for stereo equipment unless you knew it existed, and what it was good for, so decision makers won’t seek information unless they know it can be had and what it is good for. Consumer audio has a great depth of detail that needs to be known in order to select appropriate equipment. Whatever your level of expertise, there is always more to be known that is important—once you know about it. Speakers, cables, connectors, amplifiers, tuners, digital sound recovery, distortion, surround sound, home theater, frequency response. On and on goes the list, and detailed books have been written about the subject. In selecting audio equipment (or anything else for that matter), an educated consumer makes the best choice. It is exactly the same with information discovered using data exploration.

The consumers are decision makers at all levels, and in all parts of any company. They need to know that information is available, as well as the sort of information, its range of applicability, limits to use, duration of applicability, likely return, cost to acquire, and a host of other important details. As with anything else, an educated consumer makes the best use of the resource available. But unlike home audio equipment, each problem in data exploration for business is unique and has needs different from other problems. It has not yet become common that the decision maker directly explores broadly based corporate data to discover information. At the present stage of data exploration technology, it is usual to have the actual exploration done by someone familiar with the tools available—the *miner*. But how are the miner and the decision maker(s) to stay "in synch" during the process? How is the consumer, the decision maker, to become educated about reasonable expectations, reasonable return, and appropriate uses of the discovered information?

What is needed is a process. A process that works to ensure that all of the participants are engaged and educated, that sets appropriate expectations, and that ensures the most value is obtained for the effort put in. That process is the data exploration process, introduced in this chapter.

1.1 The Data Exploration Process

Data exploration is a practical multistage business process at which people work using a structured methodology to discover and evaluate appropriate problems, define solutions and implementation strategies, and produce measurable results. Each of the stages has a specific purpose and function. This discussion will give you a feel for the process: how to decide what to do at each stage and what needs to be done. This is a look at what goes in, what goes on, and what comes out of data exploration. While much of this discussion is at a conceptual level, it provides some practical "hands-on" advice and covers the major issues and interrelationships between the stages.

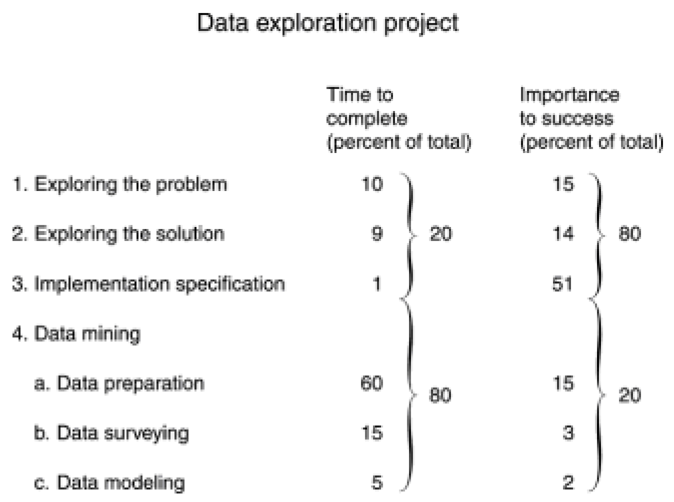
At the highest-level overview, the stages in the data exploration process are

1. Exploring the Problem Space
2. Exploring the Solution Space
3. Specifying the Implementation Method
4. Mining the Data (three parts)
    a. Preparing the Data

b. Surveying the Data

c. Modeling the Data

This is the “map of the territory” that you should keep in mind as we visit each area and discuss issues. Figure 1.1 illustrates this map and shows how long each stage typically takes. It also shows the relative importance of each stage to the success of the project. Eighty percent of the importance to success comes from finding a suitable problem to address, defining what success looks like in the form of a solution, and, most critical of all, implementing the solution. If the final results are not implemented, it is impossible for any project to be successful. On the other hand, mining—preparation, surveying, and modeling—traditionally takes most of the time in any project. However, after the importance of actually implementing the result, the two most important contributors to success are solving an appropriate problem and preparing the data. While implementing the result is of the first importance to success, it is almost invariably outside the scope of the data exploration project itself. As such, implementation usually requires organizational or procedural changes inside an organization, which is well outside the scope of this discussion. Nonetheless, implementation is critical, since without implementing the results there can be no success.


Data exploration project

	Time to complete (percent of total)		Importance to success (percent of total)	
1. Exploring the problem	10	20	15	80
2. Exploring the solution	9		14	
3. Implementation specification	1		51	
4. Data mining		80		20
a. Data preparation	60		15	
b. Data surveying	15		3	
c. Data modeling	5		2	

Figure 1.1 Stages of a data exploration project showing importance and duration of each stage.

1.1.1 Stage 1: Exploring the Problem Space

This is a critical place to start. It is also the place that, without question, is the source of most of the misunderstandings and unrealistic expectations from data mining. Quite aside from the fact that the terms “data exploration” and “data mining” are (incorrectly) used interchangeably, data mining has been described as “a worm that crawls through your data and finds golden nuggets.” It has also been described as “a method of automatically extracting unexpected hidden patterns from data." It is hard to see any analogous connection between either data exploration or data mining and metaphorical worms. As for automatically extracting hidden and unexpected patterns, there is some analogous truth to that statement. The real problem is that it gives no flavor for what goes into finding those hidden patterns, why you would look for them, nor any idea of how to practically use them when they are found. As a statement, it makes data mining appear to exist in a world where such things happen by themselves. This leads to "the expectation of magic" from data mining: wave a magic wand over the data and produce answers to questions you didn't even know you had!

Without question, effective data exploration provides a disciplined approach to identifying business problems and gaining an understanding of data to help solve them. Absolutely no magic used, guaranteed.

Identifying Problems

The data exploration process starts by *identifying the right problems to solve*. This is not as easy as it seems. In one instance, a major telecommunications company insisted that they had already identified their problem. They were quite certain that the problem was *churn*. They listened patiently to the explanation of the data exploration methodology, and then, deciding it was irrelevant in this case (since they were sure they already understood the problem), requested a model to predict churn. The requested churn model was duly built, and most effective it was too. The company's previous methods yielded about a 50% accurate prediction model. The new model raised the accuracy of the churn predictions to more than 80%. Based on this result, they developed a major marketing campaign to reduce churn in their customer base. The company spent vast amounts of money targeting at-risk customers with very little impact on churn and a disastrous impact on profitability. (Predicting churn and stopping it are different things entirely. For instance, the amazing discovery was made that unemployed people over 80 years old had a most regrettable tendency to churn. They died, and no incentive program has much impact on death!)

Fortunately they were persuaded by the apparent success, at least of the predictive model, to continue with the project. After going through the full data exploration process, they ultimately determined that the problem that should have been addressed was improving return from underperforming market segments. When appropriate models were built, the company was able to create highly successful programs to improve the value that their customer base yielded to them, instead of fighting the apparent dragon of churn. The value of finding and solving the appropriate problem was worth literally millions of dollars, and the difference between profit and loss, to this company.

Precise Problem Definition

So how is an appropriate problem discovered? There is a methodology for doing just this.

Start by defining problems in a precise way. Consider, for a moment, how people generally identify problems. Usually they meet, individually or in groups, and discuss what they feel to be precise descriptions of problems; on close examination, however, they are really general statements. These general statements need to be analyzed into smaller components that can, in principle at least, be answered by examining data. In one such discussion with a manufacturer who was concerned with productivity on the assembly line, the problem was expressed as, "I really need a model of the Monday and Friday failure rates so we can put a stop to them!" The owner of this problem genuinely thought this was a precise problem description.

Eventually, this general statement was broken down into quite a large number of applicable problems and, in this particular case, led to some fairly sophisticated models reflecting which employees best fit which assembly line profiles, and for which shifts, and so on. While exploring the problem, it was necessary to define additional issues, such as what constituted a failure; how failure was detected or measured; why the Monday and Friday failure rates were significant; why these failure rates were seen as a problem; was this in fact a quality problem or a problem with fluctuation of error rates; what problem components needed to be looked at (equipment, personnel, environmental); and much more. By the end of the problem space exploration, many more components and dimensions of the problem were explored and revealed than the company had originally perceived.

It has been said that a clear statement of a problem is half the battle. It is, and it points directly to the solution needed. That is what exploring the problem space in a rigorous manner achieves. Usually (and this was the case with the manufacturer), the exploration itself yields insights without the application of any automated techniques.

Cognitive Maps

Sometimes the problem space is hard to understand. If it seems difficult to gain insight into the structure of the problem, or there seem to be many conflicting details, it may be helpful to structure the problem in some convenient way. One method of structuring a problem space is by using a tool known as a *cognitive map* (Figures 1.2(a) and 1.2(b)). A useful tool for exploring complex problem spaces, a cognitive map is a physical picture of what are perceived as the objects that make up the problem space, together with the interconnections and interactions of the variables of the objects. It will very often show where there are conflicting views of the structure of the problem.

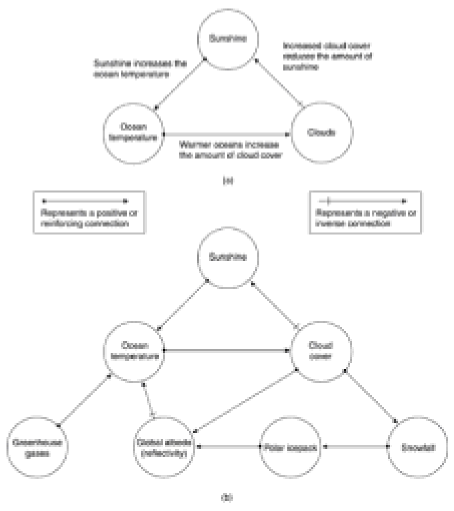

Sunshine
Sunshine increases the ocean temperature
Increased cloud cover reduces the amount of sunshine
Ocean temperature
Clouds
Warmer oceans increase the amount of cloud cover
(a)
Represents a positive or reinforcing connection
Represents a negative or inverse connection
Sunshine
Ocean temperature
Cloud cover
Greenhouse gases
Global albedo (reflectivity)
Polar icepack
Snowfall
(b)

Figure 1.2 Cognitive maps: simple (a) and complex (b).

Figure 1.2(a) shows a simple cognitive map expressing the perceived relationships among the amount of sunshine, the ocean temperature, and the level of cloud cover. Figure 1.2(b) shows a somewhat more complex cognitive map. Cloud cover and global albedo are significant in this view because they have a high number of connections, and both introduce negative feedback relationships. Greenhouse gases don't seem to be closely coupled. A more sophisticated cognitive map may introduce numerical weightings to indicate the strength of connections. Understanding the implications of the more complex relationships in larger cognitive maps benefits greatly from computer simulation.

Note that what is important is not to resolve or remove these conflicting views, but to understand that they are there and exactly in which parts of the problem they occur. They may in fact represent valid interpretations of different views of a situation held by different problem owners.

Ambiguity Resolution

While the problems are being uncovered, discovered, and clarified, it is important to use techniques of *ambiguity resolution*. While ambiguity resolution covers a wide range of areas and techniques, its fundamental purpose is to assure that the mental image of the problem in the problem owner's mind—a mental image replete with many associated assumptions—is clearly communicated to, and understood by, the problem solver—most specifically that the associated assumptions are brought out and made clear. Ambiguity resolution serves to ensure that where there are alternative interpretations, any assumptions are explicated. For a detailed treatment of ambiguity resolution, see the excellent *Exploring Requirements: Quality Before Design* by Grause and Weinberg. (See Further Reading.)

Pairwise Ranking and Building the Problem Matrix

Exploring the problem space, depending on the scope of the project, yields anything from tens to hundreds of possible problems. Something must be done to deal with these as there may be too many to solve, given the resources available. We need some way of deciding which problems are the most useful to tackle, and which promise the highest yields for the time and resources invested.

Drawing on work done in the fields of decision theory and econometrics, it is possible to use a rationale that does in fact give consistent and reliable answers as to the most appropriate and effective problems to solve: the *pairwise ranking*. Figure 1.3 illustrates the concept. Generating pairwise rankings is an extremely powerful technique for reducing comparative selections. Surprisingly, pairwise rankings will probably give different results than an intuitive ranking of a list. Here is a simple technique that you can use to experiment.

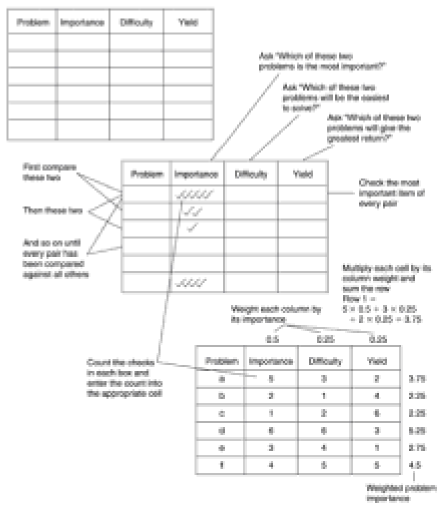

Problem
Importance
Difficulty
Yield
Ask "Which of these two problems is the most important?"
Ask "Which of these two problems will be the easiest to solve?"
Ask "Which of these two problems will give the greatest return?"
First compare these two
Then these two
And so on until every pair has been compared against all others
Check the most important item of every pair
Multiply each cell by its column weight and sum the row
Row 1 =
5 × 0.5 + 3 × 0.25
+ 2 × 0.25 = 3.75
Weigh each column by its importance
0.5
0.25
0.25
Count the checks in each box and enter the count into the appropriate cell
a 5 3 2 3.75
b 2 1 4 2.25
c 1 2 6 2.25
d 6 6 3 5.25
e 3 4 1 2.75
f 4 5 5 4.5
Weighted problem importance

Figure 1.3 Pairwise ranking method. This method is illustrative only. In practice, using a spreadsheet or a decision support software package would ease the comparison.

Create a four-column matrix. In column 1, list 10–20 books, films, operas, sports teams, or whatever subject is of interest to you. Start at the top of the list and pick your best, favorite, or highest choice, putting a “1” against it in column 2. Then choose your second favorite and enter “2” in column 2 and so on until there is a number against each choice in that column. This is an *intuitive ranking*.

Now start again at the top of the list in column 1. This time, choose which is the preferable pick between items 1 and 2, then 1 and 3, then 1 and 4, and so on to the last item. Then make your preferable picks between those labeled 2 and 3, 2 and 4, and so on. For each pair, put a check mark in column 3 against the top pick. When you have finished this, add up the check marks for each preferred pick and put the total in column 4. When you have finished, column 4 cells will contain 1, 2, 3, 4, and so on, check marks. If there is a tie in any of your choices, simply make a head-to-head comparison of the tied items. In column 4, enter a “1” for the row with the most check marks, a “2” for the second-highest number, and so on. This fourth column represents your pairwise ranking.

There are many, well-founded psychological studies that show, among other things, that a human can make judgments about 7 (plus or minus 2) items at the same time. Thus an intuitive ranking with more than 10 items will tend to be inconsistent. However, by making a comparison of each pair, you will generate a consistent ranking that gives a highly reliable indicator of where each item ranks. Look at the results. Are your listings different? Which is the most persuasive listing of your actual preferences—the intuitive ranking or the pairwise ranking?

Using the principle of the comparison technique described above with identified problems forms the problem space matrix (PSM). An actual PSM uses more than a single column of judgment rankings—“Problem,” “Importance,” “Difficulty,” “Yield,” and “Final Rank,” for example. Remember that the underlying ranking for each column is always based on the pairwise comparison method described above.

Where there are many problem owners, that is, a number of people involved in describing and evaluating the problem, the PSM uses a consensus ranking made from the individual rankings for “Importance,” “Difficulty,” and “Yield.” For the column “Importance,” a ranking is made to answer the question “Which of these two problems do you think is the most important?” The column “Difficulty” ranks the question “Given the availability of data, resources, and time, which of these two problems will be the easier to solve?” Similarly for “Yield,” the question is “If you had a solution for each of these two problems, which is likely to yield the most value to the company?” If there are special considerations in a particular application, an additional column or columns might be used to rank those considerations. For instance, you may have other columns that rank internal political considerations, regulatory issues, and so on.

The “Final Rank” is a weighted scoring from the columns “Importance,” “Difficulty,” and “Yield,” made by assigning a weight to each of these factors. The total of the weights must add up to 1. If there are no additional columns, good preliminary weightings are

Importance	0.5
Difficulty	0.25
Yield	0.25

This is because “Importance” is a subjective weighting that includes both “Difficulty” and “Yield.” The three are included for balance. However, discussion with the problem owners may indicate that they feel “Yield,” for example, is more important since benefit to the company outweighs the difficulty of solving the problem. Or it may be that time is a critical factor in providing results and needs to be included as a weighted factor. (Such a column might hold the ranks for the question, "Which of these two will be the quickest to solve?")

The final ranking is made in two stages. First, multiplying the value in each column by the weighting for that column creates a score. For this reason it is critical to construct the questions for each column so that the "best" answer is always the highest or the lowest number in all columns. Whichever method you chose, this ranks the scores from highest to lowest (or lowest to highest as appropriate).

If completed as described, this matrix represents the best selection and optimum ranking of the problems to solve that can be made. Note that this may not be the absolute best selection and ranking—just the best that can be made with the resources and judgments available to you.

Generating real-world matrixes can become fairly complex, especially if there are many problems and several problem owners. Making a full pairwise comparison of a real-world matrix having many problems is usually not possible due to the number of comparisons involved. For sizeable problems there are a number of ways of dealing with this complexity. A good primer on problem exploration techniques is *The Thinker's Toolkit* by Morgan D. Jones (see Further Reading). This mainly focuses on decision making, but several techniques are directly applicable to problem exploration.

Automated help with the problem ranking process is fairly easy to find. Any modern computer spreadsheet program can help with the rankings, and several decision support software packages also offer help. However, new decision support programs are constantly appearing, and existing ones are being improved and modified, so that any list given here is likely to quickly become out of date. As with most other areas of computer software, this area is constantly changing. There are several commercial products in this area, although many suitable programs are available as shareware. A search of the Internet using the key words "decision support" reveals a tremendous selection. It is probably more important that you find a product and method that you feel comfortable with, and will actually use, than it is to focus on the particular technical merits of individual approaches and products.

1.1.2 Stage 2: Exploring the Solution Space

After discovering the best mix of precisely defined problems to solve, and ranking them appropriately, does the miner now set out to solve them? Not quite. Before trying to find a solution, it helps to know what one looks like!

Typical outputs from simple data exploration projects include a selection from some or all of the following: reports, charts, graphs, program code, listings of records, and algebraic formulae, among others. What is needed is to specify as clearly and completely as possible what output is desired (Figure 1.4). Usually, many of the problems share a common solution.

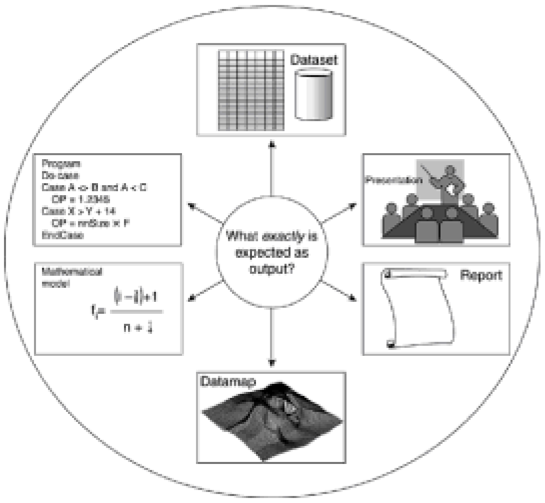


Figure 1.4 Exactly how does the output fit into the solution space?

For example, if there are a range of problems concerning fraudulent activity in branch offices, the questions to ask may include: What are the driving factors? Where is the easiest point in the system to detect it? What are the most cost-effective measures to stop it? Which patterns of activity are most indicative of fraud? And so on. In this case, the solution (in data exploration terms) will be in the form of a written report, which would include a listing of each problem, proposed solutions, and their associated rankings.

If, on the other hand, we were trying to detect fraudulent transactions of some sort, then a solution might be stated as “a computer model capable of running on a server and measuring 700,000 transactions per minute, scoring each with a probability level that this is fraudulent activity and another score for confidence in the prediction, routing any transactions above a specific threshold to an operator for manual intervention.”

It cannot be emphasized enough that in the Solution Space Exploration stage, the specified solution must be precise and complete enough that it actually specifies a real-world, implementable solution to solve the problem. Keep in mind that this specification is needed for the data exploration process, not data mining. Data mining produces a more limited result, but still one that has to fit into the overall need.

A company involved in asset management of loan portfolios thought that they had made a precise solution statement by explaining that they wanted a ranking for each portfolio such that a rational judgment could be made as to the predicted performance. This sounds like a specific objective; however, a specific objective is *not* a solution specification.

The kind of statement that was needed was something more like “a computer program to run on a Windows NT workstation terminal that can be used by trained operators and that scores portfolios and presents the score as a bar graph . . .” and so on. The point here is that the output of the data exploration process needed to be made specific enough so that the solution could be practically implemented. Without such a specific target to aim at, it is impossible to mine data for the needed model that fits with the business solution. (In reality, the target must be expected to move as a project continues, of course. But the target is still needed. If you don’t know what you’re aiming at, it’s hard to know if you’ve hit it!)

Another company wanted a model to improve the response to their mailed catalogs. Discovering what they really needed was harder than creating the model. Was a list of names and addresses needed? Simply a list of account numbers? Mailing labels perhaps? How many? How was response to be measured? How was the result to be used? It may seem unlikely, but the company had no clear definition of a deliverable from the whole process. They wanted things to improve in general, but would not be pinned down to specific objectives. It was even hard to determine if they wanted to maximize the number of responses for a given mailing, or to maximize the value per response. (In fact, it turned out—after the project was over—that what they really wanted to do was to optimize the value per page of the catalog. Much more effective models could have been produced if that had been known in advance! As it was, no clear objective was defined, so the models that were built addressed another problem they didn’t really care about.)

The problems and difficulties are compounded enormously by not specifying what success looks like in practice.

For both the problem and the solution exploration it is important to apply ambiguity resolution. This is the technique that is used to test that what was conceived as a problem is what was actually addressed. It also tests that what is presented as a solution is what was really wanted by the problem owners. Ambiguity resolution techniques seek to pinpoint any misunderstandings in communication, reveal underlying assumptions, and ensure that key points and issues are understood by everyone involved. Removing ambiguity is a crucial element in providing real-world data exploration.

1.1.3 Stage 3: Specifying the Implementation Method

At this point, problems are generated and ranked, solutions specified, expectations and specifications matched, and hidden assumptions revealed.

However, no data exploration project is conducted just to discover new insights. The point is to apply the results in a way that increases profitability, improves performance, improves quality, increases customer satisfaction, reduces waste, decreases fraud, or meets some other specified business goal. This involves what is often the hardest part of any successful data exploration project—modifying the behavior of an organization.

In order to be successful, it is not enough to simply specify the results. Very successful and potentially valuable projects have died because they were never seriously implemented. Unless everyone relevant is involved in supporting the project, it may not be easy to gain maximum benefit from the work, time, and resources involved.

Implementation specification is the final step in detailing how the various solutions to chosen problems are actually going to be applied in practice. This details the final form of the deliverables for the project. The specification needs to be a complete practical definition of the solution (what problem it addresses, what form it takes, what value it delivers, who is expected to use it, how it is produced, limitations and expectations, how long it is expected to last) and to specify five of the "six w's": who, how, what, when, and where (why is already covered in the problem specification).

It is critical at this point to get the "buy-in" of both "problem owners" and "problem holders." The problem owners are those who experience the actual problem. The problem holders are those who control the resources that allow the solution to be implemented. The resources may be in one or more of various forms: money, personnel, time, or corporate policy, to name only a few. To be effective, the defined solution *must* be perceived to be cost-effective and appropriate by the problem holder. Without the necessary commitment there is little point in moving further with the project.

1.1.4 Stage 4: Mining the Data

Geological mining (coal, gold, etc.) is not carried out by simply applying mining equipment to a lump of geology. Enormous preparation is made first. Large searches are made for terrain that is geologically likely to hold whatever is to be mined. When a likely area is discovered, detailed surveys are made to pinpoint the most likely location of the desired ore. Test mines are dug before the full project is undertaken; ore is assayed to determine its fineness. Only when all of the preparation is complete, and the outcome of the effort is a foregone conclusion, is the full-scale mining operation undertaken.

So it should be with mining data. Actually mining the data is a multistep process. The first step, preparation, is a two-way street in which both the miner is prepared and the data is prepared. It is not, and cannot be, a fully autonomous process since the objective is to prepare the miner just as much as it is to prepare the data. Much of the actual data preparation part of this first and very important step can be automated, but miner interaction with the data remains essential. Following preparation, the survey. For effective mining this too is most important. It is during the survey that the miner determines if the data is adequate—a small statement with large ramifications, and more fully explored in Chapter 11.

When the preparation and survey are complete, actually modeling the data becomes a relatively small part of the overall mining effort. The discovery and insight part of mining comes during preparation and surveying. Models are made only to capture the insights and discoveries, not to make them. The models are built only when the outcome is a foregone conclusion.

Preparing the Data for Modeling

Why prepare data? Why not just take it as it comes? The answer is that preparing data also prepares the miner so that when using prepared data, the miner produces better models, faster.

Activities that today come under the umbrella of the phrase “data mining” actually have been used for many years. During that time a lot of effort has been put forth to apply a wide variety of techniques to data sets of many different types, building both predictive and inferential models. Many new techniques for modeling have been developed over that time, such as evolution programming. In that same time other modeling tools, such as neural networks, have changed and improved out of all recognition in their capabilities. However, what has not changed at all, and what is almost a law of nature, is GIGO—garbage in, garbage out. Keeping that now-popular aphorism firmly in mind leads logically to the observation that good data is a prerequisite for producing effective models of any type.

Unfortunately, there is no such thing as a universal garbage detector! There are, however, a number of different types of problems that constantly recur when attempting to use data sets for building the types of models useful in solving business problems. The source, range, and type of these problems, the “GI” in GIGO, are explored in detail starting in Chapter 4. Fortunately, there are a number of these problems that are more or less easily remedied. Some remedies can be applied automatically, while others require some choices to be made by the miner, but the actual remedial action for a wide range of problems is fairly well established. Some of the corrective techniques are based on theoretical considerations, while others are rules of thumb based on experience. The difficulty is in application.

While methodologies and practices that are appropriate for making models using various algorithms have become established, there are no similar methodologies or practices for using data preparation techniques. Yet good data preparation is essential to practical modeling in the real world.

The data preparation tools on the accompanying CD-ROM started as a collection of practical tools and techniques developed from experience while trying to “fix” data to build decent models. As they were developed, some of them were used over and over on a wide variety of modeling projects. Their whole purpose was to help the miner produce better models, faster than can be done with unprepared data, and thus assure that the final user received cost-effective value. This set of practical tools, in the form of a computer program, and a technique of applying the program, must be used together to get their maximum benefit, and both are equally important. The accompanying demonstration software actually carries out the data manipulations necessary for data preparation. The technique is described as the book progresses. Using this technique results in the miner understanding the data in ways that modeling alone cannot reveal. Data preparation is about more than just readying the data for application of modeling tools; it is also about gaining the necessary insights to build the best possible models to solve business problems with the data at hand.

One objective of data preparation is to end with a prepared data set that is of maximum use for modeling, in which the natural order of the data is least disturbed, yet that is best enhanced for the particular purposes of the miner. As will become apparent, this is an almost totally different sort of data preparation activity than is used, say, in preparing data for data warehousing. The objective, techniques, and results used to prepare data when mining are wholly different.

The Prepared Information Environment (PIE)

A second objective of data preparation is to produce the Prepared Information Environment (PIE). The PIE is an active computer program that “envelops” the modeling tools to protect them from damaged and distorted data. The purpose and use of this very important tool in modeling is more fully described in Chapter 3. Its main purposes are to protect the modeling tool from damaged data and to maximally expose the data set’s information content to the modeling tool. One component, the Prepared Information Environment Input module (PIE-I) does this by acting as an intelligent buffer between the incoming data, manipulating the training, testing, and execution data sets before the modeling tool sees the data. Since even the output prediction variables are prepared by the PIE-I, any model predictions are predictions of the prepared values. The predictions of prepared values need to be converted back into their unmodified form, which is done by the Prepared Information Environment Output module (PIE-O).

A clear distinction has to be made between the training and testing data set, and the execution data set. On some occasions the training, testing, and execution data sets may all be drawn from the same “pool” of data that has been assembled prior to modeling. On other occasions the execution data may be impossible to obtain at the time of modeling. In the case of industrial modeling, for instance, it may be required to build a model that predicts likely time to failure for a manufactured component based on the manufacturing information collected as it is manufactured. The model, when built, validated, and verified, will be placed in service to monitor future production. However, at the time the model is being built, using already collected data, the data on next month’s or next year’s production is impossible to acquire. The same is true for stock market data, or insurance claims data, for instance, where the model is built on data already collected, but applied to future stock movements or insurance claims.

In the continuously learning model described in the Supplemental Material section at the end of this chapter, the actual data to be used for mailing was not available until it was acquired specifically for the mailing. The model built to predict likely responders to the mailing solicitation was built before the mailing data was available. The initial mailing response model was built on information resulting from previous mailings. It was known that the characteristics of the variables (described in Chapter 2) for the training data that was available were similar to those in the actual mailing data set—even though the precise data set for the mailing had not been selected.

In general, preparation of the data for modeling requires various adjustments to be made to the data prior to modeling. The model produced, therefore, is built using adjusted, prepared data. Some mechanism is needed to ensure that any new data, especially data to which the model is to be applied, is also adjusted similarly to the training data set. If this is not done, the model will be of no value as it won't work with raw data, only with data similarly prepared to that used for training.

It is the PIE that accomplishes this transformation. It may perform many other useful tasks as well, such as novelty detection, which measures how similar the current data is to that which was used for training. The various tasks and measures are discussed in detail in various parts of the book. However, a principal purpose of the PIE is to transform previously unencountered data into the form that was initially used for modeling. (This is done by the PIE-I.)

Notable too is that a predictive model's output variable(s), the one(s) that the model is trying to predict or explain, will also have been in its adjusted format, since the model was trying to predict or explain it in a prepared data set. The PIE also will transform the prepared and normalized model output into the experiential range encountered in the data before preparation—in other words, it undoes the transformations for the predicted values to get back the original range and type of values for the predicted output. (This is accomplished by the PIE-O.)

While the PIE adds great value in many other areas, its main function is allowing models trained on prepared data to be used on other data sets.

For one-shot modeling, where all of the data to be modeled and explained is present, the PIE's role is more limited. It is simply to produce a file of prepared data that is used to build the model. Since the whole of the data is present, the role of the PIE is limited to translating the output variables from the predicted adjusted value to their predicted actual expected value.

Thus, the expected output from the data preparation process is threefold: first, a prepared miner, second, a prepared data set, and third, the PIE, which will allow the trained model to be applied to other data sets and also performs many valuable ancillary functions. The PIE provides an envelope around the model, both at training and execution time, to insulate the model from the raw data problems that data preparation corrects.

Surveying the Data

Surveying the prepared data is a very important aspect of mining. It focuses on answering three questions: What’s in the data set? Can I get my questions answered? Where are the danger areas? These questions may seem similar to those posed by modeling, but there is a significant difference.

Using the survey to look at the data set is different in nature from the way modeling approaches the data. Modeling optimizes the answer for some specific and particular problem. Finding the problem or problems that are most appropriate is what the first stage of data exploration is all about. Providing those answers is the role of the modeling stage of data mining. The survey, however, looks at the general structure of the data and reports whether or not there is a useful amount of information enfolded in the data set about various areas. The survey is not really concerned with exactly what that information might be—that is the province of modeling. A most particular purpose of the survey is to find out if the answer to the problem that is to be modeled is actually in the data prior to investing much time, money, and resource in building the model.

The survey looks at all areas of the data set equally to make its estimate of what information is enfolded in the data. This affects data preparation in that such a survey may allow the data to be restructured in some way prior to modeling, so that it better addresses the problem to be modeled.

In a rich data set the survey will yield a vast amount of insight into general relationships and patterns that are in the data. It does not try to explicate them or evaluate them, but it does show the structure of the data. Modeling explores the fine structure; survey reveals the broad structure.

Given the latter fact, the search for danger areas is easier. An example of a danger area is where some bias is detectable in the data, or where there is particular sparsity of data and yet variables are rapidly changing in value. In these areas where the relationship is changing rapidly, and the data do not describe the area well, any model’s performance should be suspect. Perhaps the survey will reveal that the range in which the model predictions will be important is not well covered.

All of these areas are explored in much more detail in Chapter 11, although the perspective there is mainly on how the information provided by the survey can be used for better preparing the data. However, the essence of the data survey is to build an overall map of the territory before committing to a detailed exploration. Metaphorically speaking, it is of immense use to know where the major mountain ranges, rivers, lakes, and deserts are before setting off on a hiking expedition. It is still necessary to make the detailed exploration to find out what is present, but the map is the guide to the territory. Vacationers, paleontologists, and archeologists all use the same basic topographic map to find their way to sites that interest them. Their detailed explorations are very different and may lead them to each make changes to the local, or fine, structure of their individual maps. However, without the general map it would be impossible for them to find their way to likely places for a good vacation site, dinosaur dig, or an ancient city. The general map—the data survey—shows the way.

Modeling the Data

When considering data mining, even some of the largest companies in the U.S. have asked questions whose underlying meaning was, "What sort of problems can I solve with a neural net (or other specific technique)?" This is exactly analogous to going to an architect and asking, "What sort of buildings can I build with this power saw (or other tool of your choice)?" The first question is not always immediately seen as irrelevant, whereas the second is.

Some companies seem to have the impression that in order to produce effective models, knowledge of the data and the problem are not really required, but that the tools will do all the work. Where this myth came from is hard to imagine. It is so far from the truth that it would be funny if it were not for the fact that major projects have failed entirely due to ignorance on the part of the miner. Not that the miner was always at fault. If ordered to "find out what is in this data," an employee has little option but to do *something*. No one who expected to achieve anything useful would approach a lump of unknown substance, put on a blindfold, and whack at it with whatever tool happened to be at hand. Why this is thought possible with data mining tools is difficult to say!

Unfortunately, focusing on the data mining modeling tools as the primary approach to a problem often leads to the problem being formulated in inappropriate ways. Significantly, there may be times when data mining tools are not the right ones for the job. It is worth commenting on the types of questions that are particularly well addressed with a data-mined model. These are the questions of the "How do I . . . ?" and "Why is it that . . . ?" sort.

For instance, if your questions are those that will result in summaries, such as "What were sales in the Boston branch in June?" or "What was the breakdown by shift and product of testing failures for the last six weeks?" then these are questions that are well addressed by on-line analytical processing (OLAP) tools and probably do not need data mining. If however, the questions are more hypothesis driven, such as "What are the factors driving fraudulent usage in the Eastern sector?" or "What should be my target markets and what is the best feature mix in the marketing campaign to capture the most new customers?" then data mining, used in the context of a data exploration process, is the best tool for the job.

1.1.5 Exploration: Mining and Modeling

This brief look at the process of data exploration emphasizes that none of the pieces stands alone. Problems need to be identified, which leads to identifying potential solutions, which leads to finding and preparing suitable data that is then surveyed and finally modeled. Each part has an inextricable relationship to the other parts. Modeling, the types of tools and the types of models made, also has a very close relationship with how data is best prepared, and before leaving this introduction, a first look at modeling is helpful to set the frame of reference for what follows.

1.2 Data Mining, Modeling, and Modeling Tools

One major purpose for preparing data is so that mining can discover models. But what *is* modeling? In actual fact, what is being attempted is very simple. The ways of doing it may not be so simple, but the actual intent is quite straightforward.

It is assumed that a data set, either one immediately available or one that is obtainable, might contain information that would be of interest if we could only understand what was in it. Therein lies the rub. Since we don't understand the information that is in the data just by looking at it, some tool is needed that will turn the information enfolded in the data set into a form that is understandable. That's all. That's the modeling part of data mining—a process for transforming information enfolded in data into a form amenable to human cognition.

1.2.1 Ten Golden Rules

As discussed earlier in this chapter, the data exploration process helps build a framework for data mining so that appropriate tools are applied to appropriate data that is appropriately prepared to solve key business problems and deliver required solutions. This framework, or one similar to it, is critical to helping miners get the best results and return from their data mining projects. In addition to this framework, it may be helpful to keep in mind the 10 Golden Rules for Building Models:

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, etc.).

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 a small change in inputs).

10. Define uncertainty in the model (critical areas and ranges in the data set where the model produces low confidence predictions/insights).

In other words, rules 1–3 recapitulate the first three stages of the data exploration process. Rule 4 captures the insight that if you know what you're doing, success is more likely. Rule 5 advises to find the best tool for the job, not just a job you can do with the tool. Rule 6 says don't *just* assume, tell someone. Rule 7 says to keep trying different things until the model seems as good as it's going to get. Rule 8 means KISS (Keep It Sufficiently Simple). Rules 9 and 10 mean state what works, what doesn't, and where you're not sure.

To make a model of data is to express the relationships that change in one variable, or set of variables, has on another variable or set of variables. Another way of looking at it is that regardless of the type of model, the aim is to express, in symbolic terms, the shape of how one variable, or set of variables, changes when another variable or set of variables changes, and to obtain some information about the reliability of this relationship. The final expression of the relationship(s) can take a number of forms, but the most common are charts and graphs, mathematical equations, and computer programs. Also, different things can be done with each of these models depending on the need. *Passive models* usually express relationships or associations found in data sets. These may take the form of the charts, graphs, and mathematical models previously mentioned. *Active models* take sample inputs and give back predictions of the expected outputs.

Although models can be built to accomplish many different things, the usual objective in data mining is to produce either predictive or explanatory (also known as inferential) models.

1.2.2 Introducing Modeling Tools

There are a considerable variety of data mining modeling tools available. A brief review of some currently popular techniques is included in Chapter 12, although the main focus of that chapter is the effect of using prepared data with different modeling techniques. Modeling tools extend analysis into producing models of several different types, some mentioned above and others examined in more detail below.

Data mining modeling tools are almost uniformly regarded as software programs to be run on a computer and that perform various translations and manipulations on data sets. These are indeed the tools themselves, but it does rather leave out the expertise and domain knowledge needed to successfully use them. In any case, there are a variety of support tools that are also required in addition to the so-called data mining tools, such as databases and data warehouses, to name only two obvious examples. Quite often the results of mining are used within a complex and sophisticated decision support system. Close scrutiny often makes problematic a sharp demarcation between the actual data mining tools themselves and other supporting tools. For instance, is presenting the results in, say, an OLAP-type tool part of data mining, or is it some other activity?

In any case, since data mining is the discovery of patterns useful in a business situation, the venerable tools of statistical analysis may be of great use and value. The demarcation between statistical analysis and data mining is becoming somewhat difficult to discern from any but a philosophical perspective. There are, however, some clear pointers that allow determination of which activity is under way, although the exact tool being used may not be indicative. (This topic is also revisited in Chapter 12.)

Philosophically and historically, statistical analysis has been oriented toward verifying and validating hypotheses. These inquiries, at least recently, have been scientifically oriented. Some hypothesis is proposed, evidence gathered, and the question is put to the evidence whether the hypothesis can reasonably be accepted or not. Statistical reasoning is concerned with logical justification, and, like any formal system, not with the importance or impact of the result. This means that, in an extreme case, it is quite possible to create a result that is statistically significant—and utterly meaningless.

It is fascinating to realize that, originally, the roots of statistical analysis and data mining lie in the gaming halls of Europe. In some ways, data mining follows this heritage more closely than statistical analysis. Instead of an experimenter devising some hypothesis and testing it against evidence, data mining turns the operation around. Within the parameters of the data exploration process, data mining approaches a collection of data and asks, “What are all the hypotheses that this data supports?” There is a large conceptual difference here. Many of the hypotheses produced by data mining will not be very meaningful, and some will be almost totally disconnected from any use or value. Most, however, will be more or less useful. This means that with data mining, the inquirer has a fairly comprehensive set of ideas, connections, influences, and so on. The job then is to make sense of, and find use for, them. Statistical analysis required the inquirer first to devise the ideas, connections, and influences to test.

There is an area of statistical analysis called “exploratory data analysis” that approaches the previous distinction, so another signpost for demarcation is useful. Statistical analysis has largely used tools that enable the human mind to visualize and quantify the relationships existing within data in order to use its formidable pattern-seeking capabilities. This has worked well in the past. Today, the sheer volume of data, in numbers of data sets, let alone quantity of data, is beyond the ability of humans to sift for meaning. So, automated solutions have been called into play. These automated solutions draw largely on techniques developed in a discipline known as “machine learning.” In essence, these are various techniques by which computerized algorithms can, to a greater or lesser degree, learn which patterns actually do exist in data sets. They are not by any means as capable as a trained human mind, educated in the knowledge domain, would be. They are, however, formidably fast (compared to humans), tireless, consistent, and error-free for a particular class of errors. They are error-free in the sense that, once validated that they are indeed performing accurately, the output is consistent. Judgments about what the outputs mean remain firmly in the human domain. That is to say, while decisions as to particular actions to be taken under given circumstances can be programmed algorithmically, humans had to either explicitly program such switch points or permit the program to train and learn them. No amount of artificial intelligence reaches the level of sophistication represented by even human stupidity! In fact, appearances to the contrary, computer programs still cannot make self-motivated, intentional decisions.

Regardless of their source and how they are used (or misused), the function and purpose of modeling tools is actually very straightforward. It is to transform any of the required knowledge enfolded in a particular data set into a form useful to, or comprehensible by, humans. It may be both useful and comprehensible, but this is not necessarily so.

In marketing applications, for instance, models often have to be created where comprehensibility is not an issue. The marketing manager simply wants a model that delivers more, or more valuable, leads, customers, or orders. Why such a model works is not an issue, at least not until someone asks, “Why does that market segment produce better results?” A specific instance of this occurred with a company concerned with providing college students with funding to attend college. It had long been their practice to mail solicitations to people they felt would be appropriate candidates somewhat before the end of the school year, assuming that was the time when people were considering which college to attend and applying for financial aid. In order to investigate this further, marketing response models were made with a variety of their assumptions altered for a small subset of the mailing. Analysis of the results indicated strongly that mailing *immediately following the end of the school year* showed a stronger response. This seemed so counterintuitive to the marketers that they found it hard to accept and immediately asked why this was so. At this point a variety of different models drawing on different data sets had to be built to explore the question. (It turned out that, for the population segment for which this response was valid, colleges were explored first and the earlier solicitation had been thrown away as unwanted “junk mail” by the time financial aid for school was being considered. Early mailing meant that they weren’t in the running for that segment of the population.)

This leads to consideration of the types of models that are used.

1.2.3 Types of Models

After conducting a data exploration project and stipulating the problem set, solution set, and implementation strategies, preparing the data, surveying it, then selecting algorithms for the purpose, there still remains the process of building models and delivering the results.

First, a brief observation about modeling in general. A misconception of inexperienced modelers is that modeling is a linear process. This imagined linear process can be shown as

1. State the problem.

2. Choose the tool.

3. Get some data.

4. Make a model.

5. Apply the model.

6. Evaluate results.

On the contrary, *building any model should be a continuous process incorporating several feedback loops and considerable interaction among the components*. Figure 1.5 gives a conceptual overview of such a process. At each stage there are various checks to ensure that the model is in fact meeting the required objectives. It is a dynamic process in which various iterations converge toward the best solution. There is naturally a fair amount of human interaction and involvement in guiding the search for an optimum solution.

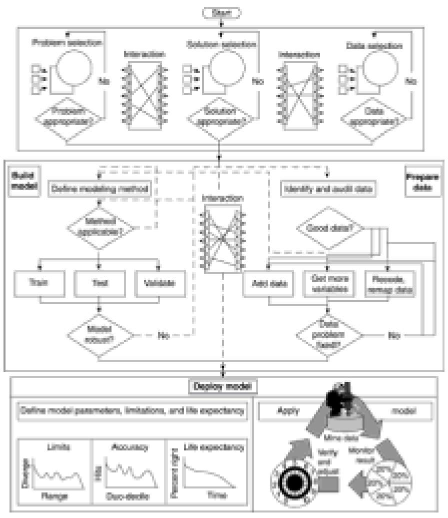


Figure 1.5 Model building outline.

The various types of models were briefly touched on previously, but discussing them together helps clarify their similarities and differences.

1.2.4 Active and Passive Models

Basically, active models actually respond in some way, whereas passive models are nonreactive.

Passive models generally answer questions and show relationships using charts, graphs, words, mathematical formulae, and so on. The example above describing why some college applicants respond better to late mailings was a passive model. It explicated in an understandable way the “why” of the relationship. It was “actionable information” in that, as a result, better marketing plans can be made and characteristics of the targeted population can be identified. A model is passive in that it does not take inputs, give outputs, change, react, or modify anything as it is used. It is simply a fixed expression, such as a statement on a piece of paper.

On the other hand, an *active model* performs one or more activities. An active model built for the college loan application, for instance, might take a specific input file and score or categorize it as to the type of response to be expected for each instance (record) in the file.

The differentiation between active and passive models may be critical to the modeler and to the application. It will have a considerable effect on which data is selected for modeling. However, when preparing the selected data set, the difference between active and passive model requirements has little if any impact on how the data is further prepared for modeling.

1.2.5 Explanatory and Predictive Models

Here, of course, the one type of model is created to explain some facet of the data, while the other is designed to predict, classify, or otherwise interpret data. These are not synonymous with active and passive.

On occasion, particularly in the arena of industrial automation, the required output from the modeling process is a passive, predictive model. For instance, in a paper mill, where paper is made, the key parts of the process were captured in the shift foreman’s experience. At shift change, the new foreman, who had enormous experience, would make various adjustments based on such measures as the taste of the process (actually tasting the slurry as a means of measuring what was happening in the mixture) at a particular stage. Each foreman knew how to tune the process to produce fine paper. Each foreman knew what was going wrong when indeed things were going wrong, and how to fix them. Yet each foreman’s recipe was different!

The business problem here was that automating such a process seemed impossible. The “rules” for making paper were embedded in the shift foremen’s heads, and extracting a useful set of rules by questioning them, although tried, proved impossible. Instead of studying the experts (shift foremen), the modeling approach was to instrument the paper mill, collect data about the process, analyze the collected data, and model the process. This type of approach is called *automated expertise capture*. This process involves watching and modeling what an expert actually does rather than questioning the expert to create a model.

It took considerable effort, but eventually successful passive predictive models were produced in the form of mathematical statements. These mathematical statements described how the process behaved, and how its behavior changed as conditions changed. To automate the paper-making process, these mathematical statements were turned into a particular sort of programming language called “ladder logic,” which is widely used in programmable logic controllers (PLCs). The passive, explanatory model was used to create the program for the PLCs. It essentially captured the expertise of the foremen and encapsulated it in succinct expressions. These, in turn, were used in machine and process automation.

Without giving detailed examples of each model type (which would properly belong in a book on modeling rather than data preparation), it can be easily seen that it is quite possible to have active-explanatory, passive-explanatory, active-predictive, or passive-predictive models.

Passive-predictive models can be exemplified in the “score cards” used to score certain credit applicants. These are really worksheets that loan officers can use. Modeling techniques have been used to improve the performance of such devices. The output is a fixed, passive worksheet printed on a form. It is, nonetheless, used as a predictive and classification tool by the user. However, note that the *output* of the modeling technique used is passive predictive.

1.2.6 Static and Continuously Learning Models

This is an interesting and important division of modeling that deserves a closer look, particularly the continuously learning models. These hold enormous promise for the application of the sophisticated techniques outlined here.

Static (One-Shot) Models

Static modeling is used to discover relationships or answer questions that are drawn from historical data. In point of fact, all data is historical. (If you have future data about, say, the stock market, please let me know!) However, in this context “historical” data can be taken to mean that the data set from which the model is built is not going to be updated with more current data. Questions leading to the building of static models might be similar to “What factors drive the failure modes in disk drive manufacture?” Once the failure modes in manufacturing are analyzed, corrective action will be applied to fix problems, and that's that. Naturally, the process will be monitored to find out how well the "fix" worked, but the data previously collected is no longer representative of production since changes were made based on the failure mode's driving factors. If any further investigation into the problem is wanted, the historical data cannot address the new issues as systemic changes were made. New data representative of the modified system's performance would have to be collected.

Although pursuing answers to problems requiring static models can be a fairly complex undertaking and draw on the full resources of the tools available, as well as heavily relying on the experience of the modeler, producing the static models themselves is fairly straightforward. An answer in a fixed form, one that does not interact with data to modify itself, is the final solution.

Inexperienced modelers frequently see the static model, or a series of static models, as how modeling should take place. While static models are certainly an appropriate solution for many problems, they are very prevalent even where more extensive techniques are more appropriate. We will now examine one alternative.

Continuously Learning Models

These types of models represent a relatively hands-off, controlling, or discovering process working in dynamic conditions. Constructing a robust continuously learning model draws on resources from outside the domain of data exploration. The core, or enabling, technology, however, is data mining directed by the data exploration process.

Continuous learning is a system using an autonomous model containing a number of internal set points. One natural example of such a system is a human being. We contain many set points that control our behavior, one of which is internal temperature. The internal temperature of a healthy human being is estimated at about 98.6º Fahrenheit. That temperature may be regarded as a set point. Our bodies seek to maintain a constant internal temperature in spite of external assaults. We may be motivated to make a number of internal and external environmental adjustments to, say, keep warm when the external temperature is falling. These include turning up the thermostat, putting on more clothes, shivering, having a hot drink, and possibly a whole host of other activities. All the time we're actively manipulating the environment, both internal and external, to maintain the specific set point for internal temperature. It is exactly this type of behavior that is used in an artificially constructed, but still self-motivated model.

In artificial continuously learning systems, the primary set points are always externally specified; natural continuously learning systems may evolve suitable set points. The system evaluates incoming data and modifies its behavior in such a way as to modify those parameters of its environment that are more or less under its control so that the system maintains the set points. It is a self-adaptive system adjusting in real time to a dynamic environment. It is continuously changing its internal structure to reflect its past experiences, and using those past experiences to modify its environment. If a continuously learning predictive model was given an identical input at different times, it may well produce totally different predictions—depending on what it had experienced, and the changes in its environment, in the interim. This is very different from a sequentially updated series of static models. The key is a continuous interaction between components.

As far as data preparation is concerned, the preparatory activities carried out when making static models tend to be manual. When continuously learning systems are deployed, however, the PIE that permits continuous, automated data preparation becomes a vital part of the whole process.

The easiest way to see what is involved in a continuously learning model is to examine a simplified actual application, and the Supplemental Material section at the end of this chapter briefly outlines a simplified application using a continuously learning model.

1.3 Summary

When discussing data mining, it is easy to think of the process only in terms of what various tools can do. This is exactly analogous to focusing on types of nails and what to do with them simply because a hammer collection is available. For sure, we will do different things with a 6-ounce ball-peen hammer, a 12-ounce claw hammer, and a 14-pound sledgehammer. However, the object of the exercise may be to knock down a wall, construct a house, repair a car door, or drive a railroad spike. It is the nature of the job to be done that determines which tool to use—not the other way around. So it is too with data mining. To obtain effective results when mining, focusing on the tools is not enough. This chapter has looked from the “100,000-foot level” at the whole process of data exploration, giving a perspective of where data mining fits within the process, and how data preparation, modeling, and the other components of mining interact. Because modeling is so closely connected with data preparation, the chapter introduced various types of models commonly produced by miners.

The key point is that data mining does not exist independently of the business problems that it needs to solve. Data mining exists to serve needs, in general the needs of a business user. The first thing to focus on is the business problem—what is the real problem, what does success look like? When that is established, then and only then is it time to select data and tools appropriate for the job.

Supplemental Material

A Continuously Learning Model Application

A major credit card issuer in the United States wanted to try innovative and more effective approaches for a solicitation program aimed at acquiring new customers. Several routes to market, including telemarketing and “take-one” programs, were used as well as direct mail. Additional marketing promotions, solicitations, and offer structures were included in the overall program, including balance transfer, affinity group marketing, and a variety of rates and payment terms. Most of this detail will not be included in this description. Although continuously learning models were used in all aspects of the solicitation, for clarity we will focus only on the direct mail portion. Before describing how a continuously learning model was built for this customer, it will be helpful to have a brief introduction to the significant considerations in credit card solicitations.

Typically, a marketing solicitation program without continuously learning techniques involves selecting the mailing list, producing the mail piece, making a bulk mailing, receiving the responses, entering the responses, and approving or declining each responding applicant. In essence, the bulk of the mailing goes out all together as far as that is possible. In practice the mailing is usually of such quantity that it is often spread over a number of days.

Since the mailing goes out, in effect, all at once, the response quickly builds to a maximum and then gradually tails off to a trickle. The sudden influx of responses necessitates hiring temporary staff, and renting office space and equipment for the staff, to cope with the sudden data entry workload. Federal requirements put great pressure on credit card issuers to “decision” an application—that is, to approve or decline the applicant—within a very short time period or face heavy financial penalties. The “decisioning” process involves studying credit references on all applicants, with the reference information almost invariably obtained from outside vendors. However, credit reference information can only legally be obtained for people who are actually offered credit.

Furthermore, the national average response rate for an unsolicited mailing program is well under 3%. The approve/decline rate is difficult to generalize since usually a variety of groups are targeted and the approve/decline rate varies enormously depending on the group actually targeted. It was estimated that at the time of this program it cost about $140 to acquire a new credit card customer by direct mail.

How the Continuously Learning Model Worked

The initial reaction of the company when approached with a discussion of the possibilities of data exploration was to say that they knew all about data mining, as they had bought a neural network package and one of their business analysts had built a model, but it didn’t work in their market. Fortunately, they were persuaded to consider the power of data exploration, not simply to mistake it for a PC-based neural network tool.

For the purposes of this explanation it is not necessary to examine the problem and solution explorations. Suffice it to say that the credit card issuer was interested in reducing the overall cost of the program, lowering the cost per acquisition, and improving the quality of the applicants and users in ways that are discussed during the course of this example.

An advantage of the traditional system of making a massive bulk mailing is that maximum return is felt shortly after the start date of the program. The bulk of new users who are going to respond typically have their credit cards activated within 60 days of program commencement. As discussed, this requires a considerable investment in temporary staff and facilities. The credit card issuer was willing to forgo this quick return for the proposed system that was designed to produce a steady stream of applicants at a preselected rate. The constancy in application level removed the need for, and cost of, temporary staff and facilities, allowing existing staff to cope with the flow rate. It also removed the sudden pressure on the “decisioning” process and additionally permitted the mailing to be routinized. Over the length of the program these changes alone saved a considerable amount of money for the card issuer.

Although it was conceived as a whole, we will consider the simplified pieces of the continuously learning system as they become relevant. The system as a whole is larger than the model itself since the whole system also includes the environment in which the model operates. The system starts with what was labeled the “slush pile.”

As was mentioned above, credit reference information can only be obtained for those people to whom credit is actually offered. In order that certain information can be obtained about people that the credit card company may wish to offer credit to, but has not yet done so, a method of “reservation numbers” was devised. Using this system, most of the information pertaining to a particular candidate (instance) is made available, but not the information about who it is. Thus you might know details such as education level, credit balance, number of children, marital status, and possibly well over 100 or more other demographic and sociographic measures. What you cannot know until an offer is made is name and address information, which makes it impossible to attach a record to an individual. In place of the missing information a unique key was supplied; called the “reservation number,” it allowed the credit information vendor to supply the relevant information when an offer of credit was to be made. The important point here is that it was possible to know a great deal about the population of potential new customers, but not to know specifically who they were until an offer was made. The information about the population was truly anonymous.

The slush pile consisted of a large number of records (instances) of credit information identified by reservation number. The pool was maintained with a minimum of about 1,000,000 records. As the records (instances) were “used,” that is, they had been selected and a solicitation made to them, the appropriate record was removed from the slush pile and replaced with a new, unsolicited credit record.

The continuously learning solicitation system began with the mailing process. A lot of preparatory work was needed to build a PIE. A result of the data preparation process, the PIE is a model of the data that allows new data drawn from the same population that was used to create the model to be transformed in “real time” into a form appropriate to be modeled. Its purpose is to convert raw data into the selected form after manual involvement in data preparation has been completed. The PIE for this application was built from credit information similar to, but not used in, this application.

Since there was no history of performance, the initial action was to make a random selection from the slush pile for the initial mailing. However, once responses from the initial mailing were received, it immediately became possible to begin building a model of who was likely to respond to the solicitation.

Information about who had actually responded to the mailing was entered into a table. Using the information about who had been solicited and who had responded, a fully automatic modeling process built models segmenting those attributes most indicative of a response to the mailing. The key features of the continuously learning system were that modeling was fully automatic—*with no human operator involvement*—and that the response was automatically optimized based on feedback information. One of the set points was externally fixed within the system—that is, the number of responses required. Another set point was variable and selected for optimization—that is, response rate as a percentage of solicitations. The environmental parameters under the “control” of this piece of the system included the selection criteria from the slush pile—that is, the characteristics of the person to whom the offer was made. Also included in the environment was which offer to make from the variety available. Thus, optimal cross-sell of additional products was automatically built in. The system, on receiving feedback information about response, used the information to update its model of the current driving factors—that is, what was working best at that instant, incorporating changes produced by competing offers, market dynamics, or social changes in the real world.

Fully automatic selection of the next batch to be mailed was made by the system based on the response model generated from previous mailings. The system automatically adjusted the number of solicitations to be mailed, based upon response levels, so that the right number of responses came back to meet the target selected in the project objectives.

The next stage in the process was that the applications were decisioned. This approve/decline information was entered into the system. The system now had additional factors in its environment to control—targeting not only people who would respond, but also those most likely to be approved. Once again, this model was automatically maintained, without human intervention, by the continuously learning system.

Following this, additional automatic environmental controls were added. The first was added when pattern of use information became available. Many credit card issuers feel a strong preference toward customers who are not “convenience users,” those who pay the balance in full when requested and, thus, never generate revenue for the issuer in the form of interest payments. Another increase in the quality of the target potential customers resulted—those who would not only respond and be approved, but also would be profitable for the card issuer. Eventually, default and fraud were modeled and added into the selection process.

This is a highly simplified description indeed. However, the system as described consists of four “sensors” feeding into a model continuously learning to recognize particular features in the environment—responsiveness, approval likeliness, convenience user tendency, and proclivity to default/fraud. The “environmental” parameters under the system’s control were the selection criteria for the 160 or so variables of the prospects in the slush pile, plus what products to offer each candidate.

Some particularly notable features of this system were that, for the duration of the program, the internal structures of the various model elements changed—dramatically in some cases. That is to say, the key indicating factors of, for example, who was likely to respond to the solicitation were dramatically different at different times. (A competing offer from another company targeted much of the original population. Any static model would have been defeated. The continuously learning model simply moved its sights and kept right on producing. Some time later the company’s marketers discovered what was going on.)

Clearly, any static model would have lost predictive power very quickly. The “half-life” of a static model, especially as market and economic conditions were changing rapidly, seemed to be about six weeks. While no full analysis of many of the underlying reasons for this shift was made, various economic, political, and social changes were happening during the solicitation period, from such things as the competing offer already mentioned, to a presidential election. Cursory examination of the parameter drift in the models indicated that these changes had an impact. In fact, competition from other credit card companies’ solicitation programs, targeting similar demographic and affinity groups, made for the most dramatic changes in the model.

In addition to actively reacting to changing conditions in order to optimize return, various pieces of business intelligence were generated. There were, in fact, a variety of different offers made, such as gold cards, preapproved and non-preapproved cards, interest rates and terms, home equity loans, lines of credit, and so on. Although not specifically requested in the specification of the system, a response surface model built of the response pattern based on the actual offer revealed what it was about different offers that different groups found attractive. This allowed the company’s marketing organization to make adjustments to terms and conditions offered to increase the appeal of the solicitation.

Although this is a very brief summary description of what a continuously learning model looks like in practice, it shows that it has a key place in a data miner’s toolkit. This particular model produced spectacular results. This system was able to achieve, among other things, response rates peaking over 10% (compared to an industry standard of well under 3%) and a greatly reduced acquisition cost (varying from time to time, of course, but under $75 at times compared to the client's previous $140). Additional benefits gained might be described (from the credit card issuer's viewpoint) as higher-quality customers—less likely to be convenience users or to default.

Chapter 2: The Nature of the World and Its Impact on Data Preparation

Overview

Data is explored to discover knowledge about the data, and ultimately, about the world. There are, however, some deep assumptions underlying this idea. It presupposes that knowledge is discoverable. In the case of using data mining as a tool for discovering knowledge, it presupposes that knowledge is discoverable in a collection of data. A reasonable assumption is that the discovered knowledge is to be usefully applied to the real world. It is therefore also assumed that the data to be mined does in fact have some persistent relationship to the world from which it was drawn. It is also assumed that any relationships that happen to be present in the assembled data can be meaningfully related back to real-world phenomena.

These crucial assumptions underpinning data exploration and data mining are usually unstated. They do, however, have a major impact on the actual process of mining data, and they affect how data is prepared for mining. Any analysis of data that is made in the hope of either understanding or influencing the world makes these assumptions. To better understand why data is manipulated in the way that it is during data preparation, and to understand the effects of the manipulations, we need to closely examine the assumptions, the nature of what data is measuring, and define the terms “data,” “information,” and “knowledge.”

Chapter 1 provided an overall framework for data exploration and put all of the components into perspective. This chapter will focus on the nature of the connection between the experiential world and the measurements used to describe it, how those measurements are turned into data, and how data is organized into data sets. Having created an organized representation of part of the world in a data set, we will also look at the nature and reasons for some of the adjustments, alterations, and reformatting that have to be applied to the data sets to prepare them for mining.

2.1 Measuring the World

The world is a place of unbelievable complexity. No matter how closely we look at some facet of the world, there is an infinite depth of detail. Yet our brains and minds construct meaningful (for us) simplicities from the stunning complexity that surrounds us. Using these simplicities we make representations of the world that we find useful, like lunch and banks. And using these simplicities, we can collect and record impressions about various facets of them, which we call data. It is this data that we then explore, at least with data mining, to understand something about the reality of the world—to discover information.

The data itself from which information is to be discovered, however rich and copious, is but a pale reflection of the real world. It doesn’t matter how much care is taken in examining the world and collecting data about it, reality is always more fluid, rich, and complex than any human can comprehend. Data never provides more than a pale and hazy shadow, a murky outline, of the true workings of the world. And yet this gossamer wisp is just enough for us to grasp at the edges of understanding. We may imagine that we control and manipulate the firm reality, but it is no more than a shadow of reality that is in our grasp. Understanding this, and understanding too the way that data connects to the world, is crucial for any data explorer. However powerful the exploring tools, or aggressive the explorer, nothing can be discovered that is beyond the limits of the data itself.

2.1.1 Objects

This is not a philosophical treatise, and I will leave discussing the true nature of the world to philosophers. The world exists in a way that humans generally agree on. It consists of *objects* that we can identify, such as cars, trees, cost-of-living adjustments, cartons of milk, beams of light, gross national products, beauty, truth, and justice. For data exploration through data mining it is these objects that form the basic material of the world to be explored. These objects actually comprise the fundamental underpinning, or the interface, that connects the activities of mining to the real world. Data mining explores the relationships that exist between these objects.

The precise definition of objects is another philosophical issue that need not concern miners. It is almost, if not actually, impossible to define what an object “really” is. It is also difficult or impossible to define the limits of an object precisely and unambiguously, since the world at very fine scale seems to appear as “shades of gray.” The miner takes a pragmatic view of the objects in the world, finding it unnecessary to define the actual objects and instead regarding an object as a collection of *features* about which measurements can be taken.

A car, for instance, is accepted by the miner as a defined object. The car possesses certain measurable features, such as the number of wheels, number of seats, color, weight, number of cylinders, fuel consumption, and a host of others. These measurements are not necessarily fixed; for instance, weight will change if fuel is added. However, they can be defined and measured with sufficient accuracy for any particular purpose, and the features can be specified as needed, such as “weight of vehicle empty.”

Clearly, objects do not have to be physical. The cost of living is a non-physical object. It has a definition and features. The features of the cost of living can be measured, such as what it may be in dollars, its rate of change per month or year, what percentage of the mean or median income it represents, and so on.

Objects in the real world relate to and interact with each other. Living objects interact with the world in noticeable and familiar ways, such as eating and breathing. Even inanimate objects interact with the world. Rocks, for example, interact with the ground on which they rest at an atomic level, and so do not sink into it. Mountains are worn down by weather, and even continents interact with the core of the earth and drift about. The cost of living changes, as does the unemployment level—driven (we say) by the economy and marketplace. All of these interactions form what philosophers have called “the great system of the world.” The features of objects captured as data form a reflection of this great system of the world. If the reflection is accurate, the features themselves, to a greater or lesser degree, represent that system. It is in this sense that data is said to represent or, sometimes, to form a system.

2.1.2 Capturing Measurements

For the data miner, objects actually consist of measurements of features. It is the groups of features that are taken as the defining characteristics of the objects, and actual instance measurements of the values of those features are considered to represent instances of the object. For instance, my car is a dark blue, two-door, six-cylinder, five-passenger vehicle. That is to say, for this particular instance of “car,” considering five features—ownership, color, door count, cylinder count, and passenger capacity—the measurements are Dorian Pyle, dark blue, 2, 6, 5.

These measurements are all taken in such a way that they have a particular type of validity. In this particular case, they were all taken at the same time, which is to say that they were true at the instant of my writing. The *validating feature* here, then, is a timestamp. This need not be the case, of course, although timestamps are very often used. To continue using cars as an example, other validating stamps might be “all 18-year-old males,” or “all Ford Escorts,” or “all red cars with four cylinders.” This would mean collecting measurements about all vehicles owned by 18-year-olds, all Ford Escorts, or all red cars with four cylinders, for instance.

There is an assumption here, then, that measurements are taken about objects under some validating circumstance. In effect, the world state is “frozen” by the validating circumstance and the measurements taken yielding a particular value. This idea of “freezing” the world’s state while taking measurements is an important one, particularly for miners. There are a variety of factors involved in taking measurements that can make the measurements seem inconsistent. Since it is very often part of mining to understand and estimate where the variability in a particular measurement comes from, as well as how reliable the measurement is, we need to look at some sources of variability.

2.1.3 Errors of Measurement

Measurement implies that there is some quantity to measure, and some device to calibrate the measurement against. A simple illustration of such a physical measurement is measuring a distance with a ruler. A nonphysical measurement might be of an opinion poll calibrated in percentage points of one opinion or another.

There are several ways in which a measurement may be in error. It may be that the quantity is not correctly compared to the calibration. For instance, the ruler may simply slip out of position, leading to an inaccurate measurement. The calibration device may be inaccurate—for instance, a ruler that is longer or shorter than the standard length. There are also inevitable errors of precision. For example, measurements of distance simply have to be truncated at some point, whether measuring to the nearest mile, foot, meter, centimeter, or angstrom unit.

Some of these errors, such as incorrect comparison, lead to a sort of "fuzz" in the measurement. Since there are likely to be as many measurements short as there are long, such errors also tend to cluster about the "correct" point. Statisticians have devised many ways to characterize this type of error, although the details are not needed here. If the calibration is in error—say, wrong ruler length—this leads to a systematic error, since all measurements made with a given ruler tend to be "off" the mark by the same amount. This is described as a *bias*.

Figure 2.1 shows the distortion, or error, that might be caused by the "fuzz" in such measurements. It shows what unbiased error might do to a measurement. Figure 2.2 shows what bias added to unbiased error might look like. These types of measurements are showing "point" measurements, so called because if taken without any error they appear as points on a graph.

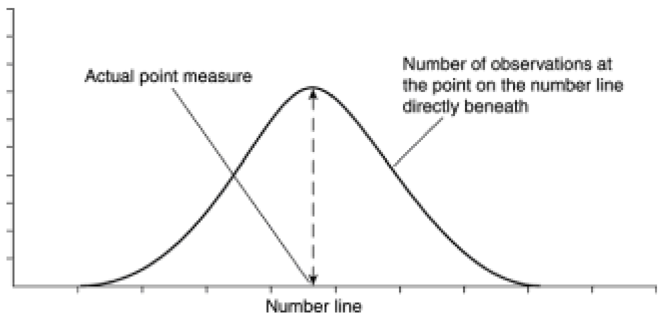


Figure 2.1 Unbiased noise spreads the measurements evenly around the measurement point. Most cluster near the actual value.

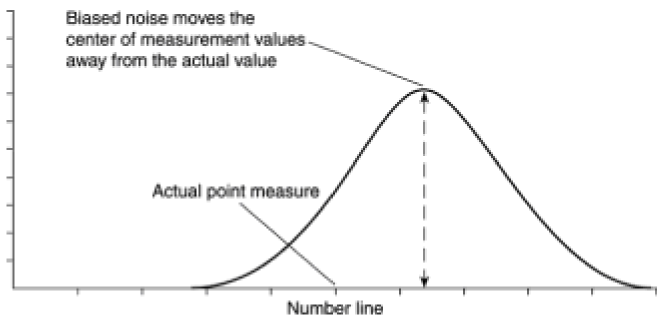


Figure 2.2 Biased noise makes most of the measurements cluster around a point that is not the true measurement.

Environmental errors are rather different in nature, but of particular importance in mining. *Environmental errors* express the uncertainty due to the nature of the world. Another way of looking at this interaction is that it expresses uncertainty due to the nature of the interactions between variables. These between-variable interactions are critically important to miners. Since there is some level of uncertainty in these interactions, they warrant a much closer inspection.

Suppose a particular potential purchaser of products from a catalog has actually made a previous purchase. The catalog company wants to measure several features of the object “purchaser” to combine them with measurements about other purchasers and create a general purchaser profile. There are many circumstances in the world that surround and influence purchasers. To make the required measurements, the world is “frozen” in its state for the particular purchaser and the surrounding circumstances captured. Several variables are measured. Each measurement is, of course, subject to the point distortion, or error, described previously.

Each fuzzy circle in Figure 2.3 represents such a single measurement. The central point of each circle represents the idealized point value, and the surrounding circle represents the unavoidable accompanying fuzz or error. Whatever the value of the actual measurement, it must be thought of as being somewhere in this fuzzy area, near to the idealized point value.

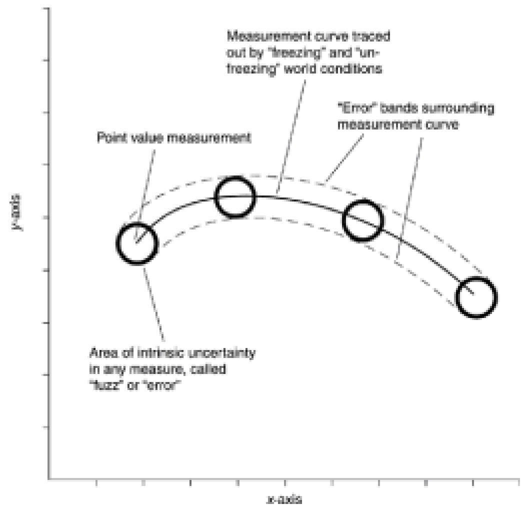

Measurement curve traced out by "freezing" and "un-freezing" world conditions
"Error" bands surrounding measurement curve
Point value measurement
y-axis
Area of intrinsic uncertainty in any measure, called "fuzz" or "error"
x-axis

Figure 2.3 Taking several point measurement values with uncertainty due to error outlines a measurement curve surrounded by an error band.

Suppose now that the world is unfrozen, conditions allowed to change minutely, and then refrozen. What happens to the measurement? If the driving factors are linearly related to the measurement, then a minute change in circumstances makes a minute change in measurement. The measurement taken under the slightly changed circumstances is slightly changed in direction and distance from the first measurement. Other minute changes in the world's state make similar minute changes in measurement. Such a series of measurements traces out the fuzzy line shown in Figure 2.3. This is the sort of change in measurement that might be traced out in the value of your bank account, say, if income varied by some small amount. The small change in income represents a change in the state of the world. The error represents the general fluctuation in bank account level due to the normal uncertainties of life. Perhaps if your income were slightly lower, the bank balance would be a little lower. An increase in income might raise the bank balance a little, but a further increase might lower it as you might then choose to put money into another account. This small change in your bank account that is associated with a small change in income demonstrates the effect of a linear relationship.

But perhaps the relationship is not linear, at least locally. What does this mean? It might mean that a minute change in some other circumstance would persuade you to use a completely different bank. Perhaps a better interest rate paid by another bank might be enough. This could mean that the overall shape of the curves would be the same, but their height would change, indicating the influence of interest rate changes. Figure 2.4 shows what this might look like.

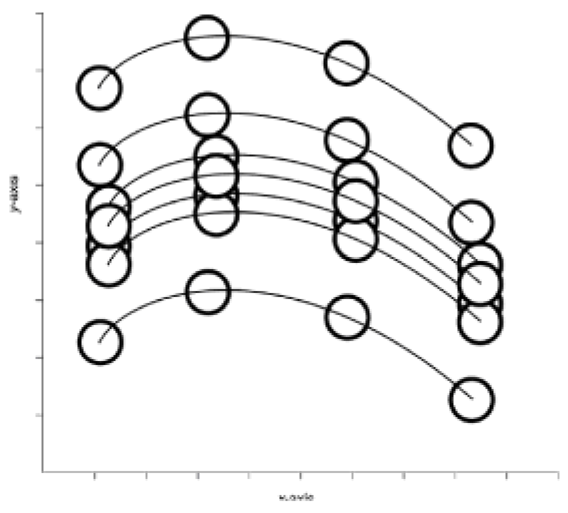


Figure 2.4 Groups and clusters of curves that result when a small change in world conditions makes a nonlinear or "step" change in the measured values.

What this figure might mean is that a small change in interest rate persuaded you to take all your money out of one bank and deposit it in another bank. For one bank the minute change in interest rate means that you completely and totally disappear as a customer, while appearing as a new customer for some other bank. The world change is small, but instead of slightly changing the bank balance up or down in the first bank, it made it disappear! The various lines in Figure 2.4 might then represent different banks, with the curves representing different balances. This "curve bundle" represents where and how the point measurements might map onto the world under the slightly different circumstances.

Some conditions, then, would be very sensitive to a minute change in circumstances due to the unfreezing/minute change/refreezing cycle, while other conditions were not so sensitive, or might be even completely unaffected by small changes. What this means for actual measurements is that, even for minute changes in the circumstances surrounding measurements, there are a variety of possible results. The changes may be undetectably small, given the nature of truncating the measurement accuracy discussed above. So the measurements traced out in state space, due possibly to the miniscule perturbations that are unavoidable in the real world, trace out not fuzzy points, but fuzzy curves. (State space will be covered in more detail later in this chapter. For now, very briefly, state space is the space in which measurement values can be plotted, like the space on a graph.)

A very important point to note for the miner here is that while many of the environmental factors may be unknowable, and certainly uncontrollable, they are subject to some limitations. For instance, it is very unlikely that any minute change in world conditions would change your deposit in a bank account into your ownership of a Swiss bank! Defining the limits, and determining the shape and size of the measurement curves, can be a critical factor in building models.

Determining the extent of the error is not so important to data preparation. What is important, and the reason for the discussion, is that during preparation it may be possible to determine where some of the components in the overall error come from, and to explore its shape. Mining to build models is concerned with addressing and, if possible, understanding the nature of the error; data preparation, with exposing and, perhaps, ameliorating it.

2.1.4 Tying Measurements to the Real World

Sometimes measurements are described as consisting of two components: the actual absolute perfect value, plus distortion. The distortion is often referred to as error. However, the distortion is actually an integral part of the measurement. Use of the term “error” has unfortunate connotations, as if there is somehow something wrong with the measurement. There seems to be an implication that if only the measurer had been more careful, the error could have been eliminated. While some part of the distortion may indeed result from a mistake on the part of the measurer, and so truly is an error in the sense of a “mistake,” much of the distortion is not only unavoidable, but is actually a critical part of what is being measured.

This use of the term “error” is emotionally loaded in ways that do a disservice to the miner. For all of the reasons discussed above, actual measurements are better envisioned as represented by curve bundles drawn in some state space. Some part of the curve will represent error in the sense of mistakes on the part of the measurer—whether human or machine. However, most of the range of the curve represents the way the feature maps onto an uncertain world. Even a perfect measurer, should such a thing exist, would still not be able to squeeze the various curves into a single point—nor even into a single curve.

Contrary to the view that there exists some perfect measurement with error, the more realistic situation is one that includes a distributed mapping of the measurement onto the real world, plus some single estimate of the location of some particular instance on the bundle of curves. Nonetheless, the term “error” is the one in general use in measurement and must be accepted. Remember that it is not to be thought of as some sort of mistake to be corrected, but as representing an essential and unavoidable part of the measurement that is integral with mapping the feature measured to the real world. Moreover, it is often the job of the miner to discover the shape of the measurement bundle. In mining, the error is often included in a feature called “noise” (looked at in detail later), although noise also includes other components.

2.2 Types of Measurements

So far this chapter has discussed measurements in general, and problems and limitations with making the measurements. Looked at more closely, there is an intuitive difference between different types of measurements. For instance, the value “1.26 feet” is obviously of a different type than the value “green.” This difference has a major impact on both the way data is prepared and the way it is modeled. Since these differences are important, the different types of measurements need to be examined in some detail.

All measurements have one feature in common: they are all made on some scale. Measurements in general map onto the world in ways represented by the measurement curve bundles. Individual instance values are not curves, but point measurements. It is usual to speak of the measurements of a particular feature of an object as a variable. A variable represents a measurement that can take on a number of particular values, with a different value possible for each instance. Conceptually, a variable is a container holding all of the measurements of a particular feature of some specific object. But different types of containers are needed to hold different types of measurements, just as tomatoes and soda both need different types of containers to hold them. The “containers” for variables are a way to classify them using descriptions such as “nominal” and “ratio” that will be discussed in a moment. Some variables consist of two components—the scale on which they are measured and the measured value itself—and others require more components. The class of variables that can be indicated by the position of a single point (value) on some particular scale are called *scalar variables*. There are other types of variables that require more than one value to define them; they are often called *vector variables*. Most of the work of the miner considers scalar variables, and these need to be examined in detail. So first, we will look at the different types of containers, and then what is in each of them.

2.2.1 Scalar Measurements

Scalar measurements come in a variety of types. Different types of measurements inherently carry different amounts of information. You can intuitively see this: just think about measuring the temperature of your coffee. By limiting the measurement to just “hot” or “cold,” you will see that this measurement contains less information than the measurements “scalding,” “too hot,” “nice and hot,” “hot,” “not hot,” “warm,” “cool,” and “cold.” The idea of information content is a very useful way to order the types of scalar measurements.

Nominal Scale Measurements

Values that are nominally scaled carry the least amount of information of the types of measurements to be considered. Nominal values essentially just name things. There is a notable difference in type or identity, but little or nothing more can be said if the scale of measurement is actually nominal. A nominal measurement is little more than a label used for purposes of identification. There is no inherent order in the nominal measurements. Nor indeed can nominally measured values even be meaningfully grouped together. They do, nonetheless, carry definite information, little though it might be.

Categorical Scale Measurements

Categorical measurements name groups of things, not individual entities. This categorization allows values to be grouped in meaningful ways. As with nominal measurements, nothing more can be said about the size or type of the differences. They are no more than labels for different groups.

For instance, ZIP codes, although they look like numbers, are really simply arbitrary labels for postal delivery zones. Listing them in their apparent numerical order is not particularly revealing. Standard industry classification (SIC) codes are very similar to ZIP codes in that, although they categorize different types of business activity, a numerical ordering seems no more nor less reasonable than an alphabetical listing of the activity represented by the number. It should also be clear that any ordering of scales such as marital status, ethnic background, or academic interest seems quite arbitrary.

However, it is possible to use a number as a categorically measured value label in order to more conveniently label and differentiate the category to which it belongs. Although formed using characters that are numerical symbols rather than letters of the alphabet, the labels remain exactly that, labels, and carry no numerical significance. Postal authorities numerically labeled ZIP codes, and the federal government numerically labeled SIC codes. Numerically labeled or not, all that can be said about the categories is that they are different in type. Numbering the measurement values is only a matter of convenience, and there is no implied ordering or ranking.

Not only do the categorical labels have no particular order, there is no information included in the categorization that indicates how different they are from each other. There is no real meaning in saying, for instance, that a plumber is twice a carpenter or three-quarters of a corporate director. You may have some personal feeling as to the amount of difference there is between different brands of boot polish. However, there is no way to determine, simply by knowing the category of the product, if the amount of difference between black and brown polish is more or less than between, say, black and red or red and brown types of polish. All that can be said, simply by knowing its category, is that there is a difference.

Categorical measurements, then, denote that there is a difference in kind or type, but are not able to quantify the difference. The scale used amounts to no more than a comprehensive listing of all of the categories into which the value can fall.

Ordinal Scale Measurements

When something more can be said about the measurement scale used, the additional information gives some sort of order to the categories that are used to label the measurement. Because there is some sort of meaningful order to the listing of the labels, this type of measurement is often known as an *ordinal measurement*.

Ordinal measurements carry far more information than either nominals or categoricals. You may not be surprised to learn that by taking a table of the actual distances between, say, major American cities, the joint distance table alone is enough to re-create the layout of the cities that show up on a map. Surprisingly, instead of the actual distances, just using a simple pairwise ranking of cities by distance is also enough to re-create the layout of the cities as seen on a map almost perfectly.

This example shows that for the purpose of making a schematic map, knowing the actual distances provides little additional information. Most of the information required to accurately create a schematic map of American cities is enfolded in a simple pairwise ranking of cities by their distance apart.

The ranking of the categories must be done subject to a very particular condition, called *transitivity*, which is actually a reasonable notion although of critical importance. Transitivity means that if A is ranked higher than B, and B higher than C, then A must be ranked higher than C. That is: If A > B and B > C, then A > C. While measuring a value using an ordinal scale adds a huge amount of information over that contained in a categorical measurement, the transitivity requirement places some constraints on how the ordinal scale can actually be built. Note that the ordinal scale does not require that anything has to be specified about the amount of the difference between each category.

For instance, at a "blind tasting" for wines, you sample several different types and styles of wine and mark down pairwise combinations of preference. Perhaps you prefer the cabernet to the merlot, and the merlot to the shiraz. If transitivity holds and you prefer the cabernet to the shiraz, the result is an ordinal listing of wine preferences: the favorites, in order, are cabernet, merlot, and shiraz. However, there is no indication of by how much you prefer the cabernet to the merlot. It may be that the difference in preference is slight: you choose cabernet 51% of the time and merlot 49% of the time; the shiraz doesn't get a look in. On the other hand, given the availability of cabernet, perhaps you will choose that every time, only considering the others when cabernet is unavailable.

The point here is that ordinal measurements do indeed carry a lot of information, but allow for no comparison of the magnitude of the differences between the categories.

Interval Scale Measurements

When there is information available not only about the order for ranking the values measured but also about the differences in size between the values, then the scale is known as an *interval scale*. This means that the scale carries with it the means to indicate the distance that separates the values measured. Interval variables are almost always measured using numbers. Because numbers are almost exclusively used when discussing interval-scaled values, measurements scaled this way are part of the group called *quantitative measurements*—that is, values that capture differences in, changes in, or the amount of the quantity of some attribute of an object.

An interval scale that almost everyone is familiar with is the temperature scale. Every day newspapers, radio, and television provide a forecast of the temperature range for the day's weather. If the low for the day is predicted to be 40 and the high 50, this provides some particular idea of what temperature you will experience. In this case the range through which the temperature is expected to move is 10º. If at some other time of year the low/high is forecast as 80 through 90, you can tell that the expected temperature range is again 10º, the same as earlier in the year. Thus the difference of 10º indicates the same amount of temperature change regardless of where it occurs on the range of the scale.

However, you cannot say, based on the interval scale used, that the low for the two days can be compared using their ratio. That is to say, 80º is not twice as hot as 40º. It is easy to see that there must be something wrong in supposing that the ratios are meaningful if instead of using Fahrenheit, you made the same comparison using the Celsius scale to measure the same temperature range.

Roughly speaking, 80ºF corresponds to 25ºC, while 40ºF corresponds to about 5ºC. However, the ratio of 80 to 40 is 2, but the ratio of 25 to 5 is 5! This means that when measuring the temperature with a Fahrenheit thermometer, you might say that it was twice as hot (ratio of 2), but your Celsius-using neighbor claims that it was five times as hot (ratio of 5)! One of these observations at least must be wrong, and in fact they are both wrong.

What is wrong, of course, is that the zero point, often called the *origin of the scale*, is not at the same temperature for the two scales. This means that the scales have differing ratios at equivalent temperatures. In fact, the zero point is arbitrarily set, which is a characteristic of interval scales. So, as far as temperature goes, scientists use a scale known as the “Absolute” or “Kelvin” scale specifically to overcome this problem. On this scale, the zero point corresponds to a true zero point so that the ratios of numbers compared on this scale have meaning.

Ratio Scale Measurements

The scale that carries the most information content is the *ratio scale.* One ratio scale measure that you are no doubt very familiar with measures the content of your bank account. It starts at a true zero point, which is to say that when the bank balance is 0 it is because there is no money in it. Also, it is denominated in currency units of equal value and size. This means that you can express meaningful ratio values of the state of your finances, knowing, for instance, that $10 is twice as much as $5, and $100 is twice $50. At any position on the scale, for any values, the ratio is a meaningful measure of properties of the scale.

As with the interval scale, ratio-scaled values are also quantitative. It is useful to consider two types of ratio-scaled measurements: those for which the scale that they are measured on must be named and those for which no scale is named. The characteristics of each type are sufficiently different that it is sometimes important to treat them differently during data preparation.

Usually it is important to know the units of a particular ratio measurement. To measure sales activity as “5” is not useful. Even if you knew that they were “4” last month, there is no reference in the numbers to indicate their significance. Knowledge of the unit of measurement is required. It means something if we stipulate that the units are millions of dollars and something else again if the units are thousands of Russian rubles or numbers of units shipped.

There is a class of ratio-scaled values that is measured *only* as numbers. These numbers are sometimes called *dimensionless*. A dimensionless number expresses a relationship that holds true without reference to the underlying measurements of the scale. For instance, consider a lighthouse standing on a rocky headland. Each lighthouse signals in a particularly distinct way such that any ship that sees the signal knows which particular lighthouse is in view just from the pattern of the signal. The lighthouse signals by showing a light in a unique pattern that is repeated over time. In any time cycle, however long or short, the light is on for a certain duration and off for another duration. Suppose that for a particular lighthouse the light is on for 10 seconds and off for 5. The ratio of on/off is 10/5, which, by division, reduces to 2/1, or 2. This measurement, sometimes known as a *duty cycle*, is dimensionless, and for this particular lighthouse it is 2. That is not 2 per anything, or 2 anythings, simply 2. The lighthouse pattern repeats once in 15 seconds, or four times per minute. So long as we consider only complete cycles, it doesn’t matter at all over how long a period the duty cycle of the lighthouse is measured; it will always be 2.

Care must be taken with measurements over time. Sometimes these measurements are assumed to be dimensionless when in fact they are not. A common discussion of ratio-scale variables discusses the distinction between “how many” and “how much.” The “how much” type is said to require the scale units, as in the sales figures just discussed. “How many” types of measurements are often said not to need such units. For instance, in stock market reports, not only is the market index quoted, but frequently the “advance/decline ratio” is given.

“The stock market was up today,” the news anchor might say, “with advances leading declines 5 to 4.” This means that five stocks went up in price for every four that did not. Now a ratio of 5/4 can be given as 1.25 and this gives the appearance of a dimensionless number. Here is a measurement of “how many” (i.e., 5/4) rather than “how much” (which is measured in “points” or dollars or some other specified unit).

However, the “gotcha” is that the count of advances to declines was taken over today. When considering the example of the lighthouse, a very important point was that, so long as we looked at complete cycles, the length of time of the observation did not matter. The advance/decline ratio applies only to the specific period over which it was measured. If a period is included that is longer or shorter by only a few minutes, it is possible that the measurement would not be 5/4. Indeed, you can be quite sure that in choosing some other period there is no reason to think that the 5/4 ratio holds true except by coincidence.

We have become so culturally accustomed to the idea of fixed and “natural” measurements of time that it is easy to overlook the fact that measurements of duration are arbitrary. By happenstance the Babylonians had a number system based on the number 60. Since it was they who made the original astronomical measurements, and because they thought there were approximately 360 days in a year, we now have 360 degrees in a circle. By a series of what they thought were convenient divisions, they arrived at a 24-hour day as standard. The hours were further divided in smaller parts, and by making a second division of the minute parts of an hour by 60 we get the “seconds,” so called because of the second division involved. Very clever and useful. However, there are alternatives.

Napoleon, in attempting to introduce the metric system, tried to “rationalize” all of the measuring systems then in use. Measures for distance and mass—the meter and gram, respectively—were adopted; however, the division of the year into 10 months and the day into 10 hours, and so on, was not accepted. The point is that all of our measurements of time are arbitrary. Some are arbitrary through human selection, but even the rotation of our planet has varied considerably through the eons. The first creatures out of the primordial soup probably experienced an 18-hour day. The slowing of planetary rotation has brought us to a day of approximately 24 hours. Because we are creatures of planet Earth, there are many cycles that are tied to days, seasons, and years. However, there is nothing inherently special about these scale units any more than any other scale unit.

Measurements in time, then, need to be considered carefully. By identifying and confirming complete cycles, returning to an identifiable identical state from time to time, dimensionless numbers may be useful. Measures based on the “how many/how much” dichotomy are suspect.

2.2.2 Nonscalar Measurements

Scalar values consist of just two component parts, the value of the measurement and the scale against which the measurement was made. In traffic court it is enough to prove that the speed of a vehicle was, or was not, some particular number of miles per hour. The speed is expressed as a number and the scale in miles per hour. Nonscalar measurements need more component parts to capture additional information. Speed is the number of miles traveled in one hour. Velocity, however, is measured as speed in a particular direction. There are at least four components in such a measurement—two scales and two measurements on those scales. Navigation at sea, for instance, is very concerned with velocity—how fast and in which direction the vessel is travelling.

Measurements such as velocity can be plotted on a two-dimensional graph in the form of a point specified by the measurements of speed on one axis, and direction on another. A line drawn on a graph from the common point that was chosen to begin the measurement representation, to the point where it ends up, is known as a *vector*. There is a great deal of literature about such vector quantities, their properties, and how to manipulate them. So far as data preparation is concerned, however, vector quantities are built out of scalar quantities. It is true that the scalar quantities are linked in particular and significant ways, but for the purposes of data preparation, the vectors can be carefully treated as multiple scalar values.

This is not to minimize the importance of vector quantities. Indeed, the concept of state space regards the instance value of multiple features as a multidimensional vector. Each record in a table, in other words, is taken as a vectoral representation. The point here is that most vectors can be thought of as being made up of separate scalar values and can be usefully treated for purposes of data preparation at the scalar level.

2.3 Continua of Attributes of Variables

So far this chapter has addressed the way in which measurements are taken using different types of scale. Collections of measured values of particular features are grouped together into variables. Because the values are collected together, it is possible to look for patterns in the way the values change with changes in the validating feature, or with changes in other variables.

When the measurements are actually taken in practice, certain patterns appear if many instances of values of a variable are considered as a whole. These aggregate collections of values begin to show a variety of different features. It is hard to characterize these elementary patterns as a part of data mining, although they are in truth the surface ripples of the deeper structure that miners will be seeking when the actual data mining tools are applied to the prepared data. Although this discussion only concerns introductory issues about data preparation, it is still true that the data preparation begins with a fairly comprehensive survey of the properties of each of the variables taken individually. It is in the appreciation of the basic types of attributes of variables that data preparation begins. Chapter 4 looks at this issue in considerable detail. Later discussion in this chapter summarizes the methods that will prepare variables for modeling.

Although described as if each of the scales were separate, actually the types blur together into a more continuous spectrum than the separate descriptions seem to imply. It is usual to describe variables as being of the same type as the scale, or features of the scale, on which they are measured. So it is convenient, say, to talk of a categorical variable, or a continuous variable. A measured value on a scale is, of course, a single point and as such cannot show any pattern. It cannot even show any of the “fuzz” of noise discussed earlier. Variables, being collections of instance values of a particular feature, all being made on a common scale, do show recognizable patterns, or attributes. It is these common attributes of variables that can be described as existing in a continuum.

2.3.1 The Qualitative-Quantitative Continuum

This continuum captures the low to high information content of the different types of scalar variables. Describing variables as qualitative or quantitative might not make it obvious that what is being described is information content. Nominal variables are at the qualitative end of the scale—that is, they separate attributes by a difference in quality. Similarly, at the other end of the scale is the ratio (quantitative) association. Information content varies continuously across the scale. Any sharp division implied by the qualitative-quantitative differentiation is not really present. So this continuum really recapitulates the differences in the scales that were discussed before, except that it considers the impact of the different scales on variables.

2.3.2 The Discrete-Continuous Continuum

This will prove to be a very important distinction about variables. In fact, the discrete-continuous distinction forms a continuum. As was done when considering scales, for ease of explanation it is easiest to look at several points along the continuum. At each of the points viewed, the distinctions are easy to draw. There are, however, no hard and fast boundaries in practice.

As a very brief introduction to the following discussion, discrete variables are considered to have a very limited set of values that they can take on, such as colors of the rainbow. Continuous values can take on any value within a range, like the temperature. To see that this is a continuum, consider your bank account—discrete or continuous? Technically, it is discrete as it is restricted to values to the nearest penny. In practice, however, the quantization, or fineness of division, is such that it would usually be more useful to consider it as a continuous value.

Single-Valued Variables (Constants)

It may seem odd to discuss a “variable” as having only a single value. Strictly speaking, since it is not varying its value, it would seem to be something other than a variable. However, variables that do not vary are often used, and very useful they are, too. Some examples of constants are the number of days in a week, inches in a foot, the distance represented by a light year, and the number of sides in a triangle. These constant values are representative of what we see as invariant, defining characteristics of an object.

They also turn up when modeling variables. Perhaps a marketing organization wants to examine all records for “the gold card upgrade program.” There may be many different marketing programs represented in the original data set. In this original data set, the variable “program name” is variable—it varies by having different values representing the different programs. The indicator for the gold card upgrade program is, say, “G”. Different letters are used to identify other programs. However, by the time only the records that are relevant to the gold card upgrade program are extracted into a separate file, the variable “program name” becomes a constant, containing only “G” in this data set. The variable is a defining feature for the object and, thus, becomes a constant.

Nonetheless, a variable in a data set that does not change its value does not contribute any information to the modeling process. Since constants carry no information within a data set, they can and should be discarded for the purposes of mining the data.

Two-Valued Variables

At least variables with two values do vary! Actually, this is a very important type of variable, and when mining, it is often useful to deploy various techniques specifically designed to deal with these *dichotomous* variables. An example of a dichotomous variable is “gender.” Gender might be expected to take on only values of male and female in normal use. (In fact, there are always at least three values for gender in any practical application: “male,” “female,” and “unknown.”)

Empty and Missing Values: A Preliminary Note

A small digression is needed here. When preparing data for modeling, there are a number of problems that need to be addressed. One of these is missing data. Dealing with the problem is discussed more fully later, but it needs to be mentioned here that even dichotomous variables may actually take on four values. These are the two values it nominally contains and the two values “missing” and “empty.”

It is often the case that there will be variables whose values are missing. A *missing* value for a variable is one that has not been entered into the data set, but for which an actual value exists in the world in which the measurements were made. This is a very important point. When preparing a data set, the miner needs to “fix” missing values, and other problems, in some way. It is critical to differentiate, if at all possible, between values that are missing and those that are empty. An *empty* value in a variable is one for which no real-world value can be supposed.

A simple example will help to make the difference clear. Suppose that a sandwich shop sells one particular type of sandwich that contains turkey with either Swiss or American cheese. In order to determine customer preferences and to control inventory, the store keeps records of customer purchases. The data structure contains a variable “gender” to record the gender of the purchaser, and a variable “cheese type” to record the type of cheese in the sandwich. “Gender” could be expected to take the values “M” for male and “F” for female. “Cheese type” could be expected to take the values “S” for Swiss and “A” for American cheese.

Suppose that during the recording of a sale, one particular customer requests a turkey sandwich with no cheese. In recording the sale the salesperson forgets to enter the customer's gender. This transaction generates a record with both fields "gender" and "cheese type" containing no entry. In looking at the problem, the miner can assume that in the real world in which the measurements were taken, the customer was either male or female, and any adjustment must be made accordingly. As for "cheese type," this value was not measured because no value exists. The miner needs a different "fix" to deal with this situation.

If this example seems contrived, it is based on an actual problem that arose when modeling a grocery store chain's data. The original problem occurred in the definition of the structure of the database that was used to collect the data. In a database, missing and empty values are called *nulls*, and there are two types of null values, one each corresponding to missing and empty values. Nulls, however, are *not* a type of measurement.

Miners seldom have the luxury of going back to fix the data structure problem at the source and have to make models with what data is available. If a badly structured data set is all that's available, so be it; the miner has to deal with it! Details of how to handle empty and missing values are provided in Chapter 8. At this point we are considering only the underlying nature of missing and empty variables.

Binary Variables

A type of dichotomous variable worth noting is the *binary* variable, which takes on only the values "0" and "1." These values are often used to indicate if some condition is true or false, or if something did or did not happen. Techniques applicable to dichotomous variables in general also apply to binary variables. However, when mining, binary variables possess properties that other dichotomous variables may not.

For instance, it is possible to take the mean, or average, of a binary variable, which measures the occurrence of the two states. In the grocery store example above, if 70% of the sandwich purchasers were female, indicated by the value "1," the mean of the binary variable would be 0.7. Certain mining techniques, particularly certain types of neural networks, can use this kind of variable to create probability predictions of the states of the outputs.

Other Discrete Variables

All of the other variables, apart from the constants and dichotomous variables, will take on at least three or more distinct values. Clearly, a sample of data that contains only 100 instances cannot have more than 100 distinct values of any variable. However, what is important is to understand the nature of the underlying feature that is being measured. If there are only 100 instances available, these represent only a sample of all of the possible measurements that can be taken. The underlying feature has the properties that are indicated by all of the measurements that could be taken. Much of the full representation of the nature of the underlying feature may not be present in the instance values actually available for inspection. Such knowledge has to come from outside the measurements, from what is known as the *domain of inquiry*.

As an example, the underlying value of a variable measuring “points” on a driving license in some states cannot take on more than 13 discrete values, 0–12 inclusive. Drivers cannot have less than 0 points, and if they get more than 12 their driving licenses are suspended. In this case, regardless of the actual range of values encountered in a particular sample of a data set, the possible range of the underlying variable can be discovered. It may be significant that a sample does, or does not, contain the full range of values available in the underlying attribute, but the miner needs to try to establish how the underlying attribute behaves.

As the density of discrete values, or the number of different values a variable can take on, increases for a given range, so the variable approaches becoming a continuous variable.

In theory, it is easy to determine the transition point from discrete to continuous variables. The theory is that if, between any two measurements, it is inherently possible to find another measurement, the variable is continuous; otherwise not. In practice it is not always so easy, theoretical considerations notwithstanding. The value of a credit card balance, for instance, can in fact take on only a specifically limited number of discrete values within a specified range. The range is specified by a credit limit at the one end and a zero balance (ignoring for the moment the possibility of a credit balance) at the other. The discrete values are limited by the fact that the smallest denomination coin used is the penny and credit balances are expressed to that level. You will not find a credit card balance of “$23.45964829.” There is, in fact, nothing that comes between $23.45 and $23.46 on a credit card statement.

Nonetheless, with a modest credit limit of $500 there are 50,000 possible values that can occur in the range of the credit balance. This is a very large number of discrete values that are represented, and this theoretically discrete variable is usually treated for practical purposes as if it were continuous.

On the other hand, if the company for which you work has a group salary scale in place, for instance, while the underlying variable probably behaves in a continuous manner, a variable measuring which of the limited number of group salary scales you are in probably behaves more like a categorical (discrete) variable.

Techniques for dealing with these issues, as well as various ways to estimate the most effective technique to use with a particular variable, are discussed later. The point here is to be aware of these possible structures in the variables.

Continuous Variables

Continuous variables, although perhaps limited as to a maximum and minimum value, can, at least in theory, take on any value within a range. The only limit is the accuracy of representation, which in principle for continuous variables can be increased at any time if desired.

A measure of temperature is a continuous variable, since the “resolution” can be increased to any amount desired (within the limit of instrumentation technology). It can be measured to the nearest degree, or tenth, or hundredth, or thousandth of a degree if so chosen. In practice, of course, there is a limit to the resolution of many continuous variables, such as a limit in ability to discriminate a difference in temperature.

2.4 Scale Measurement Example

As an example demonstrating the different types of measurement scales, and the measurements on those scales, almost anything might be chosen. I look around and see my two dogs. These are things that appear as measurable objects in the real world and will make a good example, as shown in Table 2.1.

TABLE 2.1 Title will go here

Scale Type	Measurement	Measured Value	Note
Nominal	Name	• Fuzzy • Zeus	Distinguishes one from the other.
Categorical	Breed	• Golden Retriever • Golden Retriever	Could have chosen other categories.
Categorical (Dichotomous)	Gender	• Female • Male	
Categorical (Binary)	Shots up to Date (1=Yes;0=No)	• 1 • 1	

Categorical (Missing)	Eye color	• Value exists in real world	
Categorical (Empty)	Drivers License #	• No such value in real world	
Ordinal	Fur length	• Longer • Shorter	Comparative length allowing ranking.
Interval	Date of Birth	• 1992 • 1991	
Ratio	Weight	• 78 lbs • 81 lbs	
Ratio (Dimensionless)	Height / Length	• 0.5625 • 0.625	

2.5 Transformations and Difficulties—Variables, Data, and Information

Much of this discussion has pivoted on information—information in a data set, information content of various scales, and transforming information. The concept of information is crucial to data mining. It is the very substance enfolded within a data set for which the data set is being mined. It is the reason to prepare the data set for mining—to best expose the information contained in it to the mining tool. Indeed, the whole purpose for mining data is to transform the information content of a data set that cannot be directly used and understood by humans into a form that can be understood and used.

Part of Chapter 11 takes a more detailed look at some of the technical aspects of information theory, and how they can be usefully used in the data preparation process. Information theory provides very powerful and useful tools, not only for preparing data, but also for understanding exactly what is enfolded in a data set. However, while within the confines of information theory the term "information" has a mathematically precise definition, Claude Shannon, principal pioneer of information theory, also provided a very apt and succinct definition of the word. In the seminal 1949 work *The Mathematical Theory of Communication*, Claude E. Shannon and Warren Weaver defined information as "that which reduces uncertainty." This is about as concise and practical a definition of information as you can get.

Data forms the source material that the miner examines for information. The extracted information allows better predictions of the behavior of some aspect of the world. The improved prediction means, of necessity, that the level of uncertainty about the outcome is reduced. Incorporating the information into a predictive or inferential framework provides knowledge of how to act in order to bring about some desired result. The information will usually not be perfect, so some uncertainty will remain, perhaps a great deal, and thus the knowledge will not be complete. However, the better the information, the more predictive or powerfully inferential the knowledge framework model will be.

2.6 Building Mineable Data Representations

In order to use the variables for mining, they have to be in the form of data. Originally the word “datum” was used to indicate the same concept that is indicated here, in part, by “measurement” or “value.” That is, a datum was a single instance value of a variable. Here measurement both signifies a datum, and also is extended to indicate the values of several features (variables) taken under some validating condition.

A collection of data points was called data, and the word was also used as a plural form of datum. Computer users are more familiar with using data as a singular noun, which is the style adopted here. However, there is more to the use of the term than simply a collection of individual measurements. Data, at least as a source for mining, implies that the data points, the values of the measurements, are all related in some identifiable way. One of the ways the variables have to be structured has already been mentioned—they have to have some validating phenomenon associated with a set of measurements. For example, with each instance of a customer of cellular phone service who decides to leave a carrier, a process called *churning*, the various attributes are captured and associated together.

The validating phenomenon for data is an intentional feature of the data, an integral part of the way the data is structured. There are many other intentional features of data, including basic choices such as what measurements to include and what degree of precision to use for the measurements. All of the intentional, underlying assumptions and choices form the *superstructure* for the data set. Three types of structure are discussed in the next chapter. Superstructure, however, is the only one specifically involved in turning variables into data.

Superstructure forms the framework on which the measurements hang. It is the deliberately erected scaffolding that supports the measurements and turns them into data. Putting such scaffolding in place and adding many instances of measured values is what makes a data set. Superstructure plus instance values equals data sets.

2.6.1 Data Representation

The sort of data that is amenable to mining is always available on a computer system.

This makes discussions of data representation easy. Regardless of how the internal operations of the computer system represent the data, whether a single computer or a network, data can almost universally be accessed in the form of a table. In such a table the columns represent the variables, and the records, or rows, represent instances. This representation has become such a standardized form that it needs little discussion. It is also very convenient that this standard form can also easily be discussed as a matrix, with which the table is almost indistinguishable. Not only is the table indistinguishable from a matrix for all practical purposes, but both are indistinguishable from a spreadsheet.

Spreadsheets are of limited value in actual mining due to their limited data capacity and inability to handle certain types of operations needed in data preparation, data surveying, and data modeling. For exploring small data sets, and for displaying various aspects of what is happening, spreadsheets can be very valuable. Wherever such visualization is used, the same row/column assumption is made as with a table.

So it is that throughout the book the underlying assumption about data representation is that the data is present in a matrix, table, or spreadsheet format and that, for discussion purposes, such representation is effectively identical and in every way equivalent. However, it is not assumed that all of the operations described can be carried out in any of the three environments. Explanations in the text of actual manipulations, and the demonstration code, assume only the table structure form of data representation.

2.6.2 Building Data—Dealing with Variables

The data representation can usefully be looked at from two perspectives: as data and as a data set. The terms “data” and “data set” are used to describe the different ways of looking at the representation. Data, as used here, implies that the variables are to be considered as individual entities, and their interactions or relationships to other variables are secondary. When discussing the data set, the implication is that not only the variables themselves are considered, but the interactions and interrelationships have equal or greater import. Mining creates models and operates exclusively on data sets. Preparation for mining involves looking at the variables individually as well as looking at the data set as a whole.

Variables can be characterized in a number of useful ways as described in this chapter. Having described some features of variables, we now turn our attention to the types of actions taken to prepare variables and to some of the problems that need to be addressed.

Variables as Objects

In order to find out if there are problems with the variables, it is necessary to look at a summary description and discover what can be learned about the makeup of the variable itself. This is the foundation and source material for deciding how to prepare each variable, as well as where the miner looks at the variable itself as an object and scrutinizes its key features and measurements.

Naturally it is important that the measurements about the variable are actually valid. That is to say, any inferences made about the state of the features of the variable represent the actual state of the variable. How could it be that looking at the variable wouldn't reveal the actual state of the variable? The problem here is that it may be impossible to look at all of the instances of a variable that could exist. Even if it is not actually impossible, it may be impractical to look at all of the instances available. Or perhaps there are not enough instance values to represent the full behavior of the variable. This is a very important topic, and Chapter 5 is entirely dedicated to describing how it is possible to discover if there is enough data available to come to valid conclusions. Suffice it to say, it is important to have enough representative data from which to draw any conclusions about what needs to be done.

Given that enough data is available, a number of features of the variable are inspected. Whatever it is that the features discover, each one inspected yields insight into the variable's behavior and might indicate some corrective or remedial action.

Removing Variables

One of the features measured is a count of the number of instance values. In any sample of values there can be only a limited number of different values, that being the size of the sample. So a sample of 1000 can have at most only 1000 distinct values. It may very well be that some of the values occur more than once in the sample. In some cases—1000 binary variable instances, for example—it is certain that multiple occurrences exist.

The basic information comprises the number of distinct values and the frequency count of each distinct value. From this information it is easy to determine if a variable is entirely empty—that is, that it has only a single value, that of "empty" or "missing." If so, the variable can be removed from the data set. Similarly, constants are discovered and can also be discarded.

Variables with entirely missing values and variables that contain only a single value can be discarded because the lack of variation in content carries no information for modeling purposes. Information is only carried in the pattern of change of value of a variable with changing circumstances. No change, no information.

Removing variables becomes more problematic when most of the instance values are empty, but occasionally a value is recorded. The changing value does indeed present some information, but if there are not many actual values, the information density of the variable is low. This circumstance is described as *sparsity*.

Sparsity

When individual variables are sparsely populated with instance values, the miner needs to decide when to remove them because they have insignificant value. Chapter 5 describes in some detail how to decide when to remove sparse variables. Essentially, the miner has to make an arbitrary decision about confidence levels, that is, how confident the miner needs to be in the model.

There is more to consider about sparsity, however, than can be seen by considering variables individually. In some modeling applications, sparsity is a very large problem. In several applications, such as in telecommunications and insurance, data is collected in ways that generate very sparsely populated data sets. The variable count can be high in some cases, over 7000 variables in one particular case, but with many of the variables very sparsely populated indeed. In such a case, the sparsely populated variables are not removed. In general, mining tools deal very poorly with highly sparse data. In order to be able to mine them, they need to be collapsed into a reduced number of variables in such a way that each carries information from many of the original variables. Chapter 10 discusses collapsing highly sparse data.

Since each of the instances are treated as points in state space, and state space has many dimensions, reducing the number of variables is called *dimensionality reduction*, or *collapsing dimensionality*. Techniques for dealing with less extreme sparsity, but when dimensionality reduction is still needed, are discussed in Chapter 7. State space is described in more detail in Chapter 6.

Note that it has to be the miner’s decision if a particular variable should be eliminated when some sparsity threshold is reached, or if the variable should be collapsed in dimensionality with other variables. The demonstration software makes provision for flagging variables that need to be retained and collapsed. If not flagged, the variables are treated individually and removed if they fall below the selected sparsity threshold.

Monotonicity

A *monotonic* variable is one that increases without bound. Monotonicity can also exist in the relationship between variables in which as one variable increases, the other does not decrease but remains constant, or also increases. At the moment, while discussing variable preparation, it is the monotonic variable itself that is being considered, not a monotonic relationship.

Monotonic variables are very common. Any variable that is linked to the passage of time, such as date, is a monotonic variable. The date always increases. Other variables not directly related to time are also monotonic. Social security numbers, record numbers, invoice numbers, employee numbers, and many, many other such indicators are monotonic. The range of such categorical or nominal values increases without bound.

The problem here is that they almost always have to be transformed into some nonmonotonic form if they are to be used in mining. Unless it is certain that every possible value of the monotonic variable that will be used is included in the data set, transformation is required. Transformation is needed because only some limited part of the full range of values can possibly be included in any data set. Any other data set, specifically the execution data set, will contain values of the monotonic variable that were not in the training data set. Any model will have no reference for predicting, or inferring, the meaning of the values outside its training range. Since the mined model will not have been exposed to such values, predictions or inferences based on such a model will at best be suspect.

There are a number of transformations that can be made to monotonic variables, depending on their nature. Datestamps, for instance, are often turned into seasonality information in which the seasons follow each other consecutively. Another transformation is to treat the information as a time series. Time series are treated in several ways that limit the nature of the monotonicity, say, by comparing "now" to some fixed distance of time in the past. Unfortunately, each type of monotonic variable requires specific transformations tailored to best glean information from it. Employee numbers will no doubt need to be treated differently from airline passenger ticket numbers, and those again from insurance policy numbers, and again from vehicle registration numbers. Each of these is monotonic and requires modification if they are to be of value in mining.

It is very hard to detect a monotonic variable in a sample of data, but certain detectable characteristics point to the possibility that a variable is in fact monotonic. Two measures that have proved useful in giving some indication of monotonicity in a variable (described in Chapter 5) are interstitial linearity and rate of detection. *Interstitial linearity* measures the uniformity of spacing between the sampled values, which tends to be more uniform in a monotonic variable than in some nonmonotonic ones. Rate of discovery measures the rate at which new values are experienced during random sampling of the data set. Rate of detection tends to remain uniform for monotonic variables during the whole sampling period and falls off for some nonmonotonic variables.

A problem with these metrics is that there are nonmonotonic variables that also share the characteristics that are used to detect potential monotonicity. Nonetheless, used as warning flags that the variables indicated need looking at more closely for monotonicity or other problems, the metrics are very useful. As noted, automatically modifying the variables into some different form is not possible.

Increasing Dimensionality

The usual problem in mining large data sets is in reducing the dimensionality. There are some circumstances where the dimensionality of a variable needs to be increased. One concern is to increase the dimensionality as much as is needed, but only as little as necessary, by recoding and remapping variables. Chapter 7 deals in part with these techniques. The types of variables requiring this transformation, which are almost always categorical, carry information that is best exposed in more than one dimension. A couple of examples illustrate the point.

Colors can be represented in a variety of ways. Certainly a categorical listing covers the range of humanly appreciated color through a multitude of shades. Equally well, for some purposes, the spectral frequency might be listed. However, color has been usefully mapped onto a color wheel. Such a wheel not only carries color information, but also describes color as a continuum, carrying information about what other colors are near and far from some selected category. This is very useful information. Since a circle can be drawn on a plane, such as a piece of paper, it is easy to see that any point on the circle's circumference can be unambiguously represented by two coordinates, or numbers. Mapping the color wheel onto a circle on a graph and using the two coordinates for some selected color as the instance values of two variables may form a better description of color than a categorical listing.

ZIP codes form a perennial problem in mining. Sometimes, depending on the application, it is beneficial to translate the ZIP code from the categorical list into latitude and longitude. These values translate the ZIP code into two instance values. The single variable "ZIP" translates into two variables, say, "Lat" and "Lon."

Once again, the decision of whether to expand the dimensionality of a variable must be, in many cases, left up to the miner or domain expert.

Outliers

An *outlier* is a single, or very low frequency, occurrence of the value of a variable that is far away from the bulk of the values of the variable. The question miners always ask is: "Is this a mistake?" As a general rule of thumb, if it can be established that it is a mistake, it can be rectified. (One way to do this, if the correct value cannot be found, is to treat it as a missing value, discussed later in this chapter.) The problem is what to do if it cannot be pinpointed as an error. It is a problem because, for some modeling methods in particular (some types of neural network, for instance), outliers may distort the remaining data to the point of uselessness. Figure 2.5(a) shows this sort of situation.

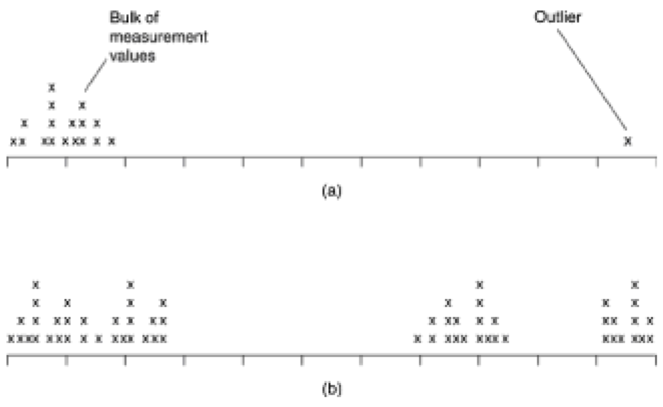


Figure 2.5 Examples of outliers: as an individual value (a) and as clumps of values (b).

Insurance data typically suffers considerably from the problem of outliers. Most insurance claims are small, but occasionally one comes in for some enormous sum. This is no error, and it must be included in modeling. How to do this without distorting the remaining data is a problem.

There is also a problem when the outliers are not individual values but clumps of values, illustrated in Figure 2.5(b). It’s actually the gaps between the clumps that can pose problems. Are these clumps, perhaps, valid measurements from differently biased instruments? Once again, it must be determined first that there is not some sort of error. Maybe some measurements were made against an incorrect calibration and are biased. However, again it might not be possible to determine that an error occurred. In general, the miner is constrained to consider that the measurements are not an error until and unless it is possible to definitely show that they are.

If indeed the outlying value is not a mistake, or is at least going to be dealt with as if it is not, how is it to be treated? Fortunately there is a way of automatically dealing with the problem if it is not a mistake. This involves remapping the variable’s values. Part of Chapter 7 deals with this remapping.

Numerating Categorical Values

Dealing correctly with categorical values is one of the most important functions of data preparation. For many modeling techniques it is necessary to translate categorical values into numbers: they simply cannot deal with untranslated categorical values. Experience shows that even modeling techniques that can deal well with untranslated categorical values benefit from a valid numeration of categoricals.

However, a na‹ve way of making the translation, one that is very commonly done, is terribly destructive of information. Simply assigning numbers to the nominals to create a numbered list is a disastrous way to proceed! To see why, consider the variable “marital status,” for instance, which might be measured as married, single, widowed, divorced, or never married. To simply assign the numbers 1, 2, 3, 4, and 5 to these is totally destructive of the natural structure of the data. If it turned out, for instance, that the natural order of this variable, when translated, was in fact (on a scale of 0–1)

Never married	0
Single	0.1
Divorced	0.15
Widowed	0.65
Married	1

then the “brute force” assignment of numbers from 1–5 not only destroyed the natural ordering of these measures, but even if they were in the right order, it would have destroyed the interval information. Interval information is contained in the distance between the numbers and may be a significant factor in modeling. Except by pure, unadulterated luck, all of the structure contained in this variable would have been destroyed. Worse than that, some meaningless artificial structure has been forced into the data quite arbitrarily!

But what is the “natural order”? The natural order can be found embedded in the system of variables. Recall that the data set reflects to some degree the system of the world. As such, the data set itself forms a system. Thus, embedded in the data set is a structure that reflects an appropriate ordering and distance for categorical values. Assigning values in accord with the system embedded in the data reveals the natural ordering. Arbitrary assignment not only destroys the order, and any information carried by the variable, but actually introduces an artificial pattern to the data.

It is hard to imagine how more damage can be done to the natural ordering of the data than by arbitrary number assignment to categoricals. If it is not intuitively clear how damaging this might be, imagine working for a company that pays you for set periods of a half-day, a day, a half-week, a week, and a month. Perhaps the rate of pay for these periods in dollars might be

Time period	**Rate of pay ($)**
Half-day	100

Day	200
Half-week	500
Week	1000
Half-month	2000
Month	4000

This gives a natural order to these measures. Now, not knowing the actual numerical values, in building a model the modeler lists these periods alphabetically for convenience and assigns numbers to them:

Day	1
Half-day	2
Half-month	3
Half-week	4
Month	5
Week	6

Would you expect this ranking to accurately reflect anything significant about the categories? Compare the relationship between the arbitrary ordering and the monetary value:

Day	1	200
Half-day	2	100
Half-month	3	2000
Half-week	4	500
Month	5	4000
Week	6	1000

It is clear that the natural order of these ordinal values has been completely destroyed. It would, for instance, be impossible to use the arbitrary value assigned to predict how much is earned in each period. Thus, the arbitrary assignment has destroyed the information carried in the ordinal labels.

Regardless of what arbitrary order is given to the measures—whether it be alphabetic, reverse alphabetic, random selection, or just the order they are encountered in the data set—the arbitrary assignment of values destroys information content at best. *At worst it introduces and creates patterns in the data that are not natural and that reflect throughout the data set, wreaking havoc with the modeling process.*

A data set reflects the real world to some extent. (If not, any model will be useless anyway!) Any variables ordinal or higher in information content are, therefore, in an appropriate ordering to some extent. The variables in a data set, because they form an interlocking system, all have specific relationships to each other. It is quite easy, at least for a computer, to reflect the ordering from the ordinal, or higher variables, into the nominal and categorical variables, thus recovering the natural ordering. This process is not perfect. If domain knowledge is available for a more appropriate ordering, this is preferable. Domain expertise reflects far more knowledge of the real world than is enfolded in any data set for mining! Most often, such domain knowledge is unavailable or unknown. Using the information at hand can help enormously.

So, natural orderings can be recovered, at least to some extent, by looking at the data. In a data set that had the pay periods listed in the above tables as categoricals, but without numeric values, it is usually possible to recover, at least to some degree, the natural order and spacing of the measures. In the event that full recovery cannot be made, it is still possible to assign a ranking and position that turn out to be neutral; that is, even if they don’t contribute much information, they at least do not distort the data. One of the key principles in data preparation is to do as little damage as possible to the natural structure in a data set.

Sometimes a nominal variable will fairly easily translate into a single numeric variable. This allows translation of the nominal or categorical, one for one, into a numeric value for modeling. This could have been done in the pay-period example described above if it was possible to recover the value and spacing information. By simply inserting the recovered value, a numeric variable replaces the nominal, one for one, when modeling.

Also note that sometimes a categorical value needs to be expanded into more than one numeric value for reasons similar to those mentioned above during the discussion of increasing the dimensionality of variables. Fortunately, discovering an appropriate numeration of categorical values can be completely automated. Chapter 6 includes a detailed discussion of the technique.

Anachronisms

An *anachronism* is, literally, something out of place in time. Temporal displacement. When mining, an *anachronistic variable* is one that creeps into the variables to be modeled, but that contains information not actually available in the data when a prediction is needed.

For example, in mining a data set to predict people who will take a money market account with a bank, various fields of interest will be set up, one entitled “investor.” This could be a binary field with a “1” indicating people who opened a money market account, and a “0” for the others. Obviously, this is a field to predict. The data set might also include a field entitled “account number” filled in with the issued account number. So far, so good. However, if “account number” is included in the predicting variables, since there is only an account number when the money market account has been opened, it is clearly anachronistic—information not available until after the state of the field to be predicted is known. (Such a model makes pretty good predictions, about 100% accurate—always a suspicious circumstance!)

“Account number” is a fairly straightforward example, but is based on a real occurrence. Easy to spot with hindsight, but when the model has 400–500 variables, it is easy to miss one. Other forms of “leakage” of after-the-fact information can easily happen. It can sometimes be hard to find where the leakage is coming from in a large model. In one telephone company churn application, the variables did not seem to be at all anachronistic. However, the models seemed to be too good to be believed. In order to get information about their customers, the phone company had built a database accumulated over time based on customer interviews. One field was a key that identified which interviewer had conducted the interview. It turned out that some interviewers were conducting general interviews, and others were conducting interviews after the customer had left, or churned. In fact, the interviewer code was capturing information about who had churned! Obviously an anachronistic variable, but subtle, and in this case hard to find.

One of the best rules of thumb is that if the results seem to good to be true, they probably are. Anachronistic variables simply have to be removed.

2.6.3 Building Mineable Data Sets

Looking at data sets involves considering the relationships between variables. There is also a natural structure to the interrelationships between variables that is just as critical to maintain as it is within variables. Mining tools work on exploring the interactions, or relationships, that exist between the collected variables. Unfortunately, simply preparing the variables does not leave us with a fully prepared data set. Two separate areas need to be looked at in the data set: exposing the information content and getting enough data.

A first objective in preparing the data set is to make things as easy as possible for the mining tool. It is to prepare the data in such a way that the information content is best revealed for the tool to see. Why is it important to make the mining tools' job easier? Actually, there are important reasons. A brief discussion follows in the next section.

Some types of relationships cause problems for modeling tools. A second objective in preparing the data set, then, is to obviate the problems where possible. We will look briefly at some of those. If it is possible to detect such potentially damaging relationships, even without being able to ameliorate them automatically, that is still very useful information. The miner may be able to take corrective or remedial action, or at least be aware of the problem and make due allowance for it. If there is some automatic action that can correct the problem, so much the better.

Exposing the Information Content

Since the information is enfolded in the data, why not let the mining tool find it?

One reason is time. Some data sets contain very complex, involved relationships. Often, these complex relationships are known beforehand. Suppose in trying to predict stock market performance it is believed that the "trend" of the market is important. If indeed that is the case, and the data is presented to a suitable modeling tool in an appropriate way, the tool will no doubt develop a "trend detector." Think, for a moment, of the complexity of calculation involved in creating a trend measurement.

A simple measurement of trend might be to determine that if the mean of the last three days' closing prices is higher than the mean of the previous three days' prices, the trend is "up." If the recent mean is lower than the older mean, the trend is "down." If the means are the same, the trend is "flat." Mathematically, such a relationship might be expressed as

$$t = \frac{p_{i-1} + p_{i-2} + p_{i-3}}{3} - \frac{p_{i-4} + p_{i-5} + p_{i-6}}{3}$$

where *t* is trend and *p* is closing price for day *i*. This is a modestly complex expression yielding a positive or negative number that can be interpreted as measuring trend. For a human it takes insight and understanding, plus a knowledge of addition, subtraction, multiplication, and division, to devise this measure. An automated learning tool can learn this. It takes time, and repeated attempts, but such relationships are not too hard. It may take a long time, however, especially if there are a large number of variables supplied to the mining tool. The tool has to explore all of the variables, and many possible relationships, before this one is discovered.

For this discussion we assumed that this relationship was in fact a meaningful one, and that after a while, a mining tool could discover it. But why should it? The relationship was already known, and it was known that it was a useful relationship. So the tool would have discovered a known fact. Apart from confirmation (which is often a valid and useful reason for mining), nothing new has yet been achieved. We could have started from this point, not worked to get here. Giving the tool this relationship to begin with would have sped up the process, perhaps very much. The more complex the relationship, the more the speed is improved.

However, a second reason for providing as much help as possible to the tool is much more important for the end result, but directly related to the time factor. The reason is noise. The longer that training continues, the more likely it is that noise will be learned along with the underlying pattern. In the training set, the noisy relationship is every bit as real as any other. The tool cannot discriminate, inside the training set, between the noise and target patterns.

The relationships in data are known as *features* of the data. The trend that, for this example, is assumed to be a valid relationship is called a *predictive* feature. Naturally, it’s desirable for the tool to learn all of the valid predictive features (or *inferential* features if it is an inferential model that is needed) without learning noise features. However, as training continues it is quite possible that the tool learns the noise and thereby misses some other feature. This obscuring of one feature by another is called *feature swamping*.

By including relevant domain knowledge, the mining tool is able to spend its time looking for other features enfolded in the data, and not busy itself rediscovering already known relationships. In fact, there is a modeling technique that involves building the best model prior to overfitting, taking a new data set, using the model to make predictions, and feeding the predictions plus new training data into another round of mining. This is done precisely to give the second pass with the tool a “leg up” so that it can spend its time looking for new features, not learning old ones.

In summary, exposing the information content is done partly to speed the modeling process, but also to avoid feature swamping. Searching for meaningful fine structure involves removing the coarser structure. In other words, if you want to find gold dust, move the rocks out of the way first!

Getting Enough Data

The discussion about preparing variables started with getting sufficient data to be sure that there were enough instance values to represent the variable’s actual features. The same is true for data sets. Unfortunately, getting enough of each variable to ensure that it is representative does not also assure that a representative sample of the data set has been captured. Why? Because now we’re interested in the interactions between variables, not just the pattern existing within a variable.

Figure 2.6 explains why there is a difference. Consider two variables, instance values of one of them plotted on the vertical axis, and the other on the horizontal axis. The marks on the axes indicate the range of the individual variables. In addition to distributing the individual values on the axes, there is a joint range of values that is shown by the ellipse.

This ellipse shows for this hypothetical example where the actual real-world values might fall. High values of variable 1 always correspond with low values of variable 2, and vice versa. It is quite possible to select joint values that fall only in some restricted part of the joint distribution, and yet still cover the full range of the individual variables. One way in which this could occur is shown in the shaded part of the ellipse. If joint values were selected that fell only inside the shaded area, it would be possible to have the full range of each variable covered and yet only cover part of the joint distribution. In fact, in the example, half of the joint distribution range is not covered at all. The actual method used to select the instance values means that there is only a minute chance that the situation used for the illustration would ever occur. However, it is very possible that simply having representative distributions for individual variables will not produce a fully representative joint distribution for the data set. In order to assure complete coverage of the joint distribution, every possible combination of variables has to be checked, and that can become impossible very quickly indeed!

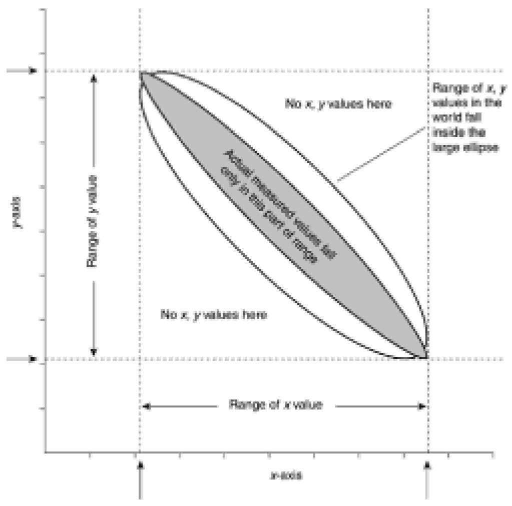
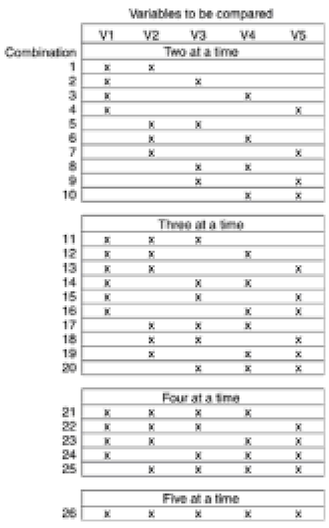

No x, y values here
Range of x, y values in the world fall inside the large ellipse
Actual measured values fall only in this part of range
y-axis
Range of y value
No x, y values here
Range of x value
x-axis

Figure 2.6 Joint occurrence of two variables may not cover the individual range of each. Values falling in only part of the full range, illustrated by half of the ellipse, may cover the full range of each variable, but not the full joint range.

The Combinatorial Explosion

With five variables, say, the possible combinations are shown in Figure 2.7. You can see that the total number of combinations is determined by taking the five variables two at a time, then three at a time, then four at a time. So, for any number of variables, the number of combinations is the sum of all combinations from two to the total number of variables. This number gets very large, very quickly! Table 2.2 shows just how quickly.



	Variables to be compared				
	V1	V2	V3	V4	V5
Combination	Two at a time				
1	x	x			
2	x		x		
3	x			x	
4	x				x
5		x	x		
6		x		x	
7		x			x
8			x	x	
9			x		x
10				x	x
	Three at a time				
11	x	x	x		
12	x	x		x	
13	x	x			x
14	x		x	x	
15	x		x		x
16	x			x	x
17		x	x	x	
18		x	x		x
19		x		x	x
20			x	x	x
	Four at a time				
21	x	x	x	x	
22	x	x	x		x
23	x	x		x	x
24	x		x	x	x
25		x	x	x	x
	Five at a time				
26	x	x	x	x	x

Figure 2.7 Combinations of five variables compared against each other, from two at a time andÿ20increasing to five at a time.

TABLE 2.2 The combinatorial explosion.

Number of variables	Number of combinations
5	26
7	120
9	502
20	1,048,555
25	33554406

This “blowup” in the number of combinations to consider is known as the *combinatorial explosion* and can very quickly defeat any computer, no matter how fast or powerful. (Calculating combinations is briefly described in the Supplemental Material section at the end of this chapter.) Because there is no practical way to check for every combination that the intervariable variability has been captured for the data set, some other method of estimating (technical talk for guessing!) if the variability has been captured needs to be used. After all, some estimate of variability capture is needed. Without such a measure, there is no way to be certain how much data is needed to build a model.

The expression of certainty is the key here and is an issue that is mentioned in different contexts many times in this book. While it may not be possible to have 100% confidence that the variability has been captured, it reduces the computational work enormously if some lesser degree of confidence is acceptable. Reducing the demanded confidence from 100% to 99%, depending on the number of variables, often changes the task from impossible to possible but time-consuming. If 98% or 95% confidence is acceptable, the estimating task usually becomes quite tractable. While confidence measures are used throughout the preparation process, their justification and use are discussed in Chapter 5. Chapter 10 includes a discussion on capturing the joint variability of multiple variables.

Missing and Empty Values

As you may recall, the difference between "empty" and "missing" is that the first has no corresponding real-world value, while the second has an underlying value that was not captured. Determining if any particular value is empty rather than missing requires domain knowledge and cannot be automatically detected. If possible, the miner should differentiate between the two in the data set. Since it is impossible to automatically differentiate between missing and empty, if the miner cannot provide discriminating information, it is perforce necessary to deal with all missing values in a similar way. In this discussion, they will all be referred to as missing.

Some mining tools use techniques that do not require the replacement of missing values. Some are able to simply ignore the missing value itself, where others have to ignore the instance (record) altogether. Other tools cannot deal with missing values at all, and have to have some default replacement for the missing value. Default replacement techniques are often damaging to the structure of the data set. The discussion on numerating categorical values discusses how arbitrary value replacement can damage information content.

The general problem with missing values is twofold. First, there may be some information content, predictive or inferential, carried by the actual pattern of measurements missing. For example, a credit application may carry useful information in noting which fields the applicant did not complete. This information needs to be retained in the data set.

The second problem is in creating and inserting some replacement value for the missing value. The objective is to insert a value that neither adds nor subtracts information from the data set. It must introduce no bias. But if it introduces no new information, why do it?

First, default replacement methods often do introduce bias. If not correctly determined, a poorly chosen value adds information to the data set that is not really present in the world, thus distorting the data. Adding noise and bias of this sort is always detrimental to modeling. If a suitable value can be substituted for the missing values, it prevents the distortion introduced by poorly chosen defaults.

Second, for those modeling tools that have to ignore the whole instance when one of the values is missing, plugging the holes allows the instance to be used. That instance may carry important information in the values that are present, and by plugging the holes that information is made available to the modeling tool.

There are several methods that can be used to determine information-neutral values to replace the missing values. Chapter 8 discusses the issues and techniques used. All of them involve caveats and require knowledgeable care in use.

Although the values of individual variables are missing, this is an issue in preparing the data set since it is only by looking at how the variable behaves vis-…-vis the other variables when it is present that an appropriate value can be determined to plug in when it is missing. Of course, this involves making a prediction, but in a very careful way such that no distortion is introduced—at least insofar as that is possible.

The Shape of the Data Set

The question of the shape of the data set is not a metaphorical one. To understand why, we have to introduce the concept of *state space*.

State space can be imagined to be a space like any other—up to a point. It is called state space because of the nature of the instances of data. Recall that each instance captures a number of measurements, one per variable, that were measured under some validating circumstance. An instance, then, represents the state of the object at validation. That is where the “state” part of the phrase comes from. It is a space that reflects the various states of the system as measured and captured in the instance values.

That’s fine, but where does “space” come from? Figure 2.6, used earlier to discuss the variability of two variables, shows a graphical representation of them. One variable is plotted on one axis, and the other variable is plotted on the other axis. The values of the combined states of the two variables can easily be plotted as a single point on the graph. One point represents both values simultaneously. If there were three variables’ values, they could be plotted on a three-dimensional graph, perhaps like the one shown in Figure 2.8. Of course, this three-dimensional object looks like something that might exist in the world. So the two- and three-dimensional representations of the values of variables can be thought of as determining points in some sort of space. And indeed they do—in state space.

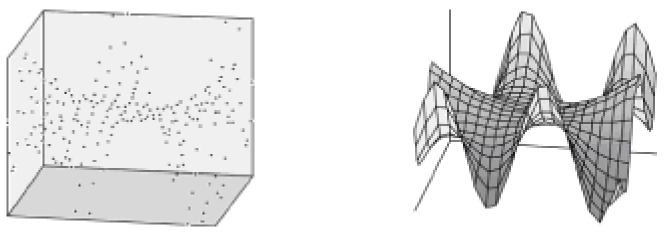


Figure 2.8 Points plotted in a 3D phase space (left) can be represented by a manifold (right).

State space can be extended to as many dimensions as there are variables. It is mathematically and computationally fairly easy to deal with state spaces of large numbers of dimensions. For description, it is very difficult to imagine what is going on in high-dimensional spaces, except by analogy with two- and three-dimensional spaces. When describing what is going on in state space, only two or three dimensions will be used here.

The left image in Figure 2.8 shows a three-dimensional state space, actually an *x*, *y*, *z* plot of the values of three variables. Wherever these points fall, it is possible to fit a sheet (a flexible two-dimensional plane) through them. If the sheet is flexible enough, it is possible to bend it about in state space until it best fits the distribution of points. (We will leave aside the issue of defining "best" here.) The right-hand image in Figure 2.8 shows how such a sheet might look. There may be some points that do not fall onto the sheet exactly when the best fit is made, making a sort of "fuzz" around the sheet.

State space is not limited to three dimensions. However, a sheet squeezed into two dimensions is called a line. What would it be called in four dimensions? Or five? Or six? A general name for the *n*-dimensional extension of a line or sheet is a *manifold*. It is analogous to a flexible sheet as it exists in three dimensions, but it can be spread into as many dimensions as required.

In state space, then, the instance values can all be represented as points defined by the values of the variables—one variable per dimension. A manifold can in some "best fit" way be spread through state space so it represents the distribution of the points. The fit of the manifold to the points may not be perfect, so that the points cluster about the manifold's surface, forming "fuzz."

The actual shape of the manifold may be exceedingly complex, and in some sense, mining tools are exploring the nature of the shape of the manifold. In the same way that the *X*, *Y* graph in two dimensions represents the relationship of one variable to another, so the manifold represents the joint behavior of the variables, one to another, and one to all of the others. However, we are now in a position to examine the question asked at the beginning of this section: What shape is the data in?

The question now becomes one of characterizing the manifold.

- If, for instance the “fuzz” is such that the manifold hardly represents the data at all over some portion of its surface, modeling in that area is not likely to produce good results.

- In another area there may be very few data points around in state space to define the shape of the manifold. Here, explore the shape as we might, the results will be poor too, but for a different reason than described above.

- Elsewhere the shape of the manifold may be well defined by the data, but have problematic shapes. For instance, it might be folded over on itself rather like a breaking wave. Many modeling tools simply cannot deal with such a shape.

- It is possible that the manifold has donutlike holes in it, or higher-dimensional forms of them anyway.

- There could be tunnels through the manifold.

There are quite a number of problems with data sets that can be described as problems with the shape of the manifold. In several of these cases, adjustments can be made. Sometimes it is possible to change the shape of the manifold to make it easier to explore. Sometimes the data can be enriched or enhanced to improve the manifold definition.

Many of these techniques for evaluating the data for problems are a part of data surveying. The data survey is made prior to modeling to better understand the problems and limitations of the data before mining. Where this overlaps with data preparation is that sometimes adjustments can be made to ameliorate problems before they arise. Chapter 6 explores the concept of state space in detail. Chapter 11 discusses the survey and those survey techniques that overlap with data preparation. In itself, making the survey is as large a topic as data preparation, so the discussion is necessarily limited.

2.7 Summary

This chapter has looked at how the world can be represented by taking measurements about objects. It has introduced the ideas of data and the data set, and various ways of structuring data in order to work with it. Problems that afflict the data and the data set (and also the miner!) were introduced. All of this data, and the data set, enfolds information, which is the reason for mining data in the first place.

The next chapter looks at the process of mining. Just as this chapter briefly examined the nature of data to provide a framework for the rest of the book, so the next chapter introduces the nature of what it is to prepare data for mining. And just as this chapter did not solve the problems discussed, so too the next chapter does not solve all of the problems of mining or data preparation! Solving the problems discussed must wait until later chapters when this introductory look at the territory is complete. This point is halfway through the introduction to the nature of the territory. We've looked at how data connects to the world, and now turn our attention to how preparation addresses data.

Supplemental Material

Combinations

The formula for determining how many combinations may be taken from *n* objects, *r* at a time, is

$$C_r^n = \frac{n!}{r!(n-r)!}$$

The symbol ! indicates that the factorial of the quantity is to be used. A factorial of any number may be found by multiplying the number by one less than itself, and one less than that, and so on from the number to 1. So

8! = 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1 = 40,320

If *n* = 5 and *r* = 3, then

$$C_3^5 = \frac{5!}{3!(5-3)!} = \frac{120}{6(2)!} = \frac{120}{12} = 10$$

In determining the full number of variable comparisons needed for 10 variables, all of the combinations of variables from 2 to 10 have to be summed:

$$C_2^{10} + C_3^{10} + C_4^{10} + C_5^{10} + C_6^{10} + C_7^{10} + C_8^{10} + C_9^{10} + C_{10}^{10}$$

A more convenient way of writing this expression is to use the summation notation:

$$\sum_{i=2}^{10} C_i^{10}$$

The sigma symbol "x" indicates repetitive addition. The "*i* = 2" indicates that in the expression to the right of the sigma, the symbol *i* should first be replaced with a "2." The "10" above the sigma indicates that the replacement should continue until 10 is reached. The expression to the right of the sigma is the notation indicating combination.

Thus it is that

$$C_2^{10} + C_3^{10} + C_4^{10} + C_5^{10} + C_6^{10} + C_7^{10} + C_8^{10} + C_9^{10} + C_{10}^{10} = \sum_{i=2}^{10} C_i^{10}$$

The only difference is that the sigma notation is more compact.

Chapter 3: Data Preparation as a Process

Overview

Data preparation has been placed in the context of data exploration, in which the problem to be solved, rather than the technology, is paramount. Without identifying the problem to solve, it is hard to define how to extract value from the data mining activities that follow. Equally important is specifying the form of a solution. Without a firm idea of what success looks like, it is hard to determine if indeed the result found, and the form that it is delivered in, have actually succeeded. Having specified what a suitable solution looks like, and collected or discovered appropriate data, you can begin the process of data mining.

Data mining is about working with data, which to a greater or lesser degree reflects some real-world activity, event, or object. In this discussion of data preparation for mining, there is a close focus on exploring more exactly what data represents, how and why it is transformed, and what can be done with and said about it. Much more will be said about data as the techniques for manipulating it are introduced. However, before examining how and why data is manipulated, a missing piece still remains to be addressed. Data needs to be prepared so that the information enfolded within it is most easily accessed by the mining tools. The missing piece, the bridge to understanding, is the explanation of what the overall process looks like. The overview of the process as a whole provides a framework and a reference to understand where each component fits into the overall design. This chapter provides the overview. Most detail is deliberately left out so that the process may be seen holistically. The questions that must arise from such a quick dash across the landscape of data preparation are answered in later chapters when each area is revisited in more detail.

Preparation of data is not a process that can be carried out blindly. There is no automatic tool that can be pointed at a data set and told to just “fix” the data. Maybe one day, when artificial intelligence techniques are a good bit more intelligent than they are today, fully automatic data preparation will become more feasible. Until that day there will remain as much art as science in good data preparation. However, just because there is art involved in data preparation does not mean that powerful techniques are not available or useful.

Because data preparation techniques cannot be completely automated, it is necessary to apply them with knowledge of their effect on the data being prepared. Understanding their function and applicability may be more important than understanding how the tools actually work. The functionality of each tool can be captured in computer code and regarded as a “black box.” So long as the tools perform reliably and as intended, knowledge of how the transformations are actually performed is far less important than understanding the appropriate use and limitations of each of the encapsulated techniques.

Art there may be, but successful practice of the art is based on understanding the overall issues and objectives, and how all the pieces relate together. Gaining that understanding of the broad picture is the purpose of this chapter. It connects the description of the data exploration process, data, data sets, and mining tools with data preparation into a whole. Later chapters discuss the detail of what needs to be done to prepare data, and how to do it. This chapter draws together these themes and discusses when and why particular techniques need to be applied and how to decide which technique, from the variety available, needs to be used.

3.1 Data Preparation: Inputs, Outputs, Models, and Decisions

The process takes inputs and yields outputs. The inputs consist of raw data and the miner's decisions (selecting the problem, possible solution, modeling tools, confidence limits, etc.). The outputs are two data sets and the Prepared Information Environment (PIE) modules. Figure 3.1 illustrates this. The decisions that have to be made concern the data, the tools to be used for mining, and those required by the solution.

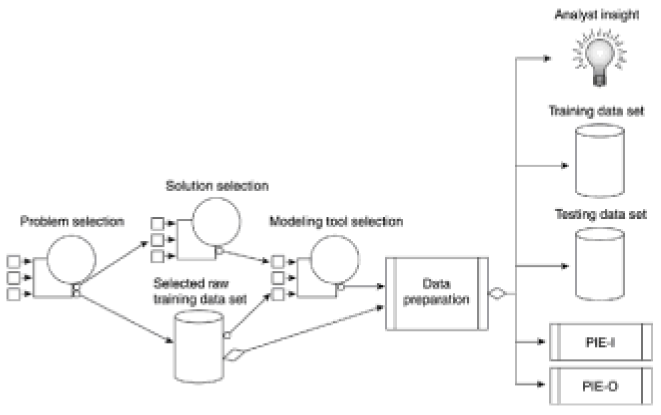


Figure 3.1 The data preparation process illustrating the major decisions, data, and process inputs and outputs.

This section explains

- What the inputs are, what the outputs are, what they do, and why they're needed
- How modeling tools affect what is done
- The stages of data preparation and what needs to be decided at each stage

The fundamental purpose of data preparation is to manipulate and transform raw data so that the information content enfolded in the data set can be exposed, or made more easily accessible. The best way to actually make the changes depends on two key decisions: what the solution requires and what the mining tool requires. While these decisions affect how the data is prepared, the inputs to and outputs from the process are not affected.

During this overview of data preparation, the actual inner workings of the preparation process will be regarded as a black box. The focus here is in what goes into and what comes out of the preparation process. By ignoring the details of the actual preparation process at this stage, it is easier to see why each of the inputs is needed, and the use of each of the output pieces. The purpose here is to try to understand the relationships between all of the pieces, and the role of each piece. With that in place, it is easier to understand the necessity of each step of the preparation process and how it fits into the whole picture.

At the very highest level, mining takes place in three steps:

1. Prepare the data

2. Survey the data

3. Model the data

Each of these steps has different requirements in the data preparation process. Each step takes place separately from the others, and each has to be completed before the next can begin. (Which doesn't mean that the cycle does not repeat when results of using the model are discovered. Getting the model results might easily mean that the problem or solution needs to be redefined, or at least that more/different/better data is found, which starts off the cycle afresh.)

3.1.1 Step 1: Prepare the Data

Figure 3.1 shows the major steps in the data preparation process. Problem selection is a decision-and-selection process affecting both solution selection and data selection. This has been extensively discussed in Chapter 1 and will not be reiterated here. Modeling tool selection is driven by the nature of the specified solution and by the data available, which is discussed later in this chapter in "Modeling Tools and Data Preparation." Chapter 12 discusses tool use and the effect of using prepared data with different techniques.

Some initial decisions have to be made about how the data is to be prepared. In part, the nature of the problem determines tool selection. If rules are needed, for example, it is necessary to select a tool that can produce them. In turn, tool selection may influence how the data is prepared. Inspection of the data may require reformatting or creating some additional features. Looking at the preliminary decisions that need to be made before applying the appropriate techniques is covered in part in this chapter and also in the next.

The miner must determine how the data is to be appropriately prepared. This is based on the nature of the problem, the tools to be used, and the types of variables in the data set. With this determined, preparation begins. Preparation has to provide at least four separate components as outputs:

- A training data set
- A testing data set
- A PIE-I (Prepared Information Environment Input module)
- A PIE-O (Prepared Information Environment Output module)

Each of these is a necessary output and has a specific function, purpose, and use. Each is needed because of the nature of data sets extracted from the real world. These four components are the absolute minimum required for mining, and it is likely that additional data sets will be needed. For example, a validation data set may also be considered essential. It is not included in the list of four essential components since valid models can be created without actually validating them at the time the miner creates them. If there is insufficient data on hand for three representative data sets, for instance, the model could be validated later when more data is available. But in some sense, each of these four components is indispensable. Why these four?

The training data set is required to build a model. A testing data set is required for the modeling tool to detect overtraining. The PIE-I is what allows the model to be applied to other data sets. The PIE-O translates the model's answers into applicable measured values. Since these are the critical output components of the data preparation process, we must look at each of these four components more closely.

A mining tool's purpose is to learn the relationships that exist between the variables in the data set. Preparation of the training data set is designed to make the information enfolded in the data set as accessible and available as possible to the modeling tool. So what's the purpose of the test data set?

Data sets are not perfect reflections of the world. Far from it. Even if they were, the nature of the measuring process necessarily captures uncertainty, distortion, and noise. This noise is integral to the nature of the world, not just the result of mistakes or poor procedures. There are a huge variety of errors that can infect data. Many of these errors have already been discussed in Chapter 2—for instance, measurement error. Some of these errors are an inextricable part of the data and cannot be removed or "cleaned." The accumulated errors, and other forms of distortion of "true" values, are called *noise*. The term "noise" comes from telephony, where the added error to the true signal is actually heard as the noise of a hiss in a telephone earpiece. AM radio also suffers from noise in the transmitted signal, especially if lightning is nearby. In general, noise simply means distortion of the original signal. Somehow a modeling tool must deal with the noise in the data.

Each modeling tool has a different way of expressing the nature of the relationships that it finds between variables. But however it is expressed, some of the relationship between variables exists because of the “true” measurement and some part is made up of the relationship caused by the noise. It is very hard, if not impossible, to precisely determine which part is made up from the underlying measurement and which from the noise. However, in order to discover the “true” underlying relationship between the variables, it is vital to find some way of estimating which is relationship and which is noise.

One problem with noise is that there is no consistent detectable pattern to it. If there were, it could be easily detected and removed. So there is an unavoidable component in the training set that should not be characterized by the modeling tool. There are ways to minimize the impact of noise that are discussed later, but there always remains some irreducible minimum. In fact, as discussed later, there are even circumstances when it is advantageous to add noise to some portion of the training set, although this deliberately added noise is very carefully constructed.

Ideally, a modeling tool will learn to characterize the underlying relationships inside the data set without learning the noise. If, for example, the tool is learning to make predictions of the value of some variable, it should learn to predict the true value rather than some distorted value. During training there comes a point at which the model has learned the underlying relationships as well as is possible. Anything further learned from this point will be the noise. Learning noise will make predictions from data inside the training set better. In any two subsets of data drawn from an identical source, the underlying relationship will be the same. The noise, on the other hand, not representing the underlying relationship, has a very high chance of being different in the two data sets. In practice, the chance of the noise patterns being different is so high as to amount to a practical certainty. This means that predictions from any data set other than the training data set will very likely be worse as noise is learned, not better. It is this relationship between the noise in two data sets that creates the need for another data set, the test data set.

To illustrate why the test data set is needed, look at Figure 3.2. The figure illustrates measurement values of two variables; these are shown in two dimensions. Each data point is represented by an X. Although an X is shown for convenience, each X actually represents a fuzzy patch on the graph. The X represents the actual measured value that may or may not be at the center of the patch. Suppose the curved line on the graph represents the underlying relationship between the two variables. The Xs cluster about the line to a greater or lesser degree, displaced from it by the noise in the relationship. The data points in the left-hand graph represent the training data set. The right-hand graph represents the test data set. The underlying relationship is identical in both data sets. The difference between the two data sets is only the noise added to the measurements. The noise means that the actual measured data points are not identically positioned in the two data sets. However, although different in values, note that by using the appropriate data preparation techniques discussed later in the book (see, for example, Chapter 11), it can be known that both data sets do adequately represent the underlying relationship even though the relationship itself is not known.

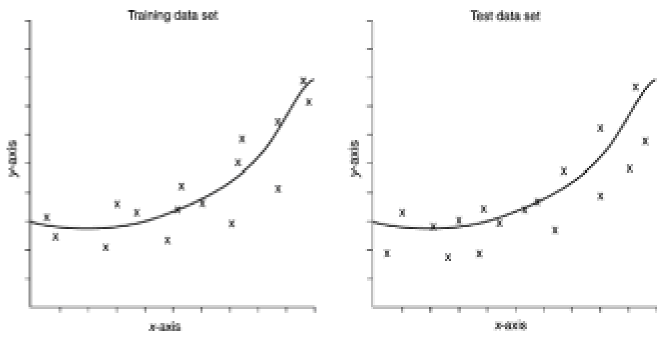


Figure 3.2 The data points in the training and test data sets with the underlying relationship illustrated by the continuous curved lines.

Suppose that some modeling tool trains and tests on the two data sets. After each attempt to learn the underlying relationship, some metric is used to measure the accuracy of the prediction in both the training and test data sets. Figure 3.3 shows four stages of training, and also the fit of the relationship proposed by the tool at a particular stage. The graphs on the left represent the training data set; the graphs on the right represent the test data set.

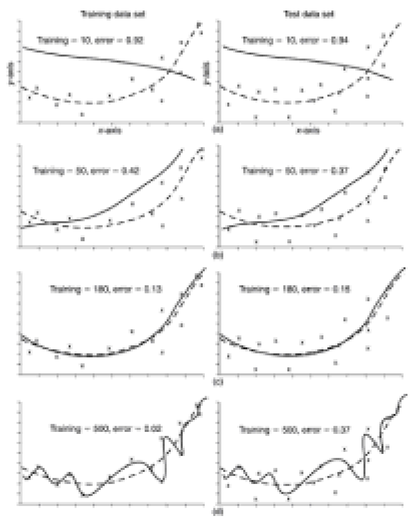


Figure 3.3 The four stages of training with training data sets (left) and test data sets (right): poor fit (a), slightly improved fit due to continued training (b), near-perfect fit (c), and noise as a result of continued training beyond best fit point (d).

In Figure 3.3(a), the relationship is not well learned, and it fits both data sets about equally poorly. After more training, Figure 3.3(b) shows that some improvement has occurred in learning the relationship, and again the error is now lower in both data sets, and about equal. In Figure 3.3(c), the relationship has been learned about as well as is possible from the data available, and the error is low, and about equal in both data sets. In Figure 3.3(d), learning has continued in the training (left) data set, and an almost perfect relationship has been extracted between the two variables. The problem is that the modeling tool has learned noise. When the relationship is tried in the test (right) data set, it does not fit the data there well at all, and the error measure has increased.

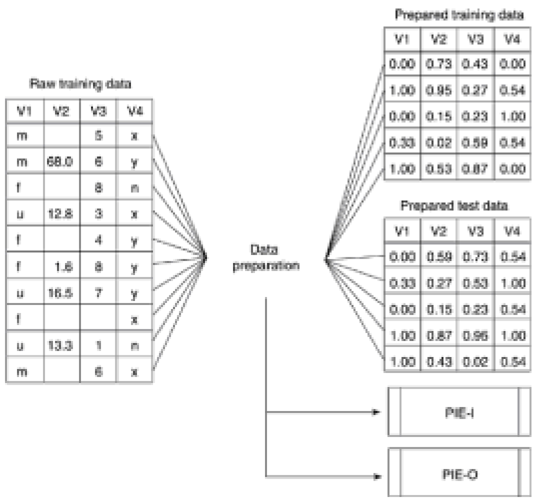
As is illustrated here, the test data set has the same underlying “true” relationships as the training data set, but the two data sets contain noise relationships that are different. During training, if the predictions are tested in both the training and test data sets, at first the predictions will improve in both. So the tool is improving its real predictive power as it learns the underlying relationships and improves its performance based on those relationships. In the example shown in Figure 3.3, real-world improvement continues until the stage shown in Figure 3.3(c). At that point the tool will have learned the underlying relationships as well as the training data set allows. Any further improvement in prediction will then be caused by learning noise. Since the noise differs between the training set and the test set, this is the point at which predictive performance will degrade in the test set. This degradation begins if training continues after the stage shown in Figure 3.3(c), and ends up with the situation shown in Figure 3.3(d). The time to stop learning is at the stage in Figure 3.3(c).

As shown, the relationships are learned in the training data set. The test data set is used as a check to try to avoid learning noise. Here is a very important distinction: the training data set is used for discovering relationships, while the test data set is used for discovering noise. The instances in the test data set are *not valid* for independently testing any predictions. This is because the test data has in fact been used by the modeling tool as part of the training, albeit for noise. In order to independently test the model for predictive or inferential power, yet another data set is needed that does not include any of the instances in either the training or test data sets.

So far, the need for two learning sets, training and test, has been established. It may be that the miner will need another data set for assessing predictive or inferential power. The chances are that all of these will be built from the same source data set, and at the same time. But whatever modifications are made to one data set to prepare it for modeling must also be made to any other data set. This is because the mining tool has learned the relationships in prepared data. The tool has to have data prepared in all data sets in an identical way. Everything done in one has to be done in all. But what do these prepared data sets look like? How does the preparation process alter the data?

Figure 3.4 shows the data view of what is happening during the data preparation process.

The raw training data in this example has a couple of categorical values and a couple of numeric values. Some of the values are missing. This raw data set has to be converted into a format useful for making predictions. The result is that the training and test sets will be turned into all numeric values (if that is what is needed) and normalized in range and distribution, with missing values appropriately replaced. These transformations are illustrated on the right side of Figure 3.4. It is obvious that all of the variables are present and normalized. (Figure 3.4 also shows the PIE-I and PIE-O. These are needed for later use.)



Raw training data

V1	V2	V3	V4
m		5	x
m	68.0	6	y
f		8	n
u	12.8	3	x
f		4	y
f	1.6	8	y
u	16.5	7	y
f			x
u	13.3	1	n
m		6	x

Prepared training data

V1	V2	V3	V4
0.00	0.73	0.43	0.00
1.00	0.95	0.27	0.54
0.00	0.15	0.23	1.00
0.33	0.02	0.59	0.54
1.00	0.53	0.87	0.00

Prepared test data

V1	V2	V3	V4
0.00	0.59	0.73	0.54
0.33	0.27	0.53	1.00
0.00	0.15	0.23	0.54
1.00	0.87	0.96	1.00
1.00	0.43	0.02	0.54

Figure 3.4 Data preparation process transforms raw data into prepared training and test sets, together with the PIE-I and PIE-O modules.

3.1.2 Step 2: Survey the Data

Mining includes surveying the data, that is, taking a high-level overview to discover what is contained in the data set. Here the miner gains enormous and powerful insight into the nature of the data. Although this is an essential, critical, and vitally important part of the data mining process, we will pass quickly over it here to continue the focus on the process of data preparation.

3.1.3 Step 3: Model the Data

In this stage, the miner applies the selected modeling tool to the training and test data sets to produce the desired predictive, inferential, or other model desired. (See Figure 3.5.) Since this book focuses on data preparation, a discussion of modeling issues, methods, and techniques is beyond the present scope. For the purposes here it will be assumed that the model is built.

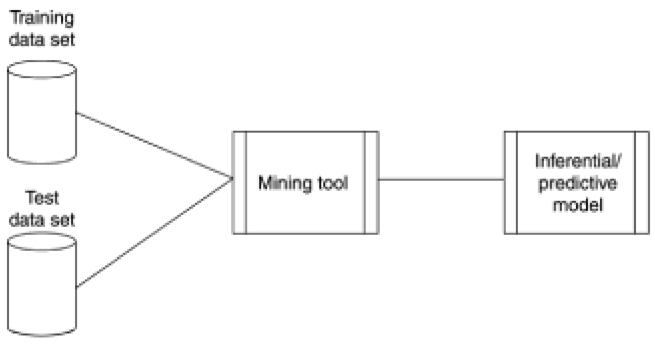
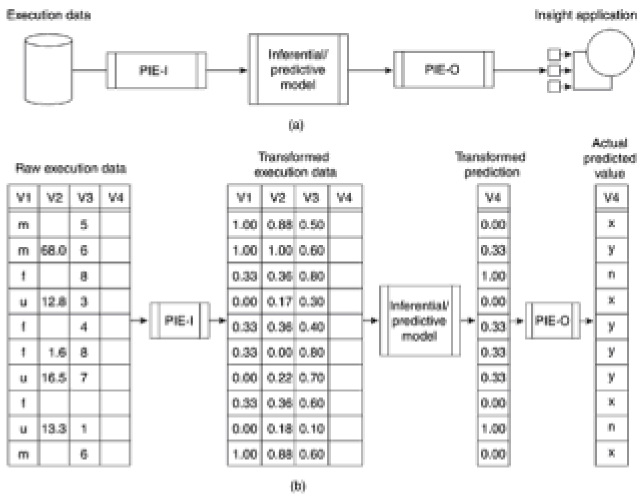


Figure 3.5 Mining the inferential or predictive model.

3.1.4 Use the Model

Having created a satisfactory model, in order to be of practical use it must be applied to "live" data, also called the *execution data*. Presumably, it is very similar in character to the training and test data. It should, after all, be drawn from the same population (discussed in Chapter 5), or the model is not likely to be applicable. Because the execution data is in its "raw" form, and the model works only with prepared data, it is necessary to transform the execution data in the same way that the training and test data were transformed. That is the job of the PIE-I: it takes execution data and transforms it as shown in Figure 3.6(a). Figure 3.6(b) shows what the actual data might look like. In the example it is variable V4 that is missing and needs to be predicted.



Raw execution data

V1	V2	V3	V4
m		5	
m	68.0	6	
f		8	
u	12.8	3	
f		4	
f	1.6	8	
u	16.5	7	
f			
u	13.3	1	
m		6	

Transformed execution data

V1	V2	V3	V4
1.00	0.88	0.50	
1.00	1.00	0.60	
0.33	0.36	0.80	
0.00	0.17	0.30	
0.33	0.36	0.40	
0.33	0.00	0.80	
0.00	0.22	0.70	
0.33	0.36	0.60	
0.00	0.18	0.10	
1.00	0.88	0.60	

Transformed prediction

V4
0.00
0.33
1.00
0.00
0.33
0.33
0.33
0.00
1.00
0.00

Actual predicted value

V4
x
y
n
x
y
y
y
x
n
x

Figure 3.6 Run-time prediction or inferencing with execution data set (a). Stages that the data goes through during actual inference/prediction process (b).

Variable V4 is a categorical variable in this example. The data preparation, however, transformed all of the variables into scaled *numeric* values. The mined model will therefore predict the result in the form of scaled numeric values. However, the prediction must be given as a categorical value. This is the purpose of the PIE-O. It "undoes" the effect of the PIE-I. In this case, it converts the mined model outputs into the desired categorical values.

The whole purpose of the two parts of the PIE is to sit between the real-world data, cleaning and preparing the incoming data stream identically with the way the training and test sets were prepared, and converting predicted, transformed values back into real-world values. While the input execution data is shown as an assembled file, it is quite possible that the real-world application has to be applied to real-time transaction data. In this case, the PIE dynamically prepares each instance value in real time, taking the instance values from whatever source supplies them.

3.2 Modeling Tools and Data Preparation

As always, different tools are valuable for different jobs. So too it is with the modeling tools available. Prior to building any model, the first two questions asked should be: What do we need to find out? and Where is the data? Deciding what to find out leads to the next two questions: Exactly what do we want to know? and In what form do we want to know it? (These are issues discussed in Chapter 1.) A large number of modeling tools are currently available, and each has different features, strengths, and weaknesses. This is certainly true today and is likely to be even more true tomorrow. The reason for the greater differences tomorrow lies in the way the tools are developing.

For a while the focus of data mining has been on algorithms. This is perhaps natural since various machine-learning algorithms have competed with each other during the early, formative stage of data exploration development. More and more, however, makers of data exploration tools realize that the users are more concerned with business problems than algorithms. The focus on business problems means that the newer tools are being packaged to meet specific business needs much more than the early, general-purpose data exploration tools. There are specific tools for market segmentation in database marketing, fraud detection in credit transactions, churn management for telephone companies, and stock market analysis and prediction, to mention only four. However, these so-called "vertical market" applications that focus on specific business needs do have drawbacks. In becoming more capable in specific areas, usually by incorporating specific domain knowledge, they are constrained to produce less general-purpose output. As with most things in life, the exact mix is a compromise.

What this means is that the miner must take even more care now than before to understand the requirements of the modeling tool in terms of data preparation, especially if the data is to be prepared "automatically," without much user interaction. Consider, for example, a futures-trading automation system. It may be intended to predict the movement, trend, and probability of profit for particular spreads for a specific futures market. Some sort of hybrid model works well in such a scenario. If past and present market prices are to be included, they are best regarded as continuous variables and are probably well modeled using a neural-network-based approach. The overall system may also use input from categorized news stories taken off a news wire. News stories are read, categorized, and ranked according to some criteria. Such categorical data is better modeled using one of the rule extraction tools. The output from both of these tools will itself need preparation before being fed into some next stage. The user sees none of the underlying technicality, but the builder of the system will have to make a large number of choices, including those about the optimal data preparation techniques to meet each objective. Categorical data and numeric data may well, and normally do, require different preparation techniques.

At the project design stage, or when directly using general-purpose modeling tools, it is important to be aware of the needs, strengths, and weaknesses of each of the tools employed. Each tool has a slightly different output. It is harder to produce humanly comprehensible rules from any neural network product than from one of the rule extraction variety, for example. Almost certainly it is possible to transform one type of output to another use—to modify selection rules, for instance, into providing a score—but it is frequently easier to use a tool that provides the type of output required.

3.2.1 How Modeling Tools Drive Data Preparation

Modeling tools come in a wide variety of flavors and types. Each tool has its strengths and weaknesses. It is important to understand which particular features of each tool affect how data is prepared.

One main factor by which mining tools affect data preparation is the sensitivity of the tool to the numeric/categorical distinction. A second is sensitivity to missing values, although this sensitivity is largely misunderstood. To understand why these distinctions are important, it is worth looking at what modeling tools try to do.

The way in which modeling tools characterize the relationships between variables is to partition the data such that data in particular partitions associates with particular outcomes. Just as some variables are discrete and some variables are continuous, so some tools partition the data continuously and some partition it discretely. In the examples shown in Figures 3.2 and 3.3 the learning was described as finding some “best-fit” line characterizing the data. This actually describes a continuous partitioning in which you can imagine the partitions are indefinitely small. In such a partitioning, there is a particular mathematical relationship that allows prediction of output value(s) depending on how far distant, and in exactly what direction (in state space), the instance value lies from the optimum. Other mining tools actually create discrete partitions, literally defining areas of state space such that if the predicting values fall into that area, a particular output is predicted. In order to examine what this looks like, the exact mechanism by which the partitions are created will be regarded as a black box.

We have already discussed in Chapter 2 how each variable can be represented as a dimension in state space. For ease of description, we'll use a two-dimensional state space and only two different types of instances. In any more realistic model there will almost certainly be more, maybe many more, than two dimensions and two types of instances. Figure 3.7 shows just such a two-dimensional space as a graph. The Xs and Os in Figure 3.7(a) show the positions of instances of two different instance types. It is the job of the modeling tool to find optimal ways of separating the instances.

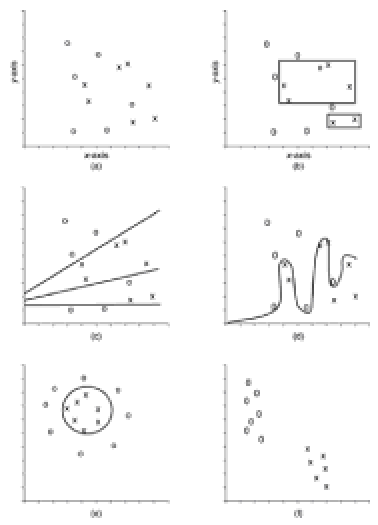


Figure 3.7 Modeling a data set: separating similar data points (a), straight lines parallel to axes of state space (b), straight lines not parallel to axes of state space (c), curves (d), closed area (e), and ideal arrangement (f).

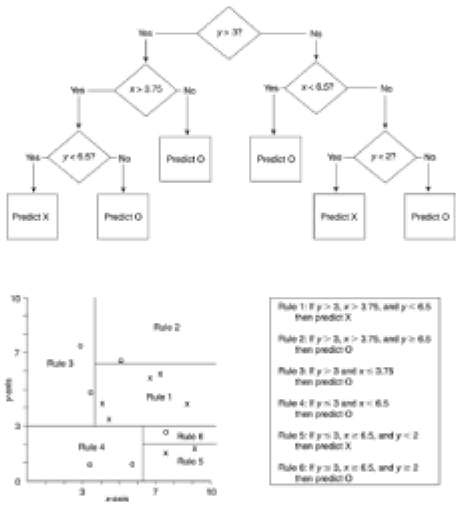
Various "cutting" methods are directly analogous to the ways in which modeling tools separate data. Figure 3.7(b) shows how the space might be cut using straight lines parallel to the axes of the graph. Figure 3.7(c) also shows cuts using straight lines, but in this figure they are not constrained to be parallel to the axes. Figure 3.7(d) shows cuts with lines, but they are no longer constrained to be straight. Figure 3.7(e) shows how separation may be made using areas rather than lines, the areas being outlined.

Whichever method or tool is used, it is generally true that the cuts get more complex traveling from Figure 3.7(b) to 3.7(e). The more complex the type of cut, the more computation it takes to find exactly where to make the cut. More computation translates into "longer." Longer can be very long, too. In large and complex data sets, finding the optimal places to cut can take days, weeks, or months. It can be a very difficult problem to decide when, or even if, some methods have found optimal ways to divide data. For this reason, it is always beneficial to make the task easier by attempting to restructure the data so that it is most easily separated. There are a number of "rules of thumb" that work to make the data more tractable for modeling tools. Figure 3.7(f) shows how easy a time the modeling tool would have if the data could be rearranged as shown during preparation! Maybe automated preparation cannot actually go as far as this, but it can go at least some of the way, and as far as it can go is very useful.

In fact, the illustrations in Figure 3.7 do roughly correspond with the ways in which different tools separate the data. They are not precisely accurate because each vendor modifies “pure” algorithms in order to gain some particular advantage in performance. It is still worthwhile considering where each sits, since the underlying method will greatly affect what can be expected to be learned from each tool.

3.2.2 Decision Trees

Decision trees use a method of logical conjunctions to define regions of state space. These logical conjunctions can be represented in the form of “If . . . then” rules. Generally a decision tree considers variables individually, one at a time. It starts by finding the variable that best divides state space and creating a “rule” to specify the split. The decision tree algorithm finds for each subset of the instances another splitting rule. This continues until the triggering of some stopping criterion. Figure 3.8 illustrates a small portion of this process.



Rule 1: If $y > 3$, $x > 3.75$, and $y < 6.5$
then predict X

Rule 2: If $y > 3$, $x > 3.75$, and $y \geq 6.5$
then predict O

Rule 3: If $y > 3$ and $x < 3.75$
then predict O

Rule 4: If $y \leq 3$ and $x < 6.5$
then predict O

Rule 5: If $y \leq 3$, $x \geq 6.5$, and $y < 2$
then predict X

Rule 6: If $y \leq 3$, $x \geq 6.5$, and $y \geq 2$
then predict O

Figure 3.8 A decision tree cutting state space.

Due to the nature of the splitting rules, it can easily be seen that the splits have to be parallel to one of the axes of state space. The rules can cut out smaller and smaller pieces of state space, but always parallel to the axes.

3.2.3 Decision Lists

Decision lists also generate “If . . . then” rules, and graphically appear similar to decision trees. However, decision trees consider the subpopulation of the “left” and “right” splits separately and further split them. Decision lists typically find a rule to well characterize some small portion of the population that is then removed from further consideration. At that point it seeks another rule for some portion of the remaining instances. Figure 3.9 shows how this might be done.

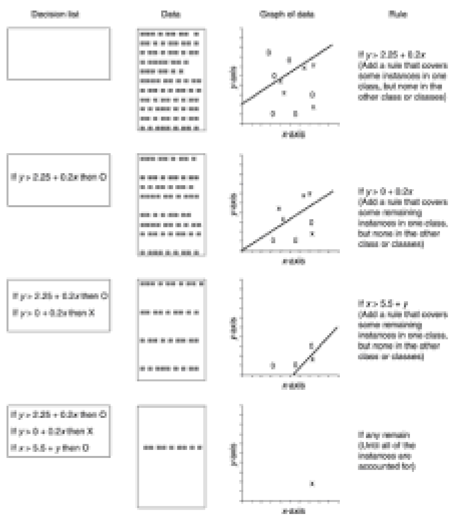
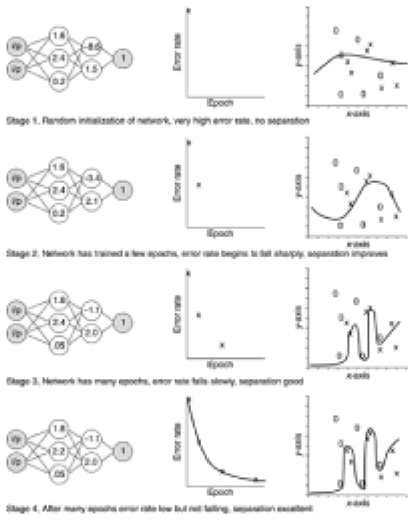

Decision list
Data
Graph of data
Rule
If y > 2.25 + 0.2x
(Add a rule that covers some instances in one class, but none in the other class or classes)
If y > 2.25 + 0.2x then O
If y > 0 + 0.2x
(Add a rule that covers some remaining instances in one class, but none in the other class or classes)
If y > 2.25 + 0.2x then O
If y > 0 + 0.2x then X
If x > 5.5 + y
(Add a rule that covers some remaining instances in one class, but none in the other class or classes)
If y > 2.25 + 0.2x then O
If y > 0 + 0.2x then X
If x > 5.5 + y then O
If any remain
(Until all of the instances are accounted for)
y-axis
x-axis

Figure 3.9 A decision list inducing rules that cover portions of the remaining data until all instances are accounted for.

(Although this is only the most cursory look at basic algorithms, it must be noted that many practical tree and list algorithms at least incorporate techniques for allowing the cuts to be other than parallel to the axes.)

3.2.4 Neural Networks

Neural networks allow state space to be cut into segments with cuts that are not parallel to the axes. This is done by having the network learn a series of “weights” at each of the “nodes.” The result of this learning is that the network produces gradients, or sloping lines, to segment state space. In fact, more complex forms of neural networks can learn to fit curved lines through state space, as shown in Figure 3.10. This allows remarkable flexibility in finding ways to build optimum segmentation. Far from requiring the cuts to be parallel to the axes, they don’t even have to be straight.



Error rate
Epoch
y-axis
x-axis
Stage 1. Random initialization of network, very high error rate, no separation
Stage 2. Network has trained a few epochs, error rate begins to fall sharply, separation improves
Stage 3. Network has many epochs, error rate falls slowly, separation good
Stage 4. After many epochs error rate low but not falling, separation excellent

Figure 3.10 Neural network training.

As the cuts become less linear, and not parallel to the axes, it becomes more and more difficult to express the rules in the form of logical conjunctions—the “If . . . then” rules. The expression of the relationships becomes more like fairly complex mathematical equations. A statistician might say they resemble “regression” equations, and indeed they do.

(Chapter 10 takes a considerably more detailed look at neural networks, although not for the purposes of predictive or inferential modeling.)

3.2.5 Evolution Programs

In fact, using a technique called *evolution programming*, it is possible to perform a type of regression known as *symbolic regression*. It has little in common with the process of finding regression equations that is used in statistical analysis, but it does allow for the discovery of particularly difficult relationships. It is possible to use this technique to discover the equation that would be needed to draw the curve in Figure 3.7(e).

3.2.6 Modeling Data with the Tools

There are more techniques available than those listed here; however, these are fairly representative of the techniques used in data mining tools available today. Demonstration versions of commercial tools based on some of these ideas are available on the CD-ROM accompanying this book. They all extend the basic ideas in ways the vendor feels enhances performance of the basic algorithm. These tools are included as they generally will benefit from having the data prepared in different ways.

Considered at a high level, modeling tools separate data using one of two approaches. The first way that tools use is to make a number of cuts in the data set, separating the total data set into pieces. This cutting continues until some stopping criterion is met. The second way is to fit a flexible surface, or at least a higher-dimensional extension of one (a manifold), between the data points so as to separate them. It is important to note that in practice it is probably impossible, with the information contained in the data set, to separate all of the points perfectly. Often, perfect separation is not really wanted anyway. Because of noise, the positioning of many of the points may not be truly representative of where they would be if it were possible to measure them without error. To find a perfect fit would be to learn this noise. As discussed earlier, the objective is for the tool to discover the underlying structure in the data without learning the noise.

The key difference to note between tools is that the discrete tools—those that cut the data set into discrete areas—are sensitive to differences in the rank, or order, of the values in the variables. The quantitative differences are not influential. Such tools have advantages and disadvantages. You will recall from Chapter 2 that a rank listing of the joint distances between American cities carries enough information to recover their geographical layout very accurately. So the rank differences do carry a very high information content. Also, discrete tools are not particularly troubled by outliers since it is the positioning in rank that is significant to them. An outlier that is in the 1000th-rank position is in that position whatever its value. On the other hand, discrete tools, not seeing the quantitative difference between values, cannot examine the fine structure embedded there. If there is high information content in the quantitative differences between values, a tool able to model continuous values is needed. Continuous tools can extract both quantitative and qualitative (or rank) information, but are very sensitive to various kinds of distortion in the data set, such as outliers. The choice of tool depends very much on the nature of the data coupled with the requirements of the problem.

The simplified examples shown in Figure 3.7 assume that the data is to be used to predict an output that is in one of two states—O or X. Typically, tools that use linear cuts do have to divide the data into such binary predictions. If a continuous variable needs to be predicted, the range of the variable has to be divided into discrete pieces, and a separate model built for predicting if the range is within a particular subrange. Tools that can produce nonlinear cuts can also produce the equations to make continuous predictions. This means that the output range does not have to be chopped up in the way that the linear cutting tools require.

These issues will be discussed again more fully later. It is also important to reiterate that, in practice, mining tool manufacturers have made various modifications so that the precise compromises made for each tool have to be individually considered.

3.2.7 Predictions and Rules

Tool selection has an important impact on exactly which techniques are applied to the unprepared data. All of the techniques described here produce output in one of two forms—predictions or rules. Data modeling tools end up expressing their learning either as a predicted number, a predicted categorical, or as a set of rules that can be used to separate the data in useful ways.

For instance, suppose that it is required as part of a solution to model the most likely value of the mortgage rate. The mortgage rate is probably best regarded as a continuous variable. Since a prediction of a continuous variable is needed, it indicates that the most appropriate tool to use for the model would be one that is capable of continuous predictions. Such a tool would probably produce some sort of equation to express the relationship of input values to the predicted value. Since a continuous value is required as output, it is advantageous, and works best, when the input values are also continuous. Thus, indications about the type of tools and some of the data preparation decisions are already made when the solution is selected.

Having decided on a predicted mortgage rate, perhaps it is required to make a model to determine if a particular prospective customer is or is not likely to respond to a solicitation with this rate. For this solution it might be most appropriate to use a model with a binary, yes/no output. The most appropriate tool is some sort of classifier that will classify records into the yes/no dichotomy required. Preparing data for a yes/no dichotomy may benefit from techniques such as binning that enhance the ability of many tools to separate the data. *Binning* is a technique of lumping small ranges of values together into categories, or "bins," for the purpose of reducing the variability (removing some of the fine structure) in a data set. For instance, customer information response cards typically ask for household income using "from-to" ranges in which household income falls. Those categories are "bins" that group ranges of income. There are circumstances in mining in which this can be useful.

Continuous and dichotomous modeling methods can be used for more than just making predictions. When building models to understand what is "driving" certain effects in the data set, the models are often used to answer questions about what features are important in particular areas of state space. Such modeling techniques are used to answer questions like "What are the underlying factors associated with fraudulent transactions in the branch offices?" Since the affecting factors may possibly be different from area to area of state space, it is important to use preparation techniques that retain as much of the fine structure—that is, the detailed fluctuations in the data set—over the full range of variability of the variables.

Looking for affecting factors is a form of inferential modeling. Examination of what is common to sets of rules is one way to discover the common themes present in particular situations, such as the branch office fraud alluded to above. The ability to give clear reasons for action are particularly important in several situations, such as credit approval or denial, where there is a legal requirement for explanation to be available. Generation of such rules can also be expressed, say, as SQL statements, if it is needed to extract parts of a data set. Perhaps a mailing list is required for all people meeting particular criteria. What is important here is to focus on how the required output affects the preparation of the input data, rather than the use to which the solution will be put.

3.2.8 Choosing Techniques

In summary, the effect that the choice of modeling tools has on data preparation is determined by the tool's characteristics. Is the tool better able to model continuous or categorical data? Since actual tools are all modifications of "pure" algorithms, this is a question that is hard to answer in its general form. Each tool has to be evaluated individually. Practically speaking, it is probably best to try several preparation techniques, preparing the data as continuous and also using several binning options to create categorized data. However, it is also important to use a mining tool that produces output appropriate to the needs of the solution. If the solution required calls for a categorical prediction, the tool needs to be able to produce such a solution and will probably benefit from categorical training, test, and execution data. The data preparation techniques discussed in this book are designed to allow preparation of the data set in a variety of ways. They allow the data to be manipulated as needed, so the miner can focus attention on deciding which are the appropriate techniques and tools to use in a particular situation.

3.2.9 Missing Data and Modeling Tools

Missing values form a very important issue in preparing data and were discussed in Chapter 2. Whenever there are missing values, it is vital that something be done about them. There are several methods for determining a suitable replacement value, but under no circumstances should the missing values be ignored or discarded. Some tools, particularly those that handle categorical values well, such as decision trees, are said to handle missing values too. Some really can; others can't. Some discrete-type modeling tools can actually elegantly ignore missing values, while others regard a missing value as just another categorical value, which is not really a satisfactory approach. Other tools, such as neural networks, require that each input be given a numeric value and any record that has a missing value has to be either completely ignored, or some default for the missing value must be created.

There are going to be problems with whatever default replacement approach is taken—very often major problems. At the very least, left untreated except by the default solution, missing values cause considerable distortion to the fabric of the data set. Not all missing values, for instance, can be assumed to represent the same value. Yet that is what a decision tree does if it assigns missing values to a separate category—assumes that they all have the same measured value. Similar distortions occur if some default numerical value is assigned. Clearly, a better solution needs to be found. Several choices are available, and the pros and cons of each method are discussed in detail in Chapter 8. For the time being, note that this is one of the issues that *must* be dealt with effectively for the best models to be built.

3.3 Stages of Data Preparation

Data preparation involves two sets of preparatory activities. The first are *nonautomated* activities that are procedural, or activities that result in a decision about the approach that the miner decides to take. There are many activities and decisions to be made in this stage that can be described as "basic preparation," and they are discussed in detail in the next chapter. The second set of activities are *automated* preparation activities. Detailed descriptions of the techniques used in the automated preparation stage, the demonstration code, and the process and decision points that go into data preparation round out the remaining chapters. What follows is a brief overview of the eight stages:

1. Accessing the data

2. Auditing the data

3. Enhancing and enriching the data

4. Looking for sampling bias

5. Determining data structure

6. Building the PIE

7. Surveying the data

8. Modeling the data

3.3.1 Stage 1: Accessing the Data

The starting point for any data preparation project is to *locate the data*. This is sometimes easier said than done! There are a considerable variety of issues that may hinder access to the nominated data, ranging from legal to connectivity. Some of these commonly encountered issues are reviewed later, but a comprehensive review of all issues is almost impossible, simply because every project provides unique circumstances. Nonetheless, locating and securing the source of data supply and ensuring adequate access is not only the first step, it is absolutely essential.

You might say, "Well, I have part of the problem licked because I have access to a data warehouse." It is a fact that data warehouses are becoming repositories of choice. More and more it is a warehouse that is to be mined. However, a warehouse is by no means essential in order to mine data. In fact, a warehouse can be positively detrimental to the mining effort, depending on how the data was loaded. Warehouses also have other drawbacks, a significant one being that they are often created with a particular structure to reflect some specific view of the enterprise. This imposed structure can color all modeling results if care is not taken to avoid bias.

3.3.2 Stage 2: Auditing the Data

Assuming that suitable data is available, the first set of basic issues that have to be addressed concern

- The source of supply
- The quantity of data
- The quality of the data

Building robust models requires data that is sufficient in quantity, and of high enough quality to create the needed model. A data audit provides a methodology for determining the status of the data set and estimates its adequacy for building the model. The reality is that the data audit does not so much assure that the model will be able to be built, but at least assures that the minimum requirements have been met.

Auditing requires examining small samples of the data and assessing the fields for a variety of features, such as number of fields, content of each field, source of each field, maximum and minimum values, number of discrete values, and many other basic metrics. When the data has been assessed for quantity and quality, a key question to ask is, Is there a justifiable reason to suppose that this data has the potential to provide the required solution to the problem? *Here is a critical place to remove the expectation of magic*. Wishful thinking and unsupported hopes that the data set that happens to be available will actually hold something of value seldom results in a satisfactory model. The answer to whether the hopes for a solution are in fact justified lies not in the data, but in the hopes! An important part of the audit, a nontechnical part, is to determine the true feasibility of delivering value with the resources available. Are there, in fact, good reasons for thinking that the actual data available can meet the challenge?

3.3.3 Stage 3: Enhancing and Enriching the Data

With a completed audit in hand, there is at least some firm idea of the adequacy of the data. If the audit revealed that the data does not really support the hopes founded on it, it may be possible to supplement the data set in various ways. Adding data is a common way to increase the information content. Many credit card issuers, for instance, will purchase information from outside agencies. Using this purchased data allows them to better assess the creditworthiness of their existing customers, or of prospects who are not yet their customers.

There are several ways in which the existing data can be manipulated to extend its usefulness. Such manipulation, for example, is to calculate price/earnings (P/E) ratios for modeling the value of share prices. So-called “fundamentalist” investors feel that this ratio has predictive value. They may be right. If they are, you may ask, “Since the price and the earnings are present in the source data, how would providing information about the P/E ratio help?" First, the P/E ratio represents an insight into the domain about what is important. This insight adds information to the modeling tool's input. Second, presenting this precalculated information saves the modeling tool from having to learn division! Modeling tools can and do learn multiplicative relationships. Indeed, they can learn relationships considerably more complicated than that. However, it takes time and system resources to discover any relationship. Adding enough domain knowledge and learning assistance about important features can boost performance and cut development time dramatically. In some cases, it turns the inability to make any model into the ability to make useful models.

3.3.4 Stage 4: Looking for Sampling Bias

Sampling bias presents some particularly thorny problems. There are some automated methods for helping to detect sampling bias, but no automated method can match reasoned thought. There are many methods of sampling, and sampling is always necessary for reasons discussed in Chapter 5. *Sampling* is the process of taking a small piece of a larger data set in such a way that the small piece accurately reflects the relationships in the larger data set. The problem is that the true relationships that exist in the fullest possible data set (called the *population*) may, for a variety of reasons, be unknowable. That means that it is impossible to actually check to see if the sample is representative of the population in fact. It is critical to bend every effort to making sure that the data captured is as representative of the true state of affairs as possible.

While sampling is discussed in many statistical texts, miners face problems not addressed in such texts. It is generally assumed that the analyst (statistician/modeler) has some control over how the data is generated and collected. If not the analyst, at least the creator or collector of the data may be assumed to have exercised suitable control to avoid sampling bias. Miners, however, sometimes face collections of data that were almost certainly gathered for purposes unknown, by processes unsure, but that are now expected to assist in delivering answers to questions unthought of at the time. With the provenance of the data unknown, it is very difficult to assess what biases are present in the data, and that, if uncorrected, will produce erroneous and inapplicable models.

3.3.5 Stage 5: Determining Data Structure (Super-, Macro-, and Micro-)

Structure refers to the way in which the variables in a data set relate to each other. It is this structure that mining sets out to explore. Bias, mentioned above, stresses the natural structure of a data set so that the distorted data is less representative of the real world than unbiased data. But structure itself has various forms: super, macro, and micro.

Superstructure refers to the scaffolding erected to capture the data and form a data set. The superstructure is consciously and deliberately created and is easy to see. When the data set was created, decisions had to be made as to exactly which measurements were to be captured, measured in which ways, and stored in which formats. Point-of-sale (POS) data, for instance, captures information about a purchasing event at the point that the sale takes place. A vast wealth of possible information could be captured at this point, but capturing it all would swamp the system. Thus, POS information typically does not include any information about the weather, the length of the checkout line, local traffic information affecting access to the store, or the sort of bag the consumer chose for carrying away purchases. This kind of information may be useful and informative, but the structure created to capture data has no place to put it.

Macrostructure concerns the formatting of the variables. For example, granularity is a macro structural feature. *Granularity* refers to the amount of detail captured in any measurement—time to the nearest minute, the nearest hour, or simply differentiating morning, afternoon, and night, for instance. Decisions about macro structure have an important impact on the amount of information that a data set carries, which, in turn, has a very significant effect on the resolution of any model built using that data set. However, macro structure is not part of the scaffolding consciously erected to hold data, but is inherent in the nature of the measurements.

Microstructure, also referred to as *fine structure*, describes the ways in which the variables that have been captured relate to each other. It is this structure that modeling explores. A basic assessment of the state of the micro structure can form a useful part of the data audit (Stage 2 above). This brief examination is a simple assessment of the complexity of the variables' interrelationships. Lack of complexity does not prevent building successful predictive models. However, if complex and unexpected results are desired, additional data will probably be needed.

3.3.6 Stage 6: Building the PIE

The first five steps very largely require assessing and understanding the data that is available. Detailed scrutiny of the data does several things:

- It helps determine the possibility, or necessity, of adjusting or transforming the data.

- It establishes reasonable expectations of achieving a solution.

- It determines the general quality, or validity, of the data.

- It reveals the relevance of the data to the task at hand.

Many of these activities require the application of thought and insight rather than of automated tools. Of course, much of the assessment is supported by information gained by application of data preparation and other discovery tools, but the result is information that affects decisions about how to prepare and use the data.

By this stage, the data's limitations are known, at least insofar as they can be. Decisions have been made based on the information discovered. Fully automated techniques for preparing the data (such as those on the CD-ROM accompanying this book) can now be used.

The decisions made so far determine the sequence of operations. In a production environment, the data set may be in any machine-accessible form. For ease of discussion and explanation, it will be assumed that the data is in the form of a flat file. Also, for ease of illustration, each operation is discussed sequentially. In practice the techniques are not likely to be applied exactly as described. It is far easier to aggregate information that will be used by several subsequent stages during one pass through the file. This description is intended as thematic, to provide an overview and introduction to preparation activities.

Data Issue: Representative Samples

A perennial problem is determining how much data is needed for modeling. One tenet of data mining is "all of the data, all of the time." That is a fine principle, and if it can be achieved, a worthwhile objective. However, for various reasons it is not a practical solution. Even if as much data as possible is to be examined, survey and modeling still require at least three data sets—a training set, a test set, and an execution set. Each data set needs to be representative. Feature enhancement, discussed in Chapters 4 and 10, may require a concentration of instances exhibiting some particular feature. Such a concentration can only be made if a subset of data is extracted from the main data set. So there is always a need to decide how large a data set is required to be an accurate reflection of the data's fine structure.

In this case, when building the PIE, it is critical that it is representative of the fine structure. Every effort must be made to ensure that the PIE itself does not introduce bias! Without checking the whole population of instances, which may be an impossibility, there is no way to be 100% certain that any particular sample is, in fact, representative. However, it is possible to be some specified amount less than 100% certain, say, 99% or 95% certain. It is these certainty measures that allow samples to be taken. Selecting a suitable level of certainty is an arbitrary decision.

Data Issue: Categorical Values

Categoricals are "numerated," or assigned appropriate numbers. Even if, in the final prepared data, the categoricals are to be modeled as categorical values, they are still numerated for estimating missing values.

Chapter 2 contains an example showing that categoricals have a natural ordering that needs to be preserved. It is an ordering that actually exists in the world and is reflected in the categorical measurements. When building predictive or inferential models, it is critical that the natural order of the categorical values be preserved insofar as that is possible. Changing this natural ordering is imposing a structure. Even imposing a random structure loses information carried by the categorical measurement. If it is not random, the situation is worse because it introduces a pattern not present in the world.

The exact method of numeration depends on the structure of the data set. In a mixed numeric/categorical data set, the numeric values are used to reflect their order into the categoricals. This is by far the most successful method, as the numeric values have an order and magnitude spacing. In comprehensive data sets, this allows a fair recovery of the appropriate ordering. In fact, it is interesting to convert a variable that is actually numeric into a categorical value and see the correct ordering and separation recovered.

Data sets that consist entirely of categorical measurements are slightly more problematic. It is certainly possible to recover appropriate orderings of the categoricals. The problem is that without numeric variables in the data set, the recovered values are not anchored to real-world phenomena. The numeration is fine for modeling and has in practice produced useful models. It is, however, a dangerous practice to use the numerated orderings to infer anything absolute about the meaning of the magnitudes. The relationships of the variables, one to another, hold true, but are not anchored back to the real world in the way that numerical values are.

It is important to note that no automated method of recovering order is likely to be as accurate as that provided by domain knowledge. Any data set is but a pale reflection of the real world. A domain expert draws on a vastly broader range of knowledge of the world than can be captured in any data set. So, wherever possible, ordered categorical values should be placed in their appropriate ordering as ordinal values. However, as it is often the case when modeling data that there is no domain expert available, or that no ordinal ranking is apparent, the techniques used here have been effective.

Data Issue: Normalization

Several types of normalization are very useful when modeling. The normalization discussed throughout this book has nothing in common with the sort of normalization used in a database. Recall that the assumption for this discussion is that the data is present as a single table. Putting data into its various normal forms in a database requires use of multiple tables. The form of normalization discussed here requires changing the instance values in specific and clearly defined ways to expose information content within the data and the data set. Although only introduced here, the exact normalization methods are discussed in detail in Chapter 7.

Some tools, such as neural networks, require range normalization. Other tools do not *require* normalization, but do benefit from having normalized data. Once again, as with other issues, it is preferable for the miner to take control of the normalization process.

Variables in the data set should be normalized both across range and in distribution. There is also very much to be learned from examining the results of normalization, which is briefly looked at in Chapter 7. In addition to range normalization, distribution normalization deals with many problems, such as removing much of the distortion of outliers and enhancing linear predictability.

Data Issue: Missing and Empty Values

Dealing with missing and empty values is very important. Unfortunately, there is no automated technique for differentiating between missing and empty values. If done at all, the miner has to differentiate manually, entering categorical codes denoting whether the value is missing or empty. If it can be done, this can produce useful results. Usually it can't be, or at any rate isn't, done. Empty and missing values simply have to be treated equally.

All modeling tools have some means of dealing with missing values, even if it is to ignore any instance that contains a missing value. Other strategies include assigning some fixed value to all missing values of a particular variable, or building some estimate of what the missing value might have been, based on the values of the other variables that are present. There are problems with all of these approaches as each represents some form of compromise.

In some modeling applications, there is high information content in noting the patterns of variables that are missing. In one case this proved to be the most predictive variable! When missing values are replaced, unless otherwise captured, information about the pattern of values that are missing is lost. A pseudo-categorical is created to capture this information that has a unique value for each missing value pattern. Only after this information has been captured are the values replaced. Chapter 8 discusses the issues and choices.

Data Issue: Displacement Series

At this point in the preparation process the data is understood, enhanced, enriched, adequately sampled, fully numerated, normalized in two dimensions (range and distribution), and balanced. If the data set is a displacement series (time series are the most common), the data set is treated with various specialized preparatory techniques. The most important action here, one that cannot be automated safely, requires inspection of the data by the miner. Detrending of displacement series can be a ruinous activity to information content if in fact the data has no real trend! Caution is an important watchword. Here the miner must make a number of decisions and perhaps smooth and/or filter to prepare the data. Chapter 9 covers the issues.

(At this point the PIE is built. This can take one of several forms—computer program, mathematical equations, or program code. The demonstration program and code included on the CD-ROM that accompanies this book produce parameters in a file that a separate program reads to determine how to prepare raw data. The previous activities have concentrated on preparing variables. That is to say, each variable has been considered in isolation from its relationship with other variables. With the variables prepared, the next step is to prepare data sets, which is to say, to consider the data as a whole.)

Data Set Issue: Reducing Width

Data sets for mining can be thought of as being made from a two-dimensional table with columns representing variable measurements, and rows representing instances, or records. *Width* describes the number of columns, whereas *depth* describes the number of rows.

One of the knottiest problems facing a miner deals with width. More variables presumably carry more information. But too many variables can bring any computational algorithm to its knees. This is referred to as the *combinatorial explosion* (discussed in Chapter 2). The number of relationships between variables increases multiplicatively as the variable count increases; that is, with 10 variables the first variable has to be compared with 9 neighbors, the second with 8 (the second was already compared with the first, so that doesn't have to be done again), and so on. The number of interactions is 9 x 8 x 7 x 6 . . . , which is 362,880 comparisons. With 13 variables the number of interactions is up to nearly 40 million. By 15 variables it is at nearly 9 billion. Most algorithms have a variety of ways to reduce the combinatorial complexity of the modeling task, but too many variables can eventually defeat any method.

Thus it is that the miner may well want to reduce the number of columns of data in a data set, if it's possible to do so without reducing its information content. There are several ways to do this if required, some more arbitrary than others. Chapter 10 discusses the pros and cons of several methods.

Data Set Issue: Reducing Depth

Depth does not have quite the devastating impact that width can have. However, while there is a genuine case in data mining for "all of the data, all of the time," there are occasions when that is not required. There is still a need for assurance that the subset of data modeled does in fact reflect all of the relationships that exist in the full data set. This requires another look at sampling. This time the sampling has to consider the interactions between the variables, not just the variability of individual variables considered alone.

Data Set/Data Survey Issue: Well- and Ill-Formed Manifolds

This is really the first data survey step as well as the last data preparation step. The data survey, discussed briefly in Chapter 11, deals with deciding what is in the data set prior to modeling. However, it forms the last part of data preparation too because if there are problems with the shape of the manifold, it may be possible to manipulate the data to ameliorate some of them. The survey is not concerned with manipulating data, but with giving the miner information that will help with the modeling.

As the last step in data preparation, this look at the manifold seeks to determine if there are problems that can be eliminated by manipulation. If the manifold is folded, for instance, there will be problems. In two dimensions a fold might look like an “S.” A vertical line drawn through the “S” will cut it in three places. The vertical line will represent a single value of one variable for which three values of the other variable existed. The problem here is that there is no additional information available to decide which of the three values will be appropriate. If, as in Figure 3.11, there is some way of “rotating” the “S” through 90 degrees, the problem might be solved. It is these sorts of problems, and others together with possible solutions, that are sought in this stage.

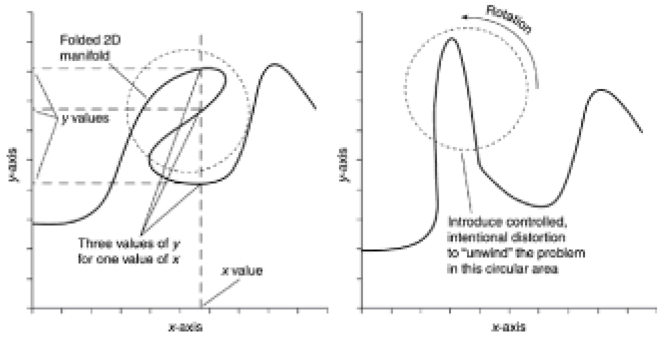

Folded 2D manifold
y values
y-axis
Three values of y for one value of x
x value
x-axis
Rotation
y-axis
Introduce controlled, intentional distortion to "unwind" the problem in this circular area
x-axis

Figure 3.11 Deliberately introduced and controlled distortion of the manifold can remove problems.

3.3.7 Stage 7: Surveying the Data

The *data survey* examines and reports on the general properties of the manifold in state space. In a fairly literal sense it produces a map of the properties of the manifold, focusing on the properties that the miner finds most useful and important. It cannot be an actual map if, as is almost invariably the case, state space exists in more than three dimensions. The modeler is interested in knowing many features, such as the relative density of points in state space, naturally occurring clusters, uncorrectable distortions and where they occur, areas of particular relative sparsity, how well defined the manifold is (its “fuzzyness”), and a host of other features. Unfortunately, it is impossible, within the confines of this book, to examine the data survey in any detail at all. Chapter 11 discusses the survey mainly from the perspective of data preparation, discussing briefly some other aspects. Inasmuch as information discovered in the data survey affects the way data is prepared, it forms a part of the data preparation process.

3.3.8 Stage 8: Modeling the Data

The whole purpose of preparation and surveying is to understand the data. Often, understanding needs to be turned into an active or passive model. As with the data survey, modeling is a topic too broad to cover here. Some deficiencies and problems only appear when modeling is attempted. Inasmuch as these promote efforts to prepare the data differently in an attempt to ameliorate the problems, modeling too has some role in data preparation. Chapter 12 looks at modeling in terms of how building the models interacts with data preparation and how to use the prepared data effectively.

3.4 And the Result Is . . . ?

Having toured the territory, even briefly, this may seem like a considerable effort, both computationally and in human time, effort, and expertise. Do the results justify the effort? Clearly, some minimal data preparation has to be done for any modeling tool. Neural networks, for instance, require all of the inputs to be numerated and range normalized. Other techniques require other minimal preparation. The question may be better framed in terms of the benefit to be gained by taking the extra steps beyond the minimum.

Most tools are described as being able to learn complex relationships between variables. The problem is to have them learn the "true" relationships before they learn noise. This is the purpose of data preparation: *to transform data sets so that their information content is best exposed to the mining tool*. It is also critical that if no good can be done in a particular data set, at least no harm be done. In the data sets provided on the CD-ROM included with this book, most are in at least mineable condition with only minimal additional preparation. Comparing the performance of the same tools on the same data sets in both their minimally prepared and fully prepared states gives a fair indication of what can be expected. Chapter 12 looks at this comparison.

There are some data sets in which there is no improvement in the prediction error rate. In these cases it is important to note that neither is there any degradation! The error rate of prediction is unaffected. This means that at least no harm is done. In most cases there is improvement—in some cases a small amount, in other cases much more. Since the actual performance is so data dependent, it is hard to say what effect will be found in any particular case. Error rates are also materially affected by the type of prediction—classification and accuracy may be very differently impacted using the same model and the same data set. (See the examples in Chapter 12.) In most cases, however error rate is determined, there is usually a significant improvement in model performance when the models are built and executed on prepared data.

However, there is far more to data preparation than just error rate improvement. Variable reduction has often sped mining time 10 to 100 times over unprepared data. Moreover, some data sets were so dirty and distorted prior to preparation that they were effectively unusable. The data preparation techniques made the data at least useable, which was a very considerable gain in itself. Not least is the enormous insight gained into the data before modeling begins. This insight can be more valuable than any improvement in modeling performance. This is where the preparation of the miner brings the benefit that the miner, through insight, builds better models than without the insight. And the effect of that is impossible to quantify.

Considering that application of these techniques can reduce the error rate in a model, reduce model building time, and yield enormous insight into the data, it is at least partly the miner's call as to where the most important benefits accrue. This brief tour of the landscape has pointed out the terrain. The remaining chapters look in detail at preparing data and addressing the issues raised here.

Chapter 4: Getting the Data—Basic Preparation

Overview

Data preparation requires two different types of activities: first, finding and assembling the data set, and second, manipulating the data to enhance its utility for mining. The first activity involves the miner in many procedural and administrative activities. The second requires appropriately applying automated tools. However, manipulating the data cannot begin until the data to be used is identified and assembled and its basic structure and features are understood. In this chapter we look at the process of finding and assembling the data and assessing the basic characteristics of the data set. This lays the groundwork for understanding how to best manipulate the data for mining.

What does this groundwork consist of? As the ancient Chinese proverb says: "A journey of a thousand miles begins with a single step." Basic data preparation requires three such steps: data discovery, data characterization, and data set assembly.

- *Data discovery* consists of discovering and actually locating the data to be used.

- *Data characterization* describes the data in ways useful to the miner and begins the process of understanding what is in the data—that is, is it reliable and suitable for the purpose?

- *Data set assembly* builds a standard representation for the incoming data so that it can be mined—taking data found to be reliable and suitable and, usually by building a table, preparing it for adjustment and actual mining.

These three stages produce the *data assay*. The first meaning of the word "assay" in the *Oxford English Dictionary* is "the trying in order to test the virtue, fitness, etc. (of a person or thing)." This is the exact intent of the data assay, to try (test or examine) the data to determine its fitness for mining. The assay produces detailed knowledge, and usually a report, of the quality, problems, shortcomings, and suitability of the data for mining. Although simple to state, assaying data is not always easy or straightforward. In practice it is frequently extremely time-consuming. In many real-world projects, this stage is the most difficult and time-consuming of the whole project. At other times, the basic preparation is relatively straightforward, quick, and easy.

As an example, imagine that First National Bank of Anywhere (FNBA) decides to run a credit card marketing campaign to solicit new customers. (This example is based on an actual mining project.) The marketing solicitations are made to "affinity groups," that is, groups of people that share some experience or interest, such as having attended a particular college or belonging to a particular country club. FNBA buys lists of names and addresses of such groups and decides to use data mining to build market segmentation and customer response models to optimize the return from the campaign. As the campaign progresses, the models will have to be updated to reflect changing market conditions and response. Various models of different types will be required, although the details have not yet been pinned down. Figure 4.1 shows an overview of the process.

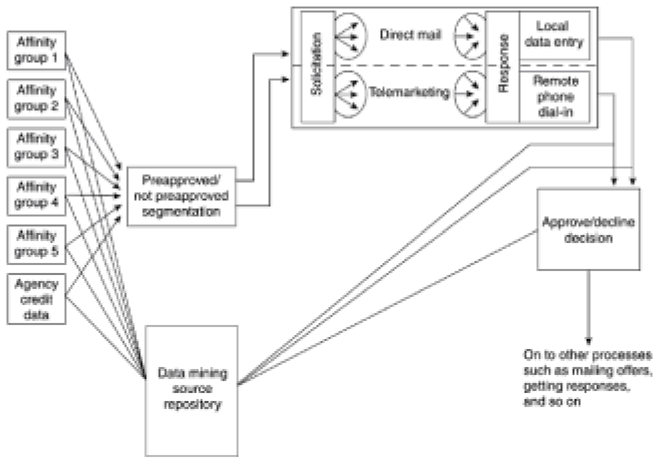


Figure 4.1 Simplified credit card direct-mail solicitation showing six different data feeds. Each data feed arrives from a different source, in a different format, at a different time and stage in the process.

4.1 Data Discovery

Current mining tools almost always require the data set to be assembled in the form of a "flat file," or table. This means that the data is represented entirely in the row and column format described in Chapter 2. Some mining tools represent that they query databases and data warehouses directly, but it is the end result of the query, an extracted table, that is usually mined. This is because data mining operations are column (variable) oriented. Databases and data warehouses are record (instance) oriented. Directly mining a warehouse or database places an unsupportable load on the warehouse query software. This is beginning to change, and some vendors are attempting to build in support for mining operations. These modifications to the underlying structural operation of accessing a data warehouse promise to make mining directly from a warehouse more practical at some future time. Even when this is done, the query load that any mining tool can levy on the warehouse will still present a considerable problem. For present practical purposes, the starting point for all current mining operations has to be regarded as a table, or flat file. "Discovering data" means that the miner needs to determine the original source from which the table will be built.

The search starts by identifying the data source. The originating data source may be a transaction processing system fed by an ATM machine or a POS terminal in a store. It may be some other record-capturing transaction or event. Whatever it is, a record is made of the original measurements. These are the founding “droplets” of data that start the process. From here on, each individual droplet of data adds to other droplets, and trickle adds to trickle until the data forms a stream that flows into a small pool—some sort of data repository. In the case of FNBA, the pools are moderately large when first encountered: they are the affinity group membership records.

The affinity group member information is likely stored in a variety of forms. The groups may well be almost unknown to each other. Some may have membership records stored on PCs, others on Macs. Some will provide their member lists on floppy disk, some on 8mm tape, some on 4mm tape, some on a Jazz drive, and others on 9-track tape. Naturally, the format for each, the field layout and nomenclature, will be equally unique. These are the initial sources of data in the FNBA project. This is not the point of data creation, but as far as the project is concerned it is the point of first contact with the raw data. The first need is to note the contact and source information. The FNBA assay starts literally with names, addresses, contact telephone numbers, media type, transmission mode, and data format for each source.

4.1.1 Data Access Issues

Before the data can be identified and assessed, however, the miner needs to answer two major questions: Is the data accessible? and How do I get it?

There are many reasons why data might not be readily accessible. In many organizations, particularly those without warehouses, data is often not well inventoried or controlled. This can lead to confusion about what data is actually available.

- *Legal issues*. There may well be legal barriers to accessing some data, or some parts of a data set. For example, in the FNBA project it is not legal to have credit information about identifiable people to whom credit is not actually going to be offered. (The law on this point is in constant change and the precise details of what is and is not legally permissible varies from time to time.) In other applications, such as healthcare, there may be some similar legal restriction or confidentiality requirement for any potential data stream.

- *Departmental access*. These restrictions are similar to legal barriers. Particularly in financial trading companies, data from one operation is held behind a “Chinese Wall” of privacy from another operation for ethical reasons. Medical and legal data are often restricted for ethical reasons.

- *Political reasons*. Data, and particularly its ownership, is often regarded as belonging to a particular department, maybe one that does not support the mining initiative for any number of reasons. The proposed data stream, while perhaps physically present, is not practically accessible. Or perhaps it is accessible, but not in a timely or complete fashion.

- *Data format*. For decades, data has been generated and collected in many formats. Even modern computer systems use many different ways of encoding and storing data. There are media format differences (9-track magnetic tape, diskettes, tape, etc.) and format differences (ASCII, EBCDIC, binary packed decimal, etc.) that can complicate assembling data from disparate sources.

- *Connectivity*. Accessing data requires that it be available online and connected to the system that will be used for mining. It is no use having the data available on a high-density 9-track tape if there is no suitable 9-track tape drive available on the mining system.

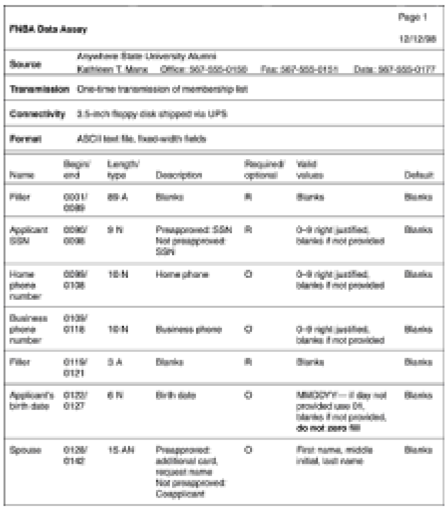
- *Architectural reasons*. If data is sourced from different database architectures, it may be extremely difficult, or unacceptably time-consuming, to translate the formats involved. Date and time information is notoriously difficult to work with. Some architectures simply have no equivalent data types to other architectures, and unifying the data representation can be a sizeable problem.

- *Timing*. The validating event (described in Chapter 2) may not happen at a comparable time for each stream. For example, merging psychographic data from one source with current credit information may not produce a useful data set. The credit information may be accurate as of 30 days ago, whereas the psychographic information is only current as of six months ago. So it is that the various data streams, possibly using different production mechanisms, may not be equally current. If a discrepancy is unavoidable, it needs to at least remain constant—that is, if psychographic information suddenly began to be current as of three months ago rather than six months ago, the relationships within the data set would change.

This is not a comprehensive listing of all possible data access issues. Circumstances differ in each mining application. However, the miner must always identify and note the details of the accessibility of each data stream, including any restrictions or caveats.

Data sources may be usefully characterized also as internal/external. This can be important if there is an actual dollar cost to acquiring outside data, or if internal data is regarded as a confidential asset of the business. It is particularly worth noting that there is *always* at least a time and effort cost to acquiring data for modeling. Identifying and controlling the costs, and getting the maximum economic benefit from each source, can be as important as any other part of a successful mining project.

FNBA has several primary data sources to define. For each source it is important to consider each of the access issues. Figure 4.2 shows part of the data assay documentation for one of the input streams.



FNBA Data Assay Page 1 12/12/98

Source	Anywhere State University Alumni Kathleen T. Mone Office: 567-555-0150 Fax: 567-555-0151 Data: 567-555-0177
Transmission	One-time transmission of membership list
Connectivity	3.5-inch floppy disk shipped via UPS
Format	ASCII text file, fixed-width fields

Name	Begin/ end	Length/ type	Description	Required/ optional	Valid values	Default
Filler	0001/ 0089	89 A	Blanks	R	Blanks	Blanks
Applicant SSN	0090/ 0098	9 N	Preapproved: SSN Not preapproved: SSN	R	0–9 right justified, blanks if not provided	Blanks
Home phone number	0099/ 0108	10 N	Home phone	O	0–9 right justified, blanks if not provided	Blanks
Business phone number	0109/ 0118	10 N	Business phone	O	0–9 right justified, blanks if not provided	Blanks
Filler	0119/ 0121	3 A	Blanks	R	Blanks	Blanks
Applicant's birth date	0122/ 0127	6 N	Birth date	O	MMDDYY — if day not provided use 01, blanks if not provided, **do not zero fill**	Blanks
Spouse	0128/ 0142	15 AN	Preapproved: additional card, request name Not preapproved: Coapplicant	O	First name, middle initial, last name	Blanks

Figure 4.2 Part of the description of one of the input streams for FNBA.

4.2 Data Characterization

After finding the source for all of the possible data streams, the *nature* of the data streams has to be characterized, that is, the data that each stream can actually deliver. The miner already knows the data format; that is to say, the field names and lengths that comprise the records in the data. That was established when investigating data access. Now each variable needs to be characterized in a number of ways so that they can be assessed according to their usefulness for modeling.

Usually, summary information is available about a data set. This information helps the miner check that the received data actually appears as represented and matches the summary provided. Most of the remainder of characterization is a matter of looking at simple frequency distributions and cross-tabs. The purpose of characterization is to understand the nature of the data, and to avoid the "GI" piece of GIGO.

4.2.1 Detail/Aggregation Level (Granularity)

All variables fall somewhere along a spectrum from detailed (such as transaction records) to aggregated (such as summaries). As a general rule of thumb, detailed data is preferred to aggregated data for mining. But the level of aggregation is a continuum. Even detailed data may actually represent an aggregation. FNBA may be able to obtain outstanding loan balances from the credit information, but not the patterns of payment that led to those balances. Describing what a particular variable measures is important. For example, if a variable is discovered to be highly predictive, during the data modeling process the strategy for using the predictions will depend on the meaning of the variables involved.

The level of detail, or granularity, available in a data set determines the level of detail that is possible for the output. Usually, the level of detail in the input streams needs to be at least one level of aggregation more detailed than the required level of detail in the output. Knowing the granularity available in the data allows the miner to assess the level of inference or prediction that the data could potentially support. It is only potential support because there are many other factors that will influence the quality of a model, but granularity is particularly important as it sets a lower bound on what is possible.

For instance, the marketing manager at FNBA is interested, in part, in the weekly variance of predicted approvals to actual approvals. To support this level of detail, the input stream requires at least daily approval information. With daily approval rates available, the miner will also be able to build inferential models when the manager wants to discover the reason for the changing trends.

There are cases where the rule of thumb does not hold, such as predicting Stock Keeping Units (SKU) sales based on summaries from higher in the hierarchy chain. However, even when these exceptions do occur, the level of granularity still needs to be known.

4.2.2 Consistency

Inconsistent data can defeat any modeling technique until the inconsistency is discovered and corrected. A fundamental problem here is that different things may be represented by the same name in different systems, and the same thing may be represented by different names in different systems. One data assay for a major metropolitan utility revealed that almost 90% of the data volume was in fact duplicate. However, it was highly inconsistent and rationalization itself took a vast effort.

The perspective with which a system of variables (mentioned in Chapter 2) is built has a huge effect on what is intended by the labels attached to the data. Each system is built for a specific purpose, almost certainly different from the purposes of other systems. Variable content, however labeled, is defined by the purpose of the system of which it is a part. The clearest illustration of this type of inconsistency comes from considering the definition of an employee from the perspective of different systems. To a payroll system, an employee is anyone who receives a paycheck. The same company's personnel system regards an employee as anyone who has an employee number. However, are temporary staff, who have employee numbers for identification purposes, employees to the payroll system? Not if their paychecks come from an external temporary agency. So to ask the two systems "How many employees are there?" will produce two different, but potentially completely accurate answers.

Problems with data consistency also exist when data originates from a single application system. Take the experience of an insurance company in California that offers car insurance. A field identifying "auto_type" seems innocent enough, but it turns out that the labels entered into the system—"Merc," "Mercedes," "M-Benz," and "Mrcds," to mention only a few examples—all represent the same manufacturer.

4.2.3 Pollution

Data pollution can occur for a variety of reasons. One of the most common is when users attempt to stretch a system beyond its original intended functionality. In the FNBA data, for instance, the miner might find “B” in the “gender” field. The “B” doesn’t stand for “Boy,” however, but for “Business.” Originally, the system was built to support personal cards, but when corporately held credit cards were issued, there was no place to indicate that the responsible party was a genderless entity.

Pollution can came from other sources. Sometimes fields contain unidentifiable garbage. Perhaps during copying, the format was incorrectly specified and the content from one field was accidentally transposed into another. One such case involved a file specified as a comma-delimited file. Unfortunately, the addresses in the field “address” occasionally contained commas, and the data was imported into offset fields that differed from record to record. Since only a few of the addresses contained embedded commas, visual inspection of parts of many thousands of records revealed no problem. However, it was impossible to attain the totals expected. Tracking down the problem took considerable time and effort.

Human resistance is another source of data pollution. While data fields are often optimistically included to capture what could be very valuable information, they can be blank, incomplete, or just plain inaccurate. One automobile manufacturer had a very promising looking data set. All kinds of demographic information appeared to be captured such as family size, hobbies, and many others. Although this was information of great value to marketing, the dealer at the point of sale saw this data-gathering exercise as a hindrance to the sales process. Usually the sales people discovered some combination of entries that satisfied the system and allowed them to move ahead with the real business at hand. This was fine for the sales process, but did the data that they captured represent the customer base? Hardly.

4.2.4 Objects

Chapter 2 explained that the world can be seen as consisting of objects about which measurements are taken. Those measurements form the data that is being characterized, while the objects are a more or less subjective abstraction. The precise nature of the object being measured needs to be understood. For instance, “consumer spending” and “consumer buying patterns” seem to be very similar. But one may focus on the total dollar spending by consumers, the other on product types that consumers seek. The information captured may or may not be similar, but the miner needs to understand why the information was captured in the first place and for what specific purpose. This perspective may color the data, just as was described for employees above.

It is not necessary for the miner to build entity-relationship diagrams, or use one of the other data modeling methodologies now available. Just understand the data, get whatever insight is possible, and understand the purpose for collecting it.

4.2.5 Relationship

With multiple data input streams, defining the relationship between streams is important. This relationship is easily specified as a common key that defines the correct association between instances in the input streams, thus allowing them to be merged. Because of the problems with possible inconsistency and pollution, merging the streams is not necessarily as easy to do as it is to describe! Because keys may be missing, it is important to check that the summaries for the assembled data set reflect the expected summary statistics for each individual stream. This is really the only way to be sure that the data is assembled as required.

Note that the data streams cannot be regarded as tables because of the potentially huge differences in format, media, and so on. Nonetheless, anyone who knows SQL is familiar with many of the issues in discovering the correct relationships. For instance, what should be done when one stream has keys not found in the other stream? What about duplicate keys in one stream without corresponding duplicates in another—which gets merged with what? Most of the SQL "join"-type problems are present in establishing the relationship between streams—along with a few additional ones thrown in for good measure.

4.2.6 Domain

Each variable consists of a particular domain, or range of permissible values. Summary statistics and frequency counts will reveal any erroneous values outside of the domain. However, some variables only have valid values in some conditional domain. Medical and insurance data typically has many conditional domains in which the values in one field, say, "diagnosis," are conditioned by values in another field, say, "gender." That is to say, there are some diagnoses that are valid only for patients of one particular gender.

Business or procedural rules enforce other conditional domains. For example, fraud investigations may not be conducted for claims of less than \$1000. A variable indicating that a fraud investigation was triggered should never be true for claims of less than \$1000.

Perhaps the miner doesn't know that such business rules exist. There are automated tools that can examine data and extract business rules and exceptions by examining data. A demonstration version of one such tool, WizRule, is included on the CD-ROM with this book. Such a rule report can be very valuable in determining domain consistency. Example 2 later in this chapter shows the use of this tool.

4.2.7 Defaults

Many data capturing programs include default values for some of the variables. Such default values may or may not cause a problem for the miner, but it is necessary to be aware of the values if possible. A default value may also be conditional, depending on the values of other entries for the actual default entered. Such conditional defaults can create seemingly significant patterns for the miner to discover when, in fact, they simply represent a lack of data rather than a positive presence of data. The patterns may be meaningful for predictive or inferential models, but if generated from the default rules inside the data capture system, they will have to be carefully evaluated since such patterns are often of limited value.

4.2.8 Integrity

Checking integrity evaluates the relationships permitted between the variables. For instance, an employee may have several cars, but is unlikely to be permitted to have multiple employee numbers or multiple spouses. Each field needs to be evaluated to determine the bounds of its integrity and if they are breached.

Thinking of integrity in terms of an acceptable range of values leads to the consideration of outliers, that is, values potentially out of bounds. But outliers need to be treated carefully, particularly in insurance and financial data sets. Modeling insurance data, as an example, frequently involves dealing with what look like outliers, but are in fact perfectly valid values. In fact, the outlier might represent exactly what is most sought, representing a massive claim far from the value of the rest. Fraud too frequently looks like outlying data since the vast majority of transactions are not fraudulent. The relatively few fraudulent transactions may seem like sparsely occurring outlying values.

4.2.9 Concurrency

When merging separate data streams, it may well be that the time of data capture is different from stream to stream. While this is partly a data access issue and is discussed in “Data Access Issues” above, it also needs to be considered and documented when characterizing the data streams.

4.2.10 Duplicate or Redundant Variables

Redundant data can be easily merged from different streams or may be present in one stream. Redundancy occurs when essentially identical information is entered in multiple variables, such as “date_of_birth” and “age.” Another example is “price_per_unit,” “number_purchased,” and “total_price.” If the information is not actually identical, the worst damage is likely to be only that it takes a longer time to build the models. However, most modeling techniques are affected more by the number of variables than by the number of instances. Removing redundant variables, particularly if there are many of them, will increase modeling speed.

If, by accident, two variables should happen to carry identical values, some modeling techniques—specifically, regression-based methods—have extreme problems digesting such data. If they are not suitably protected, they may cause the algorithm to “crash.” Such colinearity can cause major problems for matrix-based methods (implemented by some neural network algorithms, for instance) as well as regression-based methods. On the other hand, if two variables are almost colinear, it is often useful to create a new variable that expresses the difference between the nearly colinear variables.

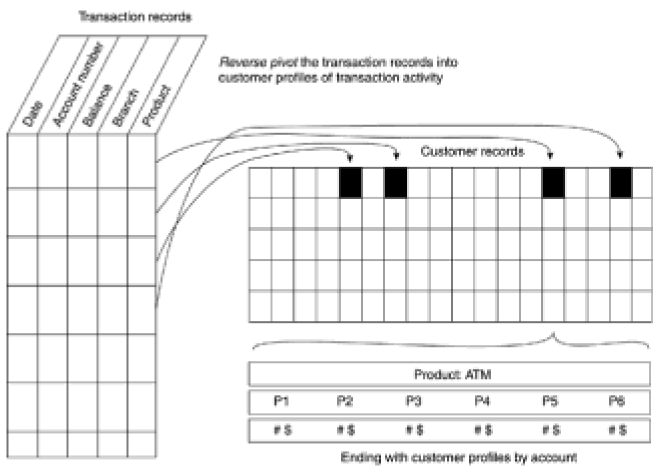
4.3 Data Set Assembly

At this point, the miner should know a considerable amount about the input streams and the data in them. Before the assay can continue, the data needs to be assembled into the table format of rows and columns that will be used for mining. This may be a simple task or a very considerable undertaking, depending on the content of the streams. One particular type of transformation that the miner often uses, and that can cause many challenges, is a *reverse pivot*.

4.3.1 Reverse Pivoting

Often, what needs to be modeled cannot be derived from the existing transaction data. If the transactions were credit card purchases, for example, the purchasing behavior of the cardholders may need to be modeled. The principal object that needs to be modeled, then, is the cardholder. Each transaction is associated with a particular account number unique to the cardholder. In order to describe the cardholder, all of the transactions for each particular cardholder have to be associated and translated into derived fields (or features) describing cardholder activity. The miner, perhaps advised by a domain expert, has to determine the appropriate derived fields that will contribute to building useful models.

Figure 4.3 shows an example of a reverse pivot. Suppose a bank wants to model customer activity using transaction records. Any customer banking activity is associated with an account number that is recorded in the transaction. In the figure, the individual transaction records, represented by the table on the left, are aggregated into their appropriate feature (Date, Account Number, etc.) in the constructed Customer Record. The Customer Record contains only one entry per customer. All of the transactions that a customer makes in a period are aggregated into that customer’s record. Transactions of different types, such as loan activity, checking activity, and ATM activity are represented. Each of the aggregations represents some selected level of detail. For instance, within ATM activity in a customer record, the activity is recorded by dollar volume and number of transactions within a period. This is represented by the expansion of one of the aggregation areas in the customer record. The “P*n*” represents a selected period, with “#” the number of transactions and “$” the dollar volume for the period. Such reverse pivots can aggregate activity into many hundreds of features.



Transaction records
Date
Account number
Balance
Branch
Product
Reverse pivot the transaction records into customer profiles of transaction activity
Customer records
Product: ATM
P1
P2
P3
P4
P5
P6
# $
# $
# $
# $
# $
# $
Ending with customer profiles by account

Figure 4.3 Illustrating the effect of a reverse pivot operation.

One company had many point-of-sale (POS) transactions and wanted to discover the main factors driving catalog orders. The POS transactions recorded date and time, department, dollar amount, and tender type in addition to the account number. These transactions were reverse pivoted to describe customer activity. But what were the appropriate derived features? Did time of day matter? Weekends? Public holidays? If so, how were they best described? In fact, many derived features proved important, such as the time in days to or from particular public holidays (such as Christmas) or from local paydays, the order in which departments were visited, the frequency of visits, the frequency of visits to particular departments, and the total amount spent in particular departments. Other features, such as tender type, returns to particular departments, and total dollar returns, were insignificant.

4.3.2 Feature Extraction

Discussing reverse pivoting leads to the consideration of feature extraction. By choosing to extract particular features, the miner determines how the data is presented to the mining tool. Essentially, the miner must judge what features might be predictive. For this reason, reverse pivoting cannot become a fully automated feature of data preparation. Exactly which features from the multitudinous possibilities are likely to be of use is a judgment call based on circumstance. Once the miner decides which features are potentially useful, then it is possible to automate the process of aggregating their contents from the transaction records.

Feature extraction is not limited to the reverse pivot. Features derived from other combinations of variables may be used to replace the source variables and so reduce the dimensionality of the data set. Even if not used to reduce dimensionality, derived features can add information that speeds the modeling process and reduces susceptibility to noise. Chapter 2 discussed the use of feature extraction as a way of helping expose the information content in a data set.

Physical models frequently require feature extraction. The reason for this is that when physical processes are measured, it is likely that very little changes from one stage to the next. Imagine monitoring the weather measured at hourly intervals. Probably the barometric pressure, wind speed, and direction change little in an hour. Interestingly, when the changes are rapid, they signify changing weather patterns. The feature of interest then is the amount of change in the measurements happening from hour to hour, rather than the absolute level of the measurement alone.

4.3.3 Physical or Behavioral Data Sets

There is a marked difference in the character of a physical data set as opposed to a behavioral data set. *Physical data sets* measure mainly physical characteristics about the world: temperature, pressure, flow rate, rainfall, density, speed, hours run, and so on. Physical systems generally tend to produce data that can be easily characterized according to the range and distribution of measurements. While the interactions between the variables may be complex or nonlinear, they tend to be fairly consistent. *Behavioral data*, on the other hand, is very often inconsistent, frequently with missing or incomplete values. Often a very large sample of behavioral data is needed to ensure a representative sample.

Industrial automation typically produces physical data sets that measure physical processes. But there are many examples of modeling physical data sets for business reasons. Modeling a truck fleet to determine optimum maintenance periods and to predict maintenance requirements also uses a physical data set. The stock market, on the other hand, is a fine example of a behavioral data set. The market reflects the aggregate result of millions of individual decisions, each made from individual motivations for each buyer or seller. A response model for a marketing program or an inferential model for fraud would both be built using behavioral data sets.

4.3.4 Explanatory Structure

Devising useful features to extract requires domain knowledge. Inventing features that might be useful without some underlying idea of why such a feature, or set of features, might be useful is seldom of value. More than that, whenever data is collected and used for a mining project, the miner needs to have some underlying idea, rationale, or theory as to why that particular data set can address the problem area. This idea, rationale, or theory forms the explanatory structure for the data set. It explains how the variables are expected to relate to each other, and how the data set as a whole relates to the problem. It establishes a reason for why the selected data set is appropriate to use.

Such an explanatory structure should be checked against the data, or the data against the explanation, as a form of “sanity check.” The question to ask is, Does the data work in the way proposed? Or does this model make sense in the context of this data?

Checking that the explanatory structure actually holds as expected for the data available is the final stage in the assay process. Many tools can be used for this purpose. Some of the most useful are the wide array of powerful and flexible OLAP (On-Line Analytical Processing) tools that are now available. These make it very easy to interactively examine an assembled data set. While such tools do not build models, they have powerful data manipulation and visualization features.

4.3.5 Data Enhancement or Enrichment

Although the assay ends with validating the explanatory structure, it may turn out that the data set as assembled is not sufficient. FNBA, for instance, might decide that affinity group membership information is not enough to make credit-offering decisions. They could add credit histories to the original information. This additional information actually forms another data stream and enriches the original data. *Enrichment* is the process of adding external data to the data set.

Note that data enhancement is sometimes confused with enrichment. *Enhancement* means embellishing or expanding the existing data set without adding external sources. *Feature extraction* is one way of enhancing data. Another method is introducing bias for a particular purpose. Adding bias introduces a *perspective* to a data set; that is, the information in the data set is more readily perceived from a particular point of view or for a particular purpose. A data set with a perspective may or may not retain its value for other purposes. Bias, as used here, simply means that some effect has distorted the measurements.

Consider how FNBA could enhance the data by adding a perspective to the data set. It is likely that response to a random FNBA mailing would be about 3%, a typical response rate for an unsolicited mailing. Building a response model with this level of response would present a problem for some techniques such as a neural network. Looking at the response data from the perspective of responders would involve increasing the concentration from 3% to, say, 30%. This has to be done carefully to try to avoid introducing any bias other than the desired effect. (Chapter 10 discusses this in more detail.) Increasing the density of responders is an example of enhancing the data. No external data is added, but the existing data is restructured to be more useful in a particular situation.

Another form of data enhancement is *data multiplication*. When modeling events that rarely occur, it may not be possible to increase the density of the rate of occurrence of the event enough to build good models. For example, if modeling catastrophic failure of some physical process, say, a nuclear power plant, or indicators predicting terrorist attacks on commercial aircraft, there is very little data about such events. What data there is cannot be concentrated enough to build a representative training data set. In this case it is possible to multiply the few examples of the phenomena that are available by carefully adding constructed noise to them. (See Chapter 10.)

Proposed enhancement or enrichment strategies are often noted in the assay, although they do not form an integral part of it.

4.3.6 Sampling Bias

Undetected sampling bias can cause the best-laid plans, and the most carefully constructed and tested model, to founder on the rocks of reality. The key word here is "undetected."

The goal of the U.S. census, for instance, is to produce an unbiased survey of the population by requiring that everyone in the U.S. be counted. No guessing, no estimation, no statistical sampling; just get out and count them. The main problem is that this is not possible. For one thing, the census cannot identify people who have no fixed address: they are hard to find and very easily slip through the census takers' net. Whatever characteristics these people would contribute to U.S. demographic figures are simply missing. Suppose, simply for the sake of example, that each of these people has an extremely low income. If they were included in the census, the "average" income for the population would be lower than is actually captured.

Telephone opinion polls suffer from the same problem. They can only reach people who have telephones for a start. When reached, only those willing to answer the pollster's questions actually do so. Are the opinions of people who own telephones different from those who do not? Are the opinions of those willing to give an opinion over the telephone different from those who are not? Who knows? If the answer to either question is "Yes," then the opinions reflected in the survey do not in fact represent the population as a whole.

Is this bias important? It may be critical. If unknown bias exists, it is a more or less unjustified assumption that the data reflects the real world, and particularly that it has any bearing on the issue in question. Any model built on such assumptions reflects only the distorted data, and when applied to an undistorted world, the results are not likely to be as anticipated.

Sampling bias is in fact impossible to detect using only the data set itself as a reference. There are automated methods of deriving measurements about the data set indicating the possible presence of sampling bias, but such measurements are no more than indicators. These methods are discussed in Chapter 11, which deals with the data survey. The assay cannot use these automated techniques since the data survey requires a fully assembled and prepared data set. This does not exist when the assay is being made.

At this stage, using the explanatory structure for the data, along with whatever domain knowledge is available, the miner needs to discover and explicate any known bias or biases that affected the collection of the data. Biasing the data set is sometimes desirable, even necessary. It is critical to note intentional biases and to seek out other possible sources of bias.

4.4 Example 1: CREDIT

The purpose of the data assay, then, is to check that the data is coherent, sufficient, can be assembled into the needed format, and makes sense within a proposed framework. What does this look like in practice?

For FNBA, much of the data comes in the form of credit histories purchased from credit bureaus. During the solicitation campaign, FNBA contacts the targeted market by mail and telephone. The prospective credit card user either responds to the invitation to take a credit card or does not respond. One of the data input streams is (or includes) a flag indicating if the targeted person responded or not. Therefore, the initial model for the campaign is a predictive model that builds a profile of people who are most likely to respond. This allows the marketing efforts to be focused on only that segment of the population that is most likely to want the FNBA credit card with the offered terms and conditions.

4.4.1 Looking at the Variables

As a result of the campaign, various data streams are assembled into a table format for mining. (The file CREDIT that is used in this example is included on the accompanying CD-ROM. Table 4.1 shows entries for 41 fields. In practice, there will usually be far more data, in both number of fields and number of records, than are shown in this example. There is plenty of data here for a sample assay.)

TABLE 4.1 Status report for the CREDIT file.

FIELD	MAX	MIN	DISTINCT	EMPTY	CONF	REQ	VAR	LIN	VAR-TYPE
AGE _INFERR	57.0	35.0	3	0	0.96	280	0.8	0.9	N
BCBAL	24251.0	0.0	3803	211	0.95	1192	251.5	0.8	N
BCLIMIT	46435.0	0.0	2347	151	0.95	843	424.5	0.9	N

BCOPEN	0.0	0.0	1	59	0.95	59	0.0	0.0	E
BEACON _C	804.0	670.0	124	0	0.95	545	1.6	1.0	N
BUYER	1.0	0.0	2	0	0.95	353	0.1	0.7	N
CHILDREN	1.0	0.0	2	0	0.95	515	0.0	0.8	N
CRITERIA	1.0	1.0	1	0	0.95	60	0.0	0.0	N
DAS _C	513.0	–202.0	604	0	0.95	437	10.3	1.0	N
DOB _MONTH	12.0	0.0	14	8912	0.95	9697	0.3	0.6	N
DOB _YEAR	70.0	0.0	42	285	0.95	879	0.5	1.0	N
EQBAL	67950.0	0.0	80	73	0.95	75	0.0	1.0	E
EQCURBAL	220000.0	0.0	179	66	0.95	67	0.0	0.0	E
EQHIGHBAL	237000.0	0.0	178	66	0.95	67	0.0	0.0	E
EQLIMIT	67950.0	0.0	45	73	0.95	75	0.0	1.0	E
EST _INC_C	87500.0	43000.0	3	0	0.95	262	1514.0	0.9	N
HOME _ED	160.0	0.0	8	0	0.95	853	3.5	0.7	N
HOME _INC	150.0	0.0	91	0	0.95	1298	0.7	0.9	N
HOME _VALUE	531.0	0.0	191	0	0.95	870	2.6	0.9	N
ICURBAL	126424.0	0.0	4322	1075	0.96	2263	397.4	0.9	N
IHIGHBAL	116545.0	0.0	4184	573	0.96	1192	951.3	0.9	N

LST _R_OPEN	99.0	0.0	100	9	0.96	482	3.6	0.9	N
MARRIED	0.0	0.0	2	0	0.95	258	0.2	0.0	C
MOF	976.0	0.0	528	0	0.95	951	3.8	0.9	N
MTCURBAL	578000.0	0.0	3973	433	0.95	919	3801.7	1.0	N
MTHIGHBAL	579000.0	0.0	1742	365	0.95	779	4019.7	0.9	N
OWN _HOME	0.0	0.0	1	0	0.95	60	0.0	0.0	N
PRCNT _PROF	86.0	0.0	66	0	0.95	579	0.8	1.0	N
PRCNT _WHIT	99.0	0.0	58	0	0.95	568	3.3	0.6	N
RBAL	78928.0	0.0	5066	18	0.97	795	600.3	0.8	N
RBALNO	14.0	0.0	14	0	0.95	642	0.1	0.9	N
RBAL _LIMIT	9.0	0.0	10	0	0.95	618	0.1	0.8	N
RLIMIT	113800.0	0.0	6067	11	0.95	553	796.3	0.9	N
ROPEN	17.0	0.0	17	0	0.96	908	0.1	0.9	N
SEX	0.0	0.0	3	0	0.95	351	0.2	0.0	C
TBALNO	370260.0	0.0	7375	9	0.95	852	2383.7	0.7	N
TOPEN	17.0	0.0	18	0	0.95	617	0.1	0.9	N
UNSECBAL	23917.0	0.0	2275	781	0.95	1349	420.1	0.8	N
UNSECLIMIT	39395.0	0.0	1596	906	0.95	1571	387.9	0.9	N
YEARS_RES	15.0	0.0	17	21	0.95	431	0.4	0.9	N

_Q_MVP	0.0	0.0	207	0	0.95	1086	0.4	0.1	C

How is this data assayed? Start looking at the basic statistics for the file. Table 4.1 shows a statistics file produced by the data preparation software on the accompanying CD-ROM for the file CREDIT. How does this file help?

First, the column headings indicate the following measurements about the variables:

- FIELD. The name of the field.
- MAX. The maximum value sampled for numeric variables.
- MIN. The minimum value sampled for numeric variables.
- DISTINCT. The number of distinct values for the variable in the sample. For example, if the field "months" was recorded with standardized three-letter abbreviations, there are a maximum of 12 valid, distinct values that the field can contain. A missing value counts toward the total number of distinct values, so the field "months" can have 13 distinct values including the value "missing." More than 13 values clearly indicates that erroneous entries are polluting the data.
- EMPTY. The number of records with missing values.
- CONF. The confidence level that the variability was captured. (Confidence levels, and how they are discovered and used, are covered in Chapter 5 and are not used in the assay.)
- REQ. The minimum number of records required to establish the confidence level. (See Chapter 5.)
- VAR. A measure of the variability in a variable. (See Chapter 5.)
- LIN. A measure of interstitial linearity (again, discussed in Chapter 5) and used in the assay. Interstitial linearity is one measure used to indicate possible problems with a variable, including monotonicity.
- VARTYPE. The type of variable detected. "N" indicates numeric, "C" indicates character, "E" indicates empty. (The demonstration code will only recognize these three types.)

Now, consider what can be learned about a few of the fields:

- AGE_INFERR. This has three discrete values, and every field has one of the three values. This is a numeric variable.

- BCOPEN. This is a completely empty variable; that is, none of the records has an entry in this field. Thus it has one distinct value (missing) in all of the records.

- BEACON_C. As a rule of thumb, if the linearity of a variable (LIN) is above 0.98, it is worth checking if the variable is monotonic. (As it happens it isn't in this case, but knowing that requires domain knowledge.)

- CRITERIA. This is shown as a numeric variable having one DISTINCT value and no variance (indicated by the 0.0 entry in VAR). This means that while all of the values are populated, they all have the same value. So this is actually a constant, not a variable, and it should be removed.

- EQBAL. What is going on here? It is shown as empty ("E" in VARTYPE) and yet it contains 80 DISTINCT values! This is a feature of the sampling process. As shown in REQ, it required 75 samples to establish the confidence level needed. Out of those 75 sampled, 73 were EMPTY, which was sufficient to establish the required level of confidence that it was indeed empty below the required threshold. From that point on, the variable was no longer sampled. This speeds the sampling process. Other variables required far more samples to establish their required confidence level. At the end of the sampling process, the data preparation software builds a fully populated sample file with prepared data. When the full sample was taken, the full range of what was found in EQBAL was noted. The 80 in DISTINCT indicates that although the variable was populated at too low a level for use at the required confidence level, it still did have some very sparse content and that sparse content did have 80 distinct values. However, since it was too empty to use, it is not included in the prepared data.

- DOB_MONTH. This variable sits almost on the edge of falling below the selected sparsity threshold. It is not quite 95% empty, the level required for rejection in this example, but it is 92% (8912/9697) empty. Because of the emptiness and distortion, the system required 785 (9697 – 8912) nonempty samples to capture its variability. Even if the miner elected to use this field in the final model, there is still the question of why there are 14 months. To discover what is possibly wrong here, another report produced by the demonstration software is needed, the "Complete Content" report. This is a very large report listing, among other things, all of the values discovered in the sample along with their frequencies. Table 4.2 shows the part of the Complete Content report that covers DOB_MONTH, the part of the interest here. From inspection of the CONTENT it seems obvious that "00" serves as a surrogate for a missing value. Adding the 646 "00" with the 8912 that are missing, this takes the variable below the sparsity threshold selected and the variable should be discarded.

TABLE 4.2 Part of the Complete Content report for the CREDIT data.

FIELD	CONTENT	CCOUNT
DOB_MONTH		8912
DOB_MONTH	00	646
DOB_MONTH	01	12
DOB_MONTH	02	7
DOB_MONTH	03	10
DOB_MONTH	04	9
DOB_MONTH	05	15
DOB_MONTH	06	14
DOB_MONTH	07	11
DOB_MONTH	08	10
DOB_MONTH	09	13
DOB_MONTH	10	10
DOB_MONTH	11	15
DOB_MONTH	12	13

• HOME_VALUE. There are no empty values. Nonetheless, it does not seem likely that 0.0, shown in MIN as the minimum value, is a reasonable home valuation! There are 191 DISTINCT values, but how many are “0.0”? The appropriate part of the Complete Content report (Table 4.3) again shows what is happening. Once again it may seem obvious that the value 000 is a surrogate of a missing value. It may be beneficial to replace the 000 with a blank so that the system will treat it as a missing value rather than treating it as if it had a valid value of 000. On the other hand, it may be that a renter, not owning a home, is shown as having a 000 home value. In that case, the value acts as a "rent/own" flag, having a completely different meaning and perhaps a different significance. Only domain knowledge can really answer this question.

TABLE 4.3 Part of the Complete Content report showing the first few values of HOME_VALUE.

FIELD	CONTENT	CCOUNT
HOME_VALUE	000	284
HOME_VALUE	027	3
HOME_VALUE	028	3
HOME_VALUE	029	3
HOME_VALUE	030	3
HOME_VALUE	031	2
HOME_VALUE	032	5

4.4.2 Relationships between Variables

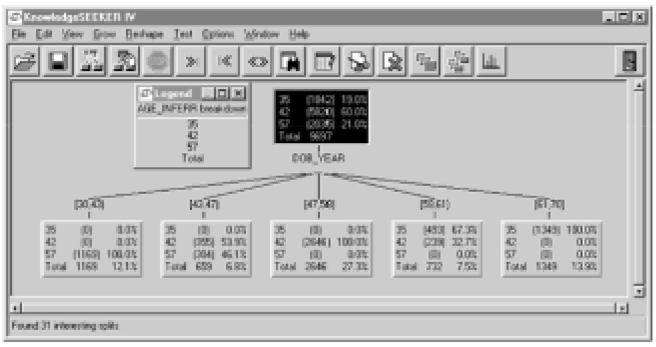
Each field, or variable, raises various questions similar to those just discussed. Is this range of values reasonable? Is the distribution of those values reasonable? Should the variable be kept or removed? Just the basic report of frequencies can point to a number of questions, some of which can only be answered by understanding the domain. Similarly, the relationship between variables also needs to be considered.

In every data mining application, the data set used for mining should have some underlying rationale for its use. Each of the variables used should have some expected relationship with other variables. These expected relationships need to be confirmed during the assay. Before building predictive or inferential models, the miner needs at least some assurance that the data represents an expected reflection of the real world. An excellent tool to use for this exploration and confirmation is a single-variable CHAID analysis. Any of the plethora of OLAP tools may also provide the needed confirmation or denial between variable relationships.

CHAID is an acronym for *chi-square automatic interaction detection*. CHAID, as its name suggests, detects interactions between variables. It is a method that partitions the values of one variable based on significant interactions between that variable and another one. KnowledgeSEEKER, a commercially available tree tool, uses the CHAID algorithm. Instead of letting it grow trees when used as an assaying tool, it is used to make single-variable analyses. In other words, after selecting a variable of interest, KnowledgeSEEKER compares that variable against only one other variable at a time. When allowed to self-select a variable predictive of another, KnowledgeSEEKER selects the one with the highest detected interaction. If two selected variables are to be compared, that can be done as well.

Using KnowledgeSEEKER to explore and confirm the internal dynamics of the CREDIT data set is revealing. As a single example, consider the variable AGE_INFERR (i.e., inferred age). If the data set truly reflects the world, it should be expected to strongly correlate with the variable DOB_YEAR.

Figure 4.4(a) shows what happened when KnowledgeSEEKER found the most highly interacting variable for AGE_INFERR. It discovered DOB_YEAR as expected. Figure 4.4(b) graphs the interaction, and it can easily be seen that while the match is not perfect, the three estimated ages do fit the year of birth very closely. But this leads to other questions: Why are both measures in the data set, and are they both needed? This seems to be a redundant pair, one of which may be beneficially eliminated. But is that really so? And if it is, which should be kept? As with so many things in life, the answer is, that depends! Only by possessing domain knowledge, and by examining the differences between the two variables and the objectives, can the miner arrive at a good answer.



```
AGE_INFERR breakdown
35
42
57
Total

35     (1842)  19.0%
42     (5820)  60.0%
57     (2035)  21.0%
Total  9697
DOB_YEAR

[20,42)               [42,47)              [47,58)                [58,61)              [61,70]
35    (0)     0.0%    35    (0)    0.0%    35    (0)     0.0%     35    (493)  67.3%   35    (1349)  100.0%
42    (0)     0.0%    42    (355)  53.9%   42    (2646)  100.0%   42    (239)  32.7%   42    (0)     0.0%
57    (1169)  100.0%  57    (304)  46.1%   57    (0)     0.0%     57    (0)    0.0%    57    (0)     0.0%
Total 1169    12.1%   Total 659    6.8%    Total 2646    27.3%    Total 732    7.5%    Total 1349    13.9%

Found 31 interesting splits
```

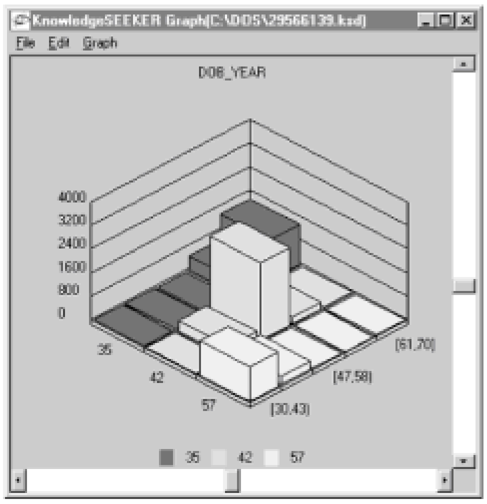


Figure 4.4 KnowledgeSEEKER tree showing interaction between AGE_INFERR and the most strongly interacting variable, DOB_YEAR (a). Graphing the detected interaction between AGE_INFERR and DOB_YEAR (b).

Exploring the data set variable by variable and finding which are the most closely interacting variables is very revealing. This is an important part of any assay. It is also important to confirm that any expected relationships, such as, say, between HOME_ED (the educational level in a home) and PRCNT_PROF (the professional level of the applicant), do in fact match expectations, even if they are not the most closely interacting. It seems reasonable to assume that professionals have, in general, a higher level of education than nonprofessionals. If, for instance, it is not true for this data set, a domain expert needs to determine if this is an error.

Some data sets are selected for particular purposes and do not in fact represent the general population. If the bias, or distortion, is intentionally introduced, then exactly why that bias is considered desirable needs to be made clear. For instance, if the application involves marketing child-related products, a data set might be selected that has a far higher predominance of child-bearing families than normally occur. This deliberately introduced distortion needs to be noted.

4.5 Example 2: SHOE

A national shoe chain wants to model customer profiles in order to better understand their market. More than 26,000 customer-purchase profiles are collected from their national chain of shoe stores. Their first question should be, Does the collected information help us understand customer motivations? The first step in answering this question is to assay the data. (This sample data set is also included on the accompanying CD-ROM.)

4.5.1 Looking at the Variables

Table 4.4 shows the variable status report from the demonstration preparation software for the SHOE data set.

TABLE 4.4 Variable Status report for the file SHOE.

FIELD	MAX	MIN	DISTINCT	EMPTY	CONF	REQ	VAR	LIN	VAR-TYPE
AGE	50.0	19.0	6	89	0.95	736	0.52	0.96	N
CITY	0.0	0.0	1150	0	0.95	659	3.54	0.67	C
GENDER	0.0	0.0	5	6	0.95	356	0.16	0.01	C
MILES _WEEK	51.0	0.0	9	322	0.95	846	0.85	0.92	N
PURCHASENU	2.0	1.0	2	0	0.95	1152	0.02	0.40	N
RACES _YEAR	10.0	0.0	7	1480	0.95	2315	0.13	0.83	N
SHOECODE	0.0	0.0	611	1	0.95	378	2.19	0.57	C
SOURCE	0.0	0.0	18	0	0.95	659	0.18	0.02	C
STATE	0.0	0.0	54	0	0.95	910	0.45	0.05	C
STORECD	0.0	0.0	564	51	0.95	389	2.10	0.50	C
STYLE	0.0	0.0	89	111	0.95	691	0.48	0.09	C
TRIATHLETE	0.0	0.0	3	62	0.95	321	0.21	0.01	C
YEARSRUNNI	10.0	0.0	7	321	0.95	1113	0.16	0.80	N
ZIP3	0.0	0.0	513	0	0.95	224	2.28	0.69	C
_Q_MVP	0.0	0.0	66	0	0.95	1035	0.25	0.05	C

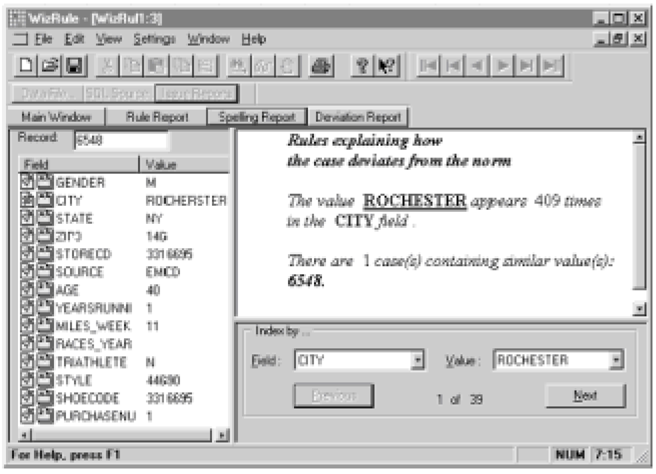
Note that there are apparently five DISTINCT values for GENDER, which indicates a possible problem. A look at the appropriate part of the Complete Content report (Table 4.5) shows that the problem is not significant. In fact, in only one case is the gender inappropriately given as “A,” which is almost certainly a simple error in entry. The entry will be better treated as missing.

TABLE 4.5 Complete Content report for the SHOE data set.

FIELD	CONTENT	CCOUNT
GENDER		45
GENDER	A	1
GENDER	F	907
GENDER	M	1155
GENDER	U	207

Any file might contain various exception conditions that are not captured in the basic statistical information about the variables. To discover these exception conditions, the miner needs a different sort of tool, one that can discover rules characterizing the data and reveal exceptions to the discovered rules. WizRule was used to evaluate the SHOE file and discovered many apparent inconsistencies.

Figure 4.5 shows one example: the “Spelling Report” screen generated for this data set. It discovered that the city name “Rochester” occurs in the file 409 times and that the name “Rocherster” is enough like it that it seems likely (to WizRule) that it is an error. Figure 4.6 shows another example. This is part of the “Rule Report” generated for the file. Rule 1 seems to have discovered possible erroneous values for the field TRIATHLETE, and it lists the record numbers in which the exception occurs.



Record: 6548

Field	Value
GENDER	M
CITY	ROCHERSTER
STATE	NY
ZIP3	146
STORECD	3316695
SOURCE	EMCD
AGE	40
YEARSRUNNI	1
MILES_WEEK	11
RACES_YEAR	
TRIATHLETE	N
STYLE	44690
SHOECODE	3316695
PURCHASENU	1

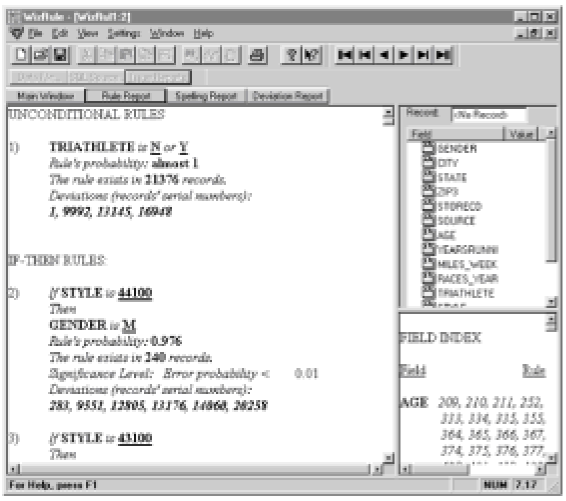
Rules explaining how
the case deviates from the norm

The value **ROCHESTER** *appears* 409 *times in the* **CITY** *field* .

There are 1 *case(s) containing similar value(s):*
6548.

Field: CITY
Value: ROCHESTER
1 of 39

Figure 4.5 WizRule Spelling Report for the table SHOE. WizRule has discovered 409 instances of “Rochester” and concludes that the value “Rocherster” (shown in the left window) is similar enough that it is likely to be an error.



UNCONDITIONAL RULES

1) **TRIATHLETE** *is* N *or* Y
Rule's probability: **almost 1**
The rule exists in ***21376*** *records.*
Deviations (records' serial numbers):
1, 9992, 13145, 16948

IF-THEN RULES:

2) *If* **STYLE** *is* **44100**
Then
GENDER *is* **M**
Rule's probability: **0.976**
The rule exists in ***240*** *records.*
Significance Level: Error probability < 0.01
Deviations (records' serial numbers):
283, 9551, 12805, 13176, 14868, 20258

3) *If* **STYLE** *is* **43100**
Then

Record: <No Record>

Field	Value
GENDER	
CITY	
STATE	
ZIP3	
STORECD	
SOURCE	
AGE	
YEARSRUNNI	
MILES_WEEK	
RACES_YEAR	
TRIATHLETE	

FIELD INDEX

Field	Rule
AGE	*209, 210, 211, 252, 333, 334, 335, 355, 364, 365, 366, 367, 374, 375, 376, 377,*

Figure 4.6 WizRule Rule Report for the SHOE file. Rule 1 has discovered four possible instance value errors in the TRIATHLETE field.

The reports produced by WizRule characterize the data and the data set and may raise many questions about it. Actually deciding what is an appropriate course of action obviously requires domain knowledge. It is often the case that not much can be done to remedy the problems discovered. This does not mean that discovering the problem has no value. On the contrary, knowing that there is a potential problem that can’t be fixed is very important to judging the value of the data.

4.5.2 Relationships between Variables

When the variables are investigated using the single-variable CHAID technique, one relationship stands out. Figure 4.7 shows a graphical output from KnowledgeSEEKER when investigating SOURCE. Its main interaction is with a variable _Q_MVP. This is a variable that does not exist in the original data set. The data preparation software creates this variable and captures information about the missing value patterns. For each pattern of missing values in the data set, the data preparation software creates a unique value and enters the value in the _Q_MVP field. This information is very useful indeed. Often the particular pattern of missing values can be highly predictive. Chapter 8 discusses missing values in more detail.

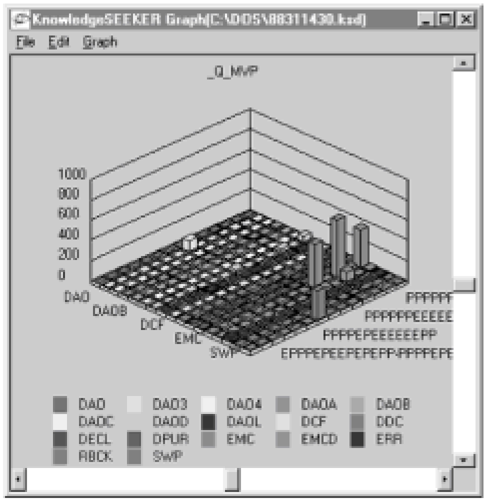


Figure 4.7 Graph showing the interaction between the variable SOURCE and the variable most interacting with it, _Q_MVP, in the file SHOE.

In this case it is clear that certain patterns are very highly associated with particular SOURCE codes. Is this significant? To know that requires domain knowledge. What is important about discovering this interaction is to try to account for it, or if an underlying explanation cannot be immediately discovered, it needs to be reported in the assay documentation.

4.6 The Data Assay

So far, various components and issues of the data assay have been discussed. The assay literally assesses the quality or worth of the data for mining. Note, however, that during the assay there was no discussion of what was to be modeled. The focus of the assay is entirely on how to get the data and to determine if the data suits the purpose. It is quite likely that issues are raised about the data during the assay that could not be answered. It may be that one variable appears to be outside its reasonable limits, or that an expected interaction between variables wasn't found. Whatever is found forms the result of the assay. It delineates what is known and what is not known and identifies problems with the data.

Creating a report about the state of the data is helpful. This report is unique to each data set and may be quite detailed and lengthy. The main purpose of the assay, however, is not to produce a voluminous report, but for the miner to begin to understand where the data comes from, what is in the data, and what issues remain to be established—in other words, to determine the general quality of the data. This forms the foundation for all preparation and mining work that follows. Most of the work of the assay involves the miner directly finding and manipulating the data, rather than using automated preparation tools. Much of the exploratory work carried out during the assay is to discover sources and confirm expectations. This requires domain expertise, and the miner will usually spend time either with a domain expert or learning sufficient domain knowledge to understand the data.

Once the assay is completed, the mining data set, or sets, can be assembled. Given assembled data sets, much preparatory work still remains to be done before the data is in optimum shape for mining. There remain many data problems to discover and resolve. However, much of the remaining preparation can be carried out by the appropriate application of automated tools. Deciding which tools are appropriate, and understanding their effect and when and how to use them, is the focus of the remaining chapters.

Chapter 5: Sampling, Variability, and Confidence

Sampling, or First Catch Your Hare!

Mrs. Beaton's famous English cookbook is alleged to have contained a recipe for Jugged Hare that started, "First. Catch your hare." It is too good a line to pass up, true or not. If you want the dish, catching the hare is the place to start. If you want to mine data, catching the "hare" in the data is the place to start. So what is the "hare" in data? The hare is the information content enfolded into the data set. Just as hare is the essence of the recipe for Jugged Hare, so information is the essence of the recipe for building training and test data sets.

Clearly, what is needed is enough data so that all of the relationships at all levels—superstructure, macrostructure, and microstructure—are captured. An easy answer would seem to be to use all the data. After all, with all of the data being used, it is a sure thing that any relationship of interest that the data contains is there to be found. Unfortunately, there are problems with the idea of using all of the data.

5.1.1 How Much Data?

One problem with trying to use all of the data, perhaps the most common problem, is simply that all of the data is not available. It is usual to call the whole of data the *population*. Strictly speaking, the data is not the population; the data is simply a set of measurements about the population of objects. Nonetheless, for convenience it is simply easier to talk about a population and understand that what is being discussed is the data, not the objects. When referring to the objects of measurement, it is easy enough to make it clear that the objects themselves are being discussed.

Suppose that a model is to be built about global forestry in which data is measured about individual trees. The population is at least all of the trees in the world. It may be, depending on the actual area of interest, all of the trees that have ever lived, or even all of the trees that could possibly live. Whatever the exact extent of the population, it is clearly unreasonable to think that it is even close to possible to have data about the whole population.

Another problem occurs when there is simply too much data. If a model of credit card transactions is proposed, most of these do actually exist on computers somewhere. But even if a computer exists that could house and process such a data set, simply accumulating all of the records would be at least ridiculously difficult if not downright impossible.

Currency of records also presents difficulties. In the case of the credit card transactions, even with the data coming in fast and furious, there would be no practical way to keep the data set being modeled reflecting the current state of the world's, or even the nation's, transactions.

For these reasons, and for any other reason that prevents having access to data about the whole population, it is necessary to deal with data that represents only some part of the population. Such data is called a *sample*.

Even if the whole of the data is available, it is still usually necessary to sample the data when building models. Many modeling processes require a set of data from which to build the model and another set of data on which to test it. Some modeling processes, such as certain decision tree algorithms, require three data sets—one to build the tree, one to prune the tree, and one to test the final result. In order to build a valid model, it is absolutely essential that each of the samples reflects the full set of relationships that are present in the whole population. If this is not the case, the model does not reflect what will be found in the population. Such a model, when used, will give inaccurate or misleading results.

So, sampling is a necessary evil. However, when preparing the data for modeling, the problem is not quite so great as when actually building the model itself. At least not in the early stages. Preparing the variables requires only that sufficient information about each individual variable be captured. Building data mining models requires that the data set used for modeling captures the full range of interactions between the variables, which is considered later, in Chapter 10. For now the focus is on capturing the variations that occur within each variable.

5.1.2 Variability

Each variable has features, many of which were discussed in Chapter 2. However, the main feature is that a variable can take on a variety of values, which is why it is called a variable! The actual values that a variable can have contain some sort of pattern and will be distributed across the variable's range in some particular way. It may be, for example, that for some parts of the range of values there are many instances bunched together, while for other parts there are very few instances, and that area of the range is particularly sparsely populated. Another variable may take on only a limited number of values, maybe only 5 or 10. Limited-value distribution is often a feature of categorical variables.

Suppose, for instance, that in a sample representative of the population, a random selection of 80 values of a numeric variable are taken as follows:

49, 63, 44, 25, 16, 34, 62, 55, 40, 31, 44, 37, 48, 65, 83, 53, 39, 15, 25, 52
68, 35, 64, 71, 43, 76, 39, 61, 51, 30, 32, 74, 28, 64, 46, 31, 79, 69, 38, 69

53, 32, 69, 39, 32, 67, 17, 52, 64, 64, 25, 28, 64, 65, 70, 44, 43, 72, 37, 31
67, 69, 64, 74, 32, 25, 65, 39, 75, 36, 26, 59, 28, 23, 40, 56, 77, 68, 46, 48

What exactly can we make of them? Is there any pattern evident? If there is, it is certainly hard to see. Perhaps if they are put into some sort of order, a pattern might be easier to see:

15, 16, 17, 23, 25, 25, 25, 25, 26, 28, 28, 28, 30, 31, 31, 31, 32, 32, 32, 32
34, 35, 36, 37, 37, 38, 39, 39, 39, 39, 40, 40, 43, 43, 44, 44, 44, 46, 46, 48
48, 49, 51, 52, 52, 53, 53, 55, 56, 59, 61, 62, 63, 64, 64, 64, 64, 64, 64, 65
65, 65, 67, 67, 68, 68, 69, 69, 69, 69, 70, 71, 72, 74, 74, 75, 76, 77, 79, 83

Maybe there is some sort of pattern here, but it is hard to tell exactly what it is or to describe it very well. Certainly it seems that some numbers turn up more often than others, but exactly what is going on is hard to tell.

Perhaps it would be easier to see any pattern if it were displayed graphically. Since the lowest number in the sample is 15, and the highest 83, that is the range of this sample. A *histogram* is a type of graph that uses columns to represent counts of features. If the sample is displayed as a histogram, some sort of pattern is easier to see, and Figure 5.1 shows a histogram of this sample. Each column in Figure 5.1 shows, by its height, the number of instances of a particular value. Each column represents one particular value. The first column on the left, for example, represents the value 15, and the column height indicates that there is one of this value in the sample.

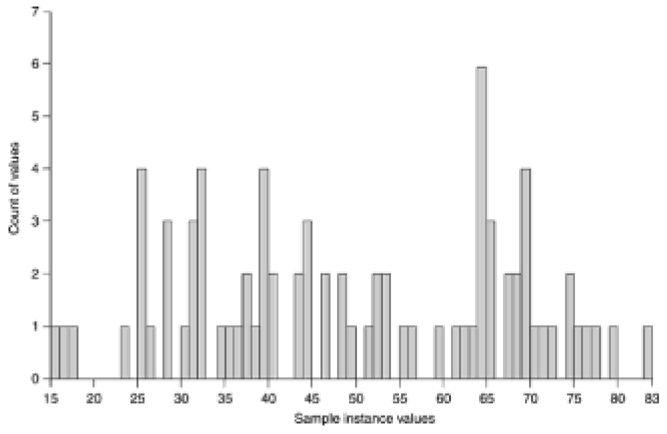


Figure 5.1 Histogram of a numeric variable sample. The column positions represent the magnitude of each of the values. The height of each column represents the count of instance values of the appropriate measured value.

The histogram in Figure 5.1 certainly makes some sort of pattern easier to see, but because of the number of columns, it is still hard to detect an overall pattern. Grouping the values together, shown in Figure 5.2, might make it easier to see a pattern. In this histogram each column represents the count of instances that are in a particular range. The leftmost column has a zero height, and a range of 0 to 9.99 (less than 10). The next column has a range from 10 to less than 20, and a height of 3. This second column aggregates the values 15, 16, and 17, which are all there are in the range of the column. In this figure the pattern is easier to see than in the previous figure.

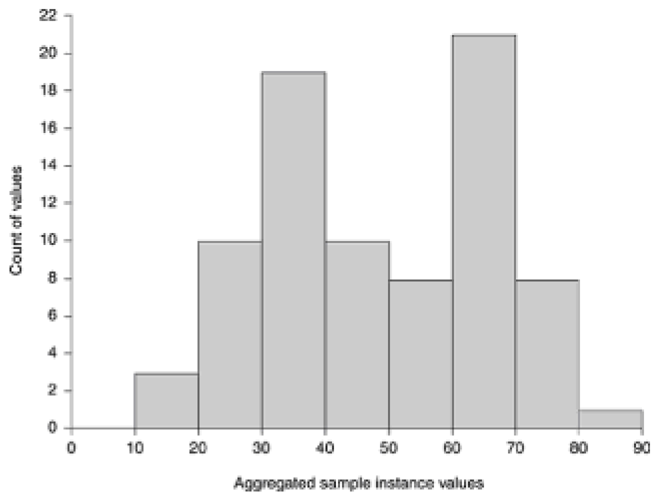


Figure 5.2 A histogram with vertical columns representing the count for a range of values.

Another way to see the distribution pattern is to use a graph that uses a continuous line, called a *curve*, instead of columns. Figure 5.3 shows a distribution curve that uses each value, just as in Figure 5.1. Again, the curve is very jagged. It would be easier to see the nature of the distribution if the curve were smoother. Curves can be easily smoothed, and Figure 5.4 shows the curve using two smoothing methods. One method (shown with the unbroken line) uses the average of three values; the other (shown with the dashed line) uses the average of five values. Smoothing does make the pattern easier to see, but it seems to be a slightly different pattern that shows up with each method. Which is the "correct" pattern shape for this distribution, if any?

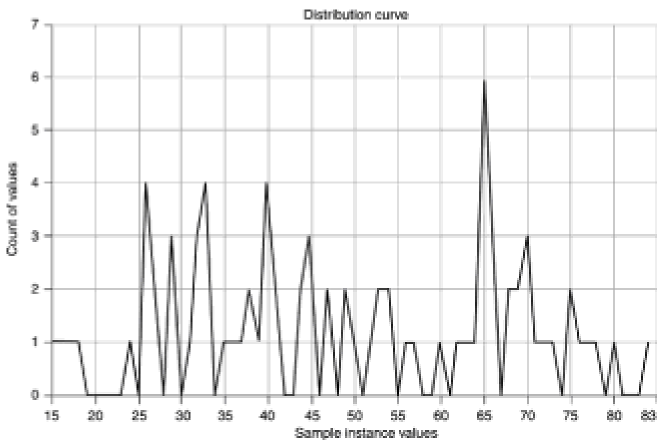


Figure 5.3 Sample value counts individually plotted and shown by a continuous line instead of using columns.

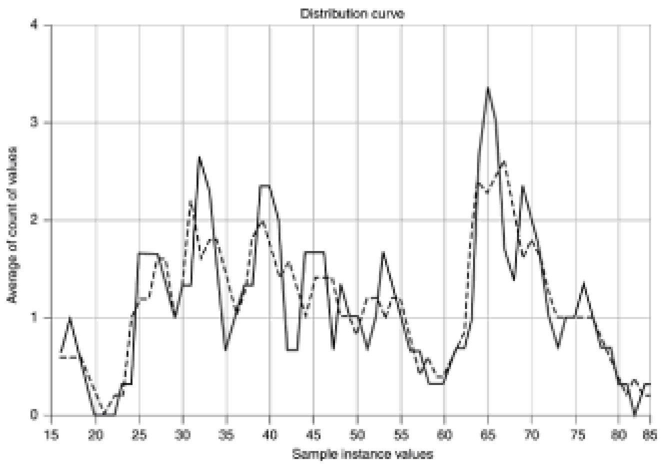


Figure 5.4 Results of two “smoothing methods, both using the average of a number of instance values. The solid line uses the average of three values, and the dashed line uses the average of five values.

There are two problems here. Recall that if this sample is indeed representative of the population, and for the purposes of this discussion we will assume that it is, then any other representative random sample drawn from the same population will show these patterns.

The first problem is that until a representative sample is obtained, and known to be representative, it is impossible to know if the pattern in some particular random sample does, in fact, represent the “true” variability of the population. In other words, if the true population distribution pattern is unknown, how can we know how similar the sample distribution curve is to the true population distribution curve?

The second problem is that, while it is obvious that there is some sort of pattern to a distribution, various ways of looking at it seem to produce slightly different patterns. Which of all these shapes, if any, is the right one to use?

5.1.3 Converging on a Representative Sample

The first problem, getting a representative sample, can be addressed by a phenomenon called *convergence*. Taking a sample starts by selecting instance values from a population, one at a time and at random. The sample starts at size 0. For any sample size a distribution curve can be created for the sample, similar to those shown in the earlier figures. In fact, although tedious for a human being, the distribution curve can be recalculated every time an instance value is added to the sample.

Suppose that the sample distribution curve is recalculated with each additional instance added. What will it look like? At first, when the number of instances in the sample is low, each addition will make a big impact on the shape of the curve. Every new instance added will make the curve “jump” up quite noticeably. Almost every instance value added to the sample will make a large change in the shape of the distribution curve. After a while, however, when the number of instances in the sample is modestly large, the overall shape of the curve will have settled down and will change little in shape as new instances are added. It will continue to increase in height because with more points in the sample, there are more points under any particular part of the curve. When there are a large number of instances in the sample, adding another instance barely makes any difference at all to the overall shape. The important point here is that the overall shape of the curve will settle down at some point.

This “settling down” of the overall curve shape is the key. As more instances are added, the actual shape of the curve becomes more like some final shape. It may never quite get there, but it gets closer and closer to settling into this final, unchanging shape. The curve can be thought of as “approaching” this ultimate shape. Things are said to converge when they come together, and in this sense the sample distribution curve converges with the final shape that the curve would have if some impossibly large number of instances were added. This impossibly large number of instances, of course, is the population. So the distribution curve in any sample converges with the distribution curve of the population as instances selected at random are added to the sample.

In fact, when capturing a sample, what is measured is not the shape of the curve, but the variability of the sample. However, the distribution curve shape is produced by the variability, so both measures represent very much the same underlying phenomenon. (And to understand what is happening, distribution curves are easier to imagine than variability.)

5.1.4 Measuring Variability

The other problem mentioned was that the distribution curve changes shape with the width of the columns, or the smoothing method. This problem is not so easy to address. What is really required instead of using column widths or smoothing is some method of measuring variability that does not need any arbitrary decision at all. Ideally, we need some method that simply allows the numbers sampled to be “plugged in,” and out comes some indication of the variability of the sample.

Statisticians have had to grapple with the problem of variability over many years and have found several measures for describing the characteristics of variables. Detailed discussion is beyond the scope of this book, but can be found in many statistical works, including those on business statistics. What they have come up with is a description of the *variability*, or *variance*, of a variable that captures the necessary variability information without being sensitive to column width or smoothing.

In many statistical texts, variability is very often described in terms of how far the individual instances of the sample are from the mean of the sample. It is, in fact, a sort of “average” distance of the instance values from the mean. It is this measure, or one derived from it, that will be used to measure variability. The measure is called the *standard deviation*. We need to look at it from a slightly different perspective than is usually found in statistics texts.

5.1.5 Variability and Deviation

Deviation is simply the name for what was described above as “a sort of average distance of instance values from the mean.” Given the same set of 80 numbers that were used before, the *mean*, often called the *arithmetic average*, or just average for short, is approximately 49.16. In order to find the distance of the instance values from the mean, it is only necessary to subtract the one from the other. To take the first five numbers as an example:

$$\begin{aligned}
49 - 49.16 &= -0.16\\
63 - 49.16 &= 13.84\\
44 - 49.16 &= -5.16\\
25 - 49.16 &= -24.16\\
16 - 49.16 &= -33.16
\end{aligned}$$

Unfortunately, the “–” signs make matters somewhat awkward. Since it is the mean that is being subtracted, the sum of all of the differences will add up to 0. That is what the mean is! Somehow it is necessary to make the “–” signs disappear, or at least to nullify their effect. For various reasons, in the days before computers, when calculations were all done by hand (perish the thought!), the easiest way for mathematicians to deal with the problem was not to simply ignore the “–” sign. Since “negative times negative is a positive,” as you may recall from school, squaring, or multiplying a number by itself, solves the problem. So finding the variance of just the first five numbers: The mean of only the first five numbers is

$(49 + 63 + 44 + 25 + 16)/5 = 39.4$

so squaring the instance value minus the mean:

$(49 – 39.4)^2 = 9.6^2 = 92.16$
$(63 – 39.4)^2 = 23.6^2 = 556.96$
$(44 – 39.4)^2 = 4.6^2 = 21.16$
$(25 – 39.4)^2 = –14.4^2 = 207.36$
$(16 – 39.4)^2 = –23.4^2 = 547.56$

and since the variance is the mean of these differences:

$(92.16 + 556.96 + 21.16 + 207.36 + 547.56)/5 = 285.04$

This number, 285.04, is the mean of the squares of the differences. It is therefore a variance of 285.04 square units. If these numbers represent some item of interest, say, percentage return on investments, it turns out to be hard to know exactly what a variance of 285.04 square percent actually means. Square percentage is not a very familiar or meaningful measure in general. In order to make the measure more meaningful in everyday terms, it is usual to take the square root, the opposite of squaring, which would give 16.88. For this example, this would now represent a much more meaningful variance of 16.88 percent.

The square root of the variance is called the *standard deviation*. The standard deviation is a very useful thing to know. There is a neat, mathematical notation for doing all of the things just illustrated:

Standard deviation = $\sqrt{(\Sigma(x - m)^2/(n - 1))}$

where

$\sqrt{}$	means to take the square root of everything under it
Σ	means to sum everything in the brackets following it
x	is the instance value
m	is the mean
n	is the number of instances

(For various technical reasons that we don't need to get into here, when the number is divided by *n*, it is known as the standard deviation of the population, and when divided by *n* – 1, as the standard deviation of the sample. For large numbers of instances, which will usually be dealt with in data mining, the difference is miniscule.)

There is another formula for finding the value of the standard deviation that can be found in any elementary work on statistics. It is the mathematical equivalent of the formula shown above, but gives a different perspective and reveals something else that is going on inside this formula—something that is very important a little later in the data preparation process:

$$s = \sqrt{((\Sigma x^2 - nm^2)/(n - 1))}$$

What appears in this formula is "Σx^2," which is the sum of the instance values squared. Notice also that "nm^2," which is the number of instances multiplied by the mean, squared. Since the mean is just the sum of the *x* values divided by the number of values (or Σx/n), the formula could be rewritten as

$$s = \sqrt{((\Sigma x^2 - (n(\Sigma x/n))^2)/(n - 1))}$$

But notice that $n(\Sigma x$/n) is the same as Σx, so the formula becomes

$$s = \sqrt{((\Sigma x^2 - (\Sigma x)^2)/(n - 1))}$$

(being careful to note that Σx^2 means to add all the values of *x* squared, whereas $(\Sigma x)^2$ means to take the sum of the unsquared *x* values and square the total).

This formula means that the standard deviation can be determined from three separate pieces of information:

1. The sum of x^2, that is, adding up all squares of the instance values

2. The sum of *x*, that is, adding up all of the instance values

3. The number of instances

The standard deviation can be regarded as exploring the relationship among the sum of the squares of the instance values, the sum of the instance values, and the number of instances. The important point here is that in a sample that contains a variety of different values, the exact ratio of the sum of the numbers to the sum of the squares of the numbers is very sensitive to the exact proportion of numbers of different sizes in the sample. This sensitivity is reflected in the variance as measured by the standard deviation.

Figure 5.5 shows distribution curves for three separate samples, each from a different population. The range for each sample is 0–100. The linear (or rectangular) distribution sample is a random sample drawn from a population in which each number 0–100 has an equal chance of appearing. This sample is evidently not large enough to capture this distribution well! The bimodal sample was drawn from a population with two “humps” that do show up in this limited sample. The normal sample was drawn from a population with a normal distribution—one that would resemble the “bell curve” if a large enough sample was taken. The mean and standard deviation for each of these samples is shown in Table 5.1.

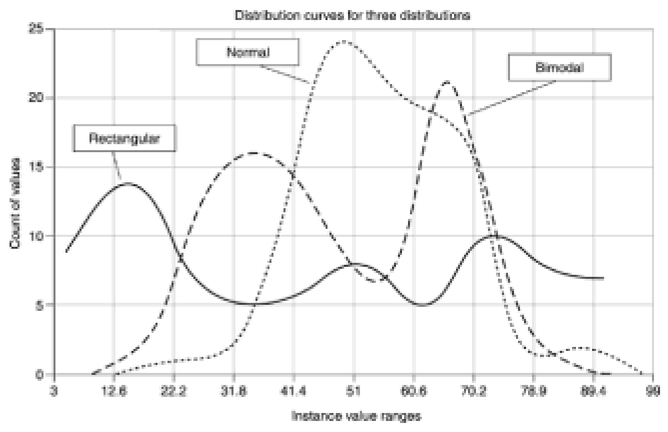


Figure 5.5 Distribution curves for samples drawn from three populations.

TABLE 5.1 Sample statistics for three distributions.

Sample distribution	**Mean**	**Standard deviation**
Linear	47.96	29.03
Bimodal	49.16	17.52
Normal	52.39	11.82

The standard deviation figures indicate that the linear distribution has the highest variance, which is not surprising as it would be expected to have the greatest average distance between the sample mean and the instance values. The normal distribution sample is the most bunched together around its sample mean and has the least standard deviation. The bimodal is more bunched than the linear, and less than the normal, and its standard deviation indicates this, as expected.

Standard deviation is a way to determine the variability of a sample that only needs to have the instance values of the sample. It results in a number that represents how the instance values are scattered about the average value of the sample.

5.2 Confidence

Now that we have an unambiguous way of measuring variability, actually capturing it requires enough instances of the variable so that the variability in the sample matches the variability in the population. Doing so captures all of the structure in the variable. However, it is only possible to be absolutely 100% certain that all of the variability in a variable has been captured if all of the population is included in the sample! But as we've already discussed, that is at best undesirable, and at worst impossible. Conundrum.

Since sampling the whole population may be impossible, and in any case cannot be achieved when it is required to split a collected data set into separate pieces, the miner needs an alternative. That alternative is to establish some acceptable degree of *confidence* that the variability of a variable is captured.

For instance, it is common for statisticians to use 95% as a satisfactory level of confidence. There is certainly nothing magical about that number. A 95% confidence means, for instance, that a judgment will be wrong 1 time in 20. That is because, since it is right 95 times in 100, it must be wrong 5 times in 100. And 5 times in 100 turns out to be 1 time in 20. The 95% confidence interval is widely used only because it is found to be generally useful in practice. "Useful in practice" is one of the most important metrics in both statistical analysis and data mining.

It is this concept of "level of confidence" that allows sampling of data sets to be made. If the miner decided to use only a 100% confidence level, it is clear that the only way that this can be done is to use the whole data set complete as a sample. A 100% sample is hardly a sample in the normal use of the word. However, there is a remarkable reduction in the amount of data needed if only a 99.99% confidence is selected, and more again for a 95% confidence.

A level of confidence in this context means that, for instance, it is 95% certain that the variability of a particular variable has been captured. Or, again, 1 time in 20 the full variability of the variable would not have been captured at the 95% confidence level, but some lesser level of variability instead. The exact level of confidence may not be important. Capturing enough of the variability is vital.

5.3 Variability of Numeric Variables

Variability of numeric variables is measured differently from the variability of nonnumeric variables. When writing computer code, or describing algorithms, it is easy to abbreviate numeric and nonnumeric to the point of confusion—"Num" and "Non." To make the difference easier to describe, it is preferable to use distinctive abbreviations. This distinction is easy when using "Alpha" for nominals or categoricals, which are measured in nonnumeric scales, and "Numeric" for variables measured using numeric scales. Where convenient to avoid confusion, that nomenclature is used here.

Variability of numeric variables has been well described in statistical literature, and the previous sections discussing variability and the standard deviation provide a conceptual overview.

Confidence in variability capture increases with sample size. Recall that as a sample size gets larger, so the sample distribution curve converges with the population distribution curve. They may never actually be identical until the sample includes the whole population, but the sample size can, in principle, be increased until the two curves become as similar as desired. If we knew the shape of the population distribution curve, it would be easy to compare the sample distribution curve to it to tell how well the sample had captured the variability. Unfortunately, that is almost always impossible. However, it is possible to measure the rate of change of a sample distribution curve as instance values are added to the sample. When it changes very little with each addition, we can be confident that it is closer to the final shape than when it changes faster. But how confident? How can this rate of change be turned into a measure of confidence that variability has been captured?

5.3.1 Variability and Sampling

But wait! There is a critical assumption here. The assumption is that a larger sample is in fact more representative of the population as a whole than a smaller one. This is not necessarily the case. In the forestry example, if only the oldest trees were chosen, or only those in North America, for instance, taking a larger sample would not be representative. There are several ways to assure that the sample is representative, but the only one that can be assured not to introduce some bias is *random sampling.* A random sample requires that any instance of the population is just as likely to be a member of the sample as any other member of the population. With this assumption in place, larger samples will, on average, better represent the variability of the population.

It is important to note here that there are various biases that can be inadvertently introduced into a sample drawn from a population against which random sampling provides no protection whatsoever. Various aspects of sampling bias are discussed in Chapters 4 and 10. However, what a data miner starts with as a source data set is almost always a sample and not the population. When preparing variables, we cannot be sure that the original data is bias free. Fortunately, at this stage, there is no need to be. (By

Chapter 10 this is a major concern, but not here.) What is of concern is that the sample taken to evaluate variable variability is representative of the original data sample. Random sampling does that. If the original data set represents a biased sample, that is evaluated partly in the data assay (Chapter 4), again when the data set itself is prepared (Chapter 10), and again during the data survey (Chapter 11). All that is of concern here is that, on a variable-by-variable basis, the variability present in the source data set is, to some selected level of confidence, present in the sample extracted for preparation.

5.3.2 Variability and Convergence

Differently sized, randomly selected samples from the same population will have different variability measures. As a larger and larger random sample is taken, the variability of the sample tends to fluctuate less and less between the smaller and larger samples. This reduction in the amount of fluctuation between successive samples as sample size increases makes the number measuring variability converge toward a particular value.

It is this property of convergence that allows the miner to determine a degree of confidence about the level of variability of a particular variable. As the sample size increases, the average amount of variability difference for each additional instance becomes less and less. Eventually the miner can know, with any arbitrary degree of certainty, that more instances of data will not change the variability by more than a particular amount.

Figure 5.6 shows what happens to the standard deviation, measured up the side of the graph, as the number of instances in the sample increases, which is measured along the bottom of the graph. The numbers used to create this graph are from a data set provided on the CD-ROM called CREDIT. This data set contains a variable DAS that is used through the rest of the chapter to explore variability capture.

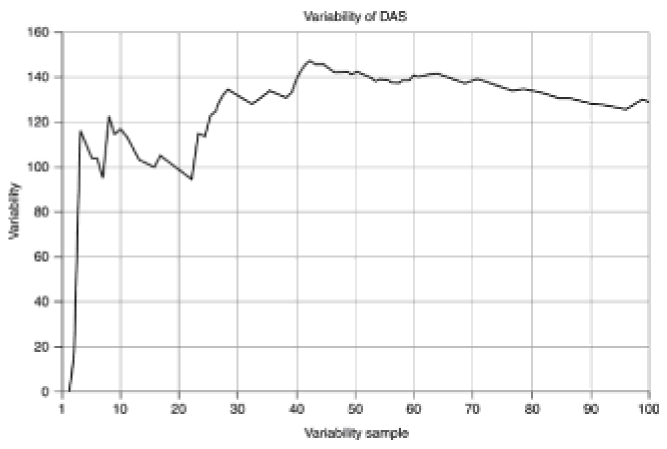


Figure 5.6 Measuring variability DAS in the CREDIT data set. Each sample contains one more instance than the previous sample. As the sample size increases, the variability seems to approach, or converge, toward about 130.

Figure 5.6 shows incremental samples, starting with a sample size of 0, and increasing the sample size by one each time. The graph shows the variability in the first 100 samples. Simply by looking at the graph, intuition suggests that the variability will end up somewhere about 130, no matter how many more instances are considered. Another way of saying this is that it has converged at about 130. It may be that intuition suggests this to be the case. The problem now is to quantify and justify exactly how confident it is possible to be. There are two things about which to express a level of confidence—first, to specify exactly the expected limits of variability, and second, to specify how confident is it possible to be that the variability actually will stay within the limits.

The essence of capturing variability is to continue to add samples until both of those confidence measures can be made at the required level—whatever that level may be. However, before considering the problem of justifying and quantifying confidence, the next step is to examine capturing variability in alpha-type variables.

5.4 Variability and Confidence in Alpha Variables

So far, much of this discussion has described variability as measured in numeric variables. Data mining often involves dealing with variables measured in nonnumeric ways. Sometimes the symbolic representation of the variable may be numeric, but the variable still is being measured nominally—such as SIC and ZIP codes.

Measuring variability in these alpha-type variables is every bit as important as in numerical variables. (Recall this is not a new variable type, just a clearer name for qualitative variables—nominals and categoricals—to save confusion.)

A measure of variability in alpha variables needs to work similarly to that for numeric variables. That is to say, increases in sample size must lead to convergence of variability. This convergence is similar in nature to that of numerical variables. So using such a method, together with standard deviation for numeric variables, gives measures of variability that can be used to sample both alpha and numeric variables. How does such a method work?

Clearly there are some alpha variables that have an almost infinite number of categories—people’s names, for instance. Each name is an alpha variable (a nominal in the terminology used in Chapter 2), and there are a great many people each with different names!

For the sake of simplicity of explanation, assume that only a limited number of alpha labels exist in a variable scale. Then the explanation will be expanded to cover alpha variables with very high numbers of distinct values.

In a particular population of alpha variables there will be a specific number of instances of each of the values. It is possible in principle to count the number of instances of each value of the variable and determine what percentage of the time each value occurs. This is exactly similar to counting how often each numeric instance value occurred when creating the histogram in Figure 5.1. Thus if, in some particular sample, “A” occurred 124 times, “B” 62 times, and “C” 99 times, then the ratio of occurrence, one to the others, is as shown in Table 5.2.

TABLE 5.2 Sample value frequency counts.

Sample distribution	**Mean**	**Standard deviation**
A	124	43.51
B	62	21.75
C	99	34.74
Total	285	100.00

If the population is sampled randomly, this proportion will not be immediately apparent. However, as the sample size increases, the relative proportion will become more and more nearly what is present in the population; that is, it converges to match that of the population. This is altogether similar to the way that the numeric variable variability converges. The main difference here is that since the values are alpha, not numeric, standard deviation can’t be calculated.

Instead of determining variability using standard deviation, which measures the way numeric values are distributed about the mean, alpha variability measures the rate of change of the relative proportion of the values discovered. This rate of change is analogous to the rate of change in variability for numerics. Establishing a selected degree of confidence that the relative proportion of alpha values will not change, within certain limits, is analogous to capturing variability for a numeric variable.

5.4.1 Ordering and Rate of Discovery

One solution to capturing the variability of alpha variables might be to assign numbers to each alpha and use those arbitrarily assigned numbers in the usual standard deviation formula. There are several problems with this approach. For one thing, it assumes that each alpha value is equidistant from one another. For another, it arbitrarily assigns an ordering to the alphas, which may or may not be significant in the variability calculation, but certainly doesn't exist in the real world for alphas other than ordinals. There are other problems so far as variability capture goes also, but the main one for sampling is that it gives no clue whether all of the unique alpha values have been seen, nor what chance there is of finding a new one if sampling continues. What is needed is some method that avoids these particular problems.

Numeric variables all have a fixed ordering. They also have fixed distances between values. (The number "1" is a fixed distance from "10"—9 units.) These fixed relationships allow a determination of the range of values in any numeric distribution (described further in Chapter 7). So for numeric variables, it is a fairly easy matter to determine the chance that new values will turn up in further sampling that are outside of the range so far sampled.

Alphas have no such fixed relationship to one another, nor is there any order for the alpha values (at this stage). So what is the assurance that the variability of an alpha variable has been captured, unless we know how likely it is that some so far unencountered value will turn up in further sampling? And therein lies the answer—measuring the rate of discovery of new alpha values.

As the sample size increases, so the rate of discovery (ROD) of new values falls. At first, when the sample size is low, new values are often discovered. As the sampling goes on, the rate of discovery falls, converging toward 0. In any fixed population of alphas, no matter how large, the more values seen, the less new ones there are to see. The chance of seeing a new value is exactly proportional to the number of unencountered values in the population.

For some alphas, such as binary variables, ROD falls quickly toward 0, and it is soon easy to be confident (to any needed level of confidence) that new values are very unlikely. With other alphas—such as, say, a comprehensive list of cities in the U.S.—the probability would fall more slowly. However, in sampling alphas, because ROD changes, the miner can estimate to any required degree of confidence the chance that new alpha values will turn up. This in turn allows an estimate not only of the variability of an alpha, but of the comprehensiveness of the sample in terms of discovering all the alpha labels.

5.5 Measuring Confidence

Measuring confidence is a critical part of sampling data. The actual level of confidence selected is quite arbitrary. It is selected by the miner or domain expert to represent some level of confidence in the results that is appropriate. But whatever level is chosen, it is so important in sampling that it demands closer inspection as to what it means in practice, and why it has to be selected arbitrarily.

5.5.1 Modeling and Confidence with the Whole Population

If the whole population of instances were available, predictive modeling would be quite unnecessary. So would sampling. If the population really is available, all that needs to be done to “predict” the value of some variable, given the values of others, is to look up the appropriate case in the population. If the population is truly present, it is possible to find an instance of measurements that represents the exact instance being predicted—not just one similar or close to it.

Inferential modeling would still be of use to discover what was in the data. It might provide a useful model of a very large data set and give useful insights into related structures. No training and test sets would be needed, however, because, since the population is completely represented, it would not be possible to *overtrain*. Overtraining occurs when the model learns idiosyncrasies present in the training set but not in the whole population. Given that the whole population is present for training, anything that is learned is, by definition, present in the population. (An example of this is shown in Chapter 11.)

With the whole population present, sampling becomes a much easier task. If the population were too large to model, a sample would be useful for training. A sample of some particular proportion of the population, taken at random, has statistically well known properties. If it is known that some event happens in, say, a 10% random sample with a particular frequency, it is quite easy to determine what level of confidence this implies about the frequency of the event in the population. When the population is not available, and even the size of the population is quite unknown, no such estimates can be made. This is almost always the case in modeling.

Because the population is not available, it is impossible to give any level of confidence in any result, based on the data itself. All levels of confidence are based on assumptions about the data and about the population. All kinds of assumptions are made about the randomness of the sample and the nature of the data. It is then possible to say that *if* these assumptions hold true, then certain results follow. The only way to test the assumptions, however, is to look at the population, which is the very thing that can’t be done!

5.5.2 Testing for Confidence

There is another way to justify particular levels of confidence in results. It relies on the quantitative discriminatory power of tests. If, for instance, book reviewers can consistently and accurately predict a top 10 best-selling book 10% of the time, clearly they are wrong 90% of the time. If a particular reviewer stated that a particular book just reviewed was certain to be a best-seller, you would be justified in being skeptical of the claim. In fact, you would be quite justified in being 10% sure (or confident) that it would be a success, and 90% confident in its failure. However, if at a convention of book reviewers, every one of hundreds or thousands of reviewers each separately stated that the book was sure to be a best-seller, even though each reviewer had only a 10% chance of success, you would become more and more convinced of the book's chance of success.

Each reviewer performs an independent reading, or test, of the book. It is this independence of tests that allows an accumulation of confidence. The question is, how much additional confidence is justified if two independent tests are made, each with a 10% accuracy of being correct in their result, and both agree? In other words, suppose that after the first reviewer assured you of the book's success, a second one did the same. How much more confident, if at all, are you justified in being as a result of the second opinion? What happens if there are third and fourth confirming opinions? How much additional confidence are you justified in feeling?

At the beginning you are 100% skeptical. The first reviewer's judgment persuades you to an opinion of 10% in favor, 90% against the proposition for top 10 success. If the first reviewer justified a 10/90% split, surely the second does too, but how does this change the level of confidence you are justified in feeling?

Table 5.3 shows that after the first reviewer's assessment, you assigned 10% confidence to success and 90% to skepticism. The second opinion (test) should also justify the assignment of an additional 10%. However, you are now only 90% skeptical, so it is 10% of that 90% that needs to be transferred, which amounts to an additional 9% confidence. Two independent opinions justify a 19% confidence that the book will be a best-seller. Similar reasoning applies to opinions 3, 4, 5, and 6. More and more positive opinions further reinforce your justified confidence of success. With an indefinite amount of opinions (tests) available, you can continue to get opinions until any particular level of confidence in success is justified.

TABLE 5.3 Reviewer assurance charges confidence level.

Reviewer number	**Start level**	**Transfer amount (start level x10%)**	**Confidence of success**	**Your remaining skeptical balance**
1	100%	10	10	90
2	90%	9	9	81

3	81%	8.1	27.1	72.9
4	72.9%	7.29	34.39	65.61
5	65.61%	6.561	40.951	59.049
6	59.049%	5.9049%	46.8559	53.1441

Of course, a negative opinion would increase your skepticism and decrease your confidence in success. Unfortunately, without more information it is impossible to say by how much you are justified in revising your opinion. Why?

Suppose each reviewer reads all available books and predicts the fate of all of them. One month 100 books are available, 10 are (by definition) on the top 10 list. The reviewer predicts 10 as best-sellers and 90 as non-best-sellers. Being consistently 10% accurate, one of those predicted to be on the best-seller list was on it, 9 were not. Table 5.4 shows the reviewer's hit rate this month.

TABLE 5.4 Results of the book reviewer's predictions for month 1.

Month 1	Best-seller	Non-best seller
Predicted bestseller	1	9
Predicted non-best-seller	9	81

Since one of the 10 best-sellers was predicted correctly, we see a 10% rate of accuracy. There were also 90 predicted to be non-best-sellers, of which 81 were predicted correctly as non-best-sellers. (81 out of 90 = 81/90 = 90% incorrectly predicted.)

In month 2 there were 200 books published. The reviewer read them all and made 10 best-seller predictions. Once again, a 10% correct prediction was achieved, as Table 5.5 shows.

TABLE 5.5 Results of the book reviewer's predictions for month 2.

Month 1	**Best-seller**	**Non-best seller**
Predicted bestseller	1	9
Predicted non-best-seller	9	181

Once again, there are 10 best-sellers and one was correctly predicted for a correct pick rate of 10%. However, there were 190 books predicted to be non-best-sellers this month, of which 181 were correctly predicted because they weren't best-sellers. However, 181 out of 190 is a correct prediction rate for non-best-sellers of 95.26%!

What is going on here? The problem is that predicting best-sellers and predicting non-best-sellers are *not* two sides of the same problem, although they look like they might be. The chances of being right about best-sellers are not the opposite of the chances of being right about non-best-sellers. This is because of the old bugaboo of knowledge of the size of the population. What changed here is the size of the population from 100 to 200. The number of best-sellers is always 10 because they are defined as being the 10 best-selling books. The number of non-best-sellers depends entirely on how many books are published that month.

However (and this is a very important point), deciding how much confidence can be justified after a given number of tests depends *only on the success ratio of the tests*. This means that if the success/fail ratio of the test is known, or assumed, knowledge of the size of the population is not needed in order to establish a level of confidence. With this knowledge it is possible to construct a test that doesn't depend on the size of the population, but only on the consecutive number of confirmatory tests.

The confidence generated in the example is based on predicting best-sellers. The number of best-sellers is a purely arbitrary number. It was just chosen to suit the needs of the selector. After all, it could have been the top 12, or 17, or any other number. The term "best-seller" was defined to suit someone's convenience. It is very likely that the success of reviewers in picking best-sellers would change if the definition of what constituted a best-seller changed. The point here is that if the chosen assumptions meet the needs of whoever selected them, then a rational assessment of confidence can be made based on those assumptions.

5.5.3 Confidence Tests and Variability

The consequence for determining variability of a variable is that the modeler must make assumptions that meet the modeler's needs. Choosing a 95% level of confidence implies saying, among other things, "If this test is wrong 95% of the time, how many times must independent tests confirm its correctness before the cumulative judgment can be accepted as correct at least 95% of the time?"

In practical terms (using the 95% level of confidence for this discussion), this implies several consequences. Key, based on the level of confidence, is that a single test for convergence of variability is incorrect 95% of the time and correct 5% of the time. From that it is possible to rationally accumulate confidence in a continuing series of positive results. (Positive results indicate variability convergence.) After some unbroken series of positive results, a level of confidence is accumulated that exceeds 95%. When that happens you can be sure that accepting the convergence as complete will only be a mistake 1 time in 20, or less.

At the end, the result is a very simple formula that is transformed a little and used in the demonstration software to know when enough is enough. That is,

$$s = e^t$$

where

s = Justified level of skepticism

e = Error rate

t = Number of positive tests

Results of this formula, using the 90% confidence level from the best-seller example, are given in Table 5.6.

TABLE 5.6 Results of the book reviewer's predictions for month 2.

Skepticism	**Error rate**	**Number of tests**
0.9	0.9	1
0.81	0.9	2

0.729	0.9	3
0.6561	0.9	4
0.59049	0.9	5
0.531441	0.9	6

This is the same series of numbers shown in the "Your remaining skeptical balance" column in Table 5.3 except that these are expressed as decimals rather than percentages.

Of course, this diminishing level of skepticism indicates the confidence that you are *wrong*. The confidence that you are right is what is left after subtracting the confidence that you are wrong! The confidence level in being right is, therefore,

$c = 1 - e^t$

where

c = Confidence

e = Error rate

t = Number of positive tests

The Supplemental Material section at the end of this chapter briefly shows the transformation from this statement into one that allows the number of consecutive true tests to be directly found from the error rate. It is this version of the formula that is used in the demonstration software. However, for those who need only to understand the concepts and issues, that section may be safely skipped.

5.6 Confidence in Capturing Variability

Capturing the variability of a variable means, in practice, determining to a selected level of confidence that the measured variability of a sample is similar to that of the population, within specified limits. The measure of sample variability closeness to population variability is measured by convergence in increasingly larger samples. In other words, converged variability means that the amount of variability remains within particular limits for enough independent tests to be convincing that the convergence is real. When the variability is converged, we are justified in accepting, to a certain level of confidence, that the variability is captured. This leads to two questions. First, exactly what is measured to determine convergence, and second, what are the “particular limits” and how they are discovered?

On the accompanying CD-ROM there is a data set CREDIT. This includes a sample of real-world credit information. One of the fields in that data set is “DAS,” which is a particular credit score rating. All of the data used in this example is available on the CD-ROM, but since the sample is built by random selection, it is very unlikely that the results shown here will be duplicated exactly in any subsequent sample. The chance of a random sampling procedure encountering exactly the same sequence of instance values is very, very low. (If it weren’t, it would be of little value as a “random” sequence!) However, the results do remain consistent in that they converge to the same variability level, for a given degree of confidence, but the precise path to get there may vary a little.

5.6.1 A Brief Introduction to the Normal Distribution

Capturing variability relies on assuming normality in the distribution of the test results, and using the known statistical properties of the normal distribution. The assumption of normality of the distribution of the test results is particularly important in estimating the probability that variability has actually converged. A brief and nontechnical examination of some facets of the normal distribution is needed before looking at variability capture.

The normal distribution is well studied, and its properties are well known. Detailed discussion of this distribution, and justification for some of the assertions made here, can be found in almost any statistical text, including those on business statistics. Specifically, the distribution of values within the range of a normally distributed variable form a very specific pattern. When variables’ values are distributed in this way, the standard deviation can be used to discover exactly how likely it is that any particular instance value will be found. To put it another way, given a normally distributed sample of a particular number of instances, it is possible to say how many instances are likely to fall between any two values.

As an example, about 68% of the instance values of a normally distributed variable fall inside the boundary values set at the mean-plus-1 standard deviation and the mean-minus-1 standard deviation. This is normally expressed as $m \pm s$, where m is the mean and s the standard deviation. It is also known, for instance, that about 95.5% of the sample’s instance values will lie within $m \pm 2s$, and 99.7% within $m \pm 3s$.

What this means is that if the distance and direction from the mean is known in standard deviation units for any two values, it is possible to determine precisely the probability of discovering instance values in that range. For instance, using tables found in any elementary statistics text, it is easy to discover that for a value of the mean-minus-1 standard deviation, approximately 0.1587 (i.e., about 16%) of the instances lie in the direction away from the mean, and therefore 0.8413 (about 84%) lie in the other direction.

The normal curve shown in Figure 5.7 plots values in the sample along the horizontal axis (labeled Standard deviation) and the probability of finding a value on the vertical axis. In a normally distributed sample of numbers, any value has some probability of occurring, but with a vanishingly small probability as the distance of the values moves far from the mean. The curve is centered on the sample mean and is usually measured in standard deviation units from the mean. The total area under the curve corresponds to a probability of 100% for finding a value in the distribution. The chance of finding a specific value corresponds to the area under the curve for that value. It is easier to imagine finding a value in some interval between two values.

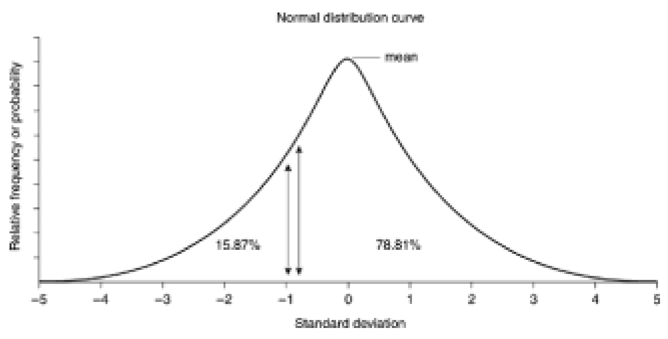


Figure 5.7 Normal distribution curve with values plotted for standard deviation (x-axis) and probability of finding a value (y-axis).

This figure shows the normal curve with the interval between –1 and –0.8 standard deviations. It can be found, by looking in standard deviation tables, that approximately 15.87% of the instance values lie to the left of the mean-minus-1 standard deviation line, 78.81% lie to the right of the –0.8 standard deviation line, which leaves 5.32% of the distribution between the two (100% – 78.81% – 15.78% = 5.32%). So, for instance, if it were known that some feature fell between these two limits consistently, then the feature is "pinned down" with a 94.68% confidence (100% – 5.32% = 94.68%).

5.6.2 Normally Distributed Probabilities

Measuring such probabilities using normally distributed phenomena is only important, of course, if the phenomena are indeed normally distributed.

Looking back at Figure 5.6 will very clearly show that the variability is not normally distributed, nor even is the convergence. Fortunately, the changes in convergence, that is, the size of the change in variance from one increment to the next increment, can easily be adjusted to resemble a normal distribution. This statement can be theoretically supported, but it is easy to intuitively see that this is reasonable.

Early in the convergence cycle, the changes tend to be large compared to later changes.

This is true no matter how long the convergence cycle continues. This means that the proportion of relatively small changes always predominates. In turn, this leads to the conclusion that the more instances that are considered, the more the later changes in variance cluster about the mean. Relatively large changes in variance, both positive and negative, are much less likely than small changes. And that is exactly what the normal distribution describes.

To be sure, this is only a descriptive insight, not proof. Unfortunately, proof of this takes us beyond the scope of this book. The Further Reading section at the end of this book has pointers to where to look for further exploration of this and many other areas.

It must be noted that the variance distribution is not actually normal since convergence can continue arbitrarily long, which can make the number of small changes in variability far outweigh the large changes. Figure 5.8 shows part of the distribution for the first 100 samples of DAS variance. This distribution is hardly normal! However, although outside the scope of this conceptual introduction, adjustment for the reduction in change of size of variance with the sample is fairly straightforward. Figure 5.9 shows the effect of making the adjustment, which approximately normalizes the distribution of changes in variance—sufficient to make the assumptions for testing for convergence workably valid. Figure 5.9 shows the distribution for 1000 variability samples.

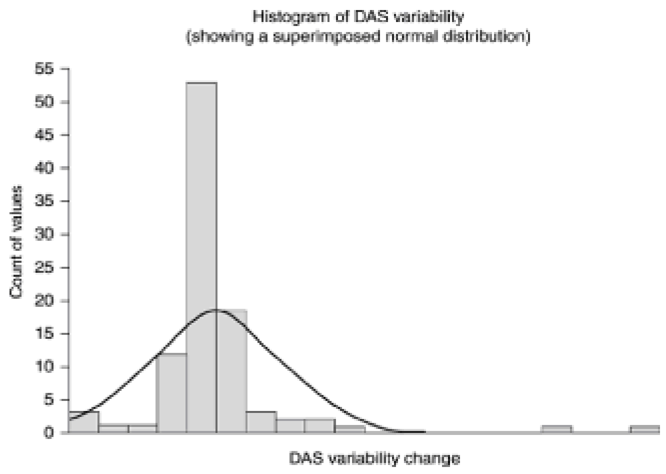


Figure 5.8 The actual distribution of the changes in variability of DAS for the first 100 samples shown in Figure 5.6.

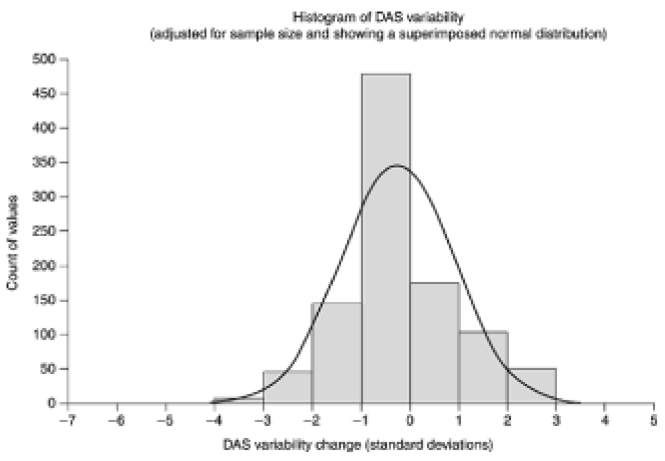


Figure 5.9 The variability of DAS for a 1000-instance sample when adjusted for sample size.

5.6.3 Capturing Normally Distributed Probabilities: An Example

After much preamble, we are at last ready to actually capture DAS variability. Recall that Figure 5.6 showed how the variance of DAS changes as additional instances are sampled. The variance rises very swiftly in the early instances, but settles down (converges) to remain, after 100 samples, somewhere about 130. Figure 5.10 shows the process of capturing variability for DAS. At first glance there is a lot on this graph! In order to get everything onto the same graph together, the figure shows the values "normalized" to fit into a range between 0 and 1. This is only for the purposes of displaying everything on the graph.

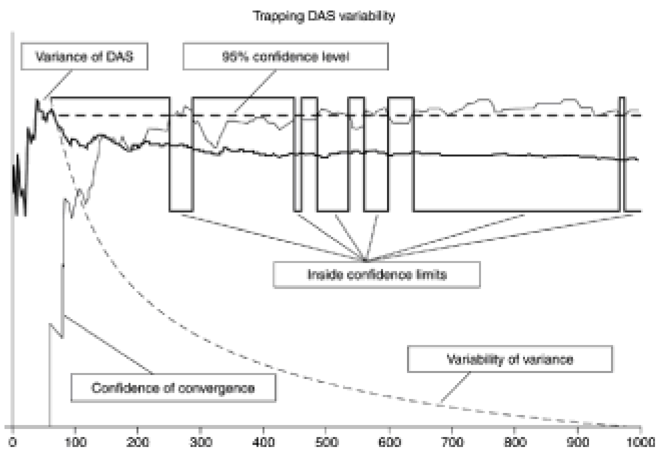


Figure 5.10 Various features shown on a common scale so that their relationships are more easily seen.

This example uses a 95% confidence level, which requires that the variability be inside 0.05 (or 5%) of its range for 59 consecutive tests. In this example, the sampling is continued long after variability was captured to see if the confidence was justified. A total of 1000 instance-value samples are used.

As the variance settles down, the confidence level that the variability has been captured rises closer to “1,” which would indicate 100% confidence. When the confidence of capture reaches 0.95, in actual data preparation, the needed confidence level is reached and sampling of this variable can stop. It means that at that point there is a 95% confidence that at least 95% of the variability has been captured.

Because the example does not stop there, the variability pops out of limits from time to time. Does this mean that the measurement of variability failed? When the variability first reached the 0.95 confidence level, the variability was 127.04. After all 1000 samples were completed, the variability was at 121.18. The second variance figure is 4.6% distant from the first time the required confidence level was reached. The variance did indeed stay within 5% for the measured period. Perhaps it might have moved further distant if more instances had been sampled, or perhaps it might have moved closer. The measurement was made at the 95% confidence interval for a 95% variability capture. So far as was measured, this confidence level remains justified.

5.6.4 Capturing Confidence, Capturing Variance

This is a complex subject, and it is easy to confuse what actually has been captured here. In the example the 95% level was used. The difficulty is in distinguishing between capturing 95% of the DAS variability, and having a 95% confidence that the variability is captured. Shouldn’t the 95% confidence interval of capturing 95% of the variability really indicate a 0.95 x 0.95 = 0.9025 confidence?

The problem here is the difficulty of comparing apples and oranges. Capturing 95% of the variability is not the same as having 95% confidence that it is captured. An example might help to clarify.

If you have an interest-bearing bank account, you have some degree of assurance, based on past history, that the interest rate will not vary more than a certain amount over a given time. Let’s assume that you think it will vary less than 5% from where it is now over the next six months. You could be said to be quite certain that you have locked in at least 95% of the current interest rate. But how certain are you? Perhaps, for the sake of this discussion, you can justify a 95% confidence level in your opinion.

So, you are 95% confident of capturing 95% of the current interest rate. By some strange coincidence, those are the numbers that we had in capturing the variability! Ask this: because 0.95 x 0.95 = 0.9025, does this mean that you are really 90.25% confident? Does it mean the interest rate is only 90.25% of what you thought it was? No. It means only that you are 95% confident of getting 95% of the present interest rate.

Remember that the 95% intervals were arbitrarily chosen. There is no inherent or intrinsic connection between them, nor any necessity that they be at the same level. For convenience in writing the demonstration software accompanying the book, they are taken to be the same. You could make changes, however, to choose other levels, such as being 99% certain that 80% of the variability has been captured. These are arbitrary choices of the user.

5.7 Problems and Shortcomings of Taking Samples Using Variability

The discussion so far has established the need for sampling, for using measurement of variability as a means to decide how much data is enough, and the use of confidence measures to determine what constitutes enough. Taken together, this provides a firm foundation to begin to determine how much data is enough. Although the basic method has now been established, there are a number of practical issues that need to be addressed before attempting to implement this methodology.

5.7.1 Missing Values

Missing or empty values present a problem. What value, if any, should be in the place of the missing value cannot yet be determined.

The practical answer for determining variability is that missing values are simply ignored as if they did not exist. Simply put, a missing value does not count as an instance of data, and the variability calculation is made using only those instances that have a measured value.

This implies that, for numerical variables particularly, the difference between a missing value and the value 0 must be distinguished. In some database programs and data warehouses, it is possible to distinguish variables that have not been assigned values. The demonstration program works with data in character-only format (.txt files) and regards values of all space as missing.

The second problem with missing values is deciding at what threshold of density the variable is not worth bothering with. As a practical choice, the demonstration program uses the confidence level here as a density measure. A 95% confidence level will generate a minimum density requirement of 5% (100 – 95). This is very low, and in practice such low-density variables probably contribute little information of value. It's probably better to remove them. The program does, however, measure the density of all variables. The cutoff occurs when an appropriate confidence level can be established that the variable is below the minimum density. For the 95% confidence level, this translates into being 95% certain that the variable is less than 5% populated.

5.7.2 Constants (Variables with Only One Value)

A problem similar in some respects to missing values is that of variables that are in fact constants. That is to say, they contain only a single value. These should be removed before sampling begins. However, they are easy to overlook. Perhaps the sample is about people who are now divorced. From a broader population it is easy to extract all those who are presently divorced. However, if there is a field answering the question “Was this person ever married?” or “Is the tax return filed jointly?” obviously the answer to the first question has to be “Yes.” It’s hard to get divorced if you’ve never married. Equally, divorced people do not file joint tax returns.

For whatever reason, variables with only one value do creep unwittingly into data sets for preparation. The demonstration program will flag them as such when the appropriate level of confidence has been reached that there is no variability in the variable.

5.7.3 Problems with Sampling

Sampling inherently has limitations. The whole idea of sampling is that the variability is captured without inspecting all of the data. The sample specifically does not examine all of the values present in the data set—that is the whole point of sampling.

A problem arises with alpha variables. The demonstration software does achieve a satisfactory representative sampling of the categorical values. However, not all the distinct values are necessarily captured. The PIE only knows how to translate those values that it has encountered in the data. (How to determine what particular value a given categorical should be assigned is explained in Chapter 6.) There is no way to tell how to translate values for the alpha values that exist in the data but were not encountered in the sample.

This is not a problem with alpha variables having a small and restricted number of values they can assume. With a restricted number of possible discrete values, sampling will find them all. The exact number sampled depends on the selected confidence level. Many real-world data sets contain categorical fields demonstrating the limitations of sampling high discrete-count categorical variables. (Try the data set SHOE on the CD-ROM.)

In general, names and addresses are pretty hopeless. There are simply too many of them. If ZIP codes are used and turn out to be too numerous, it is often helpful to try limiting the numbers by using just the three-digit ZIP. SIC codes have similar problems.

The demonstration code does not have the ability to be forced to comprehensively sample alpha variables. Such a modification would be easy to make, but there are drawbacks. The higher sampling rate can be forced by placing higher levels of confidence on selected variables. If it is known that there are high rates of categorical variable incidence, and that the sample data actually contains a complete and representative distribution of them, this will force the data preparation program to sample most or all of any number of distinct values. This feature should be used cautiously as very high confidence on high distinct-value count variables may require enormous amounts of data.

5.7.4 Monotonic Variable Detection

Monotonic variables are those that increase continuously, usually with time. Indeed, time increment variables are often monotonic if both time and date are included. Julian dates, which is a system of measurement using the number of days and fractions of days elapsed from a specified starting point (rather like *Star Trek's* star dates) are a perfect example of monotonic variables.

There are many other examples, such as serial numbers, order numbers, invoice numbers, membership numbers, account numbers, ISBN numbers, and a host of others. What they all have in common is that they increase without bound. There are many ways of dealing with these and either encoding or recoding them. This is discussed in Chapter 9. Suitably prepared, they are no longer monotonic. Here the focus is on how best to deal with the issue if monotonic variables accidentally slip into the data set.

The problem for variability capture is that only those values present in the sample available to the miner can be accessed. In any limited number of instances there will be some maximum and some minimum for each variable, including the monotonic variables. The full range will be sampled with any required degree of accuracy. The problem is that as soon as actual data for modeling is used from some other source, the monotonic variable will very likely take on values outside the range sampled. Even if the range of new variables is inside the range sampled, the distribution will likely be totally different than that in the original sample. Any modeled inferences or relationships made that rely on this data will be invalid.

This is a very tricky problem to detect. It is in the nature of monotonic variables to have a trend. That is, there is a natural ordering of one following another in sequence. However, one founding pillar of sampling is *random* sampling. Random sampling destroys any order. Even if random sampling were to be selectively abandoned, it does no good, for the values of any variable can be ordered if so desired. Such an ordering, however, is likely to be totally artificial and meaningless. There is simply no general way, from internal evidence inside any data set, to distinguish between a natural order and an artificially imposed one.

Two methods are used to provide an indication of possible monotonicity in variables. It is the nature of random sampling that neither each alone, nor both together, are perfect. They do, however, in practice provide warning and guidance that there may be something odd about a particular variable. There is a colloquial expression, to be “saved by the bell”—and these are two bells that may save much trouble: interstitial linearity and rate of discovery.

5.7.5 Interstitial Linearity

Interstitial linearity is the first method that helps signal the possibility of monotonicity. Interstices are internal spaces. Interstitial linearity measures the consistency of spaces between values of a variable. When the spaces are similar in size, interstitial linearity is high.

Monotonic variables are frequently generated as series of numbers increasing by some constant amount. Time ticks, for instance, may increase by some set amount of seconds, minutes, hours, or days. Serial numbers frequently increment by one at each step. This uniform increment, or some residual trace of it, is often found in the sample presented for modeling. Random sampling of such a series produces a subseries of numbers that also tend, on average, to have uniform interstitial spacing. This creates a distribution of numbers that tend to have a uniform density in all parts of their range. This type of distribution is known as a *linear distribution* or a *rectangular distribution*. Measuring how closely the distribution of variables approximates a linear distribution, then, can lead to an indication of monotonicity.

The demonstration software checks each variable for interstitial linearity. There are many perfectly valid variables that show high degrees of interstitial linearity. The measure varies between 0 and 1, with 1 being a perfectly linear distribution. If any variable has an interstitial linearity greater than 0.98, it is usually worth particular scrutiny.

5.7.6 Rate of Discovery

It is in the nature of monotonic variables that all of the instance values are unique. During sampling of such variables, a new instance value is discovered with every instance. (In fact, errors in data entry, or other problems, may make this not quite true, but the discovery rate will be very close to 100%.) A discovery rate of close to 100% can be indicative of monotonicity.

There are, of course, many variables that are not monotonic but have unique instance values, at least in the sample available. Men’s height measured to the nearest millimeter, for instance, would not be monotonic, but would likely have unique instance values for quite a large number of instances. However, such a variable would almost certainly not show interstitial linearity. Taken together, these measures can be useful in practice in discovering problems early.

5.8 Confidence and Instance Count

This chapter discussed using confidence as a means of estimating how many instance values are needed to capture variability. However, it is quite easy to turn this measure around, as it were, and give a confidence measure for some preselected number of instance values. It is quite possible to select a confidence level that the data simply cannot support. For instance, selecting a 99.5% confidence level may very well need more instances than the miner has available. In this case, the demonstration code estimates what level of confidence is justified for the instances available. In a small data set, selecting a 99.99% confidence level forces the sampling method to take all of the data, since such a high confidence of capture can almost never be justified. The actual confidence level justified is in the Variable Status report (mentioned in Chapter 4).

5.9 Summary

This chapter has looked at how to decide how much data the miner needs to make sure that variables have their variability represented. Variability must be captured as it represents the information content of a variable, which it is critically important to preserve. We are almost never completely assured that the variability has been captured, but it is possible to justify a degree of confidence. Data sets can be sampled because confidence levels of less than 100% work very well in real-world problems—which is fortunate since perfect confidence is almost always impossible to have! (Even if unlimited amounts of data are available, the world is still an uncertain place. See Chapter 2.) We can either select enough data to establish the needed level of confidence, or determine how much confidence is justified in a limited data set on hand. Selecting the appropriate level of confidence requires problem and domain knowledge, and cannot be automatically determined. Confidence decisions must be made by the problem owner, problem holder, domain expert, and miner.

The next actions need to expose information to modeling tools and fix various problems in the variables, and in the data set as a whole. The next chapter begins this journey.

Supplemental Material

Confidence Samples

In this section, we will use

c = Confidence

e = Error rate

n = Number of tests

s = Skepticism

(Note that the program uses the confidence factor c also as the expected error rate e. That is to say, if a 0.95 confidence level is required in the result, an assumption is made that the test may be expected to be wrong at a rate of 0.95 also. Thus, in effect, $c = e$. This is an arbitrary assumption, but seems reasonable and simplifies the number of parameters required.)

The relationship

$s = e^n$

can be transposed to

$n = \log(s)/\log(e)$

The easy way to confirm this is to plug in some numbers:

$9 = 3^2$

So the transposition supposes that

$2 = \log(9)/\log(3)$

$2 = 0.9542/0.4771$

which you will find is so.

But

$c = 1 – s$

and

$c = 1 – e^n$

since

$s = e^n$

which allows the whole formula for finding the necessary number of tests to be written as

$n = \log(1 – c)/\log(c)$

This is a very useful transformation, since it means that the number of confirmations required can be found directly by calculation.

For the 0.95, or 95%, confidence level ($c = 0.95$), and using natural logarithms for example, this becomes

$n = \log(1 - c)/\log(c)$

$n = \log(0.05)/\log(0.95)$

$n = -2.9957/-0.0513$

$n = 58.4$

This means that 59 tests of convergence agreeing that the variable is converged establishes the 95% confidence required. (Since “0.4” of a test is impossible, the 58.4 is increased to the next whole number, 59.)

This relationship is used in the demonstration software to test for convergence of variability. Essentially, the method is to keep increasing the number of instances in the sample, checking variability at each step, and wait until the variability is within the error band the required number of times.

Chapter 6: Handling Nonnumerical Variables

Overview

Given the representative sample, as described in Chapter 5, it may well consist of a mixture of variable types. Nonnumerical, or *alpha*, variables present a set of problems different from those of numerical variables. Chapter 2 briefly examined the different types of nonnumerical variables, where they were referred to as nominal and categorical. Distinguishing between the different types of alpha variables is not easy, as they blend into a continuum. Whenever methods of handling alpha variables are used, they must be effective at handling all types across the continuum.

The types of problems that have to be handled depend to some extent on the capabilities, and the needs, of the modeling tools involved. Some tools, such as decision trees, can handle alpha values in their alpha form. Other tools, such as neural networks, can handle only a numeric representation of the alpha value. The miner may need to use several different modeling tools on a data set, each tool having different capabilities and needs. Whatever techniques are used to prepare the data set, they should not distort its information content (i.e., add bias). Ideally, the data prepared for one tool should be useable by any other tool—and should give materially the same results with any tool that can handle the data.

Since all tools can handle numerical data but some tools cannot handle alpha data, the miner needs a method of transforming alpha values into appropriate numerical values.

Chapter 2 introduced the idea that the values in a data set reflect some state of the real world. It also introduced the idea that the ordering of, and spacing between, alpha variables could be recovered and expressed numerically by looking at the data set as a whole. This chapter explores how this can be done. The groundwork that is laid here is needed to cover issues other than just the numeration of alpha values, so rather than covering the same material more than once, several topics are visited, and ideas introduced, in slightly more detail than is needed just for dealing with alpha variables.

Four broad topics are discussed in this chapter. The first is the remapping of variable values, which can apply to both numeric and alpha values, but most often applies to alphas. The miner has to make decisions about the most appropriate representation of alpha variables. There are several automated techniques discussed in this chapter for discovering an appropriate numeric representation of alpha values. Unfortunately, there is no guarantee that these techniques will find even a good representation, let alone the best one. They will find the best numerical representation, given the form in which the alpha is delivered for preparation, and the information in the data set. However, insights and understanding brought by the miner, or by a domain expert, will almost certainly give a much better representation. What must be avoided at all costs is an arbitrary assignment of numbers to alpha labels. The initial stage in numerating alphas is for the miner to replace them with a numeration that has some rationale, if possible.

The second topic is a more detailed look at state space. Understanding state space is needed not only for numerating the alphas, but also for conducting the data survey and for addressing various problems and issues in data mining. Becoming comfortable with the concept of data existing in state space yields insight into, and a level of comfort in dealing with, modeling problems.

The third is *joint frequency distribution tables*. Data sets may be purely numeric, mixed alpha-numeric, or purely alpha. If the data set is purely numeric, then the techniques of this chapter are not needed—at least not for numerating alpha values. Data sets containing exclusively alpha values cannot reflect or be calibrated against associated numeric values in the data set because there are none. Numerating all alpha values requires a different technique. The one described here is based on the frequencies of occurrence of alpha values as expressed in joint frequency distribution tables.

The fourth broad topic is multidimensional scaling (MDS). Chapter 2 also introduced the idea that some alpha variables are most usefully translated into more than one single numeric value (ZIP codes into latitude and longitude, for example; see the explanation in Chapter 2). Taking that idea a step further, some technique is needed to project the values into the appropriate number of dimensions. The technique is multidimensional scaling. Although discussed in this chapter as a means to discover the appropriate dimensionality for a variable, MDS techniques can also be used for reducing the dimensionality of a data set.

6.1 Representing Alphas and Remapping

Exactly how alpha values are best represented depends very much on the needs of the modeling tool. Function-fitting modeling methods are sensitive to the form of the representation. These tools use techniques like regression and neural networks and, to some extent, symbolic regression, evolution programming, and equation evolvers. These tools yield final output that can be expressed as some form of mathematical equation (i.e., *x* = some combination and manipulation of input values). Other modeling methods, such as those sensitive mainly to the ordinality patterns in data, are usually less sensitive, although certainly not entirely insensitive, to representational issues, as they can handle alpha values directly. These include tools based on techniques like some forms of decision trees, decision lists, and some nearest-neighbor techniques.

However, all modeling tools used by data miners are sensitive to the one-to-many problem (introduced in the next section and also revisited in Chapter 11), although there are different ways of dealing with it. The one-to-many problem afflicts numeric variables as well as alpha variables. It is introduced here because for some representations it is an important consideration when remapping the alpha labels. If this problem does exist in an alpha variable, remapping using domain knowledge may prove to be the easiest way to deal with the problem.

There is an empirical way—a rule of thumb—for finding out if any particular remapping is both an improvement and robust. Try it! That is, build a few simple models with various methods. Ensure that at least the remapped values cause no significant degradation in performance over the default choice of leaving it alone. If in addition to doing no harm, it does some good, at least sometimes, it is probably worth considering. This empirical test is true too for the automated techniques described later. They have been chosen because in general, and with much practical testing, they at least do no harm, and often (usually) improve performance depending on the modeling tool used.

In general, remapping can be very useful indeed when one or more of these circumstances is true:

- The information density that can be remapped into a pseudo-variable is high.
- The dimensionality of the model is only modestly increased by remapping.
- A rational, or logical, reasoning can be given for the remapping.
- The underlying rationale for the model requires that the alpha inputs are to be represented without implicit ordering.

6.1.1 One-of-*n* Remapping

Ask a U.S. citizen or longtime resident, “How many states are there?” and you will probably get the answer “Fifty!” It turns out that for many models that deal with states, there are a lot more. Canadian provinces get squeezed in, as well as Mexican states, plus all kinds of errors. (Where, for instance, is IW?) However, sticking with just the 50 U.S. states, how are these to be represented? In fact, states are usually dealt with quite well by the automated numeration techniques described later in this chapter. However, remapping them is a classic example for neural networks and serves as a good general example of *one-of-n remapping*. It also demonstrates the problems with this type of remapping.

A one-of-*n* representation requires creating a binary-valued pseudo-variable for each alpha label value. For U.S. states, this involves creating 50 new binary pseudo-variables, one for each state. The numerical value is set to “1,” indicating the presence of the relevant particular label value, and “0” otherwise. There are 50 variables, only one of which is “on” at any one time. Only one “on” of the 50 possible, so “one-of-*n*” where in this case “*n*” is 50.

Using such a representation has advantages and disadvantages. One advantage is that the mean of each pseudo-variable is proportional to the proportion of the label in the sample; that is, if 25% of the labels were “CA,” then the mean of the pseudo-variable for the label “CA” would be 0.25. A useful feature of this is that when “State” is to be predicted, a model trained on such a representation will produce an output that is the model’s estimate of the probability of that state being the appropriate choice.

Disadvantages are several:

- Dimensionality is increased considerably. If there are many such remapped pseudo-variables, there can be an enormous increase in dimensionality that can easily prevent the miner from building a useful model.

- The density (in the state example) of a particular state may well be very low. If only the 50 U.S. states were used and each was equally represented, each would have a 2% presence. For many tools, such low levels are almost indistinguishable from 0% for each state; in other words, such low levels mean that no state can be usefully distinguished.

- Again, even when the pseudo-variables have reasonable density for modeling, the various outputs will all be “on” to some degree if they are to be predicted, estimating the probability that their output is true. This allows ranking the outputs for degree of probability. While this can be useful, sometimes it is very unhelpful or confusing to know that there is essentially a 50% chance that the answer is “NY,” and a 50% chance that the answer is “CA.”

6.1.2 *m*-of-*n* Remapping

While the na‹ve one-of-*n* remapping (one state to one variable) may cause difficulties, domain knowledge can indicate very useful remappings that significantly enhance the information content in alpha variables. Since these depend on domain knowledge, they are necessarily situation specific. However, useful remappings for state may include such features as creating a pseudo-variable for “North,” one for “South,” another for “East,” one for “West,” and perhaps others for other features of interest, such as population density or number of cities in the state. This *m*-of-*n* remapping is an advantage if either of two conditions is met. First, if the total number of additional variables is less than the number of labels, then *m*-of-*n* remapping increases dimensionality less than one-of-*n*—potentially a big advantage. Second, if the *m*-of-*n* remapping actually adds useful information, either in fact (by explicating domain knowledge), or by making existing information more accessible, once again this is an advantage over one-of-*n*.

This useful remapping technique has more than one of the pseudo-variables “on” for a single input. In one-of-*n*, one state switched “on” one variable. In *m*-of-*n*, several variables may be “on.” For instance, a densely populated U.S. state in the northeast activates several of the pseudo-variables. The pseudo-variables for “North,” “East,” and “Dense Population” would be “on.” So, for this example, one input label maps to three “on” input pseudo-variables. There could, of course, be many more than three possible inputs. In general, *m* would be “on” of the possible *n*—so it’s called an *m*-of-*n* mapping.

Another example of this remapping technique usefully groups common characteristics. Such character aggregation codings can be very useful. For instance, instead of listing the entire content of a grocery store’s produce section using individual alpha labels in a na‹ve one-of-*n* coding, it may be better to create *m*-of-*n* pseudo-variables for “Fruit,” “Vegetable,” “Root Crop,” “Leafy,” “Short Shelf Life,” and so on. Naturally, the useful characteristics will vary with the needs of the situation. It is usually necessary to ensure that the coding produces a unique pattern of pseudo-variable inputs for each alpha label—that is, for this example, a unique pattern for each item in the produce department. The domain expert must make sure, for example, either that the label “rutabaga” maps to a different set of inputs than the label “turnip,” or that mapping to the same input pattern is acceptable.

6.1.3 Remapping to Eliminate Ordering

Another use for remapping is when it is important that there be no implication of ordering among the labels. The automated techniques described in this chapter attempt to find an appropriate ordering and dimensionality of representation for alpha variables. It is very often the case that an appropriate ordering does in fact exist. Where it does exist, it should be preserved and used. However, it is the nature of the algorithms that they will always find an ordering and some dimensional representation for any alpha variable. It may be that the domain expert, or the miner, finds it important to represent a particular variable without ordering. Using remapping achieves model inputs without implicit ordering.

6.1.4 Remapping One-to-Many Patterns, or Ill-Formed Problems

The one-to-many problem can defeat any function-fitting modeling tool, and many other tools too. The problem arises when one input pattern predicts many output patterns. Since mining tools are often used to predict single values, it is convenient to discuss the problem in terms of predicting a single output value. However, since it is quite possible for some tools to predict several output values simultaneously, throughout the following discussion the single value output used for illustration must be thought of as a surrogate for any more complex output pattern. This is not a problem limited to alpha variables by any means. However, since remapping may provide a remedy for the one-to-many problem, we will look at the problem here.

Many modeling tools look for patterns in the input data that are indicative of particular output values. The essence of a predictive model is that it can identify particular input patterns and associate specific output values with them. The output values will always contain some level of noise, and so a prediction can only be to some degree approximately accurate. The noise is assumed to be "fuzz" surrounding some actual value or range of values and is an ineradicable part of the prediction. (See Chapter 2 for a further discussion of this topic.)

A severe and intractable problem arises when a single input pattern should accurately be associated with two or more discrete output values. Figure 6.1 shows a graph of data points. Modeling these points discovers a function that fits the points very well. The function is shown in the title of the graph. The fit is very good.

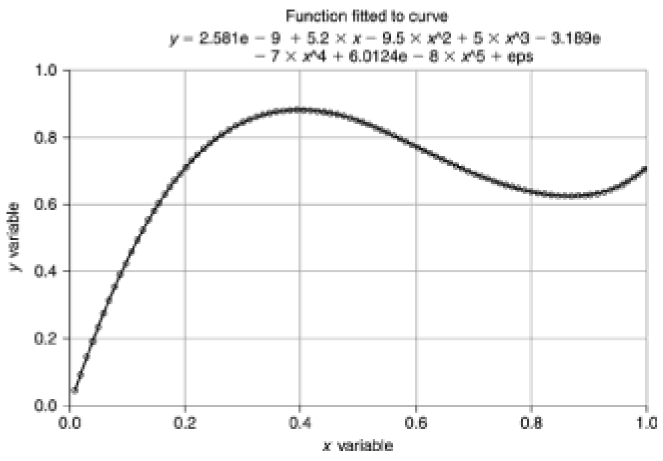


Figure 6.1 The circles show the location of the data points, and the continuous line traces the path of the fitted function. The discovered function fits the function well as there is only a single value of *y* for every value of *x*.

Figure 6.2 shows a totally different situation. Here the original curve has been reflected across the bottom-left, top-right diagonal of the curve, and fitting a function to this curve is a disaster. Why? Because for much of this curve, there is no single value of *y* for every value of *x*. Take the point *x* = 0.7, for example. There are three values of *y: y* = 0.2, *y* = 0.7, and *y* = 1.0. For a single value of *x* there are three values of *y*—and no way, from just knowing the value of *x*, to tell them apart. This makes it impossible to fit a function to this curve. The best that a function-fitting modeling tool can do is to find a function that somehow fits. The one used in this example found as its best approximation a function that can hardly be said to describe the curve very well.

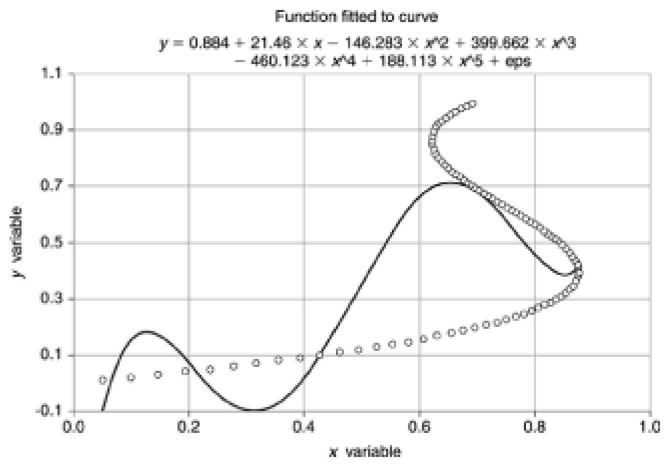


Figure 6.2 The solid line shows the best-fit function that one modeling tool could discover to fit the curve illustrated by the circles. When a curve has multiple predicted (y) values for the input value (*x*), no function can fit the curve.

In Figure 6.2 the input "pattern" (here a single number) is the *x* value. The output pattern is the *y* value. This illustrates the situation in data sets where, for some part of the range, the input pattern genuinely maps to multiple output patterns. One input, many outputs, hence the name one-to-many. Note that the problem is not noise or uncertainty in knowing the value of the output. The output values of *y* for any input values of *x* are clearly specified and can be seen on the graph. It's just that sometimes there is more than one output value associated with an input value. The problem is not that the "true" value lies somewhere between the multiple outputs, but that a function can only give a single output value (or pattern) for a unique input value (or pattern).

Does this problem occur in practice? Do data miners really have to deal with it? The curve shown in Figure 6.1 is a normalized, and for demonstration purposes, somewhat cleaned up, profit curve. The *x* value corresponds to product price, the *y* value to level of profit. As price increases, so does profit for awhile. At some critical point, as price increases, profit falls. Presumably, more customers are put off by the higher price than are offset by the higher profit margin, so overall profit falls. At some point the overall profit rises again with increase in price. Again presumably, enough people still see value in the product at the higher price to keep buying it so that the increase in price generates more overall profit. Figure 6.1 illustrates the answer to the question What level of profit can I expect at each price level over a range?

Figure 6.2 has price on the *y*-axis and profit on the *x*-axis, and illustrates the answer to the question What price should I set to generate a specific level of profit? The difficulty is that, in this example, there are multiple prices that correspond to some specific levels of profit. Many, if not most, current modeling tools cannot answer this question in the situation illustrated.

There are a number of places in the process where this problem can be fixed, *if* it is detected. And that is a very big if! It is often very hard to determine areas of multivalued output. Miners, when modeling, can overcome the problem using a number of techniques. The data survey (Chapter 11) is the easiest place to detect the problem, if it is not already known to be a problem. However, if it is recognized, and possible, by far the easiest stage in which to correct the problem is during data preparation. It requires the acquisition of some additional information that can distinguish the separate situations. This additional information can be coded into a variable, say, *z*. Figure 6.3 shows the curve in three dimensions. Here it is easy to see that there are unique *x* and *z* values for every point—problem solved!

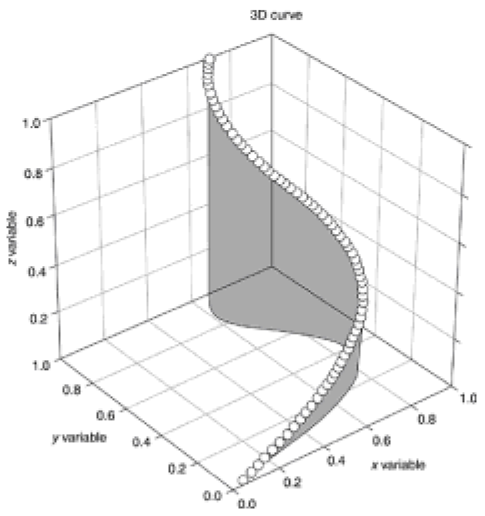


Figure 6.3 Adding a third dimension to the curve allows it to be uniquely characterized by values *x* and *z*. If there is additional information allowing the states to be uniquely defined, this is an easy solution to the problem.

Not quite. In the illustration, the variable *z* varies with *y* to make illustrating the point easy. But because *y* is unknown at prediction time, so is *z*. It’s a Catch-22! However, if additional information that can differentiate between the situations is available at preparation time, it is by far the easiest time to correct the problem.

This book focuses on data preparation. Discussing other ways of fixing the one-to-many problem is outside the present book’s scope. However, since the topic is not addressed any further here, a brief word about other ways of attacking the problem may help prevent anguish!

There is a clue in the way that the problem was introduced for this example. The example simply reflected a curve that was quite easily represented by a function. If the problem is recognized, it is sometimes possible to alleviate it by making a sort of reflection in the appropriate state space. Another possible answer is to introduce a local distortion in state space that “untwists” the curve so that it is more easily describable. Care must be taken when using these methods, since they often either require the answer to be known or can cause more damage than they cure! The data survey, in part, examines the manifold carefully and should report the location and extent of any such areas in the data. At least when modeling in such an area of the data, the miner can place a large sign “Warning—Quicksand!” on the results.

Another possible solution is for the miner to use modeling techniques that can deal with such curves—that is, techniques that can model surfaces not describable by functions. There are several such techniques, but regrettably, few are available in commercial products at this writing. Another approach is to produce separate models, one for each part of the curve that is describable by a function.

6.1.5 Remapping Circular Discontinuity

Historians and religions have debated whether time is linear or circular. Certainly scientific time is linear in the sense that it proceeds from some beginning point toward an end. For miners and modelers, time is often circular. The seasons roll endlessly round, and after every December comes a January. Even when time appears to be numerically labeled, usually ordinally, the miner should consider what nature of labeling is required inside the model.

Because of the circularity of time, specifying timelike labels has particular problems. Numbering the weeks of the year from “1” to “52” demonstrates the problem. Week 52, on a seasonal calendar, is right next to week 1, but the numbers are not adjacent. There is discontinuity between the two numbers. Data that contains annual cycles, but is ordered as consecutively numbered week labels, will find that the distortion introduced very likely prevents a modeling tool from discovering any cyclic information.

A preferable labeling might set midsummer as “1” and midwinter as “0.” For 26 weeks the “Date” flag, a *lead variable*, might travel from “0” toward “1,” and for the other 26 weeks from “1” toward “0.” A *lag variable* is used to unambiguously define the time by reporting what time it was at some fixed distance in the past. In the example illustrated in Figure 6.4, the lag variable gives the time a quarter of a year ago. These two variables provide an unambiguous indication of the time. The times shown are for solstices and equinoxes, but every instant throughout the cycle is defined by a unique pair of values. By using this representation of lead and lag variables, the model will be able to discover interactions with annual variations.

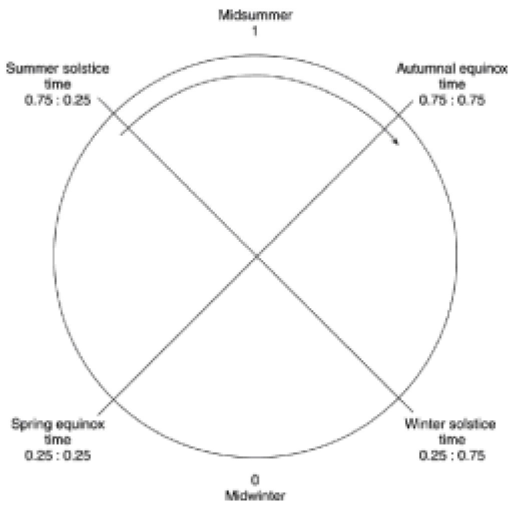

Midsummer
1
Summer solstice time 0.75 : 0.25
Autumnal equinox time 0.75 : 0.75
Spring equinox time 0.25 : 0.25
Winter solstice time 0.25 : 0.75
0
Midwinter

Figure 6.4 An annual "clock." The time is represented by two variables—one showing the time now and one showing where the time was a quarter of a year ago.

Annual variation is not always sufficient. When time is expected to be important in any model, the miner, or domain expert, should determine what cycles are appropriate and expected. Then appropriate and meaningful continuous indicators can be built. When modeling human or animal behavior, various-period circadian rhythms might be appropriate input variables. Marketing models often use seasonal cycles, but distance in days from or to a major holiday is also often appropriate. Frequently, a single cyclic time is not enough, and the model will strongly benefit from having information about multiple cycles of different duration.

Sometimes the cycle may rise slowly and fall abruptly, like "weeks to Thanksgiving." The day after Thanksgiving, the effective number of weeks steps to 52 and counts down from there. Although the immediately past Thanksgiving may be "0" weeks distant, the salient point is that once "this" Thanksgiving is past, it is immediately 52 weeks to next Thanksgiving. In this case the "1" through "52" numeration is appropriate—but it must be anchored at the appropriate time, Thanksgiving in this case. Anchoring "weeks to Thanksgiving" on January 1st, or Christmas, say, would considerably reduce the utility of the ordering.

As with most other alpha labels, appropriate numeration adds to the information available for modeling. Inappropriate labeling at best makes useful information unavailable, and at worst, destroys it.

6.2 State Space

State space is a space exactly like any other. It is different from the space normally perceived in two ways. First, it is not limited to the three dimensions of accustomed space (or four if you count time). Second, it can be measured along any ordered dimensions that are convenient.

For instance, choosing a two-dimensional state space, the dimensions could be “inches of rain” and “week of the year.” Such a state space is easy to visualize and can be easily drawn on a piece of paper in the form of a graph. Each dimension of space becomes one of the axes of the graph. One of the interesting things about this particular state space is that, unlike our three-dimensional world, the values demarking position on a dimension are bounded; that is to say, they can only take on values from a limited range. In the normal three-dimensional world, the range of values for the dimensions “length,” “breadth,” and “height” are unlimited. Length, breadth, or height of an object can be any value from the very minute—say, the Planck constant (a very minute length indeed)—to billions of light-years. The familiar space used to hold these objects is essentially unlimited in extent.

When constructing state space to deal with data sets, the range of dimensional values is limited. Modeling tools do not deal with monotonic variables, and thus these have to be transformed into some reexpression of them that covers a limited range. It is not at all a mathematical requirement that there be a limit to the size of state space, but the spaces that data miners experience almost always are limited.

6.2.1 Unit State Space

Since the range of values that a dimension can take on are limited, this also limits the “size” of the dimension. The range of the variable fixes the range of the dimension. Since the limiting values for the variables are known, all of the dimensions can be *normalized*. Normalizing here means that every dimension can be constructed so that its maximum and minimum values are the same. It is very convenient to construct the range so that the maximum value is 1 and the minimum 0. The way to do this is very simple. (Methods of normalizing ranges for numeric variables are discussed in Chapter 7.)

When every dimension in state space is constructed so that the maximum and minimum values for each range are 1 and 0, respectively, the space is known as *unit state space*—“unit” because the length of each “side” is one unit long; “state space” because each uniquely defined position in the space represents one particular *state* of the system of variables. This transformation is no more than a convenience, but making such a transformation allows many properties of unit state space to be immediately known. For instance, in a two-dimensional unit state space, the longest straight line that can be constructed is the corner-to-corner diagonal. State space is constructed so that its dimensions are all at right angles to each other—thus two-dimensional state space is rectangular. Two-dimensional unit state space not only is rectangular, but has “sides” of the same unit length, and so is square. Figure 6.5 shows the corner-to-corner diagonal line, and it immediately is clear that that the Pythagorean theorem can be used to find the length of the line, which must be 1.41 units.

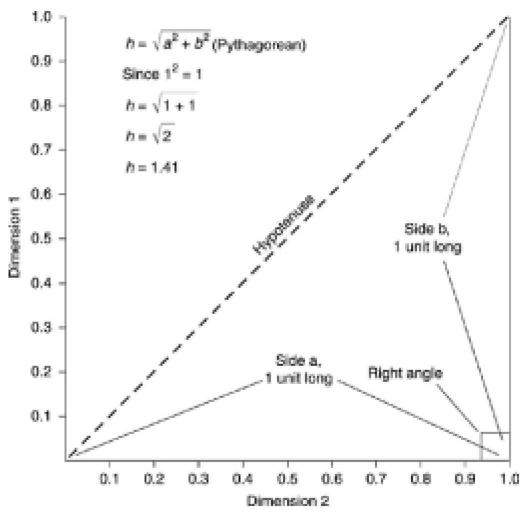


Figure 6.5 Farthest possible separation in state space.

6.2.2 Pythagoras in State Space

Two-dimensional state space is not significantly different from the space represented on the surface of a piece of paper. The Pythagorean theorem can be extended to a three-dimensional space, and in a three-dimensional unit state space, the longest diagonal line that can be constructed is 1.73 units long. What of four dimensions? In fact, there is an analog of the Pythagorean theorem that holds for any dimensionality of state space that miners deal with, regardless of the number of dimensions. It might be stated as: In any right-angled multiangle, the square on the multidimensional hypotenuse is equal to the sum of the squares on all the other sides. The length of the longest straight line that can be constructed in a four-dimensional unit state space is 2, and of a five-dimensional unit state space, 2.24. It turns out that this is just the square root of the number of sides, since the square on a unit side, the square of 1, is just 1.

This means that as more dimensions are added, the longest straight line that can be drawn increases in length. Adding more dimensions literally adds more space. In fact, the longest straight line that can be drawn in unit state space is always just the square root of the number of dimensions.

6.2.3 Position in State Space

Instead of just finding the longest line in state space, the Pythagorean theorem can be used to find the distance between any two points. The position of a point is defined by its coordinates, which is exactly what the instance values of the variables represent. Each unique set of values represents a unique position in state space. Figure 6.6 shows how to discover the distance between two points in a two-dimensional state space. It is simply a matter of finding the distance between the points on one axis and then on the other axis, and then the diagonal length between the two points is the shortest distance between the two points.

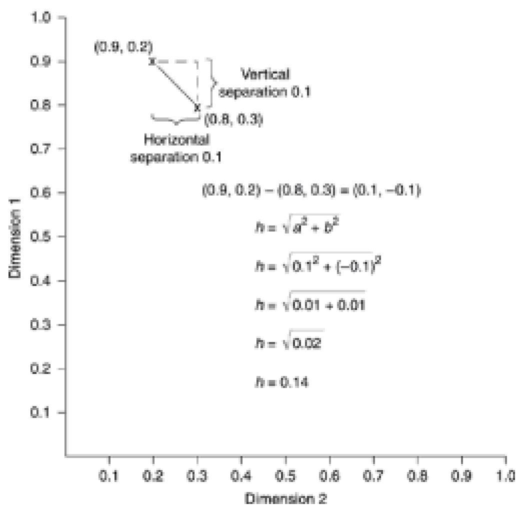


Figure 6.6 Finding the distance between two points in a 2D state space.

Just as with finding the length of the longest straight line that can be drawn in state space, so too this finding of the distance between two points can be generalized to work in higher-dimensional state spaces. But each point in state space represents a particular state of the system of variables, which in turn represent a particular state of the object or event existing in the real world that was being measured. State space provides a standard way of measuring and expressing the distance between any states of the system, whether events or objects.

Using unit state space provides a frame of reference that allows the distance between any two points in that space to be easily determined. Adding more dimensions, because it adds more space in which to position points, actually moves them apart. Consider the points shown in Figure 6.6 that are 0.1 units apart in both dimensions. If another dimension is added, unless the value of the position on that dimension is identical for both points, the distance between the points increases. This is a phenomenon that is very important when modeling data. More dimensions means more sparsity or distance between the data points in state space. A modeling tool has to search and characterize state space, and too many dimensions means that the data points disappear into a thin mist!

6.2.4 Neighbors and Associates

Points in state space that are close to each other are called *neighbors*. In fact, there is a data modeling technique called “nearest neighbor” or “*k*-nearest neighbor” that is based on this concept. This use of neighbors simply reflects the idea that states of the system that are close together are more likely to share features in common than system states further apart. This is only true if the dimensions actually reflect some association between the states of the system indicated by their positions in state space.

Consider as an example Figure 6.7. This shows a hypothetical relationship in two-dimensional unit state space between human age and height. Since height changes as people grow older up to some limiting age, there is an association between the two dimensions. Neighbors close together in state space tend to share common characteristics up to the limiting age. After the limiting age—that is, the age at which humans stop growing taller—there is no particular association between age and height, except that this range has lower and upper limits. In the age dimension, the lower limit is the age at which growth stops, and the upper limit is the age at which death occurs. In the height dimension, after the age at which growth stops, the limits are the extremes of adult height in the human population. Before growth stops, knowing the value of one dimension gives an idea of what the value of the other dimension might be. In other words, the height/age neighborhood can be usefully characterized. After growth stops, the association is lost.

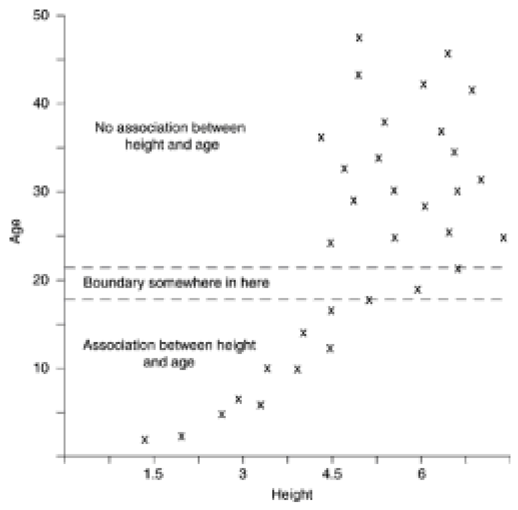


Figure 6.7 Showing the relationship between neighbors and association when there is, and is not, an association between the variables.

This simplified example is interesting because although it is simplified, it is similar to many practical data characterization problems. For sets of variables other than just human height and weight, the modeler might be interested in discovering that there are boundaries. The existence and position of such boundaries might be an unknown piece of information. The changing nature of a relationship might have to be discovered. It is clear that for some part of the range of the data in the example, one set of predictions or inferences can be made, and in another part of the same data set, wholly different inferences or predictions must be made. This change in the nature of the neighborhood from place to place can be very important. In two dimensions it is easy to see, but in high-dimensionality spaces this can be difficult to discover.

6.2.5 Density and Sparsity

Before continuing, a difference in the use of the terms *location* or *position*, and *points* or *data points*, needs to be noted.

In any space there are an infinite number of places or positions that can be specified. Even the plane represented by two-dimensional state space has an infinite number of positions on it that can be represented. In fact, even on a straight line, between any two positions there are an infinite number of other positions. This is because it is always possible to specify a location on a dimension that is between any two other locations. For instance, between the locations represented by 0.124 and 0.125 are other locations represented by 0.1241, 0.1242, 0.1243, and so on. This is a property of what is called the *number line*. It is always possible to use more precision to specify more locations. The terms *location* or *position* are used to represent a specific place in space.

Data, of course, has values—instance values—that can be represented as specifying a particular position. The instance values in a data set, representing particular states of the system, translate into representing particular positions in state space. When a particular position is actually represented by an instance value, it is referred to as a *data point* or *point* to indicate that this position represents a measured state of the system.

So the terms *location* and *position* are used to indicate a specific set of values that might or might not be represented by an instance value in the data. The terms *point* and *data point* indicate that the location represents recorded instance values and therefore corresponds to an actual measured state of the system.

Turning now to consider density, in the physical world things that are dense have more “stuff” in them per unit volume than things that are less dense. So too, some areas of state space have more data points in them for a given volume than other areas. State space density can be measured as the number of data points in a specific volume. In a dense part of state space, any given location has its nearest neighboring points packed around it more closely than in more sparsely populated parts of state space.

Naturally, in a state space of a fixed number of dimensions, the absolute mean density of the data points depends on the number of data points present and the size of the space. The number of dimensions fixes unit state space volume, but the number of data points in that volume depends only on how much data has been collected. However, given a representative sample, if there are associations among the dimensions, the relative density of one part of state space remains in the same relationship to the relative density of another part of the same space regardless of how many data points are added.

If this is not intuitive, imagine two representative samples drawn from the same population. Each sample is projected into its own state space. Since the samples are representative of the same population, both state spaces will have the same dimensions, normalized to the same values. If this were not so, then the samples would not be truly representative of the same population. Since both data sets are indeed representative of the same population, the distributions of the variables are, for all practical purposes, identical in both samples, as are the joint distributions. Thus, any given specific area common to both state spaces will have the same proportion of the total number of points in each space—not necessarily the same actual number of points, as the representative samples may be of different sizes, but the same relative number of points.

Because both representative data sets drawn from a common population have similar relative density throughout, adding them together—that is, putting all of the data points into a common state space—does not change the relative density in the common state space. As a specific example, if some defined area of both state spaces has a relative density twice the mean density, when added together, the defined area of the common state space will also have a density twice the mean—even though the mean will have changed. Table 6.1 shows an example of this.

TABLE 6.1 State space density.

	Mean density	**Specific area density**
Sample 1	20	40
Sample 2	10	20
Combined	30	60

This table shows the actual number of data points in two samples representative of the same population. The specific area density in each sample is twice the mean density even though the number of points in each sample is different. When the two samples are combined, the combined state space still has the same relative specific area density as each of the original state spaces. So it is that when looking at the density of a particular volume of space, it is *relative density* that is most usefully examined.

There are difficulties in determining density just by looking at the number of points in a given area, particularly if in some places the given volume only has one data point, or even no data points, in it. If enough data points from a representative sample are added, eventually any area will have a measurable density. Even a sample of empty space has some density represented by the points around it. The density at any position also depends on the size and shape of the area that is chosen to sample the density. For many purposes this makes it inconvenient to just look at chunks of state space to estimate density.

Another way of estimating density is to choose a point, or a position, and estimate the distance from there to each of the nearest data points in each dimension. The mean distance to neighboring data points serves as a surrogate measurement for density. For many purposes this is a more convenient measure since every point and position then has a measurable density. The series of illustrations in Figure 6.8 shows this. The difficulty, of course, is in determining exactly what constitutes a nearest neighbor, and how many to use.

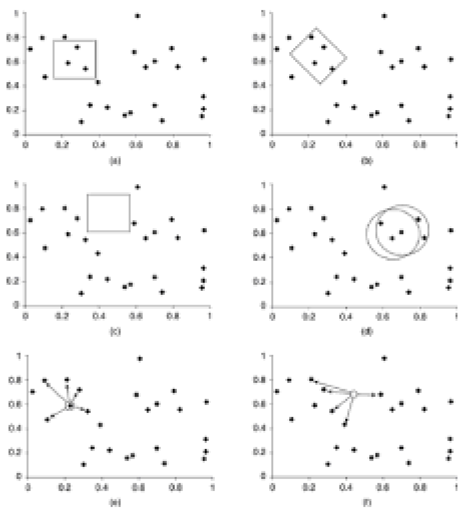


Figure 6.8 Estimating density: inside a square (a), rotating the same square (b), same square moved to an unoccupied area (c), circular area (d), distance to a number of neighbors (e), and distance to neighboring points from an empty position (f).

Figure 6.8(a) shows the density inside a square to be 3. The same square in the same location but rotated slightly could change the apparent density, as shown in Figure 6.8(b). Figure 6.8(c) shows a square in an unoccupied space, which makes deciding what the density is, or what it could be if more points were added, problematic. Using a circular area can still have large jumps in density with a small shift in position, as shown in Figure 6.8(d). Figure 6.8(e) shows that measuring the distance to a number of neighbors gives each point a unique density. Even an empty position has a measurable density by finding the distances to neighboring points, as shown in Figure 6.8(f).

A better way of estimating density determines a weighted distance from every point in state space to every other point. This gives an accurate density measure and produces a continuous density gradient for all of space. Determining the nature of any location in space uses the characteristics of every point in space, weighted by their distance. This method allows every point to “vote” on the characteristics of some selected location in space according to how near they are, and thus uses the whole data set. Distant points have little influence on the outcome, while closer points have more influence. This works well for determining nature and density of a point or location in state space, but it does necessitate that any new point added requires recalculation of the entire density structure. For highly populated state spaces, this becomes computationally intensive (slow!).

6.2.6 Nearby and Distant Nearest Neighbors

As with many things in life, making a particular set of choices has trade-offs. So it is with nearest-neighbor methods. The first compromise requires deciding on the number of nearby neighbors to actually look at. Figures 6.8(e) and 6.8(f) illustrate five neighbors near to a point or position. Using nearest neighbors to determine the likely behavior of the system for some specified location has different needs than using nearest neighbors to estimate density. When estimating system behavior, using some number of the closest neighbors in state space may provide the best estimate of system behavior. It usually does not provide the best estimate of density.

Figure 6.9(a) illustrates why this might be the case. This example shows the use of four neighbors. The closest neighbors to the point circled are all on one side of the point. Using only these points to estimate density does not reflect the distance to other surrounding points. A more intuitive view of density requires finding the nearest neighbors in “all directions” (or omnidirectionally) around the chosen point. Having only the very closest selected number of points all biased in direction leads to an overestimate of the omnidirectional density.

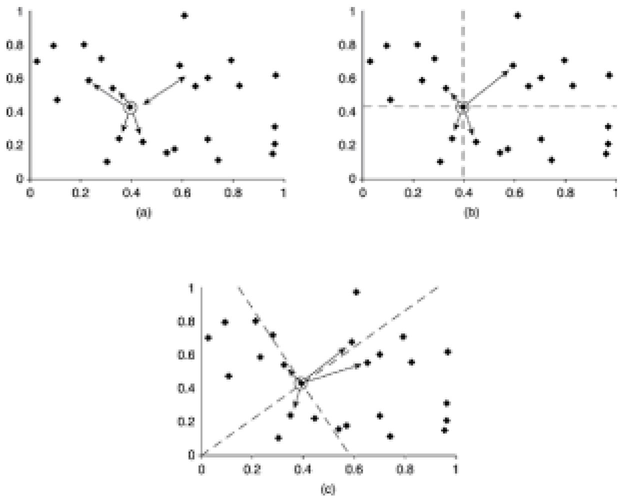


Figure 6.9 Results of estimating density with nearest neighbors: overestimate (a), better estimate by looking for nearest neighbors in specific areas (b), and change in estimate by rotating the axes of same specific areas (c).

One way around this shortcoming is to divide up the area to be searched, and to find a nearest neighbor in each division, as shown in Figure 6.9(b). Still using four neighbors, dividing space into quadrants and finding a nearest neighbor in each quadrant leads to a better estimate of the omnidirectional density. However, no compromise is perfect. As Figure 6.9(c) shows, rotating the axes of the quadrants can significantly change the estimated density.

Since “divide and conquer” provides useable estimates of density and serves to identify nearest neighbors, the demonstration code uses this both for neighbor estimation and as a quick density estimation method.

6.2.7 Normalizing Measured Point Separation

Using normalized values of dimensions facilitates building a *unit* state space. This has some convenient properties. Can distance measured between points be normalized? State space size (volume) is proportional to the number of dimensions that constitute the space. In a unit state space, the maximum possible separation between points is known—the square root of the number of dimensions. Regardless of the number of dimensions, no two points can be further separated than this distance. Similarly, no two positions can be closer together than having no separation between them. This means that the separation between points *can* be normalized. Any particular separation can be expressed as a fraction of the maximum possible separation, which comes out as a number between 0 and 1.

Density is not usually usefully expressed as a normalized quantity. Since it is relative density that is of interest, density at a point or location is usually expressed relative to the mean, or average, density. It is always possible for a particular position to be any number of times more or less dense than the average density, say, 10 or 20 times. It is quite possible to take the maximum and minimum density values found in a particular state space and normalize the range, but is it usually more useful to know the density deviation from the mean value. In any case, as more data points are added, the maximum, minimum, and mean values will change, requiring recalibration if density is to be normalized. However, as discussed above, given a representative sample data set, relative density overall will not change with additional data from the same population.

6.2.8 Contours, Peaks, and Valleys

Instead of simply regarding the points in state space as having a particular density, imagine that the density value is graduated across the intervening separation. Between a point of high density and its lower-density neighbor, the density decreases across the distance from high value to low. State space can be imagined as being permeated by a continuous gradient of density, perhaps going “down” toward the densest areas, and “up” toward the least dense areas. This up-and-down orientation conjures up the idea of a surface of some sort that represents the expression of the gradient. The surface has high points representing areas of least density and low points representing areas of most density. The slope of the surface represents the rate of change in density at that position.

Three-dimensional surfaces of this sort, surfaces such as that of the earth’s, can be mapped topographically. Such maps often show lines that are traced over the surface marking the positions of a particular constant elevation. Such lines are known as *contours*. Other sorts of contours can be traced, for example, along a ridge between two high points, or along the deepest part of a valley between two low points. A density surface can also be mapped with a variety of contours analogous to those used on a topographic map.

6.2.9 Mapping State Space

Exploring features of the density surface can reveal an enormous amount of useful, even vital information. Exploring the density map forms a significant part of the data survey. For example, tracing all of the “ridges”—that is, tracing out the contours that wend their way through the least densely populated areas of state space—leads to identifying groups of natural clusters. Each cluster of points swarms about a point of maximum density. Keeping in mind that this map ultimately represents some state of an object in the real world, the mapped clusters show the systems’ “preferred” states—or do they? Maybe they show evidence of bias in the data collection. Perhaps data about those states was for some reason preferentially collected, and they predominate simply because they were the easiest to collect. (Chapter 11 covers the practical application of this more fully. Here we are just introducing the ideas that will be used later.)

6.2.10 Objects in State Space

Sometimes a more useful metaphor for thinking of the points in state space is as a geometric object of some sort, even though when more than three dimensions are used it is hard to imagine such an object. Nonetheless, if the points in state space are thought of as “corners,” it is possible to join them with the analogs of lines and surfaces.

Three points in a two-dimensional state space could form the corners of a triangle. To construct the triangle, the points are simply joined by lines. Similarly, in three-dimensional space, points are joined by planes to form three-dimensional figures.

An interesting feature of the object analogy is that, just as with objects in familiar space, they can cast “shadows.” In the familiar world, a three-dimensional object illuminated by the sun casts a two-dimensional shadow. The shadow represents a more or less distorted image of the object. So it is in state space that higher-dimensional objects can cast lower-dimensional shadows. This ability of objects to cast shadows is one of the features used in multidimensional scaling.

6.2.11 Phase Space

Phase space is identical to state space in almost all respects, with a single exception. *Phase space* is used to represent features of objects or systems *other* than their state. Since a system state is not represented in phase space, the name of the space changes to reflect that. The reason to introduce what is essentially an identical sort of space to state space is that when numerating alpha values, a space is needed in which to represent the distances between the *labels*. Alpha labels, you will recall, do not represent states of the system, but values of a particular variable. In order to numerate alpha labels, or in other words to assign them particular numeric values indicating their order and spacing, a space has to be created in which the labels can exist. The alpha labels are arrayed in this space, each with a particular distance and direction from its neighboring labels. Finding the appropriate place to put the labels in phase space is discussed in the next section. The point is that when the appropriate positions for the labels are known, then the appropriate label values can be found.

The most important point to note here is that the name of the space does not change its properties. It simply identifies if the space is used to hold states of a system of variables (state space) or some other features (phase space).

Why the name “phase space”? Well, “phase” indicates a relationship between things. Electrical engineers are familiar with three-phase alternating-current power. This only means that three power pulses occur in a complete cycle, and that they have a specific, fixed relationship to each other. As another example, the phases of the moon represent specific, and changing, relationships between the earth, moon, and sun. So too with phase space. This is an imaginary space, identical in almost all respects to state space, except that relationships, or phases, between things are represented.

6.2.12 Mapping Alpha Values

So far, all of the discussion of state space has assumed dimensions that are numerically scaled and normalized into the range 0 to 1. Where do alpha values fit in here?

Between any two variables, whether alpha or numeric, there is some sort of relationship. As in the height/age example, characterizing the precise nature of the relationship may be difficult. In some parts of the range, the variables may allow better or worse inferences about how the values relate. Nonetheless, it is the existence of a relationship that allows any inferences to be made. Statistically, the variables may be said to be more or less independent of each other. If fully independent, it could be said that there is no relationship. Actually, it is more accurate to say that when variables are independent, knowing something about the state of one variable tells nothing about the state of the other. There is still a relationship, but it carries no useful information. As an example of complete statistical independence, flipping a coin and knowing the result tells you nothing whatever about the time at which the flip was made.

The system of variables that is used to populate state space is exactly that, a system. A system has interreacting and interlocking components. The system reflects, more or less, the real world, and the world is not a purely random phenomenon. The instance values represent snapshots of parts of the system in action. It may be that the system is not well understood; indeed, it may be that understanding the system is the whole purpose of the data exploration enterprise. Nonetheless, a system is not going to have all of its components independent of each other. If all of the components have no relation whatsoever to each other, it hardly qualifies as a system!

It is the interrelationship between the alpha values and the system of variables as a whole that allows their appropriate numeration. Numeration does not recover the actual values appropriate for an alpha variable, even if there are any. It may very well be that there are no inherently appropriate actual values. Although cities, for instance, can be ranked (say, through an opinion poll) for "quality of life," placed in order, and separated by an appropriate distance along the ranking, there is no absolute number associated with each position. The quality-of-life scale might be from 1 to 10, or 1 to 100, or 0 to 1. It could even be from 37.275 to 18.462, although that would not be intuitive to humans. What is recoverable is the appropriate order and separation. For convenience, the scale for recovery is normalized from 0 to 1, which allows them to be conveniently positioned in unit state space.

6.2.13 Location, Location, Location!

In real estate, location is all. So too when mapping alphas. The points in state space can be mapped. Alpha variables that are in fact associated with the system of variables can also be appropriately placed on this map. The values of an alpha variable are labels. The numeration method associates each label with some appropriate particular "area" on the state space map. (It is an area in two dimensions, a volume in three dimensions, and some unnamed analog in more than three. For convenience it is referred to throughout this explanation as an "area.") Discovering the appropriate location of the area is the heart of the method; having done this, the problem then is to turn the high-dimensionality position into an appropriate number. The techniques for doing that are discussed later in this chapter in the section on multidimensional scaling.

The simplest state space that can contain two variables is a two-dimensional state space. If one of the variables is numeric and one alpha, the problem of finding an appropriate value from multiple numeric dimensions does not exist since there is only a single dimension of numeric value (which means only one number) at any location. While a single numeric may not provide a particularly robust estimation of appropriate numeration of alphas, it can provide an easily understood example.

6.2.14 Numerics, Alphas, and the Montreal Canadiens

Table 6.2 shows a list of the team members on the 1997/1998 roster, together with their height and weight. There is an alpha variable present in this data set—“Position.” Unfortunately, if used as an example, when finished there is no way to tell if appropriate numerical values are assigned since the labels have no inherent ordering. With no inherent ordering to compare the recovered values against, the results cannot be checked. A convincing first example needs to be able to be checked for accuracy! So, for the purpose of explanation, a numerical variable will be labeled with alpha labels. Then, when values have been “recovered” for these labels, it is easy to compare the original values with those recovered to see if indeed an appropriate ordering and spacing have been found. With the underlying principles visible by using an example that numerates labels derived from what is actually a numeric variable, we can examine the problem of numerating “Position” as a second example.

TABLE 6.2 Montreal Canadiens roster in order of player weight.

Position	Num	Name	Height	Weight	Code	DoB	NmHt
Defense	34	Peter Popovic	6.5	235	a	10-Feb-68	1
Defense	38	Vladimir Malakhov	6.3	227	b	30-Aug-68	0.759
Forward	21	Mick Vukota	6.08	225	c	14-Sep-66	0.494
Forward	23	Turner Stevenson	6.25	220	d	18-May-72	0.699
Defense	22	Dave Manson	6.17	219	e	27-Jan-67	0.602
Forward	24	Scott Thornton	6.25	219	e	9-Jan-71	0.699
Forward	44	Jonas	6.25	215	f	29-Aug-72	0.699

		Hoglund					
Defense	5	Stephane Quintal	6.25	215	f	22-Oct-68	0.699
Defense	33	Zarley Zalapski	6.08	215	f	22-Apr-68	0.494
Forward	37	Patrick Poulin	6.08	210	g	23-Apr-73	0.494
Reserve	55	Igor Ulanov	6.08	205	h	1-Oct-69	0.494
Forward	26	Martin Rucinsky	6.08	205	h	11-Mar-71	0.494
Defense	43	Patrice Brisebois	6.17	204	j	27-Jan-71	0.602
Forward	28	Marc Bureau	6.08	202	k	19-May-66	0.494
Forward	27	Shayne Corson	6.08	199	m	13-Aug-66	0.494
Defense	52	Craig Rivet	6.17	195	n	13-Sep-74	0.602
Forward	17	Benoit Brunet	6	194	p	24-Aug-68	0.398
Forward	49	Brian Savage	6.08	191	q	24-Feb-71	0.494
Forward	25	Vincent Damphousse	6.08	191	q	17-Dec-67	0.494
Forward	71	Sebastien Bordeleau	5.92	188	r	15-Feb-75	0.301
Forward	15	Eric Houde	5.83	186	s	19-Dec-76	0.193

Forward	8	Mark Recchi	5.83	185	t	1-Feb-68	0.193
Defense	29	Brett Clark	6	182	u	23-Dec-76	0.398
Reserve	11	Saku Koivu	5.83	182	u	23-Nov-74	0.193
Goal	35	Andy Moog	5.67	177	v	18-Feb-60	0
Goal	41	Jocelyn Thibault	5.92	170	w	12-Jan-75	0.301

Example 1—A Weighty Problem

For the convenience of the example, the weights of the athletes are labeled from "a" through "w," missing out those letters that might be confused with numbers like "l" and "o." To make it easier to see what is happening, "a" represents the heaviest weight and "w" the lightest. The labels could have been assigned arbitrarily; the ordering only helps to show what is happening. (Note: It causes a problem to assign numeric values to alpha labels arbitrarily, *not* vice versa. Alpha labels are, by nature, arbitrary.) The numeric variable used in this example will be "Height." To see how well the normalized weights can be recovered from alpha labels, the weight will be converted to an alpha value. The actual weights can be compared with the recovered values to determine if the method was effective.

Table 6.2 shows the names, heights, and weights of the athletes. The heights, the numeric variable in this example, are shown in feet and decimal fractions of a foot. In order to construct a unit state space, height has to be normalized, and this is also shown. Next are the weights in pounds and their associated labels. Since some of the athletes weigh the same amount, these weights are assigned identical labels.

The athletes' heights form a one-dimensional state space, which can be easily represented by an ordered list such as the one in Table 6.3. The column on the left shows the ordered, normalized heights for each athlete, and the right-hand column shows the alpha (weight) labels. Some of the labels appear more than once for those athletes having similar weights. When "projecting" the height values onto the weight labels, since there is only a single numeric dimension, the values of most of the labels are simply the normalized height values. Where there are multiple occurrences of labels, the average of the normalized values is taken. Table 6.4 shows this.

TABLE 6.3 Normalized heights and weight code. Some codes, such as "f," are duplicated.

NmHt	WC
1	a
0.759	b
0.699	d
0.699	e
0.699	f
0.699	f
0.602	e
0.602	j
0.602	n
0.494	c
0.494	f
0.494	g
0.494	h
0.494	h
0.494	k
0.494	m
0.494	q

0.494	q
0.398	p
0.398	u
0.301	r
0.301	w
0.193	s
0.193	t
0.193	u
0	v

TABLE 6.4   Values of the weight codes.

Weight code	Code numeration
a	1
b	0.76
c	0.49
d	0.7
e	0.65
f	0.7
g	0.49
h	0.49

j	0.6
k	0.49
m	0.49
n	0.6
p	0.4
q	0.49
r	0.3
s	0.19
t	0.4
u	0.19
v	0
w	0.3

Since this simple example has only a single numeric dimension, the appropriate values are simply the single-dimensional representation in the table. Multidimensional examples are reduced using the multidimensional scaling techniques discussed later. Note that this example does not say that the weight labels are assigned the height values. It says instead that the appropriate normalized numeric value for a weight label is the matching normalized height value. (True in this case because there is only a single numeric variable in the system.)

Does this work? Figure 6.10 shows a graph of the actual normalized weights and the "recovered" normalized values for the labels. The fit is quite good. The correlation coefficient is about 0.85, where 0 indicates no predictive relationship at all, and 1 indicates a perfectly predictive relationship. Since this is a very small sample on only one numeric variable, this is a reasonable fit. It certainly isn't perfect, but it provides a reasonable and useful recovery of the appropriate weight label values and intervals. Naturally, with a larger sample, and with a true system of variables to draw upon, better mappings of numerating alpha values can be achieved.

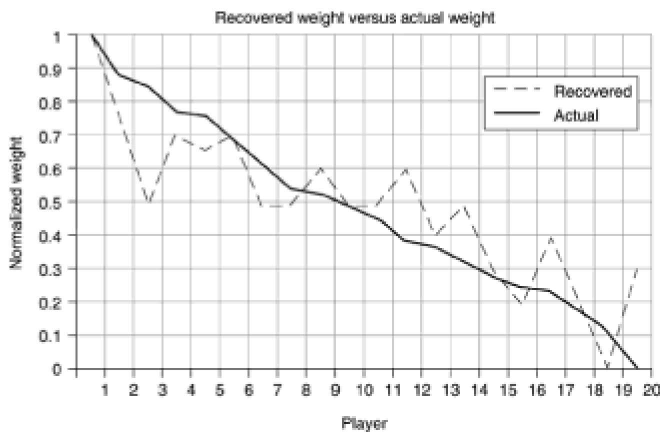


Figure 6.10 Plot of actual weights for Montreal Canadiens versus “recovered” weights. The fit is moderately good, correlation 0.85, certainly better than arbitrary assignment of numbers to the labels.

Example 2—Player Position

The variable “Position” is inherently an alpha variable. That is to say, it has no apparent inherent numeric valuation. It is exactly this sort of variable that requires appropriate numeration, and it is for these types of variables that numeration techniques are needed.

For ease of explanation, the variable “Position” will be numerated on a two-dimensional state space built from the normalized values of “Height” and “Weight.”

Plotting all of the height/weight positions shown in Table 6.5 in the state space shows the pattern, or “shape,” that each of the “Positions” makes. These positions are shown in Figure 6.11. Each of these shapes is summarized by finding its “center.” There are several ways of finding a pattern’s central location. One easy method is to find the average (mean) of the values of each label for each dimension.

TABLE 6.5 Position, normalized heights, and normalized weights for Montreal Canadiens.

Position	Height	Weight	Position	Height	Weight
Defense	1.0000	1.0000	Forward	0.4940	0.5385
Defense	0.7590	0.8769	Forward	0.4940	0.4923

Forward	0.6988	0.7692	Forward	0.4940	0.4462
Forward	0.6988	0.7538	Forward	0.4940	0.3231
Forward	0.6988	0.6923	Forward	0.4940	0.3231
Defense	0.6988	0.6923	Forward	0.3976	0.3692
Defense	0.6024	0.7538	Defense	0.3976	0.1846
Defense	0.6024	0.5231	Forward	0.3012	0.2769
Defense	0.6024	0.3846	Goal	0.3012	0.0000
Forward	0.4940	0.8462	Forward	0.1928	0.2462
Defense	0.4940	0.6923	Forward	0.1928	0.2308
Forward	0.4940	0.6154	Reserve	0.1928	0.1846
Reserve	0.4940	0.5385	Goal	0.0000	0.1077

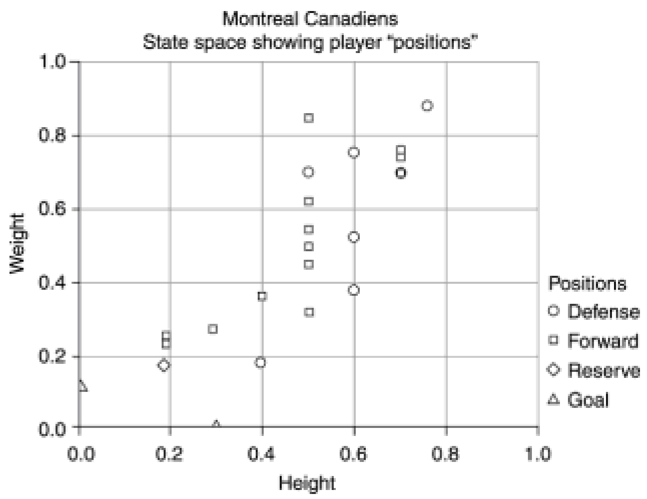


Figure 6.11 The normalized values of height and weight for each player are plotted in 2D state space. Each position type is identified. Taking the values for each alpha label type together, their outline covers an area of state space.

Using the Shape centers from Table 6.6, which are the central positions for each value (label) of the variable "Position," the variable Shape can be laid over the state space. The centers and Shape are shown in Figure 6.12. The Shape in this figure seems to be close to a straight line. Still, the points do not fall exactly on a straight line, and converting this Shape into normalized values is discussed in the section about multidimensional scaling, later in this chapter. The Shape discovered in state space is taken, placed into, and manipulated in a separate phase space.

TABLE 6.6 "Center" (mean) of each Position label.

Position	Height	Weight
Defense	0.6446	0.6385
Forward	0.4742	0.4945
Reserve	0.3434	0.3615
Goal	0.1506	0.0538

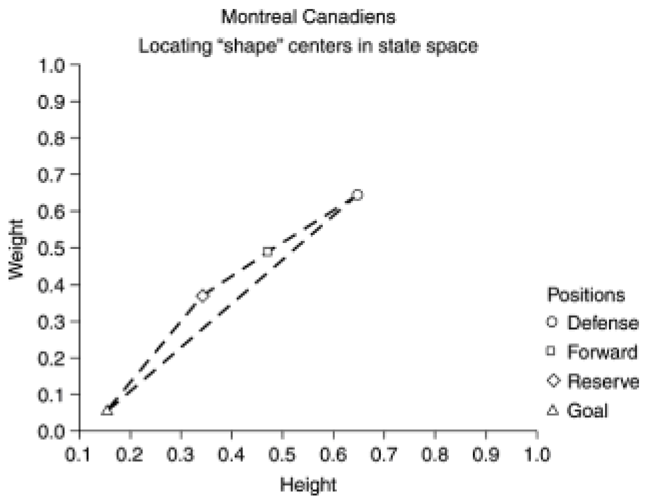


Figure 6.12 The "centers" (mean values on both dimensions) of each set of label values are located in state space. Joining the points makes a "shape." Here the shape is nearly straight line.

In this case the labels do nearly fall on a straight line. As this is the case, numerating the alpha labels can be imagined as starting with one end of the line as “0,” say with “Goal” near the zero point, and setting the other end of the line, at “Defense,” to “1.” The intervening values are set in proportion. From looking at the state space map, it seems about right to set the value of “Reserve” to about 0.4 and “Forward” to about 0.6. Usually, however, Shapes are not much like a straight line. Also, in higher dimensionalities, finding the appropriate ordering is more difficult.

6.3 Joint Distribution Tables

A different sort of problem arises if there is no numeric variable present. When there is at least one numeric variable present, it is used to set an order and spacing for the alpha variables. Without a numeric variable present, there is nothing to “calibrate” the alpha variables against. The problem is how to find any sort of logical ordering revealed by the relationships between the alpha values. The solution comes in steps. The first is to discover how the alpha values of one variable relate to the alpha values of another variable, or variables. A useful way to begin this discovery is by using a joint frequency, or joint distribution, table.

6.3.1 Two-Way Tables

A two-way table shows the joint frequency listing of alpha values between two variables. As an illustration we will return to the Montreal Canadiens. This time both height and weight will be turned into categorical values, and from these values a two-way joint frequency table is constructed. For ease of explanation, the heights are categorized as “tall,” “average,” and “short.” The weights are categorized as “heavy,” “medium,” and “light.” The categories are abbreviated “T,” “A,” “S,” and “H,” “M,” “L,” respectively.

The category boundaries are set to divide weight and height into three approximately equally sized categories as shown in Tables 6.7 and 6.8.

TABLE 6.7 Height category division criteria.

Height	Category
x6.22	T
x5.94 and x6.22	A
x5.94	S

TABLE 6.8 Weight category division criteria.

Weight	Category
x213.33	H
x191.66 and x213.33	M
x191.66	L

These three categories, or alpha labels, generate a two-way table with nine entries, one for each combination of labels. The categories for each player are shown in Table 6.9, and the cross-tabulation table is shown in Table 6.10. Figure 6.13 illustrates the distribution graphically.

TABLE 6.9 Players' names, actual height, normalized height, weights, and categories.

Name	Height	NmHt	Wt	CatHt	CatWt
Peter Popovic	6.5	1	235	T	H
Vladimir Malakhov	6.3	0.759036	227	T	H
Turner Stevenson	6.25	0.698795	220	T	H
Scott Thornton	6.25	0.698795	219	T	H
Jonas Hoglund	6.25	0.698795	215	T	H
Stephane Quintal	6.25	0.698795	215	T	H

Dave Manson	6.17	0.60241	219	A	H
Patrice Brisebois	6.17	0.60241	204	A	M
Craig Rivet	6.17	0.60241	195	A	M
Mick Vukota	6.08	0.493976	225	A	H
Zarley Zalapski	6.08	0.493976	215	A	H
Patrick Poulin	6.08	0.493976	210	A	M
Martin Rucinsky	6.08	0.493976	205	A	M
Igor Ulanov	6.08	0.493976	205	A	M
Marc Bureau	6.08	0.493976	202	A	M
Shayne Corson	6.08	0.493976	199	A	M
Vincent Damphousse	6.08	0.493976	191	A	L
Brian Savage	6.08	0.493976	191	A	L
Benoit Brunet	6	0.39759	194	A	M
Brett Clark	6	0.39759	182	A	L
Sebastien Bordeleau	5.92	0.301205	188	S	L
Jocelyn Thibault	5.92	0.301205	170	S	L
Eric Houde	5.83	0.192771	186	S	L
Mark Recchi	5.83	0.192771	185	S	L
Saku Koivu	5.83	0.192771	182	S	L
Andy Moog	5.67	0	177	S	L

TABLE 6.10 Cross-tabulation.

	H	M	L	Total
T	6	0	0	6
A	3	8	3	14
S	0	0	6	6
Total	9	8	9	26

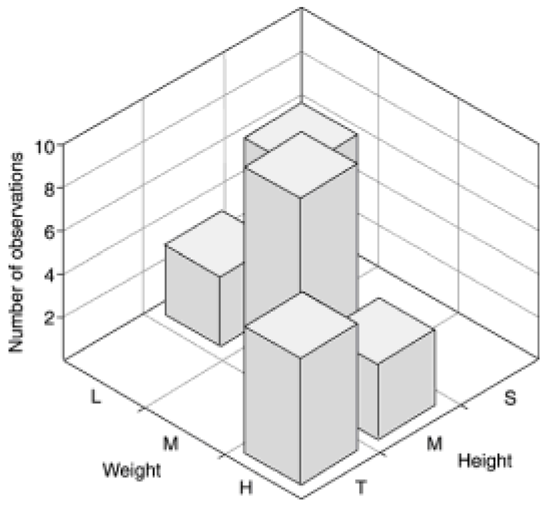


Figure 6.13 Bivariate histogram showing the joint distributions of the categories for weight and height of the Canadiens.

Notice that some of the categories overlap each other. It is these overlaps that allow an appropriate ordering for the categories to be discovered.

In this example, since the meaning of the labels is known, the ordering may appear intuitive. However, since the labels are arbitrary, and applied meaningfully only for ease in the example, they can be validly restated. Table 6.11 shows the same information as in Table 6.10, but with different labels, and reordered. Is it now intuitively easy to see what the ordering should be?

TABLE 6.11 Restated cross-tabulation.

	A	B	C	Total
X	3	3	8	14
Y	0	6	0	6
Z	6	0	0	6
Total	9	9	8	26

Table 6.11 contains exactly the same information as Table 6.10, but has made intuitive ordering difficult or impossible. It is possible to use this information to reconstruct an appropriate ordering, albeit not intuitively. For ease of understanding the previous labeling system is used, although the actual labels used, so long as consistently applied, are not important to recovering an ordering.

Restating the cross-tabulation of Table 6.10 in a different form shows how this recovery begins. Table 6.12 lists the number of players in each of the possible categories.

TABLE 6.12 Category/count tabulation.

Weight	Height	Count
H	T	6
H	A	3
H	S	0
M	T	0
M	A	8
M	S	0

L	T	0
L	A	3
L	S	6

The information in Table 6.12 represents a sort of jigsaw puzzle. Although in this example the categories in all of the tables are shown appropriately ordered to clarify explanation, the real situation is that the ordering is unknown and that needs to be discovered. What is known are the various frequencies for each of the category couplings, which are pairings here as there are only two variables. From these, the shape of the jigsaw pieces can be discovered.

Figure 6.14(a) shows the pieces that correspond to Weight = “H.” Altogether there are nine players with weight “H.” Six of them have height “T,” three of them have height “A,” and none of them have height “S.” Of the three possible pieces corresponding to H/T, H/A, and H/S, only the first two have any players in them. The figure shows the two pieces. Inside each box is a symbol indicating the label and how many players are accounted for. If the symbols are in brackets, it indicates that only part of the total number of players in the class are accounted for. Thus in the left-hand box, the top (6H) refers to six of the players with label “H,” and there remain other players with label “H” not accounted for. The lower 6T refers to all six players with height label “T.” The dotted lines at each end of the incomplete classes indicate that they need to be joined to other pieces containing members of the same class, that is, possessing the same label. The dotted lines are at each end because they could be joined together at either end. Similar pieces can be constructed for all of the label classes. These two example pieces can be joined together to form the piece shown in Figure 6.14(b).

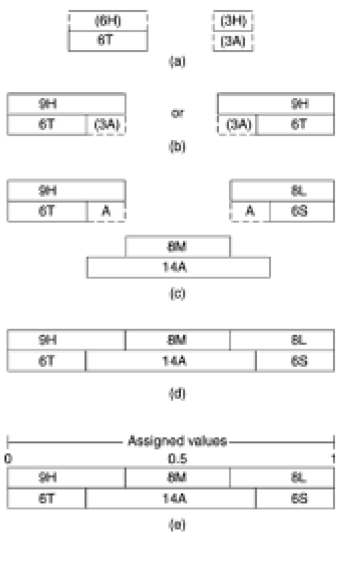


Figure 6.14 Shapes for all players with weight = "H" (a), two possible assembled shapes for the 9H/6T/3A categories (b), shapes created for each of the category combinations (c), fitting the pieces together recovers an appropriate ordering (d), and showing a straight-forward way of finding a numeration of each variable's three segments (e).

Figure 6.14(b) shows the shape of the piece for all players with Weight = "H." This is built from the two pieces in Figure 6.14(a). There are nine players with weight "H." Of these, six have height "T" and three have height "A." The appropriate jigsaw piece can be assembled in two ways; the overlapping "T" and "A" can be moved. Since the nine "H" (heavy) players cover all of the "T" (tall) players, the "H" and "T" parts are shown drawn solidly. The three "A" are left as part of some other pairing, and shown dotted. Similar shapes can be generated for the other category pairings. Figure 6.14(c) shows those.

For convenience, Figure 6.14(c) shows the pieces in position to fit together. In fact, the top and bottom sections can slide over each other to appropriate positions. Fitting them together so that the matching pieces adjoin can only be completed in two ways. Both are identical except that in one "H" and "T" are on the left, with "S" and "L" on the right. The other configuration is a mirror image.

Fitting the pieces together reveals the appropriate *order* for the values to be placed in relation to each other. This is shown in Figure 6.14(d). Which end corresponds to "0" and which to "1" on a normalized scale is not possible to determine. Since in the example there are only three values in each variable, numerating them is straightforward. The values are assigned in the normalized range of 0–1, and values are assigned as shown in Figure 6.14(e).

Having made an arbitrary decision to assign the value 0 to "H" and "T," the actual numerical relationship in this example is now inverted. This means that larger values of weight and height are estimated as lower normalized values. The *relationship* remains intact but the numbers go in the "wrong" direction. Does this matter? Not really. For modeling purposes it is finding and keeping appropriate relationships that is paramount. If it ever becomes possible to anchor the estimated values to the real world, the accuracy of the predictions of real-world values is unaffected by the direction of increase in the estimates. If the real-world values remain unknown, then, when numeric predictions are made by the final model, they will be converted back into their appropriate alpha value, which is internally consistent within the model. The alpha value predictions will be unaffected by the internal numerical representation used by the model.

Although very simplified, how well does this numeration of the alpha values work? For convenience Table 6.13 shows the normalized weights and normalized heights with the estimated valves uninverted. This makes comparison easier.

TABLE 6.13 Comparison of recovered values with normalized values.

Normalized height	Estimated height	Normalized weight	Estimated weight
1	1	1	1
0.759036	1	0.876923	1
0.698795	1	0.769231	1
0.698795	1	0.753846	1
0.698795	1	0.692308	1
0.698795	1	0.692308	1
0.60241	0.5	0.753846	1
0.60241	0.5	0.523077	0.5
0.60241	0.5	0.384615	0.5
0.493976	0.5	0.846154	1
0.493976	0.5	0.692308	1
0.493976	0.5	0.615385	0.5
0.493976	0.5	0.538462	0.5
0.493976	0.5	0.538462	0.5
0.493976	0.5	0.492308	0.5
0.493976	0.5	0.446154	0.5
0.493976	0.5	0.323077	0

0.493976	0.5	0.323077	0
0.39759	0.5	0.369231	0.5
0.39759	0.5	0.184615	0
0.301205	0	0.276923	0
0.301205	0	0	0
0.192771	0	0.246154	0
0.192771	0	0.230769	0
0.192771	0	0.184615	0
0	0	0.107692	0

6.3.2 More Values, More Variables, and Meaning of the Numeration

The Montreal Canadiens example is very highly simplified. It has a very small number of instance values and only three alpha values in each variable. In any practically modelable data set, there are always far more instances of data available and usually far more variables and alpha labels to be considered. The numeration process continues using exactly the same principles as just described. With more data and more variables, the increased interaction between the variables allows finer discrimination of values to be made.

What has using this method achieved? Only discovering an appropriate order in which to place the alpha values. While the ordering is very important, the appropriate distance between the values has not yet been discovered. In other words, we can, from the example, determine the appropriate order for the labels of height and weight. We cannot yet determine if the difference between, say, “H” and “M” is greater or less than the difference between “M” and “L.” This is true in spite of the fact that “H” is assigned a value of 1, “M” of 0.5, and “L” of 0. At this juncture, no more can be inferred from the assignment H = 1, M = 0.5, L = 0 than could be inferred from H = 1, M = 0.99, L = 0, or H = 1, M = 0.01, L = 0.

Something can be inferred about the values between variables. Namely, when normalized values are being used, both “H” and “T” should have about the same value, and “M” and “A” should have about the same value, as should “L” and “S.” This does not suggest that they share similar values in the real world, only that a consistent internal representation requires maintenance of the pattern of the relationship between them.

Even though the alpha labels are numerically ordered, it is only the ordering that has significance, not the value itself. It is sometimes possible to recover information about the appropriate separation of values in entirely alpha data sets. However, this is not always the case, as it is entirely possible that there is no meaningful separation between values. That is the inherent nature of alpha values. Steps toward recovering appropriate separation of values in entirely alpha data sets, if indeed such meaningful separation exists, are discussed in the next chapter dealing with normalizing and redistributing variables.

6.3.3 Dealing with Low-Frequency Alpha Labels and Other Problems

The joint frequency method of finding appropriate numerical labels for alpha values can only succeed when there is a sufficient and rich overlap of joint distributions. This is not always the case for all variables in all data sets. In any real-world data set, there is always enough richness of interaction among some of the variables that it is possible to numerate them using the joint frequency table approach. However, it is by no means always the case that the joint frequency distribution table is well enough populated to allow this method to work for all variables. In a very large data set, some of the cells, similar to those illustrated in Figure 6.13, are simply empty. How then to find a suitable numerical representation for those variables?

The answer lies in the fact that it is always possible to numerate some of the variables using this method. When such variables have been numerated, then they can be put into a numerical form of representation. With such a representation available in the data set, it becomes possible to numerate the remaining variables using the method discussed in the previous section dealing with state space. The alpha variables amenable to numeration using the joint frequency table approach are numerated. Then, constructing the manifold in state space using the numerated variables, values for the remaining variable instance values can be found.

6.4 Dimensionality

The preceding two parts of this chapter discussed finding an appropriate numerical representation for an alpha label value. In most cases, the discovered numeric representation, as so far discussed, is as a location on a manifold in state or phase space. This representation of the value has to be described as a position in phase space, which takes as many numbers as there are dimensions. In a 200-dimensional space, it would take a string of 200 numbers to indicate the value “gender = F,” and another similar string, with different values, to indicate “gender = M.” While this is a valid representation of the alpha values, it is hopelessly impractical and totally intractable to model. Adding 200 additional dimensions to the model simply to represent gender is impossible to deal with practically. The number of dimensions for alpha representation has to be reduced, and the method used is based on the principles of multidimensional scaling.

This explanation will use a metaphor different from that of a manifold for the points in phase space. Instead of using density to conjure up the image of a surface, each point will be regarded as being at the “corner” of a shape. Each line that can be drawn from point to point is regarded as an “edge” of a figure existing in space. An example is a triangle. The position of three points in space can be joined with lines, and the three points define the shape, size, and properties of the triangle.

6.4.1 Multidimensional Scaling

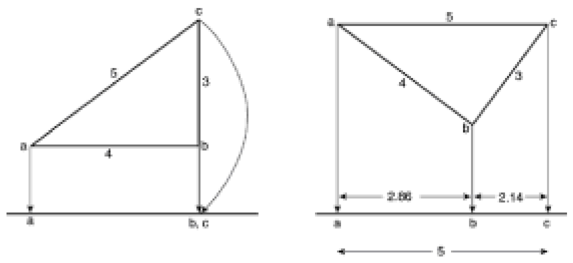
MDS is used specifically to “project” high-dimensionality objects into a lower-dimensional space, losing as little information as possible in the process. The key idea is that there is some inherent dimensionality of a representation. While the representation is made in more dimensions than is needed, not much information is lost. Forcing the representation into less dimensions than are “natural” for the representation does cause significant loss, producing “stress.” MDS aims at minimizing this stress, while also minimizing the number of dimensions the representation needs. As an example of how this is done, we will attempt to represent a triangle in one dimension—and see what happens.

6.4.2 Squashing a Triangle

A triangle is inherently a 2D object. It can be defined by three points in a state or phase space. All of the triangular points lie in a plane, which is a 2D surface. When represented in three dimensions, such as when printed on the page of this book, the triangle has some minute thickness. However, for practical purposes we ignore the thickness that is actually present and pretend that the triangle is really 2D. That is to say, mentally we can project the 3D representation of a triangle into two dimensions with very little loss of information. We do lose information about the actual triangle, say the thickness of the ink, since there is no thickness in two dimensions. Also lost is information about the actual flatness, or roughness, of the surface of the paper.

Since paper cannot be exactly flat in the real world, the printed lines of the triangle are minutely longer than they would be if the paper were exactly flat. To span the miniature hills and valleys on the paper’s surface, the line deviates ever so minutely from the shortest path between the two points. This may add, say, one-thousandth of one percent to the entire length of the line. This one-thousandth of one percent change in length of the line when the triangle is projected into 2D space is a measure of the stress, or loss of information, that occurs in projecting a triangle from three to two dimensions. But what happens if we try to project a triangle into one dimension? Can it even be done?

Figure 6.15 shows, in part, two right-angled triangles that are identical except for their orientation. The key feature of the triangles is the spacing between the points defining the vertices, or “corners.” This information, or as much of it as possible, needs to be preserved if anything meaningful is to be retained about the triangle.



Side	Original	Transform	Change
a, b	4	4	0
a, c	5	4	1
c, b	3	0	3
Total	12	8	4

Orientation 1

Side	Original	Transform	Change
a, b	4	2.86	1.14
a, c	5	5.00	0.00
c, b	3	2.14	0.86
Total	12	10	2.00

Orientation 2

	Original perimeter	New perimeter	Change	Stress
Orientation 1	12	8	4	0.333333
Orientation 2	12	10	2	0.166667

Figure 6.15 The triangle on the left undergoes more change than the triangle on the right when projected into one dimension. Stress, as measured by the change in perimeter, is 33.3% for the triangle on the left, but only 16.7% for the triangle on the right.

To project a triangle from three to two dimensions, imagine that the 3D triangle is held up to an infinitely distant light that casts a 2D shadow of the triangle. This approach is taken with the triangles in Figure 6.15 when projecting them into one dimension.

Looking at the orientation 1 triangle on the left, the three points *a*, *b*, and *c* cast their shadows on the 1D line below. Each point is projected directly to the point beneath. When this is done, point *a* is alone on the left, and points *b* and *c* are directly on top of each other. What of the original relationship is preserved here?

The original distance between points *a* and *c* was 5. The projected distance between the same points, when on the line, becomes 4. This 5 to 4 change in length means that it is reduced to 4/5 of its original length, or by 1/5, which equals 20%. This 20% distortion in the distance between points *a* and *c* represents the stress on this distance that has occurred as a result of the projection.

Each of the distances undergoes some distortion. The largest change is *c* to *b* in going from length 3 to length 0. This amount of change, 3 out of 3 units, represents a 100% distortion. On the other hand, length *a* to *b* experiences a 0% distortion—no difference in length before and after projection.

The original “perimeter,” the total distance around the “outside” of the figure was *a* to *b* = 4

b to *c* = 3

c to *a* = 5

for a total of 12. The perimeter when projected into the 1D line is

a to *b* = 4

b to *c* = 0

c to *a* = 4

for a total of 8.

So the change in perimeter length for this projection is 4, which is the difference of the before-projection total of 12 and the after-projection total of 8.

The overall stress here, then, is determined by the total amount of change in perimeter length that happened due to the projection:

change in length = 4

original length = 12

change = 4/12

or 33%. Altogether, then, projecting the triangle with orientation 1 onto a 1D line induced a 33% stress. Is this amount of stress unavoidable?

The triangle in orientation 2 is identical in size and properties to the triangle in orientation 1, except that it was rotated before making the projection. Due to the change in orientation, points *b* and *c* are no longer on top of each other when projected onto the line. In fact, the triangle in this orientation retains much more of the relationship of the distances between the points *a*, *b,* and *c*. The *a* to *b* distance retains the correct relationship to the *b* to *c* distance, although both distances lose their relationship to the *a* to *c* distance. Nonetheless, the total amount of distortion, or stress, introduced in the orientation 2 projection is much less than that produced in the orientation 1 projection. The measurements in Figure 6.15 for orientation 2 show, by reasoning similar to that above, that this projection produces a stress of 16.7%. In some sense, making the projection in orientation 2 preserves more of the information about the triangle than using orientation 1.

The important point about this example is that changing the orientation, that is, rotating the object in space, changes the amount of stress that a particular projection introduces. For most such objects this remains true. Finding an optimal orientation to reduce the stress of projection is important.

6.4.3 Projecting Alpha Values

How does this example relate to dimensionality reduction and appropriate representation for alpha labels?

When using state space to determine values for alpha labels, the method essentially finds appropriate locations to place the labels on a high-dimensionality manifold. Each label value has a more or less unique position on the manifold. Between each of these label locations is some measurable distance in state space. Using the label positions as points on the manifold, distances between each of the points can easily be discovered using the high-dimensional Pythagorean theorem extension. These points, with their distances from each other, can be plucked off the state space manifold, and the shape represented in a phase space of the same dimensionality. From here, the principle is to rotate the shape in its high-dimensional form, projecting it into a lower-dimensionality space until the minimum stress level for the projection is discovered. When the minimum stress for some particular lower dimensionality is discovered, if the stress level is still acceptable, a yet lower dimensionality is tried, until finally, for some particular lower dimensionality, the stress becomes unacceptably high. The lowest-dimensionality representation that has an acceptable level of stress is the one deemed appropriate to represent the alpha variable. (What might constitute an acceptable level of stress is discussed shortly.)

6.4.4 Scree Plots

The idea that stress changes with projection into lower numbers of dimensions can actually be graphed. If a particular shape is projected into several spaces of different dimensionality, then the amount of stress present in each space, plotted against the number of dimensions used for the projection, forms what is known as a *scree plot*. Figure 6.16 shows just such a plot.

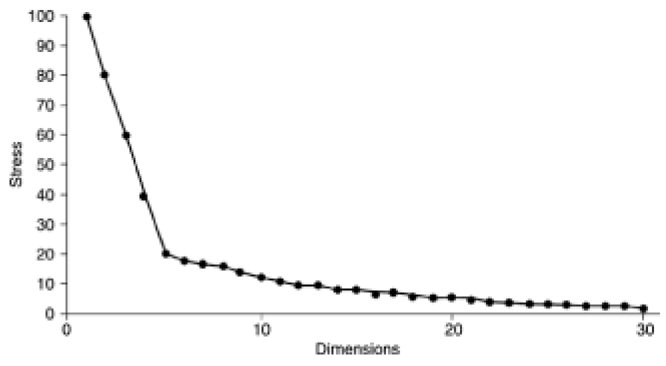


Figure 6.16 Ideal scree plot.

Starting with 30 dimensions in Figure 6.16, a high-dimensional figure is projected into progressively fewer dimensions. Not much change occurs in the level of stress occasioned by the change in dimensionality until the step from five to four dimensions. At this step there is a marked change in the level of stress, which increases dramatically with every reduction from there.

The step from five to four dimensions is called a *knee*. In dimensionalities higher than this knee, the object can be accommodated with little distortion (stress). Clearly, four dimensions are not sufficient to adequately represent the shape. It appears, from this scree plot, that five is the optimum dimensionality to use. In some sense, a five-dimensional representation is the best combination of low dimensionality with low stress.

When it works satisfactorily, finding a knee in a scree plot does provide a good way of optimizing the dimensionality of a representation. In practice, few scree plots look like Figure 6.16. Most look more like the ones shown in Figure 6.17. In practice, finding satisfactory knees in either of these plots is problematic. When satisfactory knees cannot be found, a workable way to select dimensionality is to select some acceptable level of stress and use that as a cutoff criterion.

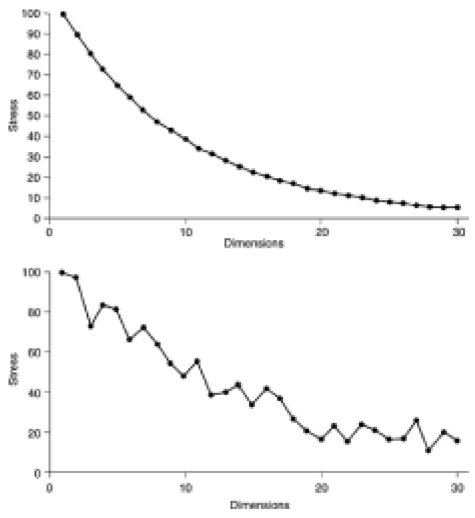


Figure 6.17 Two more realistic scree plots.

6.6 Summary

This chapter has covered a lot of ground in discussing the need for, and method of, finding justifiable numeric representations of alpha-valued variables. The concepts of methods for performing this numeration in mixed alpha-numeric and in entirely alpha data sets was discussed in detail. In all cases the information carried in the data set was used to reflect appropriate values and ordering for the individual alpha values.

We started by looking at ways that the miner can apply domain knowledge to remap alpha values to avoid problems that automated methods cannot solve alone. The conceptual groundwork of state space was discussed and this metaphor explored for its utility in representing the measured states of a system of variables, in addition to its value in numerating alpha variables. We examined the nature and features of the data representation in state space. Translating the information discovered there into insights about the data, and the objects the data represents, forms an important part of the data survey in addition to its use in data preparation. Several practical issues in providing a working data preparation computer program were also addressed.

In spite of the distance covered here, there remains much to do to the data before it is fully prepared for surveying and mining.

Chapter 7: Normalizing and Redistributing Variables

Overview

From this point on in preparing the data, all of the variables in a data set have a numerical representation. Chapter 6 explained why and how to find a suitable and appropriate numerical representation for alpha values—that is, the one that either reveals the most information, or at least does the least damage to existing information. The only time that an alpha variable's label values come again to the fore is in the Prepared Information Environment Output module, when the numerical representations of alpha values have to be remapped into the appropriate alpha representation. The discussion in most of the rest of the book assumes that the variables not only have numerical values, but are also normalized across the range of 0–1. Why and how to normalize the range of a variable is covered in the first part of this chapter.

In addition to looking at the range of a variable, its distribution may also make problems. The way a variable's values are spread, or distributed, across its range is known as its *distribution*. Some patterns in a variable's distribution can cause problems for modeling tools. These patterns may make it hard or impossible for the modeling tool to fully access and use the information a variable contains. The second topic in this chapter looks at normalizing the distribution, which is a way to manipulate a variable's values to alleviate some of these problems.

The chapter, then, covers two key topics: normalizing the range of a variable and normalizing the distribution of a variable. (Neither of these normalization methods have anything in common with putting data into the multitable structures called "normal form" in a database, data warehouse, or other data repository.) During the process of manipulation, as well as exposing information, there is useful insight to be gained about the nature of the variables and the data. Some of the potential insights are briefly discussed in this chapter, although the full exploration of these relationships properly forms part of the data survey.

7.1 Normalizing a Variable's Range

Chapter 6, discussing state space, pointed out that it was convenient to normalize variable ranges across the span of 0–1. Convenience is not an attribute to be taken lightly. Using anything less than the most convenient methods hardly contributes to easy and efficient completion of a task. However, some modeling tools *require* the range of the input to be normalized. For example, the neurons in most neural-network-based tools require data to be close to the range of 0 to 1, or –1 to +1, depending on the type of neuron. (More on neural networks in Chapter 10.) Most tools that do not actually require range normalization may benefit from it, sometimes enormously. (Chapter 2 mentioned, for instance, that exposing information and easing the learning task can reduce an effect known as *feature swamping*.)

Normalization methods represent compromises designed to achieve particular ends. Normalization requires taking values that span one range and representing them in another range. This requires remapping values from an input range to an output range. Each method of remapping may introduce various distortions or biases into the data. Some biases and distortions are deliberately introduced to better expose information content. Others are unknowingly or accidentally introduced, and damage information exposure. Some types of bias and distortion introduced in some normalization processes are beneficial only for particular types of data, or for particular modeling methods. Automated data preparation must use a method that is generally applicable to any variable range and type—one that at least does no harm to the information content of the variable. Ideally, of course, the normalization method should be beneficial.

Any method of addressing the problems has its own trade-offs and introduces biases and distortions that must be understood. Some commercial tools normalize variables. When they do, it can cause a problem if the tool uses a default method that the modeler cannot control. Exactly what might be lost in the normalization, or what distortion might be introduced, is hard to know if the normalization method is not in the modeler's control, or worse, not even known to the modeler. (The neural network model comparison between prepared data and "unprepared" data in Chapter 12 in part demonstrates this issue.)

Methods of normalization are plentiful. Some do more than one thing at a time. They not only normalize ranges, but also address various problems in the distribution of a variable. The data preparation process, as described in this book, deals with distribution problems as a separate issue (discussed later in this chapter), so normalization methods that adjust and correct simultaneously for range and distribution problems are not used. As far as range normalization goes, what the modeler needs is a method that normalizes the range of a variable, introducing as little distortion as possible, and is tolerant of out-of-range values.

Range normalization addresses a problem with a variable's range that arises because the data used in data preparation is necessarily only a sample of the population. (Chapter 5 discussed sampling.) Because a sample is used, there is a less than 100% confidence that the sample is fully representative of the population. This implies, among other things, that there is a less than 100% confidence that the maximum and minimum values of the range of a variable have been discovered. This in turn implies, with some degree of confidence, that values larger than the sample maximum, or smaller than the sample minimum, will turn up in the population—and more importantly, in other samples of the population. Since values that are outside the limits discovered in a sample are out of the range of the sample, they are called here *out-of-range* values. This only indicates that such values are out of the range discovered in the sample used for data preparation. They certainly aren’t out of the range of the population, only out of the range established in a particular sample—the training sample. Dealing with these out-of-range values presents a problem that has to be addressed. We need to look at what these problems are before considering how to fix them.

What problems turn up with out-of-range values? The answer to this question depends on the stage in the data exploration process in which the out-of-range value is encountered. The stages in which a problem might occur are during modeling: the training stage, the testing stage, and the execution stage. Preparation and survey won’t come across out-of-range values as they work with the same sample. The modeling phase might have problems with out-of-range values, and a brief review of modeling stages will provide a framework to understand what problems the out-of-range values cause in each stage.

7.1.1 Review of Data Preparation and Modeling (Training, Testing, and Execution)

Chapter 3 described the creation, use, and purpose of the PIE, which is created during data preparation. It has two components: the PIE-Input component (PIE-I) that dynamically takes a training-input or live-input data set and transforms it for use by the modeling tool, and the PIE-Output component (PIE-O) that takes the output (predictions) from a model and transforms it back into “real-world” values. A representative sample of data is required to build the PIE. However, while this representative sample *might* be the one also used to build the (predictive, inferential, etc.) model, that is not necessarily so. The modeler may choose to use a different data set for modeling, from the one used to build the PIE. Creating the model requires at least training and testing phases, followed by execution when the model is applied to “live” data.

This means that there are potentially any number of sample data sets. During training, there is one data set for building the PIE, one (probably the same one) for building a model, and one (definitely a separate one) for testing the model. At execution time, any number of data sets may be run through the PIE-I, the model, and the PIE-O in order, say, to make predictions. For example, in a transaction system scoring individual transactions for some feature, say, fraud, each transaction counts as an input execution data set. Each transaction is separately presented to the PIE-I, to the scoring model, the results to the PIE-O, with the individual output score being finally evaluated, either manually or automatically. The transactions are not prepared as a batch in advance for modeling all together, but are individually presented for evaluation as they come in.

When building the PIE, it is easy to discover the maximum and minimum values in the sample data set. So no out-of-range values can occur when building the PIE. With any other sample data set, it is always possible to encounter an out-of-range value. Since the PIE provides the modeling environment, it is the PIE that must deal with the problems.

7.1.2 The Nature and Scope of the Out-of-Range Values

Problem

Since the PIE knows the maximum and minimum values of the data sample, no out-of-range value can occur at this stage during its construction. However, what the modeler should ask is, What can I learn about the out-of-range values that are *expected* to occur in the population? The PIE is going to have to deal with out-of-range numbers when they turn up, so it needs to know the expected largest and smallest numbers it will encounter during execution. It is also useful to know how often an out-of-range number is likely to be found in the population.

There are two problems with out-of-range numbers. First, the PIE is not going to have any examples of these values, so it needs to estimate their range and frequency to determine suitable adjustments that allow for them. They are certain to turn up in the population, and the PIE will have to deal with them in some way that best preserves the information environment surrounding the model. The second problem is that the out-of-range values represent part of the information pattern in the population that the modeling tool is not going to be exposed to during training. The model can't see them during training because they aren't in the training sample. The modeler needs an estimate of the range and the proportion of values in the population that are not represented in the sample. This estimate is needed to help determine the model's range of applicability and robustness when it is exposed to real-world data. Clearly, the model cannot be expected to perform well on patterns that exist in the population when they are not modeled since they aren't in the training sample. The extent and prevalence of such patterns need to be as clearly delimited as possible.

Of course, the modeler, together with the domain expert and problem owner, will try to choose a level of confidence for selecting the sample that limits the problem to an acceptable degree. However, until a sample is taken, and the actual distribution of each variable sampled and profiled, the exact extent of the problem cannot be assessed. In any case, limiting the problem by setting confidence limits assumes that sufficient data is available to meet the confidence criteria chosen. When the training data set is limited in size, it may well be the amount of data available that is the limiting factor. In which case, the modeler needs to know the limits set by the data available. Unless the population is available for modeling, this is a problem that simply cannot be avoided.

The information about the model limits due to out-of-range values, although generated when creating the PIE modules, is generally reported as part of the data survey. It is important to note that although the information enfolded in the data in the out-of-range values is properly part of the population, the model will experience the previously unseen values as noise. Chapter 11 looks briefly at noise maps. A full survey assesses, where possible, how much noise comes from each measurable source, including out-of-range values. Unfortunately, space limitations preclude further discussion of methods for assessing noise contribution due to out-of-range values, and for separating it from noise from other sources.

7.1.3 Discovering the Range of Values When Building the PIE

How, then, does the miner determine the range and the frequency of values present in the population, but not in the sample? Recall that the data sample was determined to represent the population with a specific level of confidence. That confidence level is almost always less than 100%. A 95% confidence means that there remains a 5% confidence—that is, 1 in 20—that the sample is not representative. It doesn't need detailed analysis to see that if the range has been captured to a 95% confidence limit, out-of-range values must be quite commonly expected. Two separate features vary with the actual confidence level established. The first is the frequency of occurrence of out-of-range values. The second is the expected maximum and minimum values that exist in the population. To see that this is so, consider a population of 100 numbers ranging uniformly from 0 to 99 without duplication. Take a random sample of 10. Consider two questions: What is the chance of discovering the largest number in the population? and What is the largest value likely to be?

Probability of Discovery of Largest Value

Since there are 100 numbers, and only one can be the greatest, on any one random pick there is 1 chance in 100 that the largest number is found. Choosing 10 numbers, each selected at random, from 100 gives 10 chances in 100 for picking the largest number.

By similar reasoning, the chance of finding the largest value in a random sample of, say, 20, is 20%, as shown in Table 7.1.

TABLE 7.1 Probability of finding largest value for several numbers of picks.

Number of picks	Probability in %
1	1
2	2
5	5
10	10
15	15

Most Likely High and Low Values

But what is the largest *value* likely to be found? When making the random pick, any values at all could be chosen, each being equally likely. In this example, 10 numbers from 100 are selected (10% of the population), so every number in the population has a 10% chance of being chosen. But what is the most likely value to pick?

Imagine if numbers are selected one at a time at random and a running average of the values picked is kept. Since any number is as likely to be picked as any other, the running average is simply going to approach the average value of all the numbers in the population. If picking continues long enough, all of the numbers are chosen with equal frequency. Added together and divided by the number of picks, the result is the population average value.

In this example, the mean value of the population is 50. Does this mean that 50 is the most likely number to pick? Not exactly. There is only a 1% chance of actually choosing the value 50 in any single pick. If 10% of the population is chosen, the number 50 has a 10% chance of being in the sample. However, what it can be interpreted to mean is that if the choice of one number at random were repeated many times, the numbers chosen would seem to cluster around 50. (There would be as many values of 50 and above as there are below 50, and, on average, they would be as far above as below.) In this sense, 50 indicates the center of the cluster, and so measures the center of the place where the numbers tend to group together. That, indeed, is why the mean is called a “measure of central tendency” in statistics.

What happens when two numbers are picked, paying attention to which is larger and which is smaller? With two numbers selected, it is certain that one is larger than the other (since the population comprises the numbers 1 through 100 without duplicates). By reasoning similar to the single-number pick, the upper value will tend to be halfway between the lower value picked (whatever that is) and the largest number available (100). Similarly, the lower value will tend to be halfway between the higher value picked (whatever that is) and the lowest number available (1). So the two numbers picked will split the range into three parts. Because each value has a tendency to be as far as it can both from its neighbor, and from the extreme values in the range (1 and 100), the separations will be equal in size. In other words, the tendency for two numbers will be to split the range into three equal parts. In this example, for two choices, the expected values are about 33 and 67.

This reasoning can be extended for any number of picks where the order of the picked values is noted. The expected values are exactly the points that divide the range into *n* + 1 equally sized subranges (where *n* is the number of picks).

Table 7.2 shows the expected high and low values for a selection of numbers of picks. As the sample size increases, the expected value of the highest value found gets closer and closer to the maximum value of the population. Similarly, with increased sample size, the expected value of the lowest value found in the sample approaches the low value in the population.

TABLE 7.2 Expected values for various choices.

Number of picks	Expected low value	Expected high value
1	50	50
2	33	67
5	17	83
10	9	91
15	6	94
20	5	95

In the example, the population's extreme values are 1 and 100. Table 7.2 shows how the expected high and low values change as the number of picks changes. As the sample size increases, indicated by the number of picks, so the difference between the expected values and the extreme values in the population gets smaller. For instance, the upper-range difference at 10 picks is 100 – 91 = 9, and at 20 picks is 100 – 95 = 5. The lower range difference at 10 picks is 9 – 1 = 8, and at 20 picks is 5 – 1 = 4. (The apparent difference in the upper and lower range is due to rounding off the values. The upper and lower expected values are actually symmetrically located in the range.)

Out-of-Range Values and the PIE

The examples just given are considerably simplified. For real-world variables with real-world distributions, things are far more complex. Actual probabilities and expected values depend very much on the true distribution of a variable, among other things. This is, in any case, complicated by the fact that distributions may change over time. This example is true for a rectangular distribution (one in which every value that can occur does so with equal probability) and where no values are duplicated. In the example, the size of the population was also known, which makes determining probabilities easy.

While the probabilities vary considerably with distribution, population size, and other factors, the principles do not:

Some maximum and minimum values will be detected in a sample.

- The discovered maximum and minimum define the range of the sample.
- In the population there is always some chance of encountering an out-of-range value.
- Some specific confidence that the sample is representative of the population can be determined.
- The smaller the chance that the sample is representative, the larger the chance of encountering an out-of-range value, and the larger the gap is likely to be between sample range limit and the population limit.

That is, the less representative the sample, the more chance there is of encountering an out-of-range value in the population. And when an out-of-range value is found, the less representative the sample, the greater the expected difference will be between the sample maximum or minimum and the out-of-range value.

Knowing the confidence level that the data sample is representative does give some indication of how likely an out-of-range value is to be found in the population, and how large the gap between the detected limits and the out-of-range value might be. Having these estimates enables the normalization process to be adjusted to take account of the expected frequency of out-of-range values and the expected true range of the population values. For instance, if the sample confidence is relatively low, then many out-of-range values covering a large range can be expected. If sample confidence is high, few out-of-range values will be expected, and those that are will cover a narrower range.

To summarize: The representative sample selected to create the PIE has, for every variable, high and low values. When building the PIE, no values exceeding this range will be found, since it is the sample that produced the maximum and minimum values. The confidence level that the sample is representative gives an indication of the probability of meeting out-of-range values. The confidence level also indicates the probable size of the gap between discovered maximum or minimum, and any out-of-range value. With the frequency of occurrence and the gap size estimated, the normalization process in the PIE can be constructed accordingly.

7.1.4 Out-of-Range Values When Training

During training, the PIE is already built and in place. The PIE-I takes raw data values from the training sample and translates them into a prepared state for use by the modeling tool. What happens when the PIE-I finds an out-of-range value? As yet there has been no discussion of a method of dealing with out-of-range values. What would happen if out-of range values are not normalized into a 0–1 range, but passed through outside the normalized range? That is, an input value larger than the sample maximum would translate into a value larger than 1, perhaps 1.2. Similarly, input values smaller than the sample minimum will translate into values less than 0, maybe –0.2. (The purpose of the discussion, of course, is to examine the problems that could occur to discover how best to avoid them.)

Consequences of Ignorance I

One "solution" to an out-of-range value (adopted by some commercial modeling tools) is to simply ignore the whole instance (record) if any one of the values is out-of-range. This also takes care of the missing-value problem. (Missing values are treated as out-of-range too.) This effectively reduces the size of the sample by ignoring any data points that do not fit within the specified parameters. There are two notable problems with this approach.

The first, and less significant, problem is that reducing the number of instances in the sample reduces the level of confidence that the sample represents the population. Discarding instances is literally discarding information! Discarding, or ignoring, instances effectively reduces the size of the training set. A model created using the reduced training set cannot be as effective as one built with a more representative data set. If this were the only problem, it is easily remedied by adding more data to the training set if it is available. Adding more data again increases confidence that the sample is representative. On the other hand, if more data is not available, it may be that the information in the discarded instances is much needed, and discarding them is damaging to training.

A second, and potentially more serious problem, is introducing bias. Unless the out-of-range values occur in a truly random pattern, then obviously they do not occur at random. If they do not occur at random, then they must occur with some sort of pattern. Deleting or ignoring out-of-range instances then necessarily removes them according to some pattern. Removing instances in a pattern prevents the modeling tool from seeing the pattern. This removal of a pattern from the sample introduces distortion, or bias, to the sample. The bias can be anything from slightly damaging to disastrous—with no way to determine which! This problem is so potentially severe and undetectable that attempts must be made to avoid it at all costs.

Imagine (as really happened) using such a tool for building a model of mortgage applicants. The training sample had applicants with salaries up to, say, $100,000. When the model was run, this method ignored all applicants with salaries greater than $100,000. But the null score was interpreted as no score, and the mortgage company interpreted no score as a bad score! Until discovered (which didn't take long), this method of dealing with out-of-range variables was (to say the least) problematic. In practice, of course, it rendered the model virtually useless.

Consequences of Ignorance II

Another approach ignores the fact that the normalized range has been exceeded. It says, "Let normalized values fall outside the range if necessary." The assumptions about state space being *unit* state space will no longer hold, but this is not always a major concern since state space may only be a conceptual device for many modeling methods. Most modeling tools have at least some capacity to handle numbers outside the normalized range. But how do they handle them? And does it make a difference to the quality of the model?

Some methods do use a unit state space model. Where this is the case, these will have to deal with the out-of-limit values in a way that keeps them inside unit state space. One method is to "clip" the values that fall outside the range. If greater than 1, assign 1. If less than 0, assign 0. The problem with this method is the underlying assumption that numbers that fall outside the range are in some way equivalent to numbers that fall at the limit of the range. This ignores the fact that the numbers falling outside the range *are* in some way different and carry information to that effect. This vital information is thrown away.

Worse than throwing information away is what happens to the limit values if there is a difference that the model should reflect between limit values and out-of-range values. The limit value's information content is distorted because the model will not be able to distinguish between range limit and out-of-range values. The range limit value meaning will have to be distorted to reflect whatever aggregate meaning the out-of-range values carry, too. Projecting the information content from several values onto a single value distorts the information content of the limit value.

For example, if the out-of-range values extend up to 1.2, the range top value of 1 has to carry an "average" meaning of all the values from 1 to 1.2. Any difference that the model should reflect when the value is, say, 1.1 is lost, merged, as it were, with the meaning carried by the range top value of 1. But worse, if the model is predictive, for instance, when the input value is actually 1, the model will have to predict the "average" response of values 1 through 1.2.

Once again, the problem of bias shows up. If the occurrence of out-of-range values is not in fact random, using exactly the same argument as in the previous section, undetectable bias is introduced into the model. Just as before, the problems this introduces can range from innocuous to disastrous. Bias can invalidate the best model.

In some models, for instance, fraudulent activity falls into this out-of-range category. It is the fraudulent activity that may fall out of the modeled range, since new patterns of fraud constantly evolve. If the fraudulent activity moves some variable instance values out of their limits, and the model is constrained to ignore it, or to “merge” it with other activity, this new activity is indicated as equivalent to whatever the model found to be the activity at the range limit. This may easily be an unjustified assumption.

In one case, this “merging” behavior persuaded one model of insurance claims to assume that all building fires occurring after 9:30 at night, and started in rear rooms, scored pretty well as likely arson! In fact, this model made a number of other erratic inferences, all due to the nature of the insurance claim data set modeled and the tool used.

7.1.5 Out-of-Range Values When Testing

When testing models, many of the same problems occur as when training. Testing attempts to discover the applicability of, and limits to, the model. Whether or not the training phase experienced out-of-limit values, if no correction or allowance is made for them, their presence during testing will be dealt with in a similar, cavalier way. In one way or another they will be either ignored or clipped. For all of the reasons discussed above, this will produce a less accurate model output. (The actual output will be numeric, although the final result might be inferences, predictions, or come in some other form, depending on the type of model.)

Testing the model in ways that underestimate its limits and utility is not necessarily damaging, but will lead at least to having less confidence in the model than is perhaps justified. It will certainly help in making, to some extent, erroneous conclusions about the range, utility, and applicability of the model.

However, the model might also appear to be better and more robust than is actually the case. Ignoring instances of data in the test data set because they have out-of-range values, for instance, clearly means that the model is not tested on them. But these are the precise areas in which the model might perform most poorly, and its performance in these areas has to be included in any valid overall performance summary.

7.1.6 Out-of-Range Values When Executing

Execution is the time when a predictive model is predicting, an inferential model is inferring, a self-adaptive model is adapting, and so on. Whatever else went before, this is the time when out-of-range values are most likely to appear if they ever will! This is the phase of the data exploration project when the model is likely to be exposed to copious quantities of data, and so has the highest expectation that the fullest range of the data will appear. (It is also the time when real, applicable, and useful results are expected.) For simplicity of discussion, a predictive model will be assumed. The same principles hold for any type of model—predictive, inferential, adaptive, and so on.

A model created by training on data biased by removing problematic instances from the training data will almost certainly still be required to produce predictions for similar problematic instances in the execution data. If predicting fraud, for instance, all instances must be examined. If predicting customer segments, all customers must be predicted. The model is not considered adequate if no predictions are made for instances with problematic data. (Even people earning more than $100,000 may be good mortgage risks!) But if out-of-range values were excluded during training, the model was not exposed to such data during training. There is no reference for making a valid prediction from such data during the execution phase. In any case, the model will be more or less biased, having been trained on biased data. The execution data that the model is required to perform on will not be biased. Whatever bias is included in the model will result in biased predictions.

If, on the other hand, the out-of-range values were “trimmed” off to the limiting values during training, when the model does experience such values, they will have to be trimmed again, leading to poor predictions for any limiting conditions.

Possibly the worst scenario is that untrimmed variable values are allowed into the model. When this happens, the model is driven outside of the range of data on which it trained. In this case the model will, of course, produce predictions, but predictions that are based on no evidence. When the model is driven into areas that are outside the boundaries of the state space on which it trained, almost no valid predictions can be made. We can speculate, for instance, about the weight of 20-foot-tall human beings. Whatever extrapolation we might make, the truth is that there is no evidence to base a prediction on, for such a creature probably could not exist. Whether or not such a being could exist, and what its weight might be, is pure speculation. So it is too when a model is driven beyond the limits on which it trained.

Clipping values leaves the model no way to detect if the instance values are changing—at least at the limits of behavior. It is often the case that the distribution of the data is not stationary. Nonstationarity of a distribution simply means that the distribution does not remain constant, usually over time. The user of the model needs to monitor this, among many other things, during run time anyway. This is not a part of data preparation, but the preparation technique should at least provide support to make the monitoring easier.

7.1.7 Scaling Transformations

The discussion so far has looked at the issues surrounding finding the maximum and minimum values for each variable in a sample. Clearly, knowing the maximum and minimum values somehow allows the actual value to be scaled, or normalized, into the range 0–1. A way of doing this is to use a transforming expression that takes the input value, and, knowing the maximum and minimum values, squashes the input value into the required output range. An easy way to actually do this is with the *linear scaling transform*.

The actual expression is very straightforward:

$$v_n = \frac{v_i - \min(v_1 \ldots v_n)}{\max(v_1 \ldots v_n) - \min(v_1 \ldots v_n)}$$

where

v_n	is normalized value
v_i	is instance value

This expression takes any value and transforms it into another number. If the input value is inside the limits, the output will be between 0 and 1. Any value outside the limits will fall outside the 0–1 range, presenting a modeler with all of the problems just discussed.

Using Linear Scaling for Normalization

Although many of the problems of dealing inadequately with out-of-range values have been discussed, and the simplest method of normalizing values has been found wanting, it is still the place to start. Linear scaling is a simple, straightforward technique to use for normalizing the range of numeric values. Its big advantage is that it introduces no distortion to the variable distribution. It involves only discovering the maximum and minimum values for the range of the variable, and then finding where within the range a particular instance value falls. The formula for achieving this is given above. Given this formula, any instance value can be plugged in, and a normalized value computed. There is a one-to-one relationship between the original instance value and the normalized value. Given two instance values, with the first being twice the second, when they are normalized, the first normalized value will still be twice the second. This is true wherever in the range of the variable the two instance values occur.

The relationship between the instance values and the normalized values is called linear because if the two sets of values are plotted on a graph, the result is a straight line—as shown in Figure 7.1.

Number	Normalized
40.12	0.56
31.29	0.42
43.85	0.63
40.60	0.57
54.21	0.80
46.94	0.68
28.06	0.36
60.97	0.91
57.39	0.85
66.09	1.00
6.45	0.00
9.53	0.05
52.83	0.78
47.44	0.69
21.00	0.24
43.31	0.62
14.40	0.13
37.42	0.52
16.47	0.17
61.51	0.92

Maximum	66.09
Minimum	6.45
Difference	59.64

$$v_{norm} = \frac{v_i - \min(v_1 \ldots v_n)}{\max(v_1 \ldots v_n) - \min(v_1 \ldots v_n)}$$

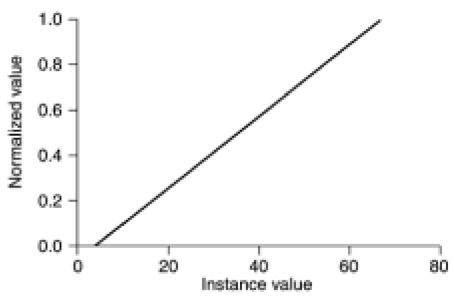


Figure 7.1 Linear scaling produces a linear relationship between instance values and normalized values.

Linear scaling works well when the maximum and minimum values are known. During data preparation, this presents a problem. The maximum and minimum values of the *sample* are known, but the true population maximum and minimum may be unknowable. As just discussed, when using the model with real-world data, it is very likely that instance values outside the sample range will be encountered. Linear scaling normalization will translate these values into numbers that fall outside the 0–1 range.

In spite of its shortcomings, which only occur at the limit of the range, linear scaling has a great strength in that it introduces no distortion in the translated values. Whatever information is in the original values is preserved unmodified in the normalized values. This is an important feature that needs to be preserved, if at all possible. If the problems that occur at the limits can be dealt with, linear scaling works well.

Making Room for Out-of-Range Values

In order to deal with the out-of-range problems, the PIE needs a method of dealing with the limit problems of linear scaling. The linear scaling transformation gives linear normalization over all of its range, but must be modified to somehow include out-of-range values. One way to do this is to reduce the part of the transformed range that holds in-range values. There is then room to squeeze the out-of-range values into space left at the upper and lower ends, still leaving some differentiation between them. But how can this be done? Theoretically, there is some chance, however small, that an arbitrarily far out-of-range number will be encountered at either end of the range. How can what is potentially an infinite range of numbers be squashed into a finite part of a 0–1 range, especially with most of the 0–1 range given over to linear scaling? Fortunately, exactly such a transform does exist, and it forms the basis of *softmax scaling*. This key transform is called the *logistic function*. Both softmax scaling and the logistic function are examined shortly. But first, what exactly is it that needs to be done?

The optimal form of normalizing transformation, if it could be guaranteed never to go out of range, is linear. Of course, that's just the problem—it can be guaranteed to go out of range with some degree of confidence. However, if we can measure, or make assumptions about, the distribution of the variable, we can then make inferences about the distance between the sample limit and the population limit. Doing this allows choosing some appropriate part of the range to be linear—and some appropriate part to accommodate the out-of-range values.

Since the idea is that the translation is linear over some part of the range, the question is, how much of the range should be linear? The sample being used for building the PIE is selected with some degree of confidence. It is this confidence that can be used to determine what part of the 0–1 range is to be held linear. The size of the expected out-of-range gap is directly proportional to the degree of confidence that there will be out-of-range values. If, for example, the selected confidence level was 98%, then 98% of the range 0–1 will be linear. The selected linear part of the range is centered, so that the linear translation range becomes, for a 98% confidence level, 0.01–0.99. The linear part of the range is squashed by 2%. The balance from the 98%, or 2%, is evenly spaced at the top and bottom of the range. This leaves 1% at each end of the range for squeezing in the out-of-limit numbers. Figure 7.2 illustrates squashing the linear part of the range.

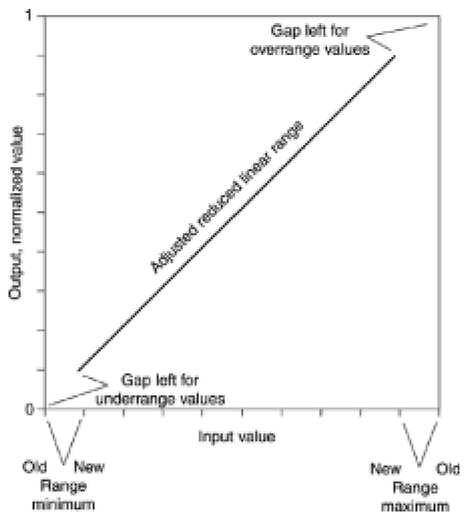

1
0
Gap left for overrange values
Adjusted reduced linear range
Gap left for underrange values
Output, normalized value
Input value
Old
New
Range minimum
New
Old
Range maximum

Figure 7.2 The linear part of the range is "compressed" so that it covers a smaller part of the output range. The "gaps" left at the top and bottom allow space in which to compress the out-of-range values.

Squashing the Out-of-Range Numbers

The problem that now remains is to fit the out-of-range numbers, potentially extending to infinity, into the minute space left for them. Consider the upper limit. First, it is important to realize that for numbers larger than the limit, the greater a number, the less likely it is that any such value will be found. When the sample was originally taken, some degree of confidence was established that the largest value had been found. Larger numbers are possible, even likely, but, as previously discussed, the greater the difference between the limit value and any larger value, the less likely it is that it will be encountered.

The transformation is made such that as the difference between the limit and the out-of-limit value grows, the smaller the increase toward the end of the range. Larger numbers produce proportionally smaller differences, and an infinitely large number produces ultimately infinitesimally small differences. In Chapter 6 it was noted that by increasing precision, it is always possible to indicate more locations on the number line. This allows an infinite number of out-of-range numbers to be mapped into space left for them. If such a transform is developed, it can be used to squash the out-of-range values above and below the linear part into the space left for them.

Looking at the upper range, a mathematical function is needed such that as the difference between the limit and overlimit values gets larger, the value increases toward, but never reaches, some boundary. Whatever its limits, the output of the squashing function can itself then be linearly squashed to fit into the gap left for it. In looking for such a function, a reciprocal makes a good starting point. A reciprocal of a number is simply one divided by the number. It starts with a value of one, and as the input number gets larger, the output value gets smaller and smaller, reaching toward, but never getting to, 0. To have this transform move in the opposite direction, subtract it from 1. It becomes 1 – 1/*v.* Table 7.3 shows the output values for various inputs.

TABLE 7.3 Values of upper-range squashing function.

v	1/*v*	1 – 1/*v*
1	1.000	0.000
2	0.500	0.500
3	0.333	0.667
5	0.200	0.800
8	0.125	0.875
13	0.077	0.923

21	0.048	0.952
34	0.029	0.971
55	0.018	0.982
89	0.011	0.989

So 1 – 1/*v* starts at 0 and moves toward 1, never quite reaching it, regardless of how large a number is input. This is shown graphically in Figure 7.3(a).

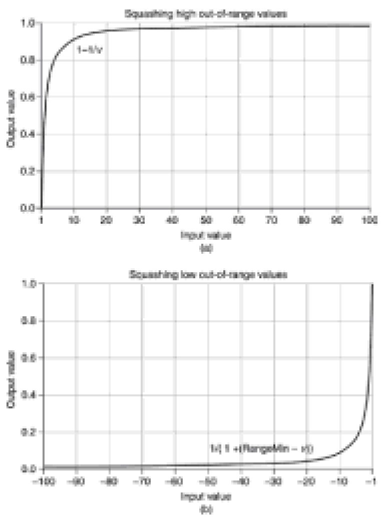


Figure 7.3 Values of 1 – 1/*v* for the range of inputs 1 through 100 (a) and values of 1/ (1+(RangeMin – *v*)) for the range of inputs –100 through –1(b).

Squashing out-of-range values into the lower space left can use the same transform. This time the out-of-range difference is discovered (RangeMin – *v*) and 1 is added to it to ensure that the difference can never be less than 1. This time *v* is the lower-range difference, but Table 7.3, in column 1/*v*, shows the values for this starting at 1 and decreasing toward, but never reaching, 0. Figure 7.3(b) shows this graphically. These two curves, one for the upper out-of-range values and one for the lower out-of-range values, need to be squashed and attached to the linear part of the transform.

Taking the linear part of the range and adding the upper and lower transforms for the out-of-range values produces a curve. The result will be a sort of “S” curve that is linear over most of the range, but squashes the over- and undervalues into the remaining space. Figure 7.4 shows the same curves squashed into the range. (The amount of the scale allocated for squashing out-of-range values is highly exaggerated to illustrate the point.)

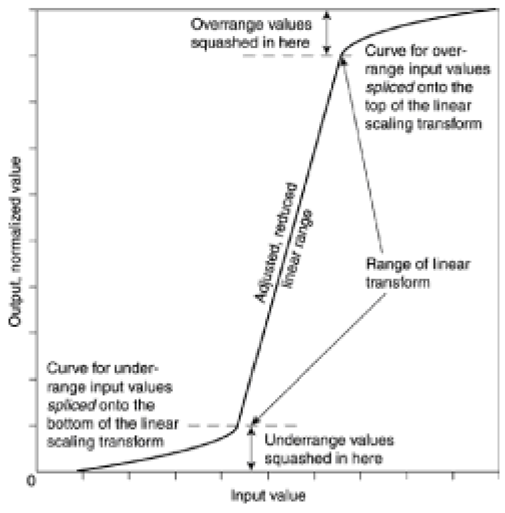

Overrange values squashed in here
Curve for over-range input values *spliced* onto the top of the linear scaling transform
Adjusted, reduced linear range
Range of linear transform
Curve for under-range input values *spliced* onto the bottom of the linear scaling transform
Underrange values squashed in here
Output, normalized value
0
Input value

Figure 7.4 The transforms for squashing overrange and underrange values are attached to the linear part of the transform. This composite "S"-shaped transform translates most of the values linearly, but also transforms any out-of-range values so that they stay within the 0–1 limits of the range.

This sort of "S" curve can be constructed to serve the purpose. Writing computer code to achieve this is somewhat cumbersome. The description shows very well the sort of effect that is needed, but fortunately there is a much easier and more flexible way to get there.

7.1.8 Softmax Scaling

Softmax scaling is so called because, among other things, it reaches "softly" toward its maximum value, never quite getting there. It also has a linear transform part of the range. The extent of the linear part of the range is variable by setting one parameter. It also reaches "softly" toward its minimum value. The whole output range covered is 0–1. These features make it ideal as a transforming function that puts all of the pieces together that have been discussed so far.

The Logistic Function

It starts with the *logistic function*. The logistic function can be modified to perform all of the work just described, and when so modified, it does it all at once so that by plugging in a variable's instance value, out comes the required, transformed value.

An explanation of the workings of the logistic function is in the Supplemental Material section at the end of this chapter. Its inner workings are a little complex, and so long as what needs to be done is clear (getting to the squashing "S" curve), understanding the logistic function itself is not necessary. The Supplemental Material can safely be skipped.

The explanation is included for interest since the same function is an integral part of neural networks, mentioned in Chapter 10. The Supplemental Material section then explains the modifications necessary to modify it to become the softmax function.

7.1.9 Normalizing Ranges

What does softmax scaling accomplish in addressing the problems of range normalization? The features of softmax scaling are as follows:

The normalized range is 0–1. It is the nature of softmax scaling that no values outside this range are possible. This keeps all normalized values inside unit state space boundaries. Since the range of input values is essentially unlimited and the output range is limited, unit state space, when softmax is normalized, is essentially infinite.

- The extent of the linear part of the normalized range is directly proportional to the level of confidence that the data sample is representative. This means that the more confidence there is that the sample is representative, the more linear the normalization of values will be.

- The extent of the area assigned for out-of-range values is directly proportional to the level of uncertainty that the full range has been captured. The less certainty, the more space to put the expected out-of-range values when encountered.

- There is always some difference in normalized value between any two nonidentical instance values, even for very large extremes.

As already discussed, these features meet many needs of a modeling tool. A static model may still be presented with out-of-range values where its accuracy and reliability are problematic. This needs to be monitored separately during execution time. (After all, softmax squashing them does not mean that the model knows what to do with them—they still represent areas of state space that the model never visited during training.) Dynamic models that continuously learn from the data stream—such as continuously learning, self-adaptive, or response-adaptive models—will have no trouble adapting themselves to the newly experienced values. (Dynamic models need to interact with a dynamic PIE if the range or distribution is not stationary—not a problem to construct if the underlying principles are understood, but not covered in detail here.)

At the limits of the linear normalization range, no modeling tool is required to aggregate the effect of multiple values by collapsing them into a single value (“clipping”).

Softmax scaling does the least harm to the information content of the data set. Yet it still leaves some information exposed for the mining tools to use when values outside those within the sample data set are encountered.

7.2 Redistributing Variable Values

Through normalization, the range of values of a variable can be made to always fall between the limits 0–1. Since this is a most convenient range to work with, it is assumed from here on that all of a variable's values fall into this range. It is also assumed that the variables fall into the linear part of the normalized range, which will be true during data preparation.

Although the range is normalized, the distribution of the values—that is, the pattern that exists in the way discrete instance values group together—has not been altered. (Distributions were discussed in Chapters 2 and 5.) Now attention needs to be turned to looking at the problems and difficulties that distributions can make for modeling tools, and ways to alleviate them.

7.2.1 The Nature of Distributions

Distributions of a variable only consist of the values that actually occur in a sample of many instances of the variable. For any variable that is limited in range, the count of possible values that can exist is in practice limited.

Consider, for example, the level of indebtedness on credit cards offered by a particular bank. For every bank there is some highest credit line that has ever been offered to any credit card customer. Large perhaps, but finite. Suppose that maximum credit line is $1,000,000. No credit card offered by this bank can possibly have a debit balance of more than $1,000,000, nor less than $0 (ignoring credit balances due, say, to overpayment). How many discrete balance amounts are possible? Since the balance is always stated to the nearest penny, and there are 100 pennies in a dollar, the range extends from 0 pennies to 100,000,000 pennies. There are no more than 100,000,000 possible discrete values in the entire range.

In general, for any possible variable, there is always a particular resolution limit. Usually it is bounded by the limits of accuracy of measurement, use, or convention. If not bounded by those, then eventually the limits of precision of representation impose a practical limit to the possible number of discrete values. The number may be large, but it is limited. This is true even for softmax normalization. If values sufficiently out of range are passed into the function, the truncation that any computer requires eventually assigns two different input values to the same normalized value. (This practical limitation should not often occur, as the way in which the scale was constructed should preclude many far out-of-range values.)

However many value states there are, the way the discrete values group together forms patterns in the distribution. Discrete value states can be close together or far apart in the range. Many variables permit identical values to occur—for example, for credit card balances, it is perfectly permissible for multiple cards to have identical balances.

A variable's values can be thought of as being represented in a one-dimensional state space. All of the features of state space exist, particularly including clustering of values. In some parts of the space the density will be higher than in other parts. Overall there will be some mean density.

7.2.2 Distributive Difficulties

One of the problems of distribution is outlying values or outlying clumps. (Figure 2.5 illustrates this.) Some modeling techniques are sensitive only to the linear displacement of the value across the range. This only means that the sensitivity remains constant across the range so that any one value is as "important" as any other value. It seems reasonable that 0.45 should be as significant as 0.12. The inferences to be made may be different—that is, each discrete value probably implies a different predicted value—but the fact that 0.45 has occurred is given the same weight as the fact that 0.12 has occurred.

Reasonable as this seems, it is not necessarily so. Since the values cluster together, some values are more common than others. Some values simply turn up more often than others. In the areas where the density is higher, values occurring in that area are more frequent than those values occurring in areas of lower density. In a sense, that is what density is measuring—frequency of occurrence. However, since some values are more common than others, the fact that an uncommon one has occurred carries a "message" that is different than a more common value. In other words, the weighting by frequency of specific values carries information.

To a greater or lesser degree, density variation is present for almost all variables. In some cases it is extreme. A binary value, for instance, has two spikes of extremely high density (one for the "0" value and one for the "1" value). Between the spikes of density is empty space. Again, most alpha variables will translate into a "spiky" sort of density, each spike corresponding to a specific label.

Figure 7.5 illustrates several possible distributions. In Figure 7.5(d) the outlier problem is illustrated. Here the bulk of the distribution has been displaced so that it occupies only half of the range. Almost half of the range (and half of the distribution) is empty.

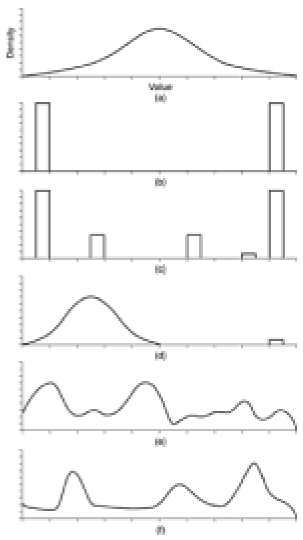


Figure 7.5 Different types of distributions and problems with the distribution of a variable's values across a normalized range: normal (a), bimodal or binary variable (b), alpha label (c), normal with outlier (d), typical actual variable A (e), and typical actual variable B (f). All graphs plot value (x) and density (y).

Many, if not most, modeling tools, including some standard statistical methods, either ignore or have difficulty with varying density in a distribution. Many such tools have been built with the assumption that the distribution is normal, or at least regular. When density is neither normal nor regular, as is almost invariably the case with real-world data sets—particularly behavioral data sets—these tools cannot perform as designed. In many cases they simply are not able to "see" the information carried by the varying density in the distribution. If possible, this information should be made accessible.

When the density variation is dissimilar between variables, the problem is only intensified. Between-variable dissimilarity means that not only are the distributions of each variable irregular, but that the irregularities are not shared by the two variables. The distributions in Figure 7.5(e) and 7.5(f) show two variables with dissimilar, irregular distributions.

There are tools that can cope well with irregular distributions, but even these are aided if the distributions are somehow regularized. For instance, one such tool for a particular data set could, when fine-tuned and adjusted, do just as well with unprepared data as with prepared data. The difference was that it took over three days of fine-tuning and adjusting by a highly experienced modeler to get that result—a result that was immediately available with prepared data. Instead of having to extract the gross nonlinearities, such tools can then focus on the fine structure immediately. The object of data preparation is to expose the maximum information for mining tools to build, or extract, models. What can be done to adjust distributions to help?

7.2.3 Adjusting Distributions

The easiest way to adjust distribution density is simply to displace the high-density points into the low-density areas until all points are at the mean density for the variable. Such a process ends up with a rectangular distribution. This simple approach can only be completely successful in its redistribution if none of the instance values is duplicated. Alpha labels, for instance, all have identical numerical values for a single label. There is no way to spread out the values of a single label. Binary values also are not redistributed using this method. However, since no other method redistributes such values either, it is this straightforward process that is most effective.

In effect, every point is displaced in a particular direction and distance. Any point in the variable's range could be used as a reference. The zero point is as convenient as any other. Using this as a reference, every other point can be specified as being moved away from, or toward, the reference zero point. The required displacements for any variable can be graphed using, say, positive numbers to indicate moving a point toward the "1," or increasing their value. Negative numbers indicate movement toward the "0" point, decreasing their value.

Figure 7.6 shows a distribution histogram for the variable "Beacon" included on the CD-ROM in the CREDIT data set. The values of Beacon have been normalized but not redistributed. Each vertical bar represents a count of the number of values falling in a subrange of 10% of the whole range. Most of the distribution shown is fairly rectangular. That is to say, most of the bars are an even height. The right side of the histogram, above a value of about 0.8, is less populated than the remaining part of the distribution as shown by the lower height bars. Because the width of the bars aggregates all of the values over 10% of the range, much of the fine structure is lost in a histogram, although for this example it is not needed.

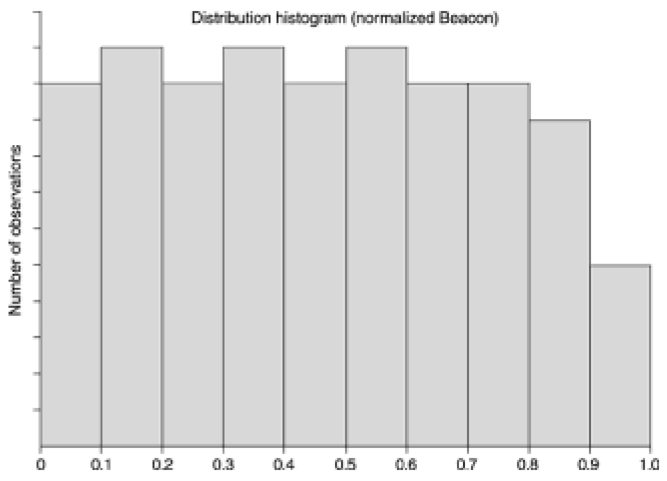


Figure 7.6 Distribution histogram for the variable Beacon. Each bar represents 10% of the whole distribution showing the relative number of observations (instances) in each bar.

Figure 7.7 shows a displacement graph for the variable Beacon. The figure shows the movement required for every point in the distribution to make the distribution more even. Almost every point is displaced toward the “1” end of the variable’s distribution. Almost all of the displaced distances being “+” indicates the movement of values in that direction. This is because the bulk of the distribution is concentrated toward the “0” end, and to create evenly distributed data points, it is the “1” end that needs to be filled.

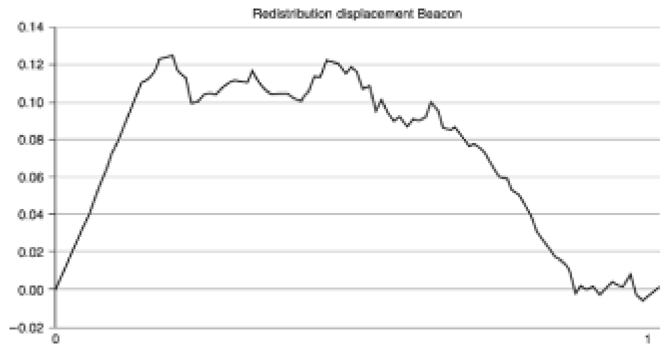


Figure 7.7 Displacement graph for redistributing the variable Beacon. The large positive “hump” shows that most of the values are displaced toward the “1” end of the normalized range.

Figure 7.8 shows the redistributed variable’s distribution. This figure shows an almost perfect rectangular distribution.

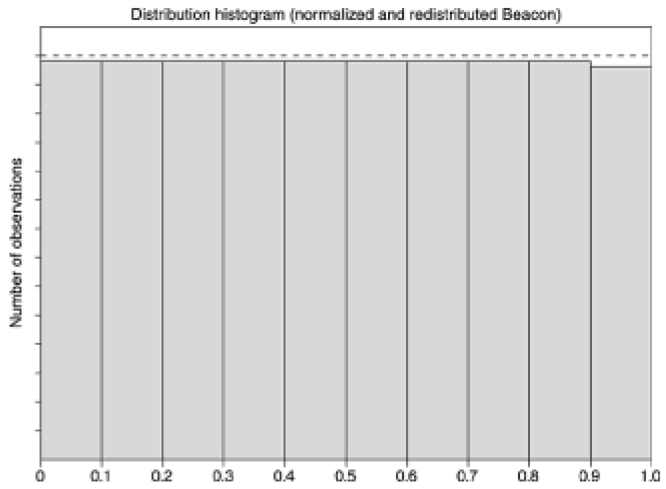


Figure 7.8 The distribution of Beacon after redistribution is almost perfectly rectangular. Redistribution of values has given almost all portions of the range an equal number of instances.

Figure 7.9 shows a completely different picture. This is for the variable DAS from the same data set. In this case the distribution must have low central density. The points low in the range are moved higher, and the points high in the range are moved lower. The positive curve on the left of the graph and the negative curve to the right show this clearly.

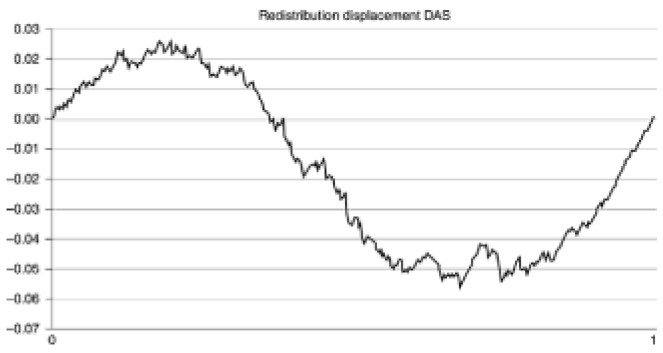


Figure 7.9 For the variable DAS, the distribution appears empty around the middle values. The shape of the displacement curve suggests that some generating phenomenon might be at work.

A glance at the graph for DAS seems to show an artificial pattern, perhaps a modified sine wave with a little noise. Is this significant? Is there some generating phenomenon in the real world to account for this? If there is, is it important? How? Is this a new discovery? Finding the answers to these, and other questions about the distribution, is properly a part of the data survey. However, it is during the data preparation process that they are first "discovered."

7.2.4 Modified Distributions

When the distributions are adjusted, what changes? The data set CARS (included on the accompanying CD-ROM) is small, containing few variables and only 392 instances. Of the variables, seven are numeric and three are alpha. This data set will be used to look at what the redistribution achieves using "before" and "after" snapshots. Only the numeric variables are shown in the snapshots as the alphas do not have a numeric form until after numeration.

Figures 7.10(a) and 7.10(b) show box and whisker plots, the meaning of which is fairly self-explanatory. The figure shows maximum, minimum, median, and quartile information. (The median value is the value falling in the middle of the sequence after ordering the values.)

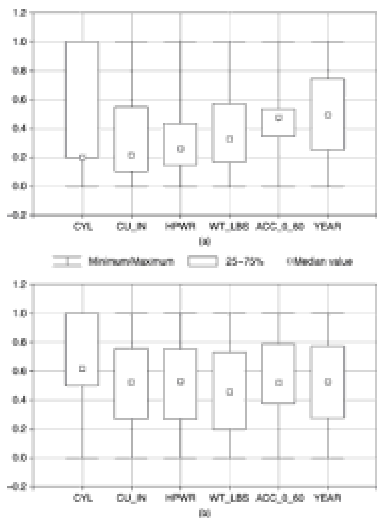


Figure 7.10 These two box and whisker plots show the before and after redistribution positions—normalized only (a) and normalized and redistributed (b)—for maximum, minimum, and median values.

Comparing the variables, before and after, it is immediately noticeable that all the median values are much more centrally located. The quartile ranges (the 25% and 75% points) have been far more appropriately located by the transformation and mainly fall near the 25% and 75% points in the range. The quartile range of the variable "CYL" (number of cylinders) remains anchored at "1" despite the transformation—why? Because there are only three values in this field—"4," "6," and "8"—which makes moving the quartile range impossible, as there are only the three discrete values. The quartile range boundary has to be one of these values. Nonetheless, the transformation still moves the lower bound of the quartile range, and the median, to values that better balance the distribution.

Figures 7.11(a) and 7.11(b) show similar figures for standard deviation, standard error, and mean. These measures are normally associated with the Gaussian or normal distributions. The redistributed variables are not translated to be closer to such a distribution. The translation is, rather, for a rectangular distribution. The measures shown in this figure are useful indications of the regularity of the adjusted distribution, and are here used entirely in that way. Once again the distributions of most of the variables show considerable improvement. The distribution of "CYL" is improved, as measured by standard deviation, although with only three discrete values, full correction cannot be achieved.

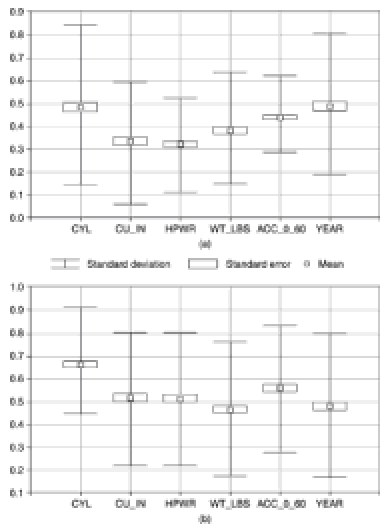


Figure 7.11 These two box and whisker plots show the before and after redistribution positions—normalized only (a) and normalized and redistributed (b)—for standard deviation, standard error, and mean values.

Table 7.4 shows a variety of measures about the variable distributions before and after transformation. “Skewness” measures how unbalanced the distribution is about its center point. In every case the measure of skewness is less (closer to 0) after adjustment than before. In a rectangular distribution, the quartile range should cover exactly half the range (0.5000) since it includes the quarter of the range immediately above and below the median point. In every case except “Year,” which was perfect in this respect to start with, the quartile range shows improvement.

TABLE 7.4 Statistical measures before and after adjustment.

BEFORE:	**Mean**	**Median**	**Lower quartile**	**Upper quartile**	**Quartile range**	**Std. dev.**	**Skew-ness**
CYL	0.4944	0.2000	0.2000	1.0000	0.8000	0.3412	0.5081
CU_IN	0.3266	0.2145	0.0956	0.5594	0.4638	0.2704	0.7017
HPWR	0.3178	0.2582	0.1576	0.4402	0.2826	0.2092	1.0873
WT_LBS	0.3869	0.3375	0.1734	0.5680	0.3947	0.2408	0.5196

ACC	0.4518	0.4706	0.3529	0.5294	0.1765	0.1624	0.3030
YEAR	0.4983	0.5000	0.2500	0.7500	0.5000	0.3070	0.0197

AFTER:	**Mean**	**Median**	**Lower quartile**	**Upper quartile**	**Quartile range**	**Std. dev.**	**Skew-ness**
CYL	0.6789	0.5998	0.4901	1.0000	0.5099	0.2290	0.2851
CU_IN	0.5125	0.5134	0.2518	0.7604	0.5086	0.2912	–0.0002
HPWR	0.5106	0.5123	0.2488	0.7549	0.5062	0.2907	–0.0359
WT_LBS	0.4740	0.4442	0.1939	0.7338	0.5400	0.2985	0.1693
ACC	0.5586	0.5188	0.3719	0.7875	0.4156	0.2799	–0.2109
YEAR	0.4825	0.5185	0.2704	0.7704	0.5000	0.3197	0.0139

The variable “Year” was distorted some small amount from an already perfectly rectangular distribution. The distortion is minor, but why did it happen? In fact, the variable “Year” is monotonic. There are a similar number of instances in each of several years. This gives the appearance of a perfectly rectangular distribution. Redistribution notices a weighting due to the monotonicity and attempts to “correct” for it. Another clue that this variable may need further investigation is that the standard deviation increases and moves further from the optimum point. The standard deviation measure for a normalized rectangular distribution is approximately 0.2889. However, altogether the adjustment is very minor and almost certainly does no harm. Being monotonic, the variable may need to be dealt with in some other way before modeling anyway.

7.3 Summary

What has been accomplished by using the techniques in this chapter? The raw values of a variable have been translated in range and distribution. This has useful benefits.

First, all values are normalized over a range of 0–1. Some modeling techniques require such a normalizing transformation; for others, it’s only a convenience. In all cases, it puts the full magnitude of the change in a variable on an equal footing for all variables in the data set.

Second, one of the limitations of sampling was dealt with: the problem that values not sampled, and outside the range of those in the sample, are sure to turn up in the population. The specific problem that unsampled out-of-range values cause for a model depends on where in the process of building or applying a model the unsampled out-of-range value is discovered. Softmax scaling, developed out of linear scaling and based on the logistic function, provides a convenient method for ensuring that all values, sampled or not, are correctly normalized. This does not overcome the out-of-range problem, but it makes it more tractable.

While looking at softmax scaling, we explored the workings of the logistic function. This is a very important function for understanding the inner workings of neural networks. Introduced here for the softmax squashing, it is also important for understanding the techniques introduced in Chapter 10. (Not absolutely necessary, as those techniques can still be applied without a full understanding of how they work.)

Third, and very important for maximum information exposure, the individual variable distributions are transformed. This transformation makes the between-variable information far more accessible to many modeling tools. Many of the problems with value clusters are removed, and almost all of the problems that outliers present are very significantly reduced, if not completely ameliorated. A miner may glean useful insights into the nature of a variable by looking at similarities, differences, and structures in the variable distributions, although looking at these is really part of the data survey and not further considered here.

By the time the techniques discussed in this chapter are applied to a data set, a suitably sized sample is selected (discussed in Chapter 5). The sample is fully represented as numeric (discussed in Chapter 6), and fully normalized in both range and distribution (this chapter). The last problem to look at in the data, before turning our attention to preparing the data set as a whole, is that some of the values may be missing or empty. Chapter 8 looks at plugging these holes. Although it is the individual variables that are considered, attention now must be turned to the data set as a whole since that is where the information needed is discovered.

Supplemental Material

The Logistic Function

The logistic function is usually written as

$$v_n = \frac{1}{1 + e^{-v_i}}$$

where

$$e^{-v_i} = \frac{1}{e^{v_i}}$$

v_n	is the normalized value
v_i	is the instance value

How does this function help? It is easier to understand what is happening by looking at each of the pieces of the function one at a time. Start with

$$e^{v_i}$$

In this piece of the function, *vi* is the instance value. The *e* represents a number, a constant, approximately 2.72. Any constant greater than 1 could be used here, but *e* is the usual choice.

$$\frac{1}{e^{v_i}}$$

This is simply the reciprocal of the previous expression. Reciprocals get smaller as the number gets larger.

$$e^{-v_i} = \frac{1}{e^{v_i}}$$

Note that these two are equivalent ways of saying the same thing. It is a little more compact to use the notation of *e* to a negative exponent.

$$1 + e^{-v_i}$$

This makes sure that the result is never less than 1, which is very important in the next step. Since this expression can never have a value of less than 1, the next expression can never have a value greater than 1.

$$v_n = \frac{1}{1 + e^{-v_i}}$$

which brings the expression full circle.

So how do each of these components behave? Table 7.5 shows how each of these components of the logistic function change as different values are plugged into the formula.

TABLE 7.5 Values of components of logistic function.

v_i	e^{v_i}	$1/e^{v_i}$	$1+1/e^{v_i}$	Logistic
–10	0.0000	22026.3176	22027.3176	0.0000
–9	0.0001	8103.0349	8104.0349	0.0001
–8	0.0003	2980.9419	2981.9419	0.0003
–7	0.0009	1096.6280	1097.6280	0.0009
–6	0.0025	403.4272	404.4272	0.0025
–5	0.0067	148.4127	149.4127	0.0067
–4	0.0183	54.5980	55.5980	0.0180
–3	0.0498	20.0855	21.0855	0.0474
–2	0.1353	7.3890	8.3890	0.1192
–1	0.3679	2.7183	3.7183	0.2689
0	1.0000	1.0000	2.0000	0.5000
1	2.7183	0.3679	1.3679	0.7311
2	7.3890	0.1353	1.1353	0.8808
3	20.0855	0.0498	1.0498	0.9526
4	54.5980	0.0183	1.0183	0.9820
5	148.4127	0.0067	1.0067	0.9933
6	403.4272	0.0025	1.0025	0.9975

7	1096.6280	0.0009	1.0009	0.9991
8	2980.9419	0.0003	1.0003	0.9997
9	8103.0349	0.0001	1.0001	0.9999
10	22026.3176	0.0000	1.0000	1.0000

Examining what is going on inside this function is easier to see graphically. Figure 7.12 illustrates the components. So that the various components can be seen on a common scale, the *vi* values (the instance values) range only between –2 and +2 on this graph. Even with this limited range, the various components vary considerably more than the normalized output. Figure 7.13 shows how the logistic function transforms inputs across the range of –10 to +10. This shows the squashing effect very clearly.

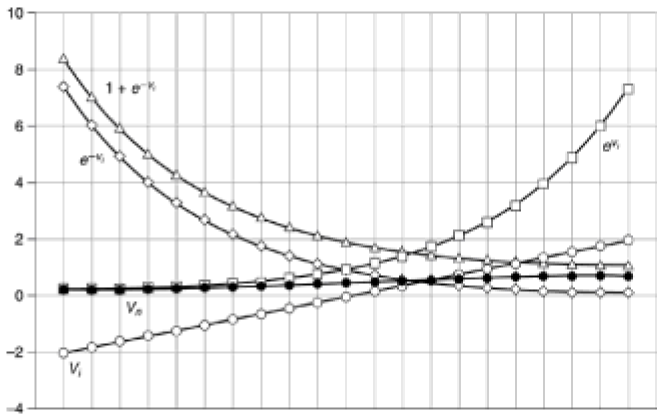


Figure 7.12 Components of the logistic function.

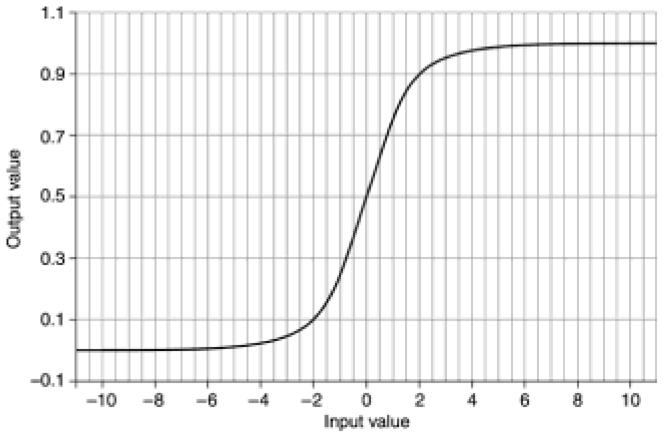


Figure 7.13 Logistic function for the range –10 through 10. The logistic function

squashes the input values into the range 0–1.

Due to rounding errors, and because only four decimal places are shown, Table 7.5 seems to show that the logistic function reaches both 0 and 1 at the extremes. This is not in fact the case. Although very close to those values at the extremes, the function actually gets there only with infinitely large positive or negative numbers.

Modifying the Linear Part of the Logistic Function Range

As it stands, the logistic function produces the needed “S” curve, but not over the needed range of values. There is also no way to select the range of linear response of the standard logistic function.

Manipulating the value of *vi* before plugging it into the logistic function allows the necessary modification to its response so that it acts as needed. To show the modification more easily, the modification made to the value of *vi* is shown here as producing *vt*. When *vt* is plugged into the logistic function in place of *vi*, the whole thing becomes known as the *softmax function*:

$$v_t = \frac{(v_i - \bar{v})}{\lambda(\sigma_v/2\pi)}$$

where

v_t	is transformed value of *vi*
x_v	is standard deviation of variable *v*
x	is linear response in standard deviations
x	is 3.14 approximately

The linear response part of the curve is described in terms of how many normally distributed standard deviations of the variable are to have a linear response. This can be discovered by either looking in tables or (as used in the demonstration software) using a procedure that returns the appropriate standard deviation for any selected confidence level. Such standard deviations are known as *z* values. Looking in a table of standard deviations for the normal curve, ±1*z* cover about 68%, ±2*z* about 95.5%, and ±3*z* about 99.7%. (The ±*n* is because the value is *n* on either side of the central point of the distribution; i.e., ±1*z* means 1*z* greater than the mean and 1*z* less than the mean.)

Chapter 8: Replacing Missing and Empty Values

Overview

The presence of missing or empty values can make problems for the modeler. There are several ways of replacing these values, but the best are those that not only are well understood by the modeler as to their capabilities, limits, and dangers, but are also in the modeler's control. Even replacing the values at all has its dangers unless it is carefully done so as to cause the least damage to the data. It is every bit as important to avoid adding bias and distortion to the data as it is to make the information that is present available to the mining tool.

The data itself, considered as individual variables, is fairly well prepared for mining at this stage. This chapter discusses a way to fill the missing values, causing the least harm to the structure of the *data set* by placing the missing value in the context of the other values that are present. To find the necessary context for replacement, therefore, it is necessary to look at the data set as a whole.

8.1 Retaining Information about Missing Values

Missing and empty values were first mentioned in Chapter 2, and the difference between missing and empty was discussed there. Whether missing or empty, many, if not most, modeling tools have difficulty digesting such values. Some tools deal with missing and empty values by ignoring them; others, by using some metric to determine "suitable" replacements. As with normalization (discussed in the last chapter), if default automated replacement techniques are used, it is difficult for the modeler to know what the limitations or problems are, and what biases may be introduced. Does the modeler know the replacement method being used? If so, is it really suitable? Can it introduce distortion (bias) into the data? What are its limitations? Finding answers to these questions, and many similar ones, can be avoided if the modeler is able to substitute the missing values with replacements that are at least neutral, that is, introduce no bias—and using a method understood, and controlled by, the modeler.

Missing values should be replaced for several reasons. First, some modeling techniques cannot deal with missing values and cast out a whole instance value if one of the variable values is missing. Second, modeling tools that use default replacement methods may introduce distortion if the method is inappropriate. Third, the modeler should know, and be in control of, the characteristics of any replacement method. Fourth, most default replacement methods discard the information contained in the missing-value patterns.

8.1.1 Missing-Value Patterns

A point to note is that replacing missing values, without elsewhere capturing the information that they were missing, actually removes information from the data set. How is this? Replacing a missing value obscures the fact that it was missing. This information can be very significant. It has happened that the pattern of missing values turned out to be the most important piece of information during modeling. Capturing this information has already been mentioned in Chapter 4. In Figure 4.7, the single-variable CHAID analysis clearly shows a significant relationship between the missing-value pattern variable, _Q_MVP, and the variable SOURCE in the SHOE data (included on the CD-ROM). Retaining the information about the pattern in which missing values occur can be crucial to building a useful model.

In one particular instance, data had been assembled into a data warehouse. The architects had carefully prepared the data for warehousing, including replacing the missing values. The data so prepared produced a model of remarkably poor quality. The quality was only improved when the original source data was used and suitably prepared for modeling rather than warehousing. In this data set, the most predictive variable was in fact the missing-value pattern—information that had been completely removed from the data set during warehousing. The application required a predictive model. With warehoused data, the correlation between the prediction and the outcome was about 0.3. With prepared data, the correlation improved to better than 0.8.

Obviously a change in correlation from 0.3 to 0.8 is an improvement, but what does this mean for the accuracy of the model? The predictive model was required to produce a score. If the prediction was within 10% of the true value, it was counted as “correct.” The prediction with a correlation of 0.3 was “correct” about 4% of the time. It was better than random guessing, but not by much. Predicting with a correlation of about 0.8 produced “correct” estimates about 22% of the time. This amounts to an improvement of about 550% in the predictive accuracy of the model. Or, again, with a 0.3 correlation, the mean error of the prediction was about 0.7038 with a standard deviation of 0.3377. With a 0.8 correlation, the mean error of the prediction was about 0.1855 with a standard deviation of 0.0890. (All variables were normalized over a range of 0–1.)

Whatever metric is used to determine the quality of the model, using the information embedded in the missing-value patterns made a large and significant difference.

8.1.2 Capturing Patterns

The *missing-value pattern* (MVP) is exactly that—the pattern in which the variables are missing their values. For any instance of a variable, the variable can either have a value (from 0–1) or not have any value. If it has no numerical value, the value is “missing.” “Missing” is a value, although not a numerical value, that needs to be replaced with a numerical value. (It could be empty, but since both missing and empty values have to be treated similarly and replaced, here they will be discussed as if they are the same.) For each variable in the data set, a flag, say, “P” for present and “E” for empty, can be used to indicate the presence or absence of a variable’s value in the instance value. Using such flags creates a series of patterns, each of which has as many flags as there are dimensions in the data. Thus a three-dimensional data set could have a maximum of eight possible MVPs as shown in Table 8.1

TABLE 8.1 Possible MVPs for a three-dimensional data set.

Pattern number	**Pattern**
1	PPP
2	PPE
3	PEP
4	PEE
5	EPP
6	EPE
7	EEP
8	EEE

The number of possible MVPs increases very rapidly with the number of dimensions. With only 100 dimensions, the maximum possible number of missing-value patterns is far more than any possible number of instance values in a data set. (There are over one nonillion, that is, 1 x 1030, possible different patterns.) The limiting number quickly becomes the maximum number of instances of data. In practice, there are invariably far fewer MVPs than there are instances of data, so only a minute fraction of the possible number of MVPs actually occur. While it is quite possible for all of the MVPs to be unique in most data sets, every practical case produces at least some repetitive patterns of missing values. This is particularly so for behavioral data sets. (Chapter 4 discussed the difference between physical and behavioral data sets.)

MVPs aren't invariably useful in a model. However, surprisingly often the MVPs do contribute useful and predictive information. The MVPs are alpha labels, so when they are extracted, the MVPs are numerated exactly as any other alpha value. Very frequently the MVPs are best expressed in more than one dimension. (Chapter 6 discusses numeration of alpha values.) Where this is the case, it is also not unusual to find that one of the multiple dimensions for MVPs is especially predictive for some particular output.

8.2 Replacing Missing Values

Once the information about the patterns of missing values is captured, the missing values themselves can be replaced with appropriate numeric values. But what are these values to be? There are several methods that can be used to estimate an appropriate value to plug in for one that is missing. Some methods promise to yield more information than others, but are computationally complex. Others are powerful under defined sets of circumstances, but may introduce bias under other circumstances.

Computational complexity is an issue. In practice, one of the most time-consuming parts of automated data preparation is replacing missing values. Missing-value estimating methods that produce mathematically optimal values can be highly complex, and vary with the type of data they are to be applied to. Highly complex methods are too time-consuming for large data sets. Even with the speed of modern computer systems, the amount of processing required simply takes too long to be reasonable, especially for time-sensitive business applications. Also, preparation techniques need to be as broadly applicable as possible. Replacement methods with advantages in specific situations, but that are unworkable or introduce bias in others, are simply not suitable for general application.

In addition to first doing as little harm as possible under all circumstances, and being computationally tractable, whatever method is used has to be applicable not only to the identified MVPs, but to any new missing values that arise in execution data sets. Chapter 7 discussed estimating the probability that the population maximum was found in any particular sample. That discussion made it clear that out-of-range values were to be expected during execution. The same is true for MVPs—and for the missing values that they represent. There is always some probability that individual variable values that were not ever missing in the sample will be found missing in the execution data set, or that MVPs occurring in the population are not actually present in a sample. The PIE-I needs to be able to deal with these values too—even though they were never encountered as missing during the building of the PIE.

8.2.1 Unbiased Estimators

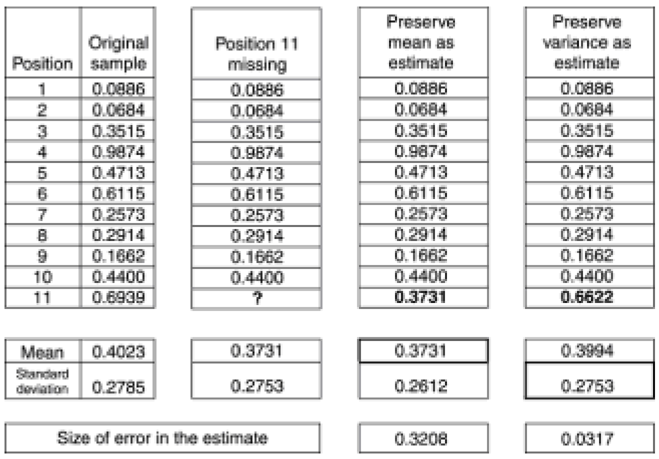
An *estimator* is a device used to make a justifiable guess about the value of some particular value, that is, to produce an *estimate*. An *unbiased* estimator is a method of guessing that doesn't change important characteristics of the values present when the estimates are included with the existing values. (Statistically, an unbiased estimator produces an estimate whose "expected" value is the value that would be estimated from the population.)

For instance, consider the numbers 1, 2, 3, *x*, 5, where "*x*" represents a missing value. What number should be plugged in as an unbiased estimate of the missing value? Ideally, a value is needed that will at least do no harm to the existing data. And here is a critical point—what does "least harm" mean exactly? If the mean is to be unbiased, the missing value needs to be 2.75. If the standard deviation is to be unbiased, the missing value needs to be about 4.659. The missing-value estimate depends as much on which characteristic is to be unbiased as it does on the actual values. Before deciding what "least harm" means in practice, it is important to discover which relationships need to be preserved, both within and between variables. Finding which are the important relationships to preserve will indicate how to find the estimate that best preserves these relationships—that is, that is least biased for these particular relationships.

8.2.2 Variability Relationships

Chapter 7 discussed variability and redistributing a variable's distribution. The transform produced a rectangular distribution insofar as the actual values allowed for it. Each variable was considered individually, and was redistributed individually without reference to the distributions of other variables. In addition to this within-variable relationship, there is also a critical between-variable relationship that exists for all of the variables. The between-variable relationship expresses the way that one variable changes its value when another variable changes in value. It is this multiple-way, between-variable relationship that will be explored by any modeling tool. Since the chosen modeling tool is going to explore these relationships between variables, it is critical to preserve them, so far as possible, when replacing missing values.

Variability forms a key concept in deciding what values to use for the replacement. Standard deviation is one measure of that variability (introduced in Chapter 5). To exemplify the underlying principles of preserving variability, consider a single variable whose values are transformed into a rectangular distribution. Figure 8.1 shows the values of such a variable. The 11 values of the original series are shown in the column headed "Original sample." Suppose that the value in series position 11 is missing and is to be replaced. Since in this example the actual series value is present, it is easy to see how well any chosen estimator preserves the relationships.



Position	Original sample	Position 11 missing	Preserve mean as estimate	Preserve variance as estimate
1	0.0886	0.0886	0.0886	0.0886
2	0.0684	0.0684	0.0684	0.0684
3	0.3515	0.3515	0.3515	0.3515
4	0.9874	0.9874	0.9874	0.9874
5	0.4713	0.4713	0.4713	0.4713
6	0.6115	0.6115	0.6115	0.6115
7	0.2573	0.2573	0.2573	0.2573
8	0.2914	0.2914	0.2914	0.2914
9	0.1662	0.1662	0.1662	0.1662
10	0.4400	0.4400	0.4400	0.4400
11	0.6939	?	**0.3731**	**0.6622**
Mean	0.4023	0.3731	0.3731	0.3994
Standard deviation	0.2785	0.2753	0.2612	0.2753
Size of error in the estimate			0.3208	0.0317

Figure 8.1 Estimating the value of position 11, given only the values in positions 1 through 10.

Position 11 has an actual value of 0.6939. The mean for the original 11-member series is 0.4023. If the value for position 11 is to be estimated, only positions 1 through 10 are used since the actual value of position 11 is assumed unknown. The third column (headed “Position 11 missing”) shows the 10 values used to make the estimate, with the mean and standard deviation for these first 10 positions shown beneath. The mean for these first 10 positions is 0.3731. Using this as the estimator and plugging it into position as an estimate of the missing value (position 11) changes the standard deviation from about 0.2753 to 0.2612. The column mean is unchanged (0.3731). Using the mean of instances 1 through 10 has least disturbed the *mean* of the series. The actual value of the “missing” value in position 11 is 0.6939, and the mean has estimated it as 0.3731—a discrepancy of 0.3208.

Suppose, however, that instead of using the mean value, a value is found that least disturbs the standard deviation. Is this a more accurate estimate, or less accurate? Position 11 in column five uses an estimate that least disturbs the standard deviation. Comparing the values in position 11, column four (value not disturbing mean as replacement) and column five (value not disturbing standard deviation as replacement), which works best at estimating the original value in column two?

The column four estimator (preserving mean) misses the mark by

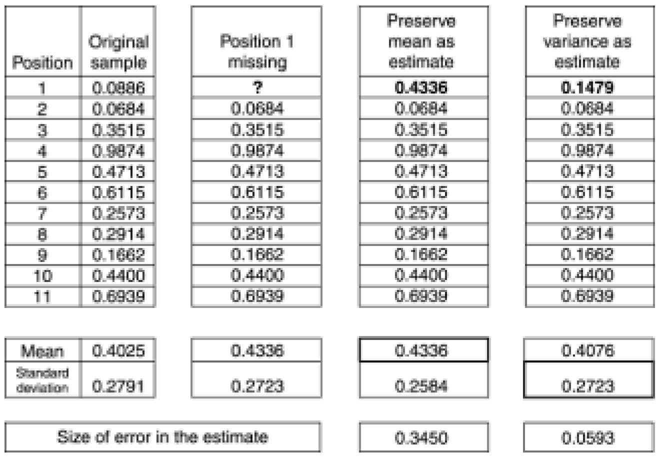
0.6939 – 0.3731 = 0.3208.

The column five estimator (preserving standard deviation) only misses the mark by

0.6622 – 0.6939 = 0.0317.

Also, preserving the standard deviation (column five) moved the new mean closer to the original mean value for all 11 original values in column one. This can be seen by comparing the mean and standard deviation for each of the columns. The conclusion is that preserving standard deviation does a much better job of estimating the "true" mean and provides a less biased estimate of the "missing" value.

Was this a convenient and coincidental fluke? Figure 8.2 shows the situation if position 1 is assumed empty. The previous example is duplicated with values for preserving mean and standard deviation shown in separate columns. As before, generating the replacement value by preserving standard deviation produces a less biased estimate.



Position	Original sample	Position 1 missing	Preserve mean as estimate	Preserve variance as estimate
1	0.0886	**?**	**0.4336**	**0.1479**
2	0.0684	0.0684	0.0684	0.0684
3	0.3515	0.3515	0.3515	0.3515
4	0.9874	0.9874	0.9874	0.9874
5	0.4713	0.4713	0.4713	0.4713
6	0.6115	0.6115	0.6115	0.6115
7	0.2573	0.2573	0.2573	0.2573
8	0.2914	0.2914	0.2914	0.2914
9	0.1662	0.1662	0.1662	0.1662
10	0.4400	0.4400	0.4400	0.4400
11	0.6939	0.6939	0.6939	0.6939

Mean	0.4025	0.4336	0.4336	0.4076
Standard deviation	0.2791	0.2723	0.2584	0.2723
Size of error in the estimate			0.3450	0.0593

Figure 8.2 Estimating the value of position 1, given only the values in positions 2 through 11.

Why is it that preserving variability as measured by standard deviation produces a better estimate, not only of the missing value, but of the original sample mean too? The reason is that the standard deviation reflects far more information about a variable than the mean alone does. The mean is simply a measure of central tendency (mentioned in Chapter 7). The standard deviation reflects not just central tendency, but also information about the variability within the variable's distribution. If the distribution is known (here made as rectangular as possible by the methods described in Chapter 7), that knowledge contributes to determining a suitable replacement value. It is this use of additional information that produces the better estimate.

Preserving variability works well for single missing values in a variable. If multiple values are missing, however, the estimator still produces a single estimate for all missing values, just as using the mean does. If both positions 1 and 11 were missing in the above example, any estimator preserving standard deviation would produce a single estimate to replace both missing values. So, there is only one single replacement value to plug into all the missing values. Does this cause any problem?

8.2.3 Relationships between Variables

Since it is important to retain the relationship between the variables as well as possible, the question becomes, Does assigning a single value to a variables' missing values maintain the between-variable relationship?

As a system of variables, there exists some relationship linking the variables' values to each other. It may be stronger or weaker depending on which variables are compared and which portions of the range are compared. Nonetheless, since there is a relationship, whatever it may be, if the value of one variable is found to be at a specific value, the values of all the other variables will be expected to be at particular values in their ranges. The linkage expresses the amount and direction of change in value of each variable. This amounts to no more, for instance, than saying, "Size of home owned increases with income level." True or not, this statement expresses a relationship between house size and income level. As one changes, so do expected values of the other.

If missing values of, say, income, are replaced with any fixed values, it does not allow for, or reflect, the value of house size associated with that missing value. No notice is taken of what might be an "appropriate" value to use for the missing value of income, given the matching value of house size. Assigning a constant value to one variable for all of its missing values will certainly distort the relationship between the variables. What is more, as already observed, if the missing values are not missing at random, then using replacements that all have the same value will not only be inappropriate, but will actually add bias, or distortion. This will show up as the nonrandom pattern of fixed values plugged into the missing values.

Replacing missing values in one variable, then, needs to take account of whatever values are actually present for the other variables in a specific instance value. Given multiple MVPs, several variables may be simultaneously missing values, but never all of the variables at once. Whatever variable values are present can be used to estimate what the appropriate level of the missing variable values should be.

There are several ways of estimating the appropriate missing values, and it is here that the demonstration software takes a shortcut to make the computations tractable. At this point in the data preparation, the modeler does not know the precise nature of the relationship between the variables. Discovering and explicating the nature of that relationship is, in fact, one of the main tasks of data mining in the first place—specifically the part called modeling that comes after preparation. It is likely that the exact relationship is not actually linear. In a linear relationship, if the value of one variable changes by a particular amount, then the value of another variable changes by another particular amount, and in a specific direction. To return to house size and income for a moment, it means that however much the house size increases for a $1,000 rise in income at $25,000 a year, it increases a similar amount for any similar $1,000 rise at any other income level, say, $50,000. This may or may not be the case. For the demonstration software, however, the relationship is assumed to be linear. In practice, this assumption introduces very little if any bias for most data sets. Nonetheless, the modeler needs to be aware of this.

A key point is that, although the replacement values are indeed predictions, it is not the accuracy of these predictions that is of most importance when replacing missing values. The key concern is that the predictions produce a workable estimate that least distorts the values that are actually present. It is this “least distortion” that is the important point. The purpose of replacing missing values is not to use the values themselves, but to make available to the modeling tools the information contained in the other variables’ values that are present. If the missing values are not replaced, the whole instance value may be ignored. If not ignored, the missing value may be replaced with some default unknown to the modeler that introduces bias and distortion. Avoiding bias is important. Although not perfect, multiple linear estimation is enormously preferable to, and produces much less bias than, any replacement with a constant value (like the mean value), or a procedure that ignores problematic instances.

However, it must be noted that a more accurate replacement can be achieved by using one of the more computationally intensive methods of determining multiple nonlinear relationships between variables. Although the demonstration software uses a modified multiple linear regression technique, which is examined next, a brief review of some possible alternatives in discussed in the Supplemental Material section at the end of this chapter. (The full data preparation and survey suite, on which the demonstration code is based, uses a nonlinear estimator optimized for least information change. This method is based on the principles discussed here. Where intervariable relationships are nonlinear or even discontinuous, such methods minimize the potential distortion to information content in a data set.)

8.2.4 Preserving Between-Variable Relationships

Regression methods are inherently mathematical and normally are themselves very sensitive to missing values. The Supplemental Material section at the end of this chapter discusses exactly how regression methods can be modified to determine appropriate missing values.

As a conceptual overview, recall that what is needed is some way of determining how the value of one variable changes as the value of another (or several others) changes. This simply means measuring what values of one variable are when another variable has particular values.

Table 8.2 shows, for instance, that if the value of *y* was missing in a particular instance, but the value of *x* was 3, then an appropriate value to plug in for *y* would be about 14. But what is needed for automated replacement is not a table to look at to find a replacement value, but a formula that relates one set of values to the other. What would this formula look like?

TABLE 8.2 Values of two variables.

Variable *x* value	**Variable *y* value**
1	10
2	12
3	14
4	16
5	18
6	20

Start by trying to find a way to predict *y* from *x*. Notice that the value of *y* increases by 2 for every increase in *x* of 1, and decreases by 2 for every decrease in *x* of 1. So notice that if $x = 0$, y must $= 8$. So the relationship must be that *y* equals 8 plus 2 times the amount of *x*. Or

$$y = 8 + 2x$$

A little mathematical manipulation of this formula gives an expression for determining values of *x* from values of *y*:

$$x = (y - 8)/2$$

These formulas represent the essence of what needs to be done to find missing values. Of course, something other than an intuitive mechanism is needed for discovering the expressions needed, and more of those details are explored in the Supplemental Material section at the end of this chapter. However, the whole method is based on preserving between-variable variability, which is no more than saying preserving the between-variable relationships, which is what the example just did.

This is a very simple example of what regression analysis achieves. How it is done in practice, why preserving variability is critical, and how the methods work are all covered in more conceptual detail also in the Supplemental Material section at the end of this chapter.

8.3 Summary

Replacing missing values in a data set is very important. However, it is at least as important to capture the patterns of values that are missing, and the information that they contain, as it is to replace the values themselves. The values that are missing must be replaced with extreme sensitivity for not disturbing the patterns in the data that already exist. Using inappropriate values easily disturbs those patterns and introduces spurious patterns, called bias or noise. Such artificially introduced patterns damage the information carried in the between-variable variability in a data set. This hides or distorts the existing patterns, thus hiding or distorting the information embedded in the data set.

Capturing the variability that is present in a data set in the form of the ratios between various values can be used to infer, or impute, appropriate missing values that do least damage to the information content of the data set. All methods of replacing missing values are a compromise. Please see the Supplemental Material section at the end of this chapter for a more detailed explanation.

From here forward, the data (variables) are prepared for modeling. What remains is to look at the issues that are still outstanding in preparing the data set as a whole.

Supplemental Material

Using Regression to Find Least Information-Damaging Missing Values

In any data set, with or without missing values, there is enfolded information. The information is carried in the relationships between the values both within a single variable (its distribution), and in the relationships with the patterns in other variables. If some of these values are missing, for reasons already discussed, they need to be replaced. But replacing values is a tricky business. It is critically important that the replacement values do not introduce a pattern into the data that is not actually present in the measured values. Such a pattern may later be “discovered” by the miner, and may actually appear to be meaningful, but it carries no real information since it is simply an artifact of the replacement process.

However, just as important as not introducing an artificial pattern into a data set by replacement values is the maintenance of the existing pattern. Some patterns, maybe critically important ones, are fragile and delicate, and great care must be taken to maintain them undistorted in the data set. How is this delicate balancing act to be accomplished? There are a number of techniques available. The one explained here is used primarily for ease of understanding. There are more complex approaches available that perform slightly better under some circumstances, and those are discussed in the second section of this supplemental material.

Regressions

Multiple linear regression is a generalized extension of ordinary linear regression, which is more accessible for explanatory purposes than multiple linear regression since it uses only two variables. The idea is simply to determine the value of one variable given the value of the other. It assumes that the variables' values change, the one with the other, in some linear way. The technique involved simply fits a manifold, in the form of a straight line, through the two-dimensional state space formed by the two variables. Figure 8.3 shows this for three of the variables in the CARS data set.

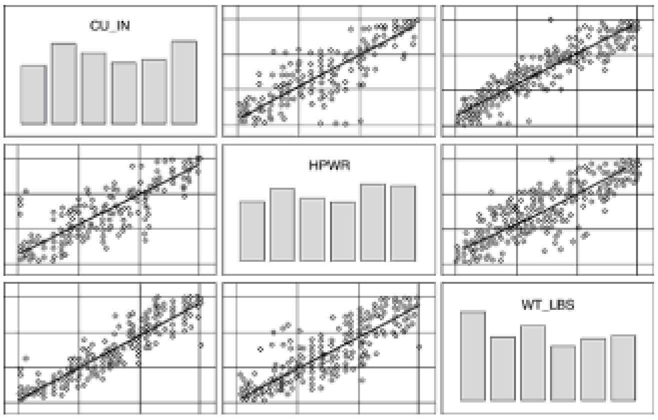


Figure 8.3 Linear regression manifolds (lines) for 2D spaces. The bar charts show the distributions of the variables CU_IN (cubic inches), HPWR (horsepower), and WT_LBS (weight).

Figure 8.3 shows the state space for each pair of variables. The bar graphs show the distribution for each of the three variables. Since they have been normalized and redistributed, the distributions are approximately rectangular. The lines drawn through each graph show the linear regression line that best fits each. For two-dimensional state spaces, the linear regression manifold is a line. In three dimensions it is a plane, and just such a plane is shown for all three variables simultaneously in Figure 8.4. In more than three dimensions, the manifold is impossible to show but still forms a rigid hyperdimensional surface that fits the points in state space.

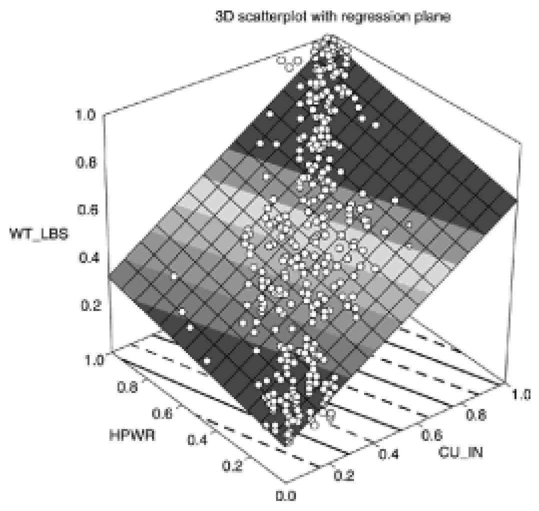


Figure 8.4 Multiple linear regression manifold in a 3D state space. The manifold is a flat plane. The flatness is characteristic of linear regressions in any number of dimensions. The variables used are the same as shown in Figure 8.3.

The linear regression technique involves discovering the *joint* variability of the two variables and using this to determine which values of the predicted variable match values of the predictor variable. *Joint variability*, the measure of the way one variable varies as another varies, allows between-variable inferences to be made of the same sort that were made within-variable in the example above. The shortcoming in the example shown in Figures 8.1 and 8.2 is that when multiple values are missing in a single variable, only a single replacement for all the missing values can be estimated. Preserving between-variable variability allows a suitable value to be found for a missing value in one variable if the value of its partner in the other variable is known. In a sense, linear regression involves finding the joint variability between two variables and preserving it for any needed replacement values. Multiple linear regression does the same thing, but weighs the contributions of several variables' joint distributions to estimate any missing value. With more variables contributing to the joint variability (more evidence if you will), it is usual to find a better estimate of any missing value than can be found by using only one other variable.

Linear Regression

Linear regression is relatively straightforward. It involves no more than discovering a specific expression for the straight line that best fits the data points in state space. The expression describing a straight line is

$$y = a + bx$$

This expression gives the appropriate *y* value for any value of *x*. The expression *a* is a constant that indicates where the straight line crosses the *y*-axis in state space, and *b* is an expression that indicates how much the line increases (or decreases if it is a negative number) its position for an increase in *x*. In other words, to find a value for *y*, start at *a* and go up (or down) by amount *b* for every unit of *x*. Figure 8.5 illustrates the expression for a straight line. Linear regression involves finding the appropriate *a* and *b* values to use in the expression for a straight line that make it best fit through the points in state space. The linear regression comes in two parts, one to find *b* and the other to find *a* when *b* is known. These two expressions look like this:

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

$$a = \bar{y} - b\bar{x}$$

where

$\bar{y}$ is the mean value of y

$\bar{x}$ is the mean value of x

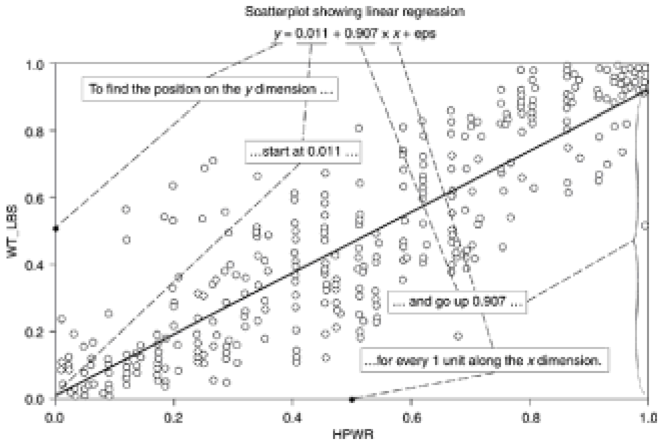


Figure 8.5 Showing the straight-line equation found that best fits the point distribution for the two variables WT_LBS and HPWR from the CARS data set.

Looking at these two expressions shows that the pieces of information required are

n	the number of instance values
Σx	the sum of all the *x* values
Σy	the sum of all the *y* values
Σx^2	the sum of all the values of *x* squared
Σxy	the sum of all *x* values multiplied by all *y* values

Calculating these quantities is simple only if all of the values are actually present. Table 8.3 illustrates the problem.

TABLE 8.3 The effect of missing values (?.??) on the summary values of *x* and *y*.

n	*x*	*y*	*x2*	*y2*	*xy*
1	0.55	0.53	0.30	0.28	0.29
2	0.75	0.37	0.56	0.14	0.28
3	0.32	0.83	0.10	0.69	0.27
4	0.21	0.86	0.04	0.74	0.18
5	0.43	0.54	0.18	0.29	0.23
Sum	**2.26**	**3.13**	**1.20**	**2.14**	**1.25**
1	0.55	0.53	0.30	0.28	0.29
2	*?.??*	0.37	*?.??*	0.14	*?.??*
3	0.32	0.83	0.10	0.69	0.27
4	0.21	*?.??*	0.04	*?.??*	*?.??*
5	0.43	0.54	0.18	0.29	0.23
Sum	**?.??**	**?.??**	**?.??**	**?.??**	**?.??**

The problem is what to do if values are missing when the complete totals for all the values are needed. Regressions simply do not work with any of the totals missing. Yet if any single number is missing, it is impossible to determine the necessary totals. Even a single missing *x* value destroys the ability to know the sums for *x*, x^2, and *xy*! What to do?

Since getting the aggregated values correct is critical, the modeler requires some method to determine the appropriate values, even with missing values. This sounds a bit like pulling one's self up by one's bootstraps! Estimate the missing values to estimate the missing values! However, things are not quite so difficult.

In a representative sample, for any particular joint distribution, the *ratios* between the various values x*x* and x*x*², and x*y* and x*y*² remain constant. So too do the ratios between x*x* and x*xy* and x*y* and x*xy*. When these ratios are found, they are the equivalent of setting the value of *n* to 1. One way to see why this is so is because in any representative sample the ratios are constant, regardless of the number of instance values—and that includes *n* = 1. More mathematically, the effect of the number of instances cancels out. The end result is that when using ratios, *n* can be set to unity. In the linear regression formulae, values are multiplied by *n*, and multiplying a value by 1 leaves the original value unchanged. When multiplying by *n* = 1, the *n* can be left out of the expression. In the calculations that follow, that piece is dropped since it has no effect on the result.

The key to building the regression equations lies in discovering the needed ratios for those values that are jointly present. Given the present and missing values that are shown in the lower part of Table 8.3, what are the ratios?

Table 8.4 shows the ratios determined from the three instance values where *x* and *y* are both present. Using the expressions for linear regression and these ratios, what is the estimated value for the missing *y* value from Table 8.3?

TABLE 8.4 Ratios of the values that are present in the lower part of Table 8.3.

	x*x*2	**x*y*2**	**x*xy***
Ratio x*x* to:	0.45		0.61
Ratio x*y* to:		0.66	0.42

In addition to the ratios, the sums of the *x* and *y* values that are present need to be found. But since the ratios scale to using an *n* of 1, so too must the sums of *x* and *y*—which is identical to using their mean values. The mean values of variable *x* and of variable *y* are taken for the values of each that are jointly present as shown in Table 8.5.

TABLE 8.5 Mean values of *x* and *y* for estimating missing values.

n	***x***	***y***

1	*0.55*	*0.53*
2		0.37
3	*0.32*	*0.83*
4	0.21	
5	*0.43*	*0.54*
Sum	1.30	1.90
Mean	0.43	0.63

For the linear regression, first a value for *b* must be found. Because ratios are being used, the ratio must be used to yield an appropriate value of x$x^2$and x*xy* to use for any value of x*x*. For example, since the ratio of x*x* to xx^2 is 0.45, then given an x*x* of 0.43, the appropriate value of xx^2 is 0.43 x 0.45 = 0.1935—that is, the actual value multiplied by the ratio. Table 8.6 shows the appropriate values to be used with this example of a missing *x* value.

TABLE 8.6 Showing ratio-derived estimated values for x*x*2 and x*xy.*

Est x*x*	**Est xx^2**	**Est x*xy***
0.43	0.43 x 0.45 = 0.1935	0.43 x 0.61 = 0.2623

Plugging these values into the expression to find *b* gives

$$b = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

$$b = \frac{0.2623 - 0.43 \times 0.63}{0.1935 - (0.43)^2}$$

$$b = \frac{0.2623 - 0.2709}{0.1935 - 0.1849}$$

$$b = \frac{-0.0086}{0.0086}$$

$$b = -1.0$$

So *b* = –1. The negative sign indicates that values of *y* will decrease as values of *x* increase. Given this value for *b*, *a* can be found:

$$a = \bar{y} - b\bar{x}$$
$$a = 0.63 - (-1)0.43$$
$$a = 0.63 + 0.43$$
$$a = 1.06$$

The *a* value is 1.06. With suitable values discovered for *a* and *b*, and using the formula for a straight line, an expression can be built that will provide an appropriate estimate for any missing value of *y*, given a value of *x*. That expression is

$y = a + b$x

$y = 1.06 + (–1)x$

$y = 1.06 – x$

Table 8.7 uses this expression to estimate the values of *y*, given *x*, for all of the original values of *x*.

TABLE 8.7 Derived estimates of *y* given an *x* value using linear regression based on ratios.

Original *x*	Original *y*	Estimated *y*	Error
0.55	0.53	0.51	0.02
0.75	0.37	0.31	0.06

0.32	0.83	0.74	0.09
0.21	0.86	0.85	0.01
0.43	0.54	0.63	0.09

These estimates of *y* are quite close to the original values in this example. The error, the difference between the original value and the estimate, is small compared to the actual value.

Multiple Linear Regression

The equations used for performing multiple regression are extensions of those already used for linear regression. They are built from the same components as linear regression—x*x*, xx^2, x*xy*—for every pair of variables included in the multiple regression. (Each variable becomes *x* in turn, and for that *x*, each of the other variables becomes *y* in turn.) All of these values can be estimated by finding the ratio relationships for those variables' values that are jointly present in the initial sample data set. With this information available, good linear estimates of the missing values of any variable can be made using whatever variable instance values are actually present.

With the ratio information known for all of the variables, a suitable multiple regression can be constructed for any pattern of missing values, whether it was ever experienced before or not. Appropriate equations for the instance values that are present in any instance can be easily constructed from the ratio information. These equations are then used to predict the missing values.

For a statistician trying to build predictions, or glean inferences from a data set, this technique presents certain problems. However, the problems facing the modeler when replacing data are very different, for the modeler requires a computationally tractable method that introduces as little bias as is feasible when replacing missing values. The missing-value replacements themselves should contribute no information to the model. What they do is allow the information that is present (the nonempty instance values) to be used by the modeling tool, adding as little extraneous distortion to a data set as possible.

It may seem strange that the replacement values should contribute no information to a data set. However, any replacement value can only be generated from information that is already present in the form of other instance values. The regression equations fit the replacement value in such a way that it least distorts the linear relationships already discovered. Since the replacement value is derived exclusively from information that is already present in the data set, it can only reexpress the information that is already present. New information, being new, changes what is already known to a greater or lesser degree, actually defining the relationship. Replacement values should contribute as little as possible to changing the shape of the relationships that already exist. The existing relationship is what the modeler needs to explore, not some pattern artificially constructed by replacing missing values!

Alternative Methods of Missing-Value Replacement

Preserving joint variability between variables is far more effective at providing unbiased replacement values than methods that do not preserve variability. In practice, many variables do have essentially linear between-variable relationships. Even where the relationship is nonlinear, a linear estimate, for the purpose of finding a replacement for a missing value, is often perfectly adequate. The minute amount of bias introduced is often below the noise level in the data set anyway and is effectively unnoticeable.

Compared to finding nonlinear relationships, discovering linear relationships is both fast and easy. This means that linear techniques can be implemented to run fast on modern computers, even when the dimensionality of a data set is high. Considering the small amount of distortion usually associated with linear techniques, the trade-offs in terms of speed and flexibility are heavily weighted in favor of their use. The replacement values can be generated dynamically (on the fly) at run time and substituted as needed.

However, there are occasions when the relationship is clearly nonlinear, and when a linear estimate for a replacement value may introduce significant bias. If the modeler knows that the relationship exists, some special replacement procedure for missing values can be used. The real problem arises when a significantly nonlinear relationship exists that is unknown to the modeler and domain expert. Mining will discover this relationship, but if there are missing values, linear estimates for replacements will produce bias and distortion. Addressing these problems is outside the scope of the demonstration software, which is intended only to illustrate the principles involved in data preparation.

There are several possible ways to address the problem. Speed in finding replacement values is important for deployed production systems. In a typical small direct marketing application, for instance, a solicitation mailing model may require replacing anything from 1 million to 20 million values. As another example, large-scale, real-time fraud detection systems may need from tens to hundreds of millions of replacement values daily.

Tests of Nonlinearity: Extending the Ratio Method of Estimation

There are tests to determine nonlinearity in a relationship. One of the easiest is to simply try nonlinear regressions and see if the fit is improved as the nonlinearity of the expression increases. This is certainly not foolproof. Highly nonlinear relationships may well not gradually improve their fit as the nonlinearity of the expression is increased.

An advantage of this method is that the ratio method already described can be extended to capture nonlinear relationships. The level of computational complexity increases considerably, but not as much as with some other methods. The difficulty is that choosing the degree of nonlinearity to use is fairly arbitrary. There are robust methods to determine the amount of nonlinearity that can be captured at any chosen degree of nonlinearity without requiring that the full nonlinear multiple regressions be built at every level. This allows a form of optimization to be included in the nonlinearity estimation and capture. However, there is still no guarantee that nonlinearities that are actually present will be captured. The amount of data that has to be captured is quite considerable but relatively modest compared with other methods, and remains quite tractable.

At run time, missing-value estimates can be produced very quickly using various optimization techniques. The missing-value replacement rate is highly dependent on many factors, including the dimensionality of the data set and the speed of the computer, to name only two. However, in practical deployed production systems, replacement rates exceeding 1000 replacements per second, even in large or high-throughput data sets, can be easily achieved on modern PCs.

Nonlinear Submodels

Another method of capturing the nonlinearities is to use a modeling tool that supports such a model. Neural networks work well (described briefly in Chapter 10). In this case, for each variable in the data set, a subsample is created that has no missing values. This is required as unmodified neural networks do not handle missing values—they assume that all inputs have some value. A predictive model for every variable is constructed from all of the other variables, and for the MVPs. When a missing value is encountered, the appropriate model is used to predict its value from the available variable values.

There are significant drawbacks to such a method. The main flaw is that it is impossible to train a network for every possible pattern of missing values. Training networks for all of the detected missing patterns in the sample may itself be an enormous task. Even when done, there is no prediction possible when the population produces a previously unencountered MVP, since there is no network trained for that configuration. Similarly, the storage requirements for the number of networks may be unrealizable.

A modification of this method builds fewer models by using subsets of variables as inputs. If the subset inputs are carefully selected, models can be constructed that among them have a very high probability that at least one of them will be applicable. This approach requires constructing multiple, relatively small networks for each variable. However, such an approach can become intractable very quickly as dimensionality of the data set increases.

An additional problem is that it is hard to determine the appropriate level of complexity. Missing-value estimates are produced slowly at run time since, for every value, the appropriate network has to be looked up, loaded, run, and output produced.

Autoassociative Neural Networks

Autoassociative neural networks are briefly described in Chapter 10. In this architecture, all of the inputs are also used as predicted outputs. Using such an architecture, only a single neural network need be built. When a missing value(s) is detected, the network can be used in a back-propagation mode—but not a training mode, as no internal weights are adjusted. Instead, the errors are propagated all the way back to the inputs. At the input, an appropriate weight can be derived for the missing value(s) so that it least disturbs the internal structure of the network. The value(s) so derived for any set of inputs reflects, and least disturbs, the nonlinear relationship captured by the autoassociative neural network.

As with any neural network, its internal complexity determines the network's ability to capture nonlinear relationships. Determining that any particular network has, in fact, captured the extant nonlinear relationship is difficult. The autoassociative neural network approach has been used with success in replacing missing values for data sets of modest dimensionality (tens and very low hundreds of inputs), but building such networks for moderate- to high-dimensionality data sets is problematic and slow. The amount of data required to build a robust network becomes prohibitive, and for replacement value generation a robust network that actually reflects nonlinearities is needed.

At run time, replacement values can be produced fairly quickly.

Nearest-Neighbor Estimators

Nearest-neighbor methods rely on having the training set available at run time. The method requires finding the point in state space best represented by the partially complete instance value, finding the neighbors nearest to that point, and using some metric to derive the missing values. It depends on the assumption that representative near neighbors can be found despite the fact that one or more dimensional values is missing. This can make it difficult to determine a point in state space that is representative, given that its position in the dimensions whose value is missing is unknown. Nonetheless, such methods can produce good estimates for missing values. Such methods are inherently nonlinear so long as representative near neighbors can be found.

The main drawbacks are that having the training data set available, even in some collapsed form, may require very significant storage. Lookup times for neighbors can be very slow, so finding replacement values too is slow.

Chapter 9: Series Variables

Overview

Series variables have a number of characteristics that are sufficiently different from other types of variables that they need examining in more detail. Series variables are always at least two-dimensional, although one of the dimensions may be implicit. The most common type of series variable is a *time series*, in which a series of values of some feature or event are recorded over a period of time. The series may consist of only a list of measurements, giving the appearance of a single dimension, but the ordering is by time, which, for a time series, is the implicit variable.

The series values are always measured on one of the scales already discussed, nominal through ratio, and are presented as an ordered list. It is the ordering, the expression of the implied variable, that requires series data to be prepared for mining using techniques in addition to those discussed for nonseries data. Without these additional techniques the miner will not be able to best expose the available information. This is because series variables carry additional information within the ordering that is not exposed by the techniques discussed so far.

Up to this point in the book we have developed precise descriptions of features of nonseries data and various methods for manipulating the identified features to expose information content. This chapter does the same for series data and so has two main tasks:

1. Find unambiguous ways to describe the component features of a series data set so that it can be accurately and completely characterized

2. Find methods for manipulating the unique features of series data to expose the information content to mining tools

Series data has features that require more involvement by the miner in the preparation process than for nonseries data. Where miner involvement is required, fully automated preparation tools cannot be used. The miner just has to be involved in the preparation and exercise judgment and experience. Much of the preparation requires visualizing the data set and manipulating the series features discussed. There are a number of excellent commercial tools for series data visualization and manipulation, so the demonstration software does not include support for these functions. Thus, instead of implementation notes concluding the chapter discussing how the features discussed in the chapter are put into practice, this chapter concludes with a suggested checklist of actions for preparing series data for the miner to use.

9.1 Here There Be Dragons!

Mariners and explorers of old used fanciful and not always adequate maps. In unexplored or unknown territory, the map warned of dragons—terrors of the unknown. So it is when preparing data, for the miner knows at least some of the territory. Many data explorers have passed this way. A road exists. Signposts point the way. Maybe the dragons were chased away, but still be warned. “Danger, quicksand!” Trouble lurks inside series data; the road of data preparation is rocky and uncertain, sometimes ending mired in difficulties. It is all too easy to seriously damage data, render it useless, or worse, create wonderful-looking distortions that are but chimera that melt away when exposed to the bright light of reality. Like all explorers faced with uncertainty, the miner needs to exercise care and experience here more than elsewhere. The road is rough and not always well marked. Unfortunately, the existing signposts, with the best of intentions, can still lead the miner seriously astray. Tread this path with caution!

9.2 Types of Series

Nonseries multivariable measurements are taken without any particular note of their ordering. Ordering is a critical feature of a series. Unless ordered, it’s not a series. One of the variables (called the displacement variable, and described in a moment) is always monotonic—either constantly increasing or constantly decreasing. Whether there is one or several other variables in the series, their measurements are taken at defined points on the range of the monotonic variable. The key ordering feature is the change in the monotonic variable as its values change across part or all of its range.

Time series are by far the most common type of series. Measurements of one variable are taken at different times and ordered such that an earlier measurement always comes before a later measurement. For a time series, time is the *displacement variable*—the measurements of the other variable (or variables) are made as time is “displaced,” or changed. The displacement variable is also called the *index* variable. That is because the points along the displacement variable at which the measurements are taken are called the *index points*.

Dimensions other than time can serve as the displacement dimension. Distance, for instance, can be used. For example, measuring the height of the American continent above sea level at different points on a line extending from the Atlantic to the Pacific produces a distance displacement series.

Since time series are the most common series, where this chapter makes assumptions, a time series will be assumed. The issues and techniques described about time series also apply to any other displacement series. Series, however indexed, share many features in common, and techniques that apply to one type of series usually apply to other types of series. Although the exact nature of the displacement variable may make little difference to the preparation and even, to some degree, the analysis of the series itself, it makes all the difference to the interpretation of the result!

9.3 Describing Series Data

Series data differs from the forms of data so far discussed mainly in the way in which the data enfolds the information. The main difference is that the ordering of the data carries information. This ordering, naturally, precludes random sampling since random sampling deliberately avoids, and actually destroys, any ordering. Preserving the ordering is the main reason that series data has to be prepared differently from nonseries data.

There is a large difference between preparing data for modeling and actually modeling the data. This book focuses almost entirely on how to prepare the data for modeling, leaving aside almost all of the issues about the actual modeling, insofar as is practical. The same approach will apply to series data. Some of the tools needed to address the data preparation problems may look similar, indeed are similar, to those used to model and glean information and insight into series data. However, they are put to different purposes when preparing data. That said, in order to understand some of the potential problems and how to address them, some precise method of describing a series is needed. A key question is, What are the features of series data?

To answer this question, the chapter will first identify some consistent, recognizable, and useful features of series data. The features described have to be consistent and recognizable as well as useful. The useful features are those that best help the miner in preparing series data for modeling. The miner also needs these same features when modeling. This is not surprising, as finding the best way to expose the features of interest for modeling is the main objective of data preparation.

9.3.1 Constructing a Series

A series is constructed by measuring and recording a feature of an object or event at defined index points on a displacement dimension.

This statement sufficiently identifies a series for mining purposes. It is not a formal definition but a conceptual description, which also includes the following assumptions:

1. The feature or event is recorded as numerical information.

2. The index point information is either recorded, or at least the displacements are defined.

3. The index, if recorded, is recorded numerically.

It is quite possible to record a time series using alpha labels for the nondisplacement dimension, but this is extremely rare. Numerating such alpha values within the series is possible, although it requires extremely complex methods. While it is very unusual indeed to encounter series with one alpha dimension, it is practically unknown to find a series with an alpha-denominated displacement variable. The displacement dimension has to be at least an ordinal variable (ratio more likely), and these are invariably numerical. Because series with all dimensions numerical are so prevalent, we will focus entirely on those.

It is also quite possible to record multivariable series sharing a common displacement variable, in other words, capturing several features or events at each index mark. An example is collecting figures for sales, backlog, new orders, and inventory level every week. “Time” is the displacement variable for all the measurements, and the index point is weekly. The index point corresponds to the validating event referred to in Chapter 2. There is no reason at all why several features should not be captured at each index, the same as in any nonseries multidimensional data set. However, just as each of the variables can be considered separately from each other during much of the nonseries data preparation process, so too can each series variable in a multidimensional series be considered separately during preparation.

9.3.2 Features of a Series

By its nature a series has some implicit pattern within the ordering. That pattern may repeat itself over a period. Often, time series are thought of by default as repetitive, or cyclic, but there is no reason that any repeating pattern should in fact exist. There is, for example, a continuing debate about whether the stock market exhibits a repetitive pattern or is simply the result of a random walk (touched on later). Enormous effort has been put into detecting any cyclic pattern that may exist, and still the debate continues. There is, nonetheless, a pattern in series data, albeit not necessarily a repeating one. One of the objectives of analyzing series data is to describe that pattern, identify it as recognizable if possible, and find any parts that are repetitive. Preparing series data for modeling, then, must preserve the nature of the pattern that exists. Preparation also includes putting the data into a form in which the desired information is best exposed to a modeling tool. Once again, a warning: this is not always easy!

Before looking at how series data may be prepared, and what problems may be detected and corrected, the focus now turns to finding some way to unambiguously describe the series.

9.3.3 Describing a Series—Fourier

Jean Baptiste Joseph Fourier was not a professional mathematician. Nonetheless, he exerted an influence on mathematicians and scientists of his day second only to that of Sir Isaac Newton. Until Fourier revealed new tools for analyzing data, several scientists lamented that the power of mathematics seemed to be just about exhausted. His insights reinvigorated the field. Such is the power of Fourier’s insight that its impact continues to reverberate in the modern world today. Indeed, Fourier provided the key insights and methods for developing the tools responsible for building the modern technology that we take for granted.

To be fair, the techniques today brought under the umbrella description of Fourier analysis were not all entirely due to Fourier. He drew on the work of others, and subsequent work enormously extends his original insight. His name remains linked to these techniques, and deservedly so, because he had the key insights from which all else flowed.

One of the tools he devised is a sort of mathematical prism. Newton used a prism to discover that white light consists of component parts (colors). Fourier's prism scatters the information in a series into component parts. It is a truly amazing device that hinges on two insights: waves can be added together, and adding enough simple sine and cosine waves of different frequencies, phases, and amplitudes together is sufficient to create any series shape. *Any* series shape!

When adding waveforms together, several things can be varied. The three key items are

- The frequency, or how many times a waveform repeats its pattern in a given time

- The phase, that is, where the peaks and troughs of a wave occur in relation to peaks and troughs of other waves

- The amplitude, or distance between the highest and lowest values of a wave

Figure 9.1 shows how two waveforms can be added together to produce a third. The frequency simply measures how many waves, or cycles, occur in a given time. The top two waveforms are symmetrical and uniform. It is easy to see where they begin to repeat the previous pattern. The two top waveforms also have different frequencies, which is shown by the identified wavelengths tracing out different lengths on the graph. The lower, composite waveform cannot, just by looking at it, positively be determined to have completed a repeating pattern in the width of the graph.

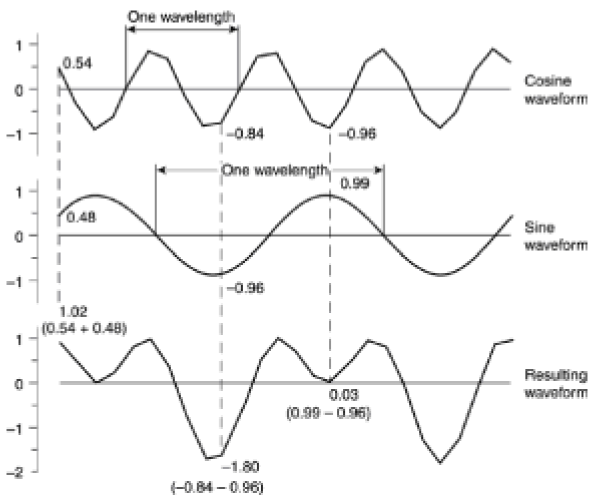


Figure 9.1 The addition of two waveforms shows how to create a new waveform. The values of the consine and sine waveforms are added together and the result plotted. The resulting wave-form may look nothing like the source that created it.

Figure 9.2 shows two waveforms that are both complete cycles, and both are identical in length. The waveforms illustrate the sine and the cosine functions:

y = sine(*x*°)

and

y = cosine(*x*°)

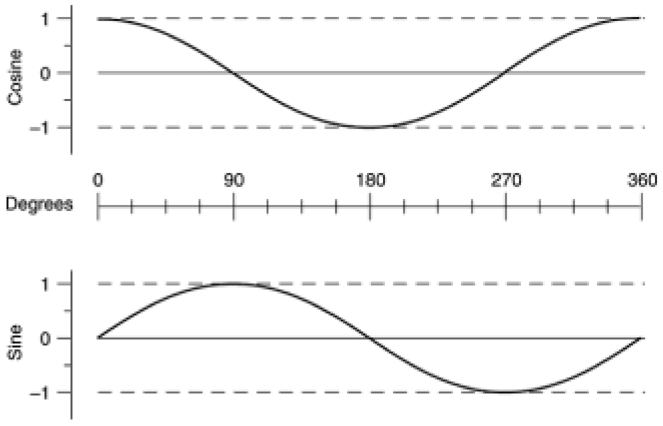


Figure 9.2 Values of the functions “sine” and “cosine” plotted for the number of degrees shown on the central viniculum.

When used in basic trigonometry, both of these functions return specific values for any number of degrees. They are shown plotted over 360°, the range of a circle. Because 0° represents the same circular position as 360° (both are due north on a compass, for example), this has to represent one complete cycle for sine and cosine waveforms—they begin an identical repetition after that point. Looking at the two waveforms shows that the sine has identical values to the cosine, but occurring 90° later (further to the right on the graph). The sine is an identical waveform to the cosine, shifted 90°. “Shifted” here literally means moved to the right by a distance corresponding to 90°. This shift is called a *phase shift*, and the two waveforms are said to be 90° out of phase with each other.

The two upper images in Figure 9.3 show the effect of changing amplitude. Six sine and cosine waveforms, three of each, are added together. The frequencies of each corresponding waveform in the two upper images are identical. All that has changed is the amplitude of each of the waveforms. This makes a very considerable difference to the resulting waveform shown at the bottom of each image. The lower two images show the amplitudes held constant, but the frequency of each contributing waveform differs. The resulting waveforms—the lower waveform of each frame—show very considerable differences.

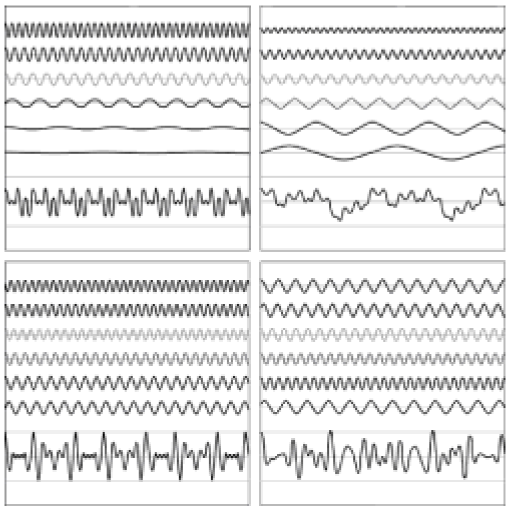


Figure 9.3 These four images show the result of summing six waveforms. In the top two images, the frequencies of the source waveforms are the same; only their amplitude differs. In both of the two lower images, all waveforms have similar amplitude.

It was Fourier’s insight that by combining enough of these two types of waveforms, varying their amplitude, phase, and frequency as needed, any desired resultant waveform can be built. Fourier analysis is the “prism” that takes in a complex waveform and “splits” it into its component parts—just as a crystal prism takes in white light and splits it into the various colors. And just as there is only one rainbow of colors, so too, for any specific input waveform, there is a single “rainbow” of outputs.

A Fourier analysis provides one way of uniquely describing a series. In Figure 9.4, Fourier analysis illustrates the prism effect—splitting a composite waveform into components. Also in Figure 9.4, Fourier synthesis shows how the reverse effect works—reassembling the waveform from components.

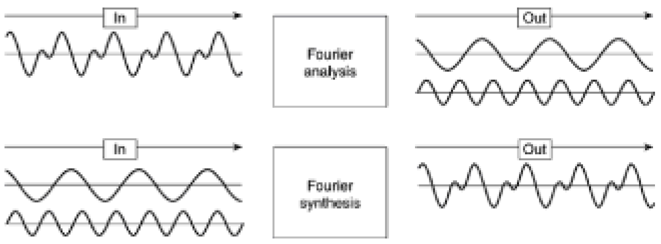


Figure 9.4 Fourier analysis takes in a complex waveform and yields an expression giving the appropriate component sine and cosine expressions, together with their amplitude, frequency, and phase, to re-create the analyzed waveform. Fourier synthesis takes the analyzed expression and yeilds the composite waveform.

9.3.4 Describing a Series—Spectrum

A *spectrum* is normally thought of as an array of colors—“the colors of the spectrum.” For light, that exactly describes a spectrum, but an infinite variety of different spectra exist. When sunlight passes through a prism, it breaks into a band showing the array of all colors possible from white light. If, however, a colored light beam passes through a prism, the resulting spectrum does not show all of the possible colors equally brightly. Depending on the exact color of the original colored light beam, all of the possible colors may be present. The brightest intensity of color in the spectrum will correspond to the apparent color of the incoming beam. Because light is a form of energy, the energy of the brightest portion of the spectrum contains the most energy. For light, a spectrum shows the energy distribution of an incoming light beam—brightest color, highest energy.

Fourier analysis also allows a spectrum to be generated. The previous section explained that any shape of waveform can be built out of component parts—various individual sine and cosine waveforms of specific frequency, phase, and amplitude. (Fourier analysis produces information about which [sine/cosine] waveforms are present in the series, which frequencies are present, and how strong [amplitude] each of the component parts is.) Figure 9.5 shows a cosine waveform in the upper image, with the associated “power” spectrum below. The power in this case relates to the most prevalent component waveform. The cosine waveform is “pure” in the sense that it consists of entirely one wavelength and is uniform in amplitude. When producing the spectrum for this waveform, there is a single spike in the spectrum that corresponds to the frequency of the waveform. There are no other spikes, and most of the curve shows zero energy, which is to be expected from a pure waveform.

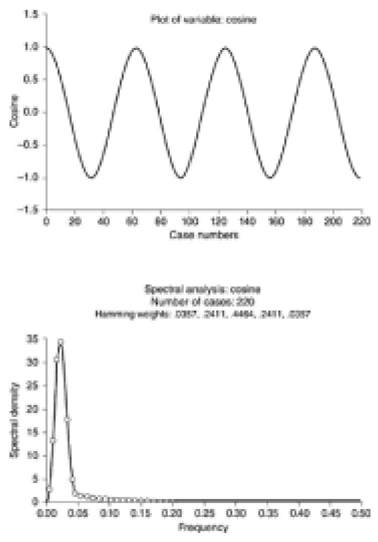


Figure 9.5 A pure cosine waveform of uniform amplitude and fequency (top) and the frequency spectrum for this waveform (bottom).

What happens if there are several frequencies present? Figure 9.6, in the top image, shows a composite waveform created from the six waveforms shown above it. When a spectral analysis is made of the composite waveform, shown in the lower image, there are six spikes. The height of each spike corresponds to the amplitude of each component waveform, and the position along the *x* (horizontal) axis of the graph corresponds to the frequency of the component waveform. The spectrum shows clearly that there are six component frequencies, declining approximately in amplitude from left to right. Inspection of the upper image reveals that the upper six waveforms, the components of the lower composite waveform, increase in frequency from top to bottom, and also decrease in amplitude from top to bottom. So the spectrum accurately reflects the way the analyzed composite waveform was actually constructed. In this example, Figures 9.5 and 9.6 show a spectrum for single and composite waveforms that consist of “clean” components. What does a spectrum look like for a noisy signal?

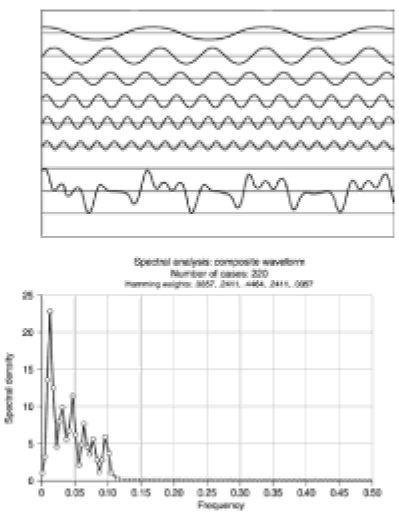


Figure 9.6 Six components of a composite waveform (top) the composite waveform itself is shown as the lowest waveform—and the power spectrum for the composite waveform (bottom).

Figure 9.7 uses the same composite signal built of six components. Considerable noise is added to the waveform. The top-left image graphs the noise. This is a random waveform, varying in value between x1. The bottom-left image shows the power spectrum for the all-noise waveform. The power is distributed fairly evenly along the bottom of the graph. This indicates that there is a fairly equal amount of power present at all wavelengths (frequencies). However, even though this is randomly generated noise, some frequencies, by chance, have more power than others, as shown by the fact that the graphed power spectrum has peaks and valleys. Since this is known to be random noise, the peaks of power are known positively to be spurious in this case. (They are present as shown in the sample of noise generated for the example, but another, identically generated random sample would have minor fluctuations in totally unpredictable places.)

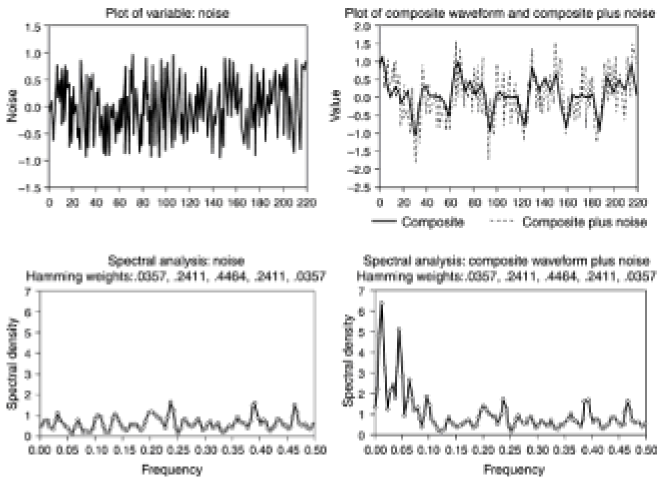


Figure 9.7 Composite signal built of six components showing graph of the random noise (top left), composite waveform with random noise (top right), and two power spectrums (lower left and right).

In any case, the level of power is low relative to the lower-right image of Figure 9.7. The composite waveform, together with the noise added to it, is shown in the upper-right image. The power spectrum in the lower-right image still shows the peaks corresponding to the six component waveforms, but the noise obscures exactly how many there are and precisely where they are located. Adding noise, in other words, "blurred" the original waveform.

So far, all of the waveforms examined have no trend. A *trend* is a noncyclic, monotonically increasing or decreasing component of the waveform. Figure 9.8 shows the composite waveform with an increasing trend in the top image. The bottom image shows the spectrum for such a trended waveform. The power in the trend swamps the detail. The peak at 0 on the *x*-axis is very large compared with the power shown in the other spectra. (Most of the spectral images share a common vertical scale within each figure. The vertical scale of this image has to be much larger than in other images in other figures to show the amplitude of the energy present.) Clearly this causes a problem for the analysis, and dealing with it is discussed shortly.

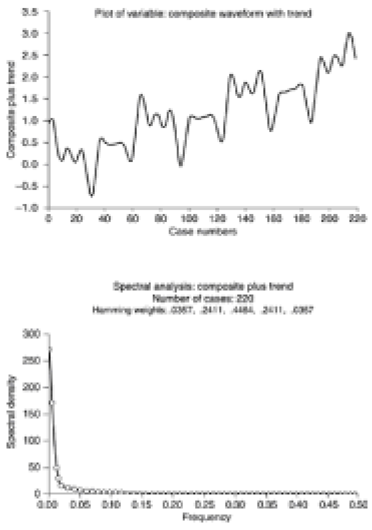


Figure 9.8 Adding a trend to the composite waveform makes it rise overall over time. The rise appears quite modest (top), but the power in the trend component completely swamps any other detail in the power spectrum (bottom).

In these examples, almost all of the waveforms discussed are produced by sine and cosine functions. Distorting noise, not deliberately produced by sine or cosine functions, is added, but the underlying waveforms are regularly cyclical. Figure 9.9 looks at the spectral analysis of a waveform generated at random. This is the type of shape known as a “random walk.” The random walk shown in the top image starts at 0. From whatever point it is at, it moves from that position either up or down at random, and for a distance of between 0–1 chosen at random, to reach its next position. Each step in distance and direction that the waveform takes, starts wherever the last one ended. (Although removing trend has not been discussed yet, because the random walk shows an apparent trend that causes a spectral analysis problem, the waveform has been detrended before the spectrum was produced. Detrending is discussed later in this chapter.)

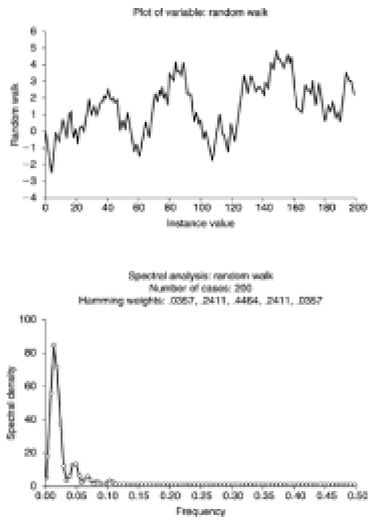


Figure 9.9 A waveform generated by a random selection of the next direction and distance in which to move from every point (top) and a spectral analysis, which shows high energy at some frequencies (bottom).

The spectral analysis in the lower image shows that one frequency predominates in this random walk. Indeed, Fourier analysis describes this waveform as an assemblage of sine and cosine functions. Does this mean that the random walk is predictable? Unfortunately not. Describing an existing waveform, and predicting some future shape for a waveform, are entirely different activities. This waveform is randomly shaped. Random numbers have a distribution. That distribution may be rectangular, normal, or some other shape. It may change over time. It may even be completely unknown. Regardless, it is nonetheless present, and, since a distribution has some structure, even truly random numbers drawn from this distribution carry evidence of the structure of the distribution. This shows up in the spectral analysis. In this case, no prediction at all can be made about the future progress of this particular series. Some inferences about various probabilities can be made, but no valid predictions.

But surely, if inferences can be made, predictions can be made too! As an example of the difference between an *inference* and a *prediction*, observe that each discrete step in the random walk is never more than one unit in each direction. Furthermore, by adding the assumption that the future will continue to be like the past, we can infer that each step will continue to be less than one unit in each direction. This still does not lead to a prediction of any future value. The best that can be said is that next direction and distance of change will remain less than one unit, positive or negative. We can estimate some probability of any particular value being the next value based on the observed past distribution of step direction and sizes. Even then, unless one knows the causative mechanism that is generating the size and direction of the positive and negative steps, that too can change at any time, invalidating the assumption. Even though there is a structure, the next-step direction, for instance, is no more certain than the outcome of the flip of a fair coin.

How to tell, then, if this is a random walk, or a genuine cyclic pattern? That too is a very fraught question. There are tests for randomness, and some “fingerprints” of randomness are discussed later in this chapter. Even so, it is very easy to “discover” meaningless patterns. Aspects of series data preparation, even as opposed to analysis, depend to some extent on discovering patterns. Meaningless patterns are worse than useless, and as discussed later, they may be positively damaging. Here indeed lurk dragons, chimera, and quicksand!

9.3.5 Describing a Series—Trend, Seasonality, Cycles, Noise

A form of time series analysis known as *classical decomposition* looks at the series as being built from four separate components: trend, seasonality, cycles, and noise. Three of these components turned up during the discussion of spectral analysis. Regarding the components as separate entities helps in dealing with data preparation problems. For instance, in the example used in Figure 9.8, adding a trend component to a cyclical component swamped any other information in the power spectra. In building a description of how power spectra describe waveforms, it was convenient and natural to describe the components in terms of trend, cycles, and noise. Since these components will be manipulated in preparing series data, a slightly closer look at each of them is useful, as well as a look at what “seasonality” might be.

Trend moves in a consistent direction. That is, it is monotonically unidirectional—either never decreasing or never increasing. If never decreasing, it is increasing, and although the rate at which it increases may vary over time, even to 0, it never falls below 0. Similarly, if decreasing, the rate at which it decreases may vary over time, but while it may be flat (fails to actually decrease), it never actually increases. Should the trend line moving in one direction change to the other direction, it is regarded as a different trend, or perhaps as a long period cycle.

Note that what is regarded as a trend over one time period may be a cycle over a different period. For instance, sales, fraud, or some other measure may increase in an upward trend over a year or two. Over 10 years that same trend may be seen to be a part of a larger cyclical pattern such as the business or economic cycle. This piece of the overall pattern, then, would be seen as a trend over 2 years, and a cycle over 10. This distinction is valid, not arbitrary, as trend components have to be treated differently from cyclical components. So it is true that the difference between finding a trend and finding a cycle may depend entirely on the period that the data covers.

Seasonality reflects the insight that, regardless of any other trend, cycle, or noise influence, certain seasons are inherently different. In time series, seasons are often exactly that—seasons of the year. For example, regardless of economic conditions and other factors, consumers spend more in late December than at other times of the year. Although it appears to be a cyclic effect, it isn't. It is caused by a phenomenon that is local to the season: Christmas. Although Christmas *occurs* cyclically, it is not a cyclic event itself. That is to say, Christmas doesn't wax and wane over the course of the year. What is the level of "Christmasness" in, say, June, July, or August? Is the change in Christmasness part of a cycle during those months? The answer is no, since Christmas is not a cyclical phenomenon but a seasonal one. For instance, to understand how a June sales campaign performed relative to a December campaign, the effect of Christmas occurring in December needs to be removed before attempting to make a fair comparison.

What in a time series are called seasonal effects do occur in other types of displacement series, although they are usually much harder to intuitively understand. Where they do occur, it usually requires a domain expert to identify the seasonalities.

Cycles are fluctuations in the level of the series that have some identifiable repetitive form and structure. Cycles represent the "what goes around, comes around" part of the series. So long as what is going around and coming around can be positively identified, and it reoccurs over some defined period—even if the period itself changes over time—it forms a cycle. Cycles aren't necessarily based on, or thought of as, a collection of sine and cosine waves. That is only the way that Fourier analysis looks at cycles—indeed, at whole waveforms. Very useful, but not necessary. It is only necessary to be able to identify the shape of the repetitive component.

Noise has also been discussed. Chapter 2 looked at some of the sources of noise that in series data can appear in a number of guises. Figure 9.7 shows it as a distortion added to a signal, hiding the underlying structure of the signal. There it is seen as a sort of "fog." Figure 9.9, on the other hand, shows noise as the generating process of a waveform. Seeing through the murk of Figure 9.7 is the problem there. Detecting that it is murk at all, and that it's all murk, is the problem in Figure 9.9!

Noise is the component that is left after the trend, cyclic, and seasonal components have been extracted. It is irregular, for if it were not, it would be characterized as something else. Even so, it has characteristics. Noise comes in different types, colors actually. There is gray noise, pink noise, white noise, and blue noise, to name but a few. The shades are named by analogy with the color spectrum. The color appellation describes, by analogy, the frequency distribution of the noise. Blue light is at the high-frequency end of the color spectrum. So too, blue noise has a power distribution weighted towards the higher frequencies. White light's energy is evenly distributed at all frequencies—so too with white noise. Just as with blue and white noise, the other noise-shaded frequencies share their relation with the color spectrum.

Noise can be generated from a variety of sources. In the physical world, different processes tend to generate noise with different "signatures"—characteristic frequency distributions. So it is too with noise in behavioral data. Because a noise source may have a characteristic signature, which may be seen by looking at a power spectrum, noise sources can sometimes be identified. If the noise characteristics are known and constant, it can make filtering a waveform out of the noise much easier. (A brief introduction to simple filtering techniques is discussed shortly.)

9.3.6 Describing a Series—Autocorrelation

Correlation measures how values of one variable change as values of another variable change. There are many types of correlation. *Linear correlation* measures the closeness to linear of the between-variable relationship. (Chapter 8 discussed linear relationships in terms of linear regression.) If two variables vary in such a way that the relationship is exactly linear, then, knowing of the linear relationship, and knowing the value of one of the variables, a 100% confident prediction can be made of the value of the other. For instance, the "two times table" has a perfectly correlated, linear relationship between the number to be multiplied and the result. Knowing the value of one variable, "the number to be multiplied," is enough to completely define the value of the other variable, "the result."

Correlation is expressed as a number ranging between +1 and –1. A correlation of x1 indicates perfect predictability. The linear correlation between the two variables in the two times table example is +1. When positive, a linear correlation of ±1 says not only that the two variables are completely linearly related, but also that they move in the same direction. As one becomes more positive, so does the other. A linear correlation of –1 indicates a perfectly predictable relationship, but the values of the variables move in opposite directions—one getting more positive, the other more negative. The "minus two times table" (multiplying any number by –2) would have such a correlation.

Figure 8.5 in the last chapter explained how the equation for a straight line can be interpreted. However, the line on that graph does not join all of the points, which are shown by small circles. The data points cluster about the line, but the fit is certainly not perfect. Knowing one variable's value gives some idea of the value of the other variable, but not an exact idea. Under these circumstances the correlation is less than 1, but since there is some relationship, the correlation is not 0. A correlation of 0 means that knowing the value of one variable tells nothing about the value of the other variable.

As a general guide, until the correlation gets outside the range of between +0.3 to –0.3, any connection is tenuous at best. Not until correlations get to be greater than 0.8, or less than –0.8, do they begin to indicate a good fit. A difficulty in understanding what correlation says about the “goodness of fit” between two variables is that the relationship between correlation and the goodness of fit itself is not linear! When trying to understand what correlation reveals about the goodness of fit between two variables, perhaps a more useful measure is the amount of “explanatory power” one variable has about the value of another. This explanatory power represents a linear relationship of how well one variable’s value explains another variable’s value. This value, commonly denoted by the symbol r^2 (technically known as the *sample coefficient of determination*), is the square of the correlation. The square of a number between 0 and 1 is *smaller* than the original number. So a correlation of 0.3 represents a r^2 of 0.3 x 0.3 = 0.09, a very small value that indicates little explanatory power indeed. Even a correlation of 0.8 only represents an explanatory power of 0.8 x 0.8 = 0.64. Note that r^2 can never have a negative value since the square of any number is always positive and r^2 ranges in value from 0 to 1. Correlation values are most commonly used and quoted, but it is well to keep in mind that the strength of the connection is perhaps more intuitively represented by the square of the correlation.

Having looked at correlation, autocorrelation follows naturally. *Autocorrelation* literally means correlation with self. When used as a series description, it is a measure of how well one part of the series correlates with another part of the same series some fixed number of steps away. The distance between index points on the series is called the *lag*. An autocorrelation with a lag of one measures the correlation between every point and its immediate neighbor. A lag of two measures the correlation between every point and its neighbor two distant. Figure 9.10 shows how the data points can be placed into the columns of a matrix so that each column has a different lag from the first column. Using this matrix, it is easy to find the linear correlation between each column, each corresponding to a different lag.

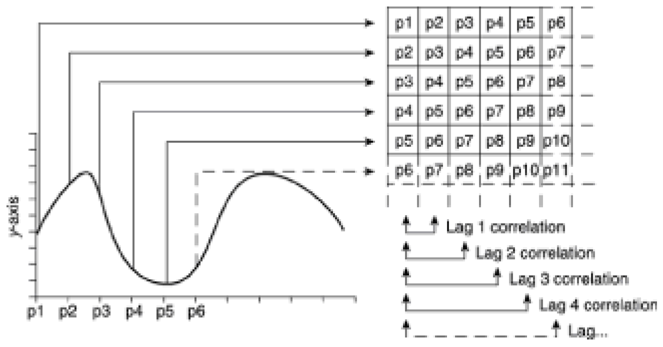


Figure 9.10 Building a matrix to find linear correlation for many lags. Every additional column is one point further lagged from the first column. Finding the correlation between every column gives the data needed to build a correlogram.

The result of building a multiple lag autocorrelation is a correlogram. A *correlogram* measures, and shows in graphical form, the correlation for each of many different lags. This is done by plotting the linear correlation for a lag of one, then for a lag of two, then three, and so on. Figure 9.11 shows a series of the waveforms used so far, together with their associated correlograms. Comparison between the correlograms and the spectra for the same waveforms shows that different features about the waveforms are emphasized by each descriptive technique.

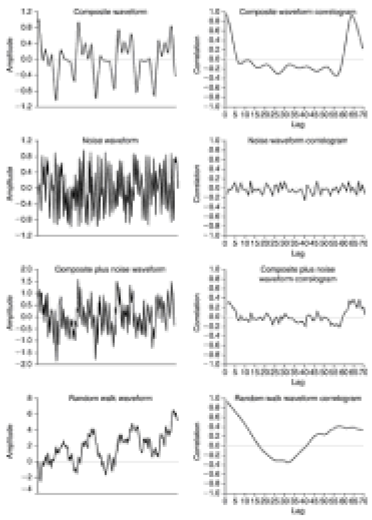


Figure 9.11 Waterforms and their correlograms.

9.4 Modeling Series Data

Given these tools for describing series data, how do they help with preparing the data for modeling? There are two main approaches to modeling series data. One uses extensions to the descriptive tools just discussed, and attempts to understand and explicate the within-variable and between-variable interactions in terms of the series itself. The other approach decomposes the data, using something like a reverse pivot (described in Chapter 4), and uses nonseries tools to further analyze the data.

Both of these approaches try to understand the interactions of the data over time. The first approach tries to understand the data in time, whereas the second tries to understand the effect of time. For instance:

- Approach 1 shows how sales/fraud/claims vary over time.
- Approach 2 shows what the effect of the trend/season/cycle/noise of sales/fraud/claims has on inferences/predictions about backlog/losses/payments.

Deciding which approach, or deciding to use both approaches, does have an impact on data preparation. However, it mainly affects data set assembly, rather than other aspects of preparing series data. In general, the same problems need to be addressed regardless of which approach is used. For instance, and this is discussed more fully later, a trend is a monotonic component of a series. Monotonicity is a problem that needs to be addressed whether or not it is a series that is being modeled, and regardless of which type of series model is being built.

9.5 Repairing Series Data Problems

Series data shares most of the problems of nonseries data, plus some additional ones of its own. Where the problems are similar, some of the preparation techniques are similar too. Some are not. Even those that are similar may need to be implemented differently. The important point for the miner to remember about series data is maintaining the information contained in the ordering.

9.5.1 Missing Values

While a poorly constructed series can contain empty values, it is very unlikely. Missing values cause a major problem in series data, and it is very unlikely that any series would be constructed to permit empty values. (Chapter 2 discusses the difference between missing and empty values.) In any case, whether missing or empty, series modeling techniques fare even worse with values that are absent than nonseries techniques.

There are two dimensions of a series in which a value could be missing: the feature variable, and the index variable. Index variable problems are addressed next. Here attention will be confined to a value missing in the feature variable.

When replacing a missing value in nonseries data, joint variability is preserved between variables, and a suitable value for the replacement is found using the information that is contained in whatever variable values are present. If the series is a multivariable series, that technique works here too. Instead of multiple regression, a multiple autoregression is used to find a replacement value. The concept represents a straightforward extension of multiple regression, described in Chapter 8, combined with autocorrelation described earlier in this chapter.

Autoregression measures the self-similarity of the waveform across different lags. Just as with single linear or multiple regression, so too can autoregressions be determined using the ratio method (Chapter 8). Using multiple autoregression techniques for replacing missing values can provide a robust estimate for missing values. However, chimerical dragons rear their heads when doing this!

Often, time series, both in physical and in behavioral data sets, tend to have contiguous missing values—that is, runs of values all missing. This can easily happen if the collection mechanism either fails or is intermittent in operation. Then there may occur runs of data with periods of no data in between. Filling in these holes with self-similar patterns from other parts of the series reinforces the apparent self-similarity. The estimated missing values will be smoothed (smoothing is mentioned later) by the replacement process itself since they represent some sort of aggregated estimate of the missing value. Unfortunately for missing-value replacement, smoothing enhances any regular patterns in the data that are being used to make the estimate. So replacing missing values necessarily enhances a pattern that is discovered elsewhere in the series and used to replace the missing runs of data.

This means that not only is a missing run replaced with an aggregate pattern borrowed from elsewhere, it is an aggregate pattern *enhanced* by smoothing! When the prepared data is later modeled, this enhanced pattern may be, almost certainly will be, “discovered.” Depending on the length of the replaced run, it may predominate in a spectral analysis or correlogram, for instance.

One way to ameliorate the problem to some extent is to add noise to the replacement values. A glance at the correlograms in Figure 9.11 shows that the noise added to the composite waveform considerably masks the strength of the correlations at various lags. But how much noise to add? One estimate for additional noise to add is to use the same level that is estimated to exist in the values that are present. And what is that level of noise? That question, although examined later, does not always have an easy answer.

This is a very tough chimerical dragon to slay! The miner needs to look carefully at which patterns are being reinforced by the replacement of missing values, and be very circumspect in deciding later that these are at all meaningful when they are later “discovered” during modeling. It is, unfortunately, quite possible that the training, test, and evaluation data sets all suffer from the same replacement problem, and thus, seem to confirm the “discovery” of the pattern. This chimera can be very persuasive; caution is the watchword!

9.5.2 Outliers

Chapter 2 first mentioned outliers—a few values that lie far from the bulk of the range. Outliers occur in time series too. They come as individual occurrences and, sometimes, in “runs”—clusters of consecutive values of the same order of magnitude as each other, but as a group lying well outside the range of the other values. The miner will need to ask hard questions about why the outliers exist. Are they really significant? What sort of process could account for them? Can they be translated back into the normal range if they are indeed errors?

If no rationale can account for the outliers, with all of the same caveats mentioned for missing values, as a last resort replace the outliers, exactly as for missing values.

9.5.3 Nonuniform Displacement

Usually, although not invariably, displacement steps are spaced uniformly across the indexing dimension—measurements, in other words, taken in regular and uniform increments. Many of the analytical techniques assume this uniformity and won’t work well, or at all, if the displacement is not constant. Since almost all techniques, including spectral analysis and correlogramming, assume uniform displacement between indexes, the values must be adjusted to reflect what they would have been had they been taken with uniform displacement.

Figure 9.12 shows a sine wave sampled at nonuniform displacements. Graphing it as if the displacements were uniform clearly shows jagged distortion. The less uniform the displacement intervals, the worse the situation becomes. Since the jagged waveform appears affected by noise, techniques for removing the noise work well to estimate the original waveform as it would have been if sampled at uniform displacements.

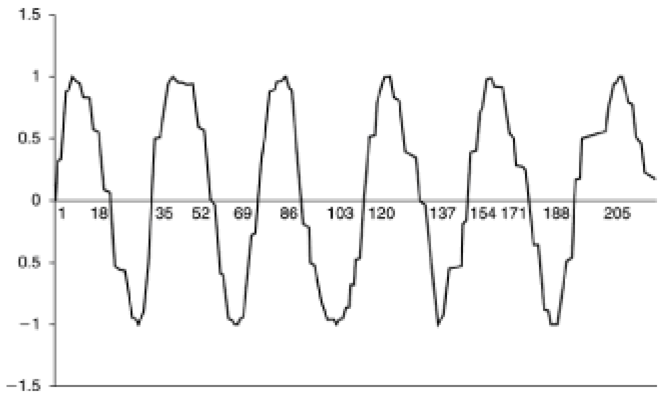


Figure 9.12 A sine wave that is sampled with a nonconstant displacement. When reproduced as if the intervals were constant, the waveform becomes distorted.

9.5.4 Trend

Trend is the monotonically increasing or decreasing component of a waveform. Leaving a trend present in a waveform causes problems for almost all modeling methods. If the trend is nonlinear, then removing the linear component leaves only the nonlinear component of the trend. The nonlinear component then appears cyclic. Figure 9.13 shows how this can be done. In the left image, performing a linear regression linearly approximates the nonlinear trend. Subtracting the linear component from the trend is shown in the right image. This leaves the cyclic part of the trend, but there is no trend remaining—or at least, stated differently, what there is of the trend now appears to be cyclic.

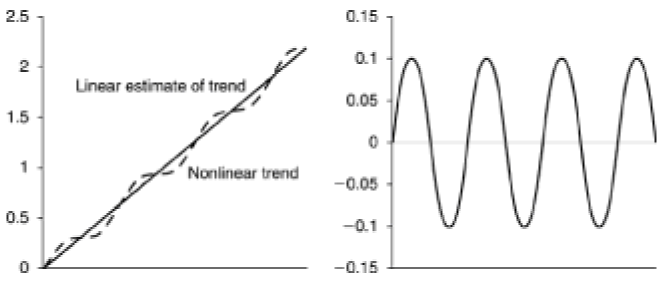


Figure 9.13 A nonlinear trend and a linear estimates found by linear regression (left). Subtracting the linear part of the trend from the nonlinear part turns the nonlinear part of the trend, in the case, into a cyclic representation (right).

Detrending a series is obviously, at least on occasion, an absolute necessity. (Figure 9.8 showed how trend can swamp the spectrum.) None of the modeling techniques discussed can deal with the problem, whether it is called monotonicity or trend. But detrending has dangers. Figure 9.14 shows a cosine waveform. This is a perfectly cyclic waveform that oscillates uniformly about the 0 point. A cosine waveform actually has a completely flat trend component since, if extended far enough, it has a completely symmetrical distribution about the 0 point. Here, as much of the waveform as is available has been linearly detrended. The “discovered” linear trend shown apparently has a downward slope. Subtracting this slope from the “trended” waveform distorts the waveform. Using such a distorted waveform in modeling the data leads to many problems, and bad—frequently useless—models. Detrending nontrended data can do enormous damage. But can the miner avoid it?

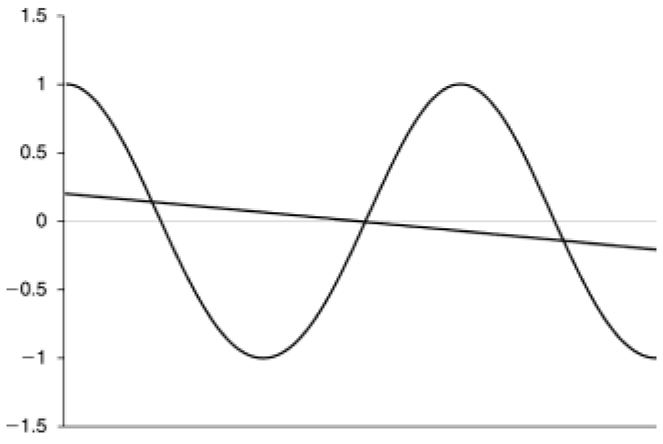


Figure 9.14 A cosine waveform and straight line indicating a trend even though the waveform actually has no trend. This is caused by using less than a complete waveform.

The problem in Figure 9.14 is that only partial cycles were used. In the example, which was constructed specifically to show the problem, this is obvious. In real-world data it is often very much harder, impossible even, to determine if apparent trend is an artifact of the data or real. There is no substitute for looking at the data in the form of data plots, correlograms, spectra, and so on. Also, as always, the miner should try to determine if there is a rationale for any discovery. Always the miner should ask, “In this case, is there some reason to expect the existence of the discovered trend?” More than that, if there is a positive known reason that precludes the existence of trend, data should not be detrended—even if it appears to have a trend. Look back at the random walk, say, Figure 9.11. This waveform was entirely generated by additive noise processes—accumulated errors. Suppose that this additive error had contaminated the series that otherwise contained no trend. Detrending this type of waveform can make the cyclic information impossible for a modeling tool to discover. Over the long run, such additive noise as is shown averages to 0—that is, over the long haul it is trendless. Including trend as part of the model not only hides the cyclic information, but also adds a nonexistent trend to the predictions. On the other hand, it may be almost impossible to work with a waveform that is not detrended.

Deep quicksand here! The only real answer is to experiment! Survey the data extensively, trended and detrended. If access to surveying software is difficult, at least build multiple models and work extensively with the data, both trended and detrended.

9.6 Tools

Removing trend involves identifying and removing a component of the overall waveform. Doing this makes the remaining waveform more convenient or tractable to handle, or better reveals information of concern for modeling. But removing trend is really just a special case in a set of manipulations for exposing series information. These manipulations—filtering, moving averages, and smoothing—are the miner’s basic series manipulation tools.

9.6.1 Filtering

A *filter* is a device that selectively holds some things back and lets other pass. In the case of series data, the filtering is performed on different components of the overall waveform. Since a complex waveform can be thought of as being constructed from simpler waveforms, each of a separate single frequency, the components can be thought of as those simpler waveforms. A wide array of filters can be constructed. High-pass filters, for example, let only high frequencies through, “holding back” the lower frequencies. Actually the “holding back” is known as attenuation. *Attenuation* means “to make less,” and the lower-frequency amplitude is actually reduced, rather than held back, leaving the higher frequencies more visible. If selected components at different frequencies are attenuated, their amplitude is reduced. Changing the amplitude of component cycles changes the shape of the waveform, as shown in Figure 9.3. By using filters, various parts of the frequency spectrum can be removed from the overall waveform and investigated separately from the effects of the remaining frequencies.

While it is possible to construct complex mathematical structures to perform the necessary filtering, the purpose behind filtering is easy to understand and to see.

Figure 9.8 showed the spectrum of a trended waveform. Almost all of the power in this spectrum occurs at the lowest frequency, which is 0. With a frequency of 0, the corresponding waveform to that frequency doesn't change. And indeed, that is a linear trend—an unvarying increase or decrease over time. At each uniform displacement, the trend changes by a uniform amount. Removing trend corresponds to low-frequency filtering at the lowest possible frequency—0. If the trend is retained, it is called *low-pass filtering* as the trend (the low-frequency component) is "passed through" the filter. If the trend is removed, it would be called *high-pass filtering* since all frequencies but the lowest are "passed through" the filter.

In addition to the zero frequency component, there are an infinite number of possible low-frequency components that are usefully identified and removed from series data. These components consist of fractional frequencies. Whereas a zero frequency represents a completely unvarying component, a fractional frequency simply represents a fraction of the whole cycle. If the first quarter of a sine wave is present in a composite waveform, for example, that component would rise from 0 to 1, and look like a nonlinear trend.

Some of the more common fractional frequency components include exponential growth curves, logistic function curves, logarithmic curves, and power-law growth curves, as well as the linear trend already discussed. Figure 9.15 illustrates several common trend lines. Where these can be identified, and a suitable underlying generating mechanism proposed, that mechanism can be used to remove the trend. For instance, taking the logarithm of all of the series values for modeling is a common practice for some series data sets. Doing this removes the logarithmic effect of a trend. Where an underlying generating mechanism cannot be suggested, some other technique is needed.

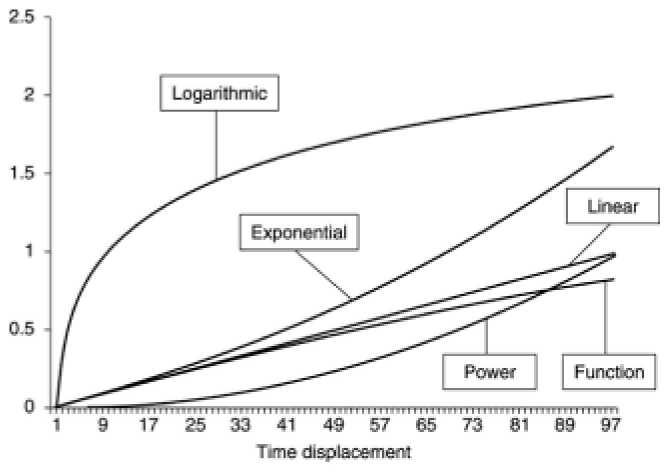


Figure 9.15 Several low-frequency components commonly discovered in a series data that can be beneficially identified and removed.

9.6.2 Moving Averages

Moving averages are used for general-purpose filtering, for both high and low frequencies. Moving averages come in an enormous range and variety. To examine the most straightforward case of a simple moving average, pick some number of samples of the series, say, five. Starting at the fifth position, and moving from there onward through the series, use the average of that position plus the previous four positions instead of the actual value. This simple averaging reduces the variance of the waveform. The longer the period of the average, the more the variance is reduced. With more values in the weighting period, the less effect any single value has on the resulting average.

TABLE 9.1 Log-five SMA

Position	Series value	SMA5	SMA5 range
1	0.1338		
2	0.4622		
3	0.1448	0.2940	1-5
4	0.6538	0.3168	2-6
5	0.0752	0.3067	3-7
6	0.2482	0.3497	4-8
7	0.4114	0.3751	5-9
8	0.3598	0.4673	6-10
9	0.7809		
10	0.5362		

9.1 shows a lag-five *simple moving average (SMA)*. The values are shown in the column “Series value,” with the value of the average in the column “SMA5.” Each moving average value is the average of the two series values above it, the one series value opposite and the two next series values, making five series values in all. The column “SMA5 range” shows which positions are included in any particular moving average value.

One drawback with SMAs, especially for long period weightings, is that the average cannot begin to be calculated until the number of periods in the weighting has passed. Also, the average value refers to the data point that is at the center of the weighting period. (Table 9.1 plots the average of positions 1–5 in position 3.) With a weighting period of, say, five days, the average can only be known as of two days ago. To know the moving average value for today, two days have to pass.

Another potential drawback is that the contribution of each data point is equal to that of all the other data points in the weighting period. It may be that the more distant past data values are less relevant than more recent ones. This leads to the creation of a *weighted moving average (WMA)*. In such a construction, the data values are weighted so that the more recent ones contribute more to the average value than earlier ones. Weights are chosen for each point in the weighting period such that they sum to 1.

Table 9.2 shows the weights for constructing the lag-five WMA that is shown in Table 9.3. The “v_{-4} indicates that the series value four steps back is used, and the weight “0.066” indicates that the value with that lag is multiplied by the number 0.066, which is the weight. The lag-five WMA’s value is calculated by multiplying the last five series values by the appropriate weights.

TABLE 9.2 Weight for calculating a lag-five WMA.

Log	Weight
V_{-4}	0.576766
V_{-3}	0.423234
V_{-2}	0.576766
V_{-1}	0.423234
V_0	0.576766

Wt total	1.000

TABLE 9.3 Log-five WMA

Position	Series value	WMA5
1	0.1338	
2	0.4622	
3	0.1448	
4	0.6538	0.2966
5	0.0752	0.2833
6	0.2482	0.3161
7	0.4114	0.3331
8	0.3598	0.4796
9	0.7809	0.5303
10	0.5362	

Table 9.3 shows the actual average values. Because of the weights, it is difficult to “center” a WMA. Here it is shown “centered” one advanced on the lag-five SMA. This is done because the weights favor the most recent values over the past values—so it should be plotted to reflect that weighting.

Exponential moving averages (EMAs) solve the delay problem. Such averages consist of two parts, a “head” and a “tail.” The tail value is the previous average value. The head value is the current data value. The average's value is found by moving the tail some way closer to the head, but not all of the way. A weight is applied to decide how far to move the tail toward the head. With light tail weights, the tail follows the head quite closely, and the average behaves much like a short weighting period simple moving average. With heavier tail weights, the tail moves more slowly, and it behaves somewhat like a longer-period SMA. The head weight and the tail weight taken together must always sum to a value of 1.

No two averages behave in exactly the same way, but for EMAs, obviously the heavier the head weight, the "faster" the EMA value will move—that is to say, the more closely it follows the value of the series. For comparison, the EMA weights shown in Table 9.4 approximate the lag-five SMA.

TABLE 9.4 Head and tail weights to approximate a lag-five SMA.

Head weight	0.576766
Head weight	0.423234

Table 9.5 shows the actual values for the EMA. In this table, position 1 of the EMA is set to the starting value of the series. The formula for determining the present value of the EMA is

$$V_{EMA0} = (V_{s0} \times W_h) + (V_{EMA-1} \times W_t)$$

where

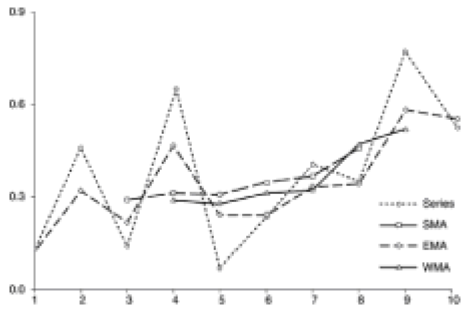
V_{EMA0}	is the value of the current EMA
V_{s0}	is the current series value
W_h	is the head weight
V_{EMA-1}	is the last value of the EMA
W_t	is the tail weight

TABLE 9.5 Values of the EMA

Position	Series value	EMA	Head	Tail
1	0.1338	0.1338		
2	0.4622	0.3232	0.2666	0.0566
3	0.1448	0.2203	0.0835	0.1956
4	0.6538	0.4703	0.3771	0.0613
5	0.0752	0.2424	0.0434	0.2767
6	0.2482	0.2458	0.1432	0.0318
7	0.4114	0.3413	0.2373	0.1051
8	0.3598	0.3519	0.2075	0.1741
9	0.7809	0.5993	0.4504	0.1523
10	0.5362	0.5629	0.3092	0.3305

This formula, with these weights, specifies that the current average value is found by multiplying the current series value by 0.576766, and the last value of the average by 0.423243. The results are added together. The table shows the value of the series, the current EMA, and the head and the tail values.

Figure 9.16 illustrates the moving averages discussed so far, and the effects of changing the way they are constructed. The series itself changes value quite abruptly, and all of the averages change more slowly. The SMA is the slowest to change of the averages shown. The WMA moves similarly to the SMA, but clearly responds more to the recent values, exactly as it is constructed to do.



Moving average formulae:

$$\text{SMA} = \frac{\sum d_{n-2} \dots d_{n+2}}{5}$$

$$\text{WMA} = d_{n-2}w_1 + d_{n-1}w_2 + d_n w_3 + d_{n+1}w_4 + d_{n+2}w_5$$

$$w_1 = 0.066 \; w_2 = 0.132 \; w_3 = 0.198 \; w_4 = 0.264 \; w_5 = 0.34$$

$$\text{EMA}_n = (d_n \times 0.576766) + (\text{EMA}_{n-1} \times 0.423234)$$

where

d_n is series data point n

w_n is weight for data point n

Figure 9.16 Various moving averages and the effects of changing weights showing SMAs, WMAs (weights shown separately), and EMAs (weights included in formula). The graph illustrates the data shown in Tables 9.1, 9.2, and 9.5.

The EMA is the most responsive to the actual series value of the three averages shown. Yet the weights were chosen to make it approximate the lag-five SMA average. Since they seem to behave so differently, in what sense are these two approximately the same? Over a longer series, with this set of weights, the EMA tends to be centered about the value of the lag-five SMA. A series length of 10, as in the examples, is not sufficient to show the effect clearly.

In general, as the lag periods get longer for SMAs and WMAs, or the head weights get lighter (so the tail weights get heavier) for the EMAs, the average reacts more slowly to changes in the series. Slow changes correspond to longer wavelengths, and longer wavelengths are the same as lower frequencies. It is this ability to effectively change the frequency at which the moving average reacts that makes them so useful as filters.

Although specific moving averages are constructed for specific purposes, for the examples that follow later in the chapter, an EMA is the most convenient. The convenience here is that given a data value (head), the immediate EMA past value (tail), and the head and tail weights, then the EMA needs no delay before its value is known. It is also quick and easy to calculate.

Moving averages can be used to separate series data into two frequency domains—above and below the threshold set by the reactive frequency of the moving average. How does this work in practice?

Moving Averages as Filters—Removing Noise

The composite-plus-noise waveform, first shown in Figure 9.7, seems to have a slower cycle buried in higher-frequency noise. That is, buried in the rapid fluctuations, there appears to be some slower fluctuation. Since this is a waveform built especially for the example, this is in fact the case. However, nonmanufactured signals often show this type of noise pattern too. Discovery of the underlying signal starts by trying to remove some of the noise. Using an EMA, the high frequencies can be separated from the lower frequencies.

High frequencies imply an EMA that moves fast. The speed of reaction of an EMA is set by adjusting its weights. In this case, the head weight is set at 0.44 so that it moves very fast. However, because of the tail weight, it cannot follow the fastest changes in the waveform—and the fastest changes are the highest frequencies. The path of the EMA itself represents the waveform without the higher frequencies. To separate out just the high frequencies, subtract the EMA from the original waveform. The difference is the high-frequency component missing from the EMA trace. Figure 9.17 shows the original waveform, waveform plus noise, EMA, and high frequencies remaining after subtraction. Using an EMA with a head weight of 0.44 better resembles the original signal than the noisy version because it has filtered out the high frequencies. Subtracting the EMA from the noisy signal leaves the high frequencies removed by the EMA (top).

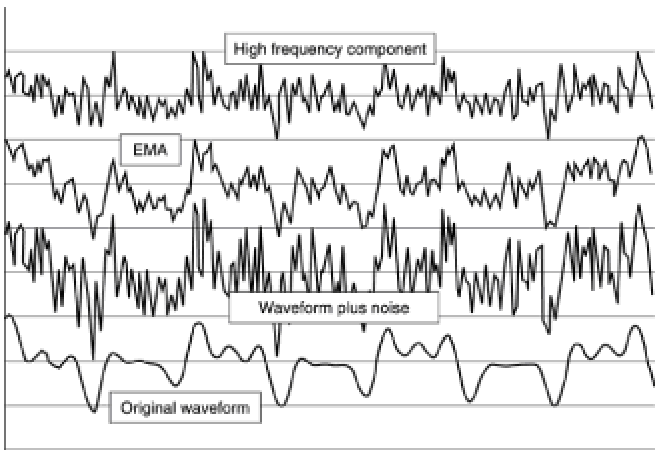


Figure 9.17 The original waveform, waveform plus noise, EMA, and high frequencies remaining after subtraction.

It turns out that with this amount of weighting, the EMA is approximately equivalent to a three-sample SMA (SMA3). An SMA3 has its value centered over position two, the middle position. Doing this for the EMA used in the example recovers the original composite waveform with a correlation of about 0.8127, as compared to the correlation for the signal plus noise of about 0.6.

9.6.3 Smoothing 1—PVM Smoothing

There are many other methods for removing noise from an underlying waveform that do not use moving averages as such. One of these is *peak-valley-mean (PVM)* smoothing. Using PVM, a *peak* is defined as a value higher than the previous and next values. A *valley* is defined as a value lower than the previous and next values. PVM smoothing uses the mean of the last peak and valley (i.e., (P + V)/2) as the estimate of the underlying waveform, instead of a moving average. The PVM retains the value of the last peak as the current peak value until a new peak is discovered, and the same is true for the valleys. This is the shortest possible PVM and covers three data points, so it is a lag-three PVM. It should be noted that PVMs with other, larger lags are possible.

Figure 9.18 shows in the upper image the peak, valley, and mean values. The lower image superimposes the recovered waveform on the original complex waveform without any noise added. Once again, as with moving averages, the recovered waveform needs to be centered appropriately. Centering again is at position two of three, halfway along the lag distance, as from there it is always the last and next positions that are being evaluated. The recovery is quite good, a correlation a little better than 0.8145, very similar to the EMA method.

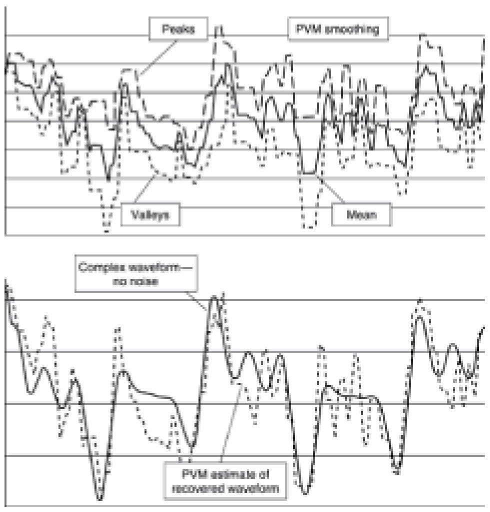


Figure 9.18 PVM smoothing: the peak, valley, and mean values for the composite-plus-noise waveform (top) and the mean estimate superimposed on the actual composite waveform (bottom).

9.6.4 Smoothing 2—Median Smoothing, Resmoothing, and Hanning

Median smoothing uses “windows.” A *window* is a group of some specific number of contiguous data points. It corresponds to the lag distance mentioned before. The only difference between a window and a lag is that the data in a window is manipulated in some way, say, changed in order. A lag implies that the data is not manipulated. As the window moves through the series, the oldest data point is discarded, and a new one is added. When *median smoothing*, use the median of the values in the window in place of the actual value. A *median* is the value that comes in the middle of a list of values ordered by value. When the window is an even length, use as the median value the average of the two middle values in the list. In many ways, median smoothing is similar to average smoothing except that the median is used instead of the average. Using the median makes the smoothed value less sensitive to extremes in the window since it is always the middle value of the ordered values that is taken. A single extreme value will never appear in the middle of the ordered list, and thus does not affect the median value.

Resmoothing is a technique of smoothing the smoothed values. One form of resmoothing continues until there is no change in the resmoothed waveform. Other resmoothing techniques use a fixed number of resmooths, but vary the window size from smoothing to smoothing.

Hanning is a technique borrowed from computer vision, where it is used for image smoothing. Essentially it is a form of weighted averaging. The window is three long, left in the original order, so it is really a lag. The three data points are multiplied by the weights 0.25, 0.50, 0.25, respectively. The hanning operation removes any final spikes left after smoothing or resmoothing.

There are very many types of resmoothing. A couple of examples of the technique will be briefly examined. The first, called “3R2H,” is a median smooth with a window of three, repeated (the “R” in the name) until no change in the waveform occurs; then a median smoothing with a window length of two; then one hanning operation. When applied to the example waveform, this smoothing has a correlation with the original waveform of about 0.8082.

Another, called “4253H” smoothing, has four median smoothing operations with windows of four, two, five, and three, respectively, followed by a hanning operation. This has a correlation with the original example waveform of about 0.8030. Although not illustrated, both of these smooths produce a waveform that appears to be very similar to that shown in the lower image of Figure 9.18.

Again, although not illustrated, these techniques can be combined in almost any number of ways. Smoothing the PVM waveform and performing the hanning operation, for example, improves the fit with the original slightly to a correlation of about 0.8602.

9.6.5 Extraction

All of these methods remove noise or high-frequency components. Sometimes the high-frequency components are not actually noise, but an integral part of the measurement. If the miner is interested in the slower interactions, the high-frequency component only serves to mask the slower interactions. Extracting the slower interactions can be done in several ways, including moving averages and smoothing. The various smoothing and filtering operations can be combined in numerous ways, just as smoothing and hanning the PVM smooth shows. Many other filtering methods are also available, some based on very sophisticated mathematics. All are intended to separate information in the waveform into its component parts.

What is extracted by the techniques described here comes in two parts, high and lower frequencies. The first part is the filtered or smoothed part. The remainder forms the second part and is found by subtracting the first part, the filtered waveform, from the original waveform. When further extraction is made on either, or both, of the extracted waveforms, this is called *reextraction*. There seems to be an endless array of smoothing and resmoothing, extraction and reextraction possibilities!

Waveforms can be separated in high-, middle-, and low-frequency components—and then the separated components further separated. Here is where the miner must use judgment. Examination of the extracted waveforms is called for—indeed, it is essential. The object of all filtering and smoothing is to separate waveforms with pattern from noise. The time to stop is when the extraction provides no additional separation. But how does the miner know when to stop?

This is where the spectra and correlograms are very useful. The noise spectrum (Figure 9.7) and correlogram (Figure 9.11) show that noise, at least of the sort shown here, has a fairly uniform spectrum and uniformly low autocorrelation at all lags. There still might be useful information contained in the waveform, but the chance is small. This is a good sign that extra effort will probably be better placed elsewhere. But what of the random walk? Here there is a strong correlation in the correlogram, and the spectrum shows clear peaks. Is there any way to determine that this is random walking?

9.6.6 Differencing

Differencing a waveform provides another powerful way to look at the information it contains. The method takes the difference between each value and some previous value, and analyzes the differences. A lag value determines exactly which previous value is used, the lag having the same meaning as mentioned previously. A lag of one, for instance, takes the difference between a value and the immediately preceding value.

The actual differences tend to appear noisy, and it is often very hard to see any pattern when the difference values are plotted. Figure 9.19 shows the lag-one difference plot for the composite-plus-noise waveform (left). It is hard to see what, if anything, this plot indicates about the regularity and predictability of the waveform! Figure 9.19 also shows the lag-one difference plot for the complex waveform without noise added (right). Here it is easy to see that the differences are regular, but that was easy to see from the waveform itself too—little is learned from the regularity shown.

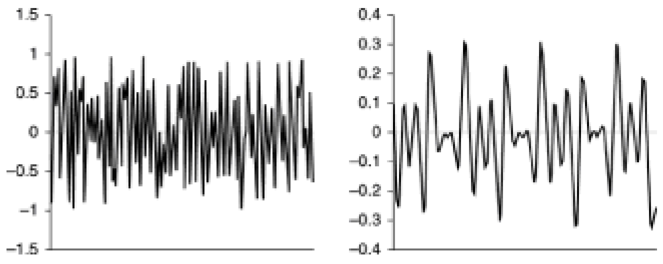


Figure 9.19 Log-one difference plots: composite-plus-noise waveform differences (left) and pattern of differences for the composite waveform without noise (right).

Forward Differencing

Looking at the spectra and correlograms of the lag-one difference plots, however, does reveal information. When first seen, the spectra and correlograms shown in Figure 9.20 look somewhat surprising. It is worth looking back to compare them with the nondifferenced spectra for the same waveforms in Figures 9.6, 9.7, and 9.9, and the nondifferenced correlograms in Figure 9.11.

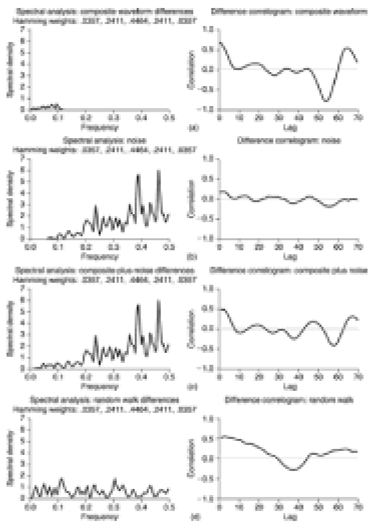


Figure 9.20 Differences spectra and correlograms for various waveforms.

Figure 9.20(a) shows that the differenced composite waveform contains little spectral energy at any of the frequencies shown. What energy exists is in the lower frequencies as before. The correlogram for the same waveform still shows a high correlation, as expected.

In Figure 9.20(b), the noise waveform, the differencing makes a remarkable difference to the power spectrum. High energy at high frequencies—but the correlogram shows little correlation at any lag.

Although the differenced noise spectrum in Figure 9.20(b) is remarkably changed, it is nothing like the spectrum for the differenced random walk in 9.20(d). Yet both of these waveforms were created from random noise. What is actually going on here?

Randomness Detector?

What is happening that makes the random waveforms produce such different spectra? The noise power spectrum (shown in Figure 9.7) is fairly flat. Differencing it, as shown in Figure 9.20(b), amplified—made larger—the higher frequencies. In fact, the higher the frequency, the more the amplification. At the same time, differencing attenuated—made smaller—the lower frequencies. So differencing serves as a high-pass filter.

What of the random walk? The random walk was actually constructed by taking random noise, in the form of numbers in the range of –1 to +1, and adding them together step by step. When this was differenced, back came the original random noise used to generate it. In other words, creating a walk, or "undifferencing," serves to amplify the low frequencies and attenuate the high frequencies—exactly the opposite of differencing! Building the random walk obviously did something that hid the underlying nature of the random noise used to construct it. When differenced, the building process was undone, and back came a spectrum characteristic of noise. So, to go back to the question, "Is there a way to tell that the random walk is generated by a random process?" the answer is a definite "maybe." Differencing can at least give some clues that the waveform was generated by some process that, at least by this test, looks random.

There is no way to tell from the series itself if the random walk is in fact random. That requires knowing the underlying process in the real world that is actually responsible for producing the series. The numbers used here, for instance, were not actually random, but what is known as pseudo-random. (Genuinely random numbers turn out to be fiendishly difficult to come by!) A computer algorithm was used that has an internal mechanism that produces a string of numbers that pass certain tests for randomness. However, the sequence is actually precisely defined, and not random at all. Nonetheless, it looks random, and lacking an underlying explanation, which may or may not be predictive, it is at least known to have some of the properties of a random number. Simply finding a spectrum indicating possible randomness only serves as a flag that more tests are needed. If it eventually passes enough tests, this indeed serves as a practical definition of randomness. What constitutes "enough" tests depends on the miner and the needs of the application. But nonetheless, the working definition of randomness for a series is simply one that passes all the tests of randomness and has no underlying explanation that shows it to be otherwise.

Reverse Differencing (Summing)

Interestingly, discovering a way to potentially expose random characteristics used the reverse process of differencing. Building the random walk required adding together random distance and direction steps generated by random noise. It turns out that creating any series in a similar way is the equivalent of reverse differencing! (This, of course, is summing—the exact opposite of taking a difference. "Reverse differencing" seems more descriptive.) Without going into details, the power spectrum and correlogram for the reverse-differenced composite-plus-noise waveform is shown in Figure 9.21. The power spectrum shows the low-frequency amplification, high-frequency attenuation that is the opposite effect of forward differencing. The correlogram is interesting as the correlation curve is much stronger altogether when the high-frequency components are attenuated. In this case, the reverse-differenced curve becomes very highly autocorrelated—in other words, highly predictable.

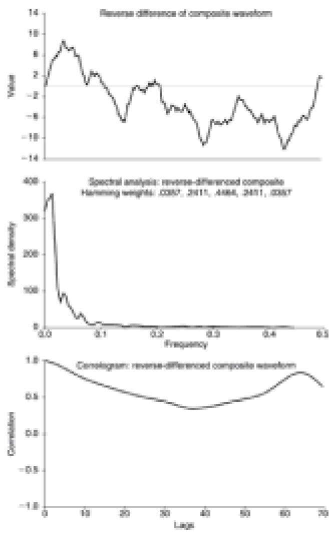


Figure 9.21 Effects of reverse differencing. Low frequencies are enhanced, and high frequencies are attenuated.

Just as differencing can yield insights, so too can summing. Linearly detrending the waveform before the summing operation may help too.

9.7 Other Problems

So far, the problems examined have been specific to series data. The solutions have focused on ways of extracting information from noisy or distorted series data. They have involved extracting a variety of waveforms from the original waveform that emphasize particular aspects of the data useful for modeling. But whatever has been pulled out, or extracted, from the original series, it is still in the form of another series. It is quite possible to look at the distribution of values in such a series exactly as if it were not a series. That is to say, taking care not to actually lose the indexing, the variable can be treated exactly as if it were a nonseries variable. Looking at the series this way allows some of the tools used for nonseries data to be applied to series data. Can this be done, and where does it help?

9.7.1 Numerating Alpha Values

As mentioned in the introduction to this chapter, numeration of alpha values in a series presents some difficulties. It can be done, but alpha series values are almost never found in practice. On the rare occasions when they do occur, numerating them using the nonseries techniques already discussed, while not providing an optimal numeration, does far better than numeration without any rationale. Random or arbitrary assignment of values to alpha labels is always damaging, and is just as damaging when the data is a series. It is not optimal because the ordering information is not fully used in the numeration. However, using such information involves projecting the alpha values in a nonlinear phase space that is difficult to discover and computationally intense to manipulate. Establishing the nonlinear modes presents problems because they too have to be constructed from the components cycle, season, trend, and noise. Accurately determining those components is not straightforward, as we have seen in this chapter. This enormously compounds the problem of in-series numeration.

The good news is that, with time series in particular, it seems easier to find an appropriate rationale for numerating alpha values from a domain expert than for nonseries data. Reverse pivoting the alphas into a table format, and numerating them there, is a good approach. However, the caveat has to be noted that since alpha numerated series occur so rarely, there is little experience to draw on when preparing them for mining. This makes it difficult to draw any hard and fast general conclusions.

9.7.2 Distribution

As far as distributions are concerned, a series variable has a distribution that exists without reference to the ordering. When looked at in this way, so long as the ordering—that is, the index variable—is not disturbed, the displacement variable can be redistributed in exactly the same manner as a nonseries variable. Chapter 7 discussed the nature of distributions, and reasons and methods for redistributing values. The rationale and methods of redistribution are similar for series data and may be even more applicable in some ways. There are time series methods that require the variables' data to be centered (equally distributed above and below the mean) and normalized. For series data, the distribution should be normalized after removing any trend.

When modeling series data, the series should, if possible, be what is known as stationary. A *stationary* series has no trend and constant variance over the length of the series, so it fluctuates uniformly about a constant level.

Redistribution Modifying Waveform Shape

Redistribution as described in Chapter 7, when applied to series variables' data, goes far toward achieving a stationary series. Any series variable can be redistributed exactly as described for nonseries. However, this is not always an unambiguous blessing! (More dragons.) Whenever the distribution of a variable is altered, the transform required is captured so that it can always be undone. Indeed, the PIE-O has to undo any transformation for any output variables. However, it may be that the exact shape of the waveform is important to the modeling tool. (Only the modeler is in a position to know for sure if this is the case at modeling time.) If so, the redistribution may introduce unwanted distortion. In Figure 9.22, the top-left image shows a histogram of the distribution of values for the sine wave. Redistribution creates a rectangular distribution, shown in the top-right image. But redistribution changes the nature of the shape of the wave! The lower image shows both a sine wave and the wave shape after redistribution. Redistribution is intended to do exactly what is seen here—all of the nonlinearity has been removed. The curved waveform is translated into a linear representation—thus the straight lines. This may or may not cause a problem. However, the miner must be aware of the issue.

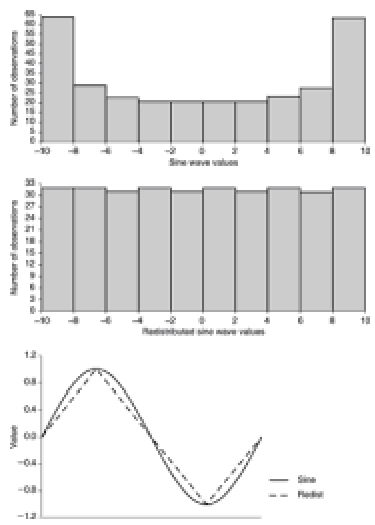


Figure 9.22 Redistributing the distribution linearizes the nonlinear waveform. As the distribution of a pure sine wave is adjusted to be nearer rectangular, so the curves are straightened. If maintaining the wave shape is important, some other transform is required.

Distribution Maintaining Waveform Shape

Redistribution goes a long way toward equalizing the variance. However, some other method is required if the wave shape needs to be retained. If the variance of the series changes as the series progresses, it may be possible to transform the values so that the variance is more constant. Erratic fluctuations of variance over the length of the series cause more problems, but may be helped by a transformation. A "Box-Cox" transformation (named after the people who first described it) may work well. The transform is fairly simple to apply, and is as follows:

$$v_t = \frac{v_n^{\lambda} - 1}{\lambda}$$

where

v_t is transformed value

v_n is original value

λ is a user-selected value

When the changing variance is adjusted, the distribution still has to be balanced. A second transform accomplishes this. The second transform subtracts the mean of the transformed variable from each transformed value, and divides the result by the standard deviation. The formula for this second transformation is

$$v_s = \frac{v_{t_1} - \overline{v}_{t_1}}{\sigma_{v_{t_1}}}$$

where

v_{t_1} is the value after the first transform

v_s is standardized value

$\overline{v}_{t_1}$ is mean value of variable v_{t_1}

$\sigma_{v_{t_1}}$ is standard deviation of variable v_{t_1}

The index, or displacement, variable should not be redistributed, even if it is of unequal increments.

9.7.3 Normalization

Normalization over the range of 0 to 1 needs no modification. The displacement variable can be normalized using exactly the same techniques (described in Chapter 7) that work for nonseries data.

9.7 Other Problems

So far, the problems examined have been specific to series data. The solutions have focused on ways of extracting information from noisy or distorted series data. They have involved extracting a variety of waveforms from the original waveform that emphasize particular aspects of the data useful for modeling. But whatever has been pulled out, or extracted, from the original series, it is still in the form of another series. It is quite possible to look at the distribution of values in such a series exactly as if it were not a series. That is to say, taking care not to actually lose the indexing, the variable can be treated exactly as if it were a nonseries variable. Looking at the series this way allows some of the tools used for nonseries data to be applied to series data. Can this be done, and where does it help?

9.7.1 Numerating Alpha Values

As mentioned in the introduction to this chapter, numeration of alpha values in a series presents some difficulties. It can be done, but alpha series values are almost never found in practice. On the rare occasions when they do occur, numerating them using the nonseries techniques already discussed, while not providing an optimal numeration, does far better than numeration without any rationale. Random or arbitrary assignment of values to alpha labels is always damaging, and is just as damaging when the data is a series. It is not optimal because the ordering information is not fully used in the numeration. However, using such information involves projecting the alpha values in a nonlinear phase space that is difficult to discover and computationally intense to manipulate. Establishing the nonlinear modes presents problems because they too have to be constructed from the components cycle, season, trend, and noise. Accurately determining those components is not straightforward, as we have seen in this chapter. This enormously compounds the problem of in-series numeration.

The good news is that, with time series in particular, it seems easier to find an appropriate rationale for numerating alpha values from a domain expert than for nonseries data. Reverse pivoting the alphas into a table format, and numerating them there, is a good approach. However, the caveat has to be noted that since alpha numerated series occur so rarely, there is little experience to draw on when preparing them for mining. This makes it difficult to draw any hard and fast general conclusions.

9.7.2 Distribution

As far as distributions are concerned, a series variable has a distribution that exists without reference to the ordering. When looked at in this way, so long as the ordering—that is, the index variable—is not disturbed, the displacement variable can be redistributed in exactly the same manner as a nonseries variable. Chapter 7 discussed the nature of distributions, and reasons and methods for redistributing values. The rationale and methods of redistribution are similar for series data and may be even more applicable in some ways. There are time series methods that require the variables' data to be centered (equally distributed above and below the mean) and normalized. For series data, the distribution should be normalized after removing any trend.

When modeling series data, the series should, if possible, be what is known as stationary. A *stationary* series has no trend and constant variance over the length of the series, so it fluctuates uniformly about a constant level.

Redistribution Modifying Waveform Shape

Redistribution as described in Chapter 7, when applied to series variables' data, goes far toward achieving a stationary series. Any series variable can be redistributed exactly as described for nonseries. However, this is not always an unambiguous blessing! (More dragons.) Whenever the distribution of a variable is altered, the transform required is captured so that it can always be undone. Indeed, the PIE-O has to undo any transformation for any output variables. However, it may be that the exact shape of the waveform is important to the modeling tool. (Only the modeler is in a position to know for sure if this is the case at modeling time.) If so, the redistribution may introduce unwanted distortion. In Figure 9.22, the top-left image shows a histogram of the distribution of values for the sine wave. Redistribution creates a rectangular distribution, shown in the top-right image. But redistribution changes the nature of the shape of the wave! The lower image shows both a sine wave and the wave shape after redistribution. Redistribution is intended to do exactly what is seen here—all of the nonlinearity has been removed. The curved waveform is translated into a linear representation—thus the straight lines. This may or may not cause a problem. However, the miner must be aware of the issue.

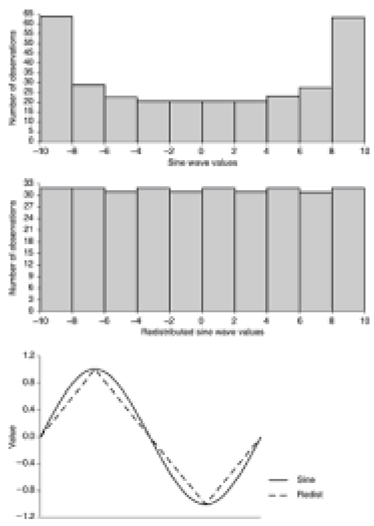


Figure 9.22 Redistributing the distribution linearizes the nonlinear waveform. As the distribution of a pure sine wave is adjusted to be nearer rectangular, so the curves are straightened. If maintaining the wave shape is important, some other transform is required.

Distribution Maintaining Waveform Shape

Redistribution goes a long way toward equalizing the variance. However, some other method is required if the wave shape needs to be retained. If the variance of the series changes as the series progresses, it may be possible to transform the values so that the variance is more constant. Erratic fluctuations of variance over the length of the series cause more problems, but may be helped by a transformation. A “Box-Cox” transformation (named after the people who first described it) may work well. The transform is fairly simple to apply, and is as follows:

$$v_t = \frac{v_n^{\lambda} - 1}{\lambda}$$

where

v_t is transformed value

v_n is original value

λ is a user-selected value

When the changing variance is adjusted, the distribution still has to be balanced. A second transform accomplishes this. The second transform subtracts the mean of the transformed variable from each transformed value, and divides the result by the standard deviation. The formula for this second transformation is

$$v_s = \frac{v_{t_1} - \bar{v}_{t_1}}{\sigma_{v_{t_1}}}$$

where

v_{t_1} is the value after the first transform

v_s is standardized value

$\bar{v}_{t_1}$ is mean value of variable v_{t_1}

$\sigma_{v_{t_1}}$ is standard deviation of variable v_{t_1}

The index, or displacement, variable should not be redistributed, even if it is of unequal increments.

9.7.3 Normalization

Normalization over the range of 0 to 1 needs no modification. The displacement variable can be normalized using exactly the same techniques (described in Chapter 7) that work for nonseries data.

9.8 Preparing Series Data

A lot of ground was covered in this chapter. A brief review will help before pulling all the pieces together and looking at a process for actually preparing series data.

- Series come in various types, of which the most common by far is the time series. All series share a common structure in that the ordering of the measurements carries information that the miner needs to use.

- Series data can be completely described in terms of its four component parts: trend, cycles, seasonality, and noise. Alternatively, series can also be completely described as consisting of sine and cosine waveforms in various numbers and of various amplitudes, phases, and frequencies. Tools to discover the various components include Fourier analysis, power spectra, and correlograms.

- Series data are modeled either to discover the effects of time or to look at how the data changes in time.

- Series data shares all the problems that nonseries data has, plus several that are unique to series.
  - — Missing values require special procedures, and care needs to be taken not to insert a pattern into the missing values by replicating part of a pattern found elsewhere in the series.
  - — Nonuniform displacement is dealt with as if it were any other form of noise.
  - — Trend needs special handling, exactly as any other monotonic value.
- Various techniques exist for filtering out components of the total waveform. They include, as well as complex mathematical devices for filtering frequencies,
  - — Moving averages of various types. A moving average involves using lagged values over the series data points and using all of the lagged values in some way to reestimate the data point value. A large variety of moving average techniques exist, including simple moving averages (SMAs), weighted moving averages (WMAs), and exponential moving averages (EMAs).
  - — Smoothing techniques of several types. Smoothing is a windowing technique in which a window of adjustable length selects a particular subseries of data points for manipulation. The window slides over the whole series and manipulates each separate subset of data points to reestimate the window's central data point value. Smoothing techniques include peak-valley-mean (PVM), median smoothing, and Hanning.
  - — Resmoothing is a smoothing technique that involves either reapplying the same smoothing technique several times until no change occurs, or applying different window sizes or techniques several times.
- Differencing and reverse differencing (summing) offer alternative ways of looking at high- or low-frequency components of a waveform. Differencing and summing also transform waveforms in ways that may give clues to underlying randomness.
- The series data alone cannot ever be positively determined to contain a random component, although additional tests can raise the confidence level that detected noise is randomly generated. Only a rationale or causal explanation external to the data can confirm random noise generation.
- Components of a waveform can be separated out from the original waveform using one or several of the above techniques. These components are themselves series that

express some part of the information contained in the whole original series. Having the parts separated aids modeling by making the model either more understandable or more predictive, or meets some other need of the miner.

9.8.1 Looking at the Data

Series data must be looked at. There is positively no substitute whatsoever for looking at the data—graphing it, looking at correlograms, looking at spectra, differencing, and so on. There are a huge variety of other powerful tools used for analyzing series data, but those mentioned here, at least, must be used to prepare series data. The best aid that the miner has is a powerful series data manipulation and visualization tool, and preferably one that allows on-the-fly data manipulation, as well as use of the tools discussed. The underlying software used here to manipulate data and produce the images used for illustration was Statistica. (The accompanying CD-ROM includes a demonstration version.) This is one of several powerful statistical software packages that easily and quickly perform these and many other manipulations. Looking at the information revealed, and becoming familiar with what it means, is without any doubt the miner’s most important tool in preparing series data. It is, after all, the only way to look for dragons, chimera, and quicksand, not to mention the marked rocky road!

9.8.2 Signposts on the Rocky Road

So how should the miner use these tools and pointers when faced with series data? Here is a possible plan of attack.

- *Plot the data* Not only at the beginning should the data be plotted—plot everything. Keep plotting. Plot noise, plot smoothed, look at correlograms, look at spectra—and keep doing it. Work with it. Get a feel for what is in the data. Simply play with it. Video games with series data! This is not a frivolous approach. There is no more powerful pattern recognition tool known than the one inside the human head. Look and think closely about what is in the data and what it might mean. Although stated first, this is a continuous activity for all the stages that follow.

- *Fill in missing values* After a first look at the data, decide what to do about any that is missing. If possible, find any missing values. Seek them out. Digging them up, if at all possible, is a far better alternative than making them up! If they positively are not available, build autoregressive models and replace them.

Now—build models before and after replacement! The “before” models will use subsets of the data without any missing values. Build “after” models of the same sample length as the “before” models, but include the replaced values. If possible, build several “after” models with the replaced values at the beginning, middle, and end of the series. At least build “after” models with the replaced values at different places in the modeled series.

What changes? What is the relative strength of the patterns “discovered” in the “after” models that are not in the “before” models? If specific strong patterns only appear in the “after” models, try diluting them by adding neutral white noise to the replaced values. Look again. Try again. Keep trying until the replacement values appear to make no noticeable difference to the pattern density.

- *Replace outliers* Are there outliers? Are they really outliers, or just extremes of the range? Can individual outliers be accounted for as measurement error? Are there runs of outliers? If so, what process could cause them? Can the values be translated into the normal range?

Just as with missing values, work at finding out what caused the outliers and finding the accurate values. If they are positively not available, replace them exactly as for missing values.

- *Remove trend* Linear trend is easy to remove. Try fitting some other fractional frequency trend lines if they look like they might fit. If uncertain, fit a few different trend lines—log, square, exponential—see what they look like. Does one of them seem to fit some underlying trend in the data? If so, subtract it from the data. When graphed, does the data fit the horizontal axis better? If yes, fine. If no, keep trying.

- *Adjust variance* Subtract out the trend. Eyeball the variance, that is, the way the data scatters along the horizontal axis. Is it constant? Does it increase or decrease as the series progresses? If it isn’t constant, try Box-Cox transforms (or other transforms if they feel more comfortable). Get the variance as uniform as possible.

- *Smooth* Try various smoothing techniques, if needed. Unless there is some good reason to expect sharp spikes in the data, use hanning to get rid of them. Look at the spectrum. Look at the correlogram. Does smoothing help make what is happening in the data clearer? Subtract out the high frequencies and look at them. What is left in there? Any pattern? Mainly random? Extract what is in the noise until what is left seems random.

Now start again using forward and reverse differencing. Same patterns found? If not, why not? If so, why? Which makes most sense?

- *Account for seasonalities* Are there seasonal effects? What are they? Can they be identified? Subtract them out. If possible, create a separate variable for each, or a separate alpha label if more appropriate (so that when building a model, the model “knows” about the seasonalities).

- *Extract main cycle* Look for the main underlying “heartbeat” in the data. Smooth and filter until it seems clear. Extract it from the data.

- *Extract minor cycles* Look at what is left in the “noise.” Smooth it again. PVM works well very often. Look at spectra and correlograms. Any pattern? If so, extract it.

Look at the main waveform—the heartbeat. Look with a spectrum. What are the main component frequencies? Does this make sense? Is it reasonable to expect this set of frequencies? If it cannot be explained, is there at least no reason that it shouldn’t be so?

- *Redistribute and normalize* Redistribute and normalize the values if strictly maintaining the waveform shape is not critical; otherwise, just normalize.

- *Model or reverse pivot* If modeling the series, time to start. Otherwise, reverse pivot. For the reverse pivot, build an array of variables of different lags that are *least* correlated with each other. Try a variable for each of the main frequencies. Try a variable for the main cycle. Try a variable for the noise level. Survey the results. (When modeling, build several quick models to see which looks like it might work best.)

9.9 Implementation Notes

Familiarity with what displacement series look like and hands-on experience are the best data preparation tool that a miner can find for preparing series data. As computer systems become ever more powerful, it appears that there are various heuristic and algorithmic procedures that will allow automated series data preparation. Testing the performance of these procedures awaits the arrival of yet more powerful, low-cost computer systems. This has already happened with nonseries data preparation algorithms as the demonstration code shows. What is here today is a stunning array of automated tools for letting the miner *look* at series data in a phenomenal number of ways. This chapter hardly scratched the surface of the full panoply of techniques available. The tools that are available put power to *look* into the miners’ hands that has never before been available. The ability to *see* has to be found by experience.

Since handling series data is a very highly visual activity, and since fully automated preparation is so potentially damaging to data, the demonstration software has no specific routines for preparing displacement series data. Data visualization is a broad field in itself, and there are many highly powerful tools for handling data that have superb visualization capability. For small to moderate data sets, even a spreadsheet can serve as a good place to start. Spectral analysis is difficult, although not impossible, and correlograms are fairly easy. Moving on from there are many other reasonably priced data imaging and manipulation tools. Data imaging is so broad and deep a field, it is impossible to begin to cover the topic here.

This chapter has dealt exclusively with displacement series data. The miner has covered sufficient ground to prepare the series data so that it can be modeled. Having prepared such data, it can be modeled using the full array of the miner’s tools. As with previous chapters, attention is now turned back to looking at data without considering the information contained in any ordering. It is time to examine the data set as a whole.

Chapter 10: Preparing the Data Set

Overview

In this chapter, the focus of attention is on the data set itself. Using the term "data set" places emphasis on the interactions *between* the variables, whereas the term "data" has implied focusing on individual variables and their instance values. In all of the data preparation techniques discussed so far, care has been taken to expose the information content of the individual variables to a modeling tool. The issue now is how to make the information content of the data set itself most accessible. Here we cover using sparsely populated variables, handling problems associated with excessive dimensionality, determining an appropriate number of instances, and balancing the sample.

These are all issues that focus on the data set and require restructuring it as a whole, or at least, looking at groups of variables instead of looking at the variables individually.

10.1 Using Sparsely Populated Variables

Why use sparsely populated variables? When originally choosing the variables to be included in the data set, any in which the percentage of missing values is too high are usually discarded. They are discarded as simply not having enough information to be worth retaining. Some forms of analysis traditionally discard variables if 10 or 20% of the values are missing. Very often, when data mining, this discards far too many variables, and the threshold is set far lower. Frequently, the threshold is set to only exclude variables with more than 80 or 90% of missing values, if not more. Occasionally, a miner is constrained to use extremely sparsely populated variables—even using variables with only fractions of 1% of the values present. Sometimes, almost all of the variables in a data set are sparsely populated. When that is the case, it is these sparsely populated variables that carry the only real information available. The miner either uses them or gives up mining the data set.

For instance, one financial brokerage data set contained more than 700 variables. A few were well populated: account balance, account number, margin account balance, and so on. The variables carrying the information to be modeled, almost all of the variables present, were populated at below 10%, more than half below 2%; and a full one-third of all the variables present were populated below 1%—that is, with less than 1% of the values present. These were fields like "trades-in-corn-last-quarter," "open-contracts-oil," "June-open-options-hogs-bellies," "number-stop-loss-per-cycle," and other specialized information. How can such sparsely populated variables be used?

The techniques discussed up to this point in this book will not meet the need. For the brokerage data set just mentioned, for instance, the company wanted to predict portfolio trading proclivity so that the brokers could concentrate on high-value clients. Traditional modeling techniques have extreme difficulty in using such sparse data. Indeed, the brokerage analysts had difficulty in estimating trading proclivity with better than a 0.2 correlation. The full data preparation tool set, of which a demonstration version is on the accompanying CD-ROM, produced models with somewhat better than a 0.4 correlation when very sparsely populated variables were not included. When such variables were included, prepared as usual (see Chapter 8), the correlation increased to just less than 0.5. However, when these variables were identified and prepared as very sparsely populated, the correlation climbed to better than 0.7.

Clearly, there are occasions when the sparsely populated variables carry information that must be used. The problem is that, unless somehow concentrated, the information density is too low for mining tools to make good use of it. The solution is to increase the information density to the point where mining tools can use it.

10.1.1 Increasing Information Density Using Sparsely Populated Variables

When using very sparsely populated variables, missing-value replacement is not useful. Even when the missing values are replaced so they can be used, the dimensionality of state space increases by every sparsely populated variable included. Almost no information is gained in spite of this increase, since the sparsely populated variable simply does not carry much information. (Recall that replacing a missing value adds no information to the data set.) However, sometimes, and for some applications, variables populated with extreme sparsity have to be at least considered for use.

One solution, which has proved to work well when sparsely populated variables have to be used, collapses the sparse variables into fewer composite variables. Each composite variable carries information from several to many sparsely populated variables. If the sparsely populated variables are alpha, they are left in that form. If they are not alpha, categories are created from the numerical information. If there are many discrete numeric values, their number may have to be reduced. One method that works well is to “bin” the values and assign an alpha label to each bin. Collapsing the numeric information needs a situation-specific solution. Some method needs to be devised by the miner, together with a domain expert if necessary, to make sure that the needed information is available to the model.

It may be that several categories can occur simultaneously, so that simply creating a label for each individual variable category is not enough. Labels have to be created for each category combination that occurs, not just the categories themselves.

10.1.2 Binning Sparse Numerical Values

Binning is not a mysterious process. It only involves dividing the range of values into subranges and using subrange labels as substitutes for the actual values. Alpha labels are used to identify the subranges. This idea is very intuitive and widely used in daily life. Coffee temperature, say, may be binned into the categories “scalding,” “too hot,” “hot,” “cool,” and “cold.” These five alpha labels each represent part of the temperature range of coffee. These bins immediately translate into alpha labels. If the coffee temperature is in the range of “hot,” then assign the label “hot.” Figure 10.1 shows how binning works for coffee temperature.

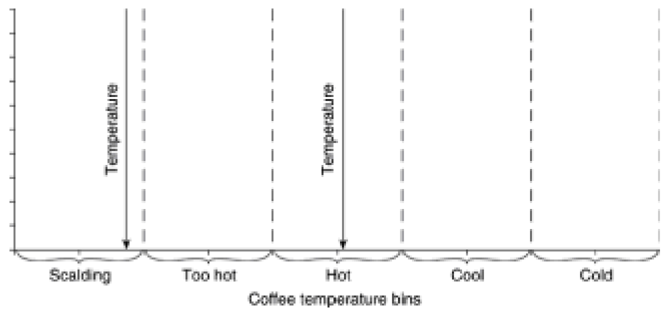


Figure 10.1 Binning coffee temperature to assign alpha labels. The left bar represents coffee assigned the label “Scalding” even though it falls close to the edge of the bin, near the boundary between ”Scalding” and “Too hot.” Likewise, the centrally located bar is assigned the label “Hot.”

This method of assigning labels to numerical values extends to any numerical variable. Domain knowledge facilitates bin boundaries assignment, appropriately locating where meaningful boundaries fall. For coffee temperature, both the number of bins and the bin boundaries have a rationale that can be justified. Where this is not the case, arbitrary boundaries and bin count have to be assigned. When there is no rationale, it is a good idea to assign bin boundaries so that each bin contains approximately equal numbers of labels.

10.1.3 Present-Value Patterns (PVPs)

Chapter 8 discussed missing-value patterns (MVPs). MVPs are created when most of the values are present and just a few are missing. Very sparsely populated variables present almost exactly the reverse situation. Most of the values are missing, and just a few are present. There is one major difference. With MVPs, the values were either missing or present. With PVPs, it is not enough to simply note the fact of the presence of a label; the PVP must also note the indentity of the label. The miner needs to account for this difference. Instead of simply noting, say, “P” for present and “A” for absent, the “P” must be a label of some sort that reflects which label or labels are present in the sparse variables. The miner must map every unique combination of labels present to a unique label and use that second label. This collapses many variables into one. Figure 10.2 shows this schematically, although for illustration the density of values in each variable in the figure is enormously higher than this method would ever be applied to in practice.

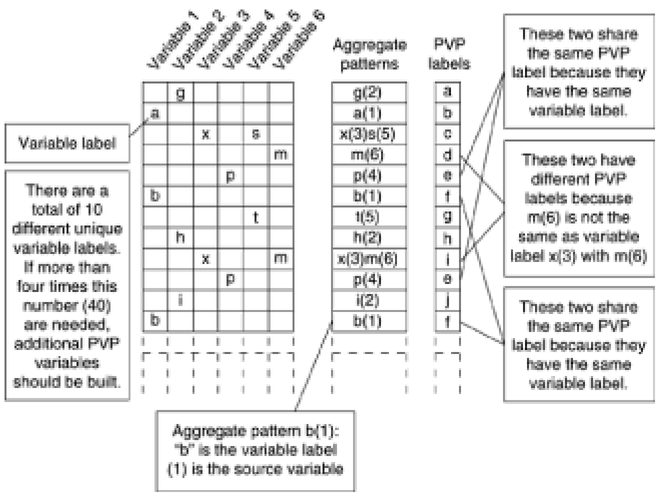

Variable 1
Variable 2
Variable 3
Variable 4
Variable 5
Variable 6
Aggregate patterns
PVP labels
These two share the same PVP label because they have the same variable label.
Variable label
There are a total of 10 different unique variable labels. If more than four times this number (40) are needed, additional PVP variables should be built.
g
a
x
s
m
p
b
t
h
x
m
p
i
b
g(2)
a(1)
x(3)s(5)
m(6)
p(4)
b(1)
t(5)
h(2)
x(3)m(6)
p(4)
i(2)
b(1)
a
b
c
d
e
f
g
h
i
e
j
f
These two have different PVP labels because m(6) is not the same as variable label x(3) with m(6)
These two share the same PVP label because they have the same variable label.
Aggregate pattern b(1): "b" is the variable label (1) is the source variable

Figure 10.2 Sparsely populated variables generate unique patterns for those values that are present. Every PVP has a unique label. The density of values shown in the figure is far above the fractional percent density for which this method of compression is designed.

Note that PVPs are only created for variables that are very sparsely populated—usually at less than 1%. Creating PVPs for more populous variables results in a proliferation of alpha labels that simply explodes! In fact, if there are too many PVPs in any one created variable, subsets of the sparsely populated variables should be collapsed so that the label count does not get too high. What are too many PVPs? As a rule of thumb, if the number of PVPs is more than four times the total number of individual variable labels, use multiple PVP variables. Where multiple PVP variables are needed, select groups of sparsely populated variables so that the PVP label count is minimized.

Building PVP patterns this way does lose some information. Binning itself discards information in the variables for a practical gain in usability. However, using PVPs makes much of the information in very sparsely populated variables available to the mining tool, where it would otherwise be completely discarded. The created PVP variable(s) is numerated exactly as any other alpha variable. Chapter 6 discusses numerating alpha values.

10.2 Problems with High-Dimensionality Data Sets

The dimensionality of a data set is really a count of the number of variables it contains. When discussing state space (Chapter 6), each of the variables was referred to as a "dimension." Very large state spaces—those with large numbers of dimensions—present problems for all mining tools. Why problems? For one reason, no matter how fast or powerful the mining tool, or the computer running the tool, there is always some level of dimensionality that will defeat any effort to build a comprehensive model. Even a massively parallel processor, totally dedicated to the project and running a highly advanced and optimized mining toolset, could not deal with a multiterabyte, 7000+ variable data set required on one mining project.

Another reason that high dimensionality presents difficulties for mining tools is that as the dimensionality increases, the size (multidimensional volume) of state space increases. This requires more data points to fill the space to any particular density. Low-density state spaces leave more “wiggle room” between the data points where the shape of the manifold is undefined. In these spaces there is more probability of overfitting than in more populous state spaces. (Chapter 3 discusses overfitting.) To reduce the risk of overfitting, more instances are needed—lots more instances. Just how many is discussed later in this chapter.

What seems to be another problem with increasing dimensionality, but is actually the same problem in different clothing, is the combinatorial explosion problem. “Combinatorial” here refers to the number of different ways that the values of the variables can be combined. The problem is caused by the number of possible unique system states increasing as the multiple of the individual variable states. For instance, if three variables have two, three, and four possible states each, there are 2 x 3 x 4 = 24 possible system states. When each variable can take tens, hundreds, thousands, or millions of meaningful discrete states, and there are hundreds or thousands of variables, the number of possible discrete, meaningful system states becomes very large—*very* large! And yet, to create a fully representative model, a mining tool ideally needs at least one example of each meaningful system state represented in state space. The number of instances required can very quickly become impractical, and shortly thereafter impossible to assemble.

There seem to be three separate problems here. First, the sheer amount of data defeats the hardware/software mining tools. Second, low density of population in a voluminous state space does not well define the shape of the manifold in the spaces between the data points. Third, the number of possible combinations of values requires an impossibly huge amount of data for a representative sample—more data than can actually be practically assembled. As if these three (apparently) separate problems weren’t enough, high dimensionality brings with it other problems too! As an example, if variables are “colinear”—that is, they are so similar in information content as to carry nearly identical information—some tools, particularly those derived from statistical techniques, can have extreme problems dealing with some representations of such variables. The chance that two such variables occur together goes up tremendously as dimensionality increases. There are ways around this particular problem, and around many other problems. But it is much better to avoid them if possible.

What can be done to alleviate the problem? The answer requires somehow reducing the amount of data to be mined. Reducing the number of instances doesn’t help since large state spaces need more, not less, data to define the shape of the manifold than small ones. The only other answer requires reducing the number of dimensions. But that seems to mean removing variables, and removing variables means removing information, and removing information is a poor answer since a good model needs all the information it can get. Even if removing variables is absolutely required in order to be able to mine at all, how should the miner select the variables to discard?

10.2.1 Information Representation

The real problem here is very frequently with the data representation, not really with high dimensionality. More properly, the problem is with *information* representation. Information representation is discussed more fully in Chapter 11. All that need be understood for the moment is that the values in the variables carry information. Some variables may duplicate all or part of the information that is also carried by other variables. However, the data set as a whole carries within it some underlying pattern of information distributed among its constituent variables. It is this information, carried in the weft and warp of the variables—the intertwining variability, distribution patterns, and other interrelationships—that the mining tool needs to access.

Where two variables carry identical information, one can be safely removed. After all, if the information carried by each variable is identical, there has to be a correlation of either +1 or –1 between them. It is easy to re-create one variable from the other with perfect fidelity. Note that although the information carried is identical, the form in which it is carried may differ. Consider the two times table. The instance values of the variable "the number to multiply" are different from the corresponding instance values of the variable "the answer." When connected by the relationship "two times table," both variables carry identical information and have a correlation of +1. One variable carries information to perfectly re-create instance values of the other, but the actual content of the variables is not at all similar.

What happens when the information shared between the variables is only partially duplicated? Suppose that several people are measured for height, weight, and girth, creating a data set with these as variables. Suppose also that any one variable's value can be derived from the other two, but not from any other one. There is, of course, a correlation between any two, probably a very strong one in this case, but not a perfect correlation. The height, weight, and girth measurements are all different from each other and they can all be plotted in a three-dimensional state space. But is a three-dimensional state space needed to capture the information? Since any two variables serve to completely specify the value of the third, one of the variables isn't actually needed. In fact, it only requires a two-dimensional state space to carry all of the information present. Regardless of which two variables are retained in the state space, a transformation function, suitably chosen, will perfectly give the value of the third. In this case, the information can be "embedded" into a two-dimensional state space without any loss of either predictive or inferential power. Three dimensions are needed to capture the variables' values—but only two dimensions to capture the information.

To take this example a little further, it is very unlikely that two variables will perfectly predict the third. Noise (perhaps as measurement errors and slightly different muscle/fat/bone ratios, etc.) will prevent any variable from being perfectly correlated with the other two. The noise adds some unique information to each variable—but is it wanted? Usually a miner wants to discard noise and is interested in the underlying relationship, not the noise relationship. The underlying relationship can still be embedded in two dimensions. The noise, in this example, will be small compared to the relationship but needs three dimensions. In multidimensional scaling (MDS) terms (see Chapter 6), projecting the relationship into two dimensions causes some, but only a little, stress. For this example, the stress is caused by noise, not by the underlying information.

Using MDS to collapse a large data set can be highly computationally intensive. In Chapter 6, MDS was used in the numeration of alpha labels. When using MDS to reduce data set dimensionality, instead of alpha label dimensionality, discrete system states have to be discovered and mapped into phase space. There may be a very large number of these, creating an enormous "shape." Projecting and manipulating this shape is difficult and time-consuming. It can be a viable option. Collapsing a large data set is always a computationally intensive problem. MDS may be no slower or more difficult than any other option.

But MDS is an "all-or-nothing" approach in that only at the end is there any indication whether the technique will collapse the dimensionality, and by how much. From a practical standpoint, it is helpful to have an incremental system that can give some idea of what compression might achieve as it goes along. MDS requires the miner to choose the number of variables into which to attempt compression. (Even if the number is chosen automatically as in the demonstration software.) When compressing the whole data set, a preferable method allows the miner to specify a required level of confidence that the information content of the original data set has been retained, instead of specifying the final number of compressed variables. Let the required confidence level determine the number of variables instead of guessing how many might work.

10.2.2 Representing High-Dimensionality Data in Fewer Dimensions

There are dimensionality-reducing methods that work well for linear between-variable relationships. Methods such as principal components analysis and factor analysis are well-known ways of compressing information from many variables into fewer variables. (Statisticians typically refer to these as data reduction methods.)

Principal components analysis is a technique used for concentrating variability in a data set. Each of the dimensions in a data set possesses a variability. (Variability is discussed in many places; see, for example, Chapter 5.) Variability can be normalized, so that each dimension has a variability of 1. Variability can also be redistributed. A *component* is an artificially constructed variable that is fitted to all of the original variables in a data set in such a way that it extracts the highest possible amount of variability.

The total amount of variability in a specific data set is a fixed quantity. However, although each original variable contributes the same amount of variability as any other original variable, redistributing it concentrates data set variability in some components, reducing it in others. With, for example, 10 dimensions, the variability of the data set is 10. The first *component,* however, might have a variability not of 1—as each of the original variables has—but perhaps of 5. The second component, constructed to carry as much of the remaining variability as possible, might have a variability of 4. In principal components analysis, there are always in total as many components as there are original variables, but the remaining eight variables in this example now have a variability of 1 to share between them. It works out this way: there is a total amount of variability of 10/10 in the 10 original variables. The first two components carry 5/10 + 4/10 = 9/10, or 90% of the variability of the data set. The remaining eight components therefore have only 10% of the variability to carry between them.

Inasmuch as variability is a measure of the information content of a variable (discussed in Chapter 11), in this example, 90% of the information content has been squeezed into only two of the specially constructed variables called components. Capturing the full variability of the data set still requires 10 components, no change over having to use the 10 original variables. But it is highly likely that the later components carry noise, which is well ignored. Even if noise does not exist in the remaining components, the benefit gained in collapsing the number of variables to be modeled by 80% may well be worth the loss of information.

The problem for the miner with principal component methods is that they only work well for *linear* relationships. Such methods, unfortunately, actually damage or destroy nonlinear relationships—catastrophic and disastrous for the mining process! Some form of nonlinear principal components analysis seems an ideal solution. Such techniques are now being developed, but are extremely computationally intensive—so intensive, in fact, that they themselves become intractable at quite moderate dimensionalities. Although promising for the future, such techniques are not yet of help when collapsing information in intractably large dimensionality data sets.

Removing variables is a solution to dimensionality reduction. Sometimes this is required since no other method will suffice. For instance, in the data set of 7000+ variables mentioned before, removing variables was the only option. Such dimensionality mandates a reduction in the number of dimensions before it is practical to either mine or compress it with any technique available today. But when discarding variables is required, selecting the variables to discard needs a rationale that selects the least important variables. These are the variables least needed by the model. But how are the least needed variables to be discovered?

10.3 Introducing the Neural Network

One problem, then, is how to squash the information in a data set into fewer variables without destroying any nonlinear relationships. Additionally, if squashing the data set is impossible, how can the miner determine which are the least contributing variables so that they can be removed? There is, in fact, a tool in the data miner's toolkit that serves both dimensionality reduction purposes. It is a very powerful tool that is normally used as a modeling tool. Although data preparation uses the full range of its power, it is applied to totally different objectives than when mining. It is introduced here in general terms before examining the modifications needed for dimensionality reduction. The tool is the standard, back-propagation, artificial neural network (BP-ANN).

The idea underlying a BP-ANN is very simple. The BP-ANN has to learn to make predictions. The learning stage is called *training*. Inputs are as a pattern of numbers—one number per network input. That makes it easy to associate an input with a variable such that every variable has its corresponding input. Outputs are also a pattern of numbers—one number per output. Each output is associated with an output variable. Each of the inputs and outputs is associated with a "neuron," so there are input neurons and output neurons. Sandwiched between these two kinds of neurons is another set of neurons called the *hidden layer*, so called for the same reason that the cheese in a cheese sandwich is hidden from the outside world by the bread. So too are the hidden neurons hidden from the world by the input and output neurons. Figure 10.3 shows schematically a typical representation of a neural network with three input neurons, two hidden neurons, and one output neuron. Each of the input neurons connects to each of the hidden neurons, and each of the hidden neurons connects to the output neuron. This configuration is known as a fully connected ANN.

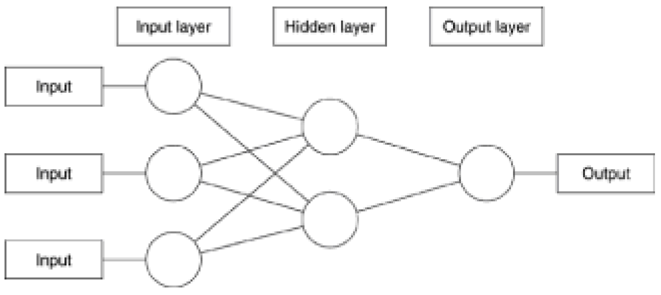


Figure 10.3 A three-input, one-output neural network with two neurons in the hidden layer.

The BP-ANN is usually in the form of a fully autonomous algorithm—often a compiled and ready-to-run computer program—which the miner uses. Use of a BP-ANN usually requires the miner only to select the input and output data that the network will train on, or predict about, and possibly some learning parameters. Seldom do miners write their own BP-ANN software today. The explanation here is to introduce the features and architecture of the BP-ANN that facilitate data compression and dimensionality reduction. This gives the miner an insight about why and how the information compression works, why the compressed output is in the form it is, and some insight into the limitations and problems that might be expected.

10.3.1 Training a Neural Network

Training takes place in two steps. During the first step, the network processes a set of input values and the matching output value. The network looks at the inputs and estimates the output—ignoring its actual value for the time being.

In the second step, the network compares the value it estimated and the actual value of the output. Perhaps there is some error between the estimated and actual values. Whatever it is, this error reflects back through the network, from output to inputs. The network adjusts itself so that, if those adjustments were used, the error would be made smaller. Since there are only neurons and connections, where are the adjustments made? Inside the neurons.

Each neuron has input(s) and an output. When training, it takes each of its inputs and multiplies them by a weight specific to that input. The weighted inputs merge together and pass out of the neuron as its response to these particular inputs. In the second step, back comes some level of error. The neuron adjusts its internal weights so that the actual neuron output, for these specific inputs, is closer to the desired level. In other words, it adjusts to reduce the size of the error.

This reflecting the output error backwards from the output is known as propagating the error backwards, or *back-propagation*. The back-propagation referred to in the name of the network only takes place during training. When predicting, the weights are frozen, and only the forward-propagation of the prediction takes place.

Neural networks, then, are built from neurons and interconnections between neurons. By continually adjusting its internal neuron weightings to reduce the error of each neuron's predictions, the neural network eventually learns the correct output for any input, if it is possible. Sometimes, of course, the output is not learnable from the information contained in the input. When it is possible, the network learns (in its neurons) the relationship between inputs and output. In many places in this book, those relationships are described as curved manifolds in state space. Can a neural network learn any conceivable manifold shape? Unfortunately not. The sorts of relationship that a neural network can learn are those that can be described by a function—but it is potentially any function! (A *function* is a mathematical device that produces a single output value for every set of input values. See Chapter 6 for a discussion of functions, and relationships not describable by functions.) Despite the limitation, this is remarkable! How is it that changing the weights inside neurons, connected to other neurons in layers, can create a device that can learn what may be complex nonlinear functions? To answer that question, we need to take a much closer look at what goes on inside an artificial neuron.

10.3.2 Neurons

Neurons are so called because, to some extent, they are modeled after the functionality of units of the human brain, which is built of biochemical neurons. The neurons in an artificial neural network copy some of the simple but salient features of the way biochemical neurons are believed to work. They both perform the same essential job. They take several inputs and, based on those inputs, produce some output. The output reflects the state and value of the inputs, and the error in the output is reduced with training.

For an artificial neuron, the input consists of a number. The input number transfers across the inner workings of the neuron and pops out the other side altered in some way. Because of this, what is going on inside a neuron is called a *transfer function*. In order for the network as a whole to learn nonlinear relationships, the neuron's transfer function has to be nonlinear, which allows the neuron to learn a small piece of an overall nonlinear function. Each neuron finds a small piece of nonlinearity and learns how to duplicate it—or at least come as close as it can. If there are enough neurons, the network can learn enough small pieces in its neurons that, as a whole, it learns complete, complex nonlinear functions.

There are a wide variety of neuron transfer functions. In practice, by far the most popular transfer function used in neural network neurons is the logistic function. (See the Supplemental Material section at the end of Chapter 7 for a brief description of how the logistic function works.) The logistic function takes in a number of any value and produces as its output a number between 0 and 1. But since the exact shape of the logistic curve can be changed, the exact number that comes out depends not only on what number was put in, but on the particular shape of the logistic curve.

10.3.3 Reshaping the Logistic Curve

First, a brief note about nomenclature. A function can be expressed as a formula, just as the formula for determining the value of the logistic function is

$$g = \frac{1}{1 + e^{-y}}$$

For convenience, this whole formula can be taken as a given and represented by a single letter, say *g*. This letter *g* stands for the logistic function. Specific values are input into the logistic function, which returns some other specific value between 0 and 1. When using this sort of notation for a function, the input value is shown in brackets, thus:

$y = g(10)$

This means that *y* gets whatever value comes out of the logistic function, represented by *g*, when the value 10 is entered. A most useful feature of this shorthand notation is that any valid expression can be placed inside the brackets. This nomenclature is used to indicate that the value of the expression inside the brackets is input to the logistic function, and the logistic function output is the final result of the overall expression. Using this notation removes much distraction, making the expression in brackets visually prominent.

10.3.4 Single-Input Neurons

A neuron uses two internal weight types: the *bias weight* and *input weights*. As discussed elsewhere, a bias is an offset that moves all other values by some constant amount. (Elsewhere, bias has implied noise or distortion—here it only indicates offsetting movement.) The bias weight moves, or biases, the position of the logistic curve. The input weight modifies an input value—effectively changing the shape of the logistic curve. Both of these weight types are adjustable to reduce the back-propagated error.

The formula for this arrangement of weights is exactly the formula for a straight line:

$$y_n \text{ x } a_0 + b_n x_n$$

So, given this formula, exactly what effect does adjusting these weights have on the logistic function's output? In order to understand each weight's effects, it is easiest to start by looking at the effect of each type of weight separately. In the following discussion a one-input neuron is used so there is a single-bias weight and a single-input weight. First, the bias weight.

Figure 10.4 shows the effect on the logistic curve for several different bias weights. Recall that the curve itself represents, on the *y* (vertical) axis, values that come out of the logistic function when the values on the *x* (horizontal) axis represent the input values. As the bias weight changes, the position of the logistic curve moves along the horizontal *x*-axis. This does not change the range of values that are translated by the logistic function—essentially it takes a range of 10 to take the function from 0 to 1. (The logistic function never reaches either 0 or 1, but, as shown, covers about 99% of its output range for a change in input of 10, say –5 to +5 with a bias of 0.)

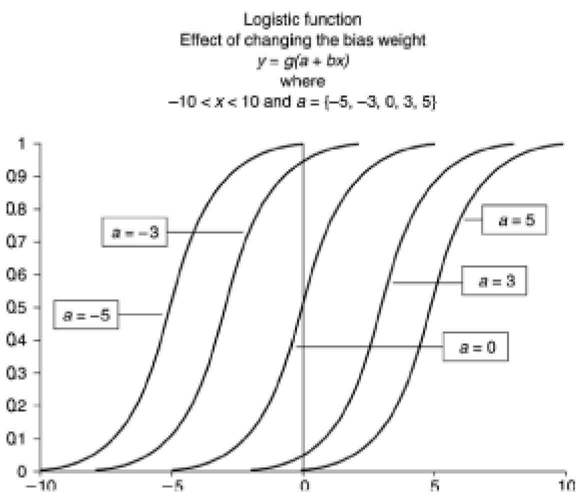


Figure 10.4 Changing the bias weight *a* moves the center of the logistic curve along the *x*-axis. The center of the curve, value 0.5, is positioned at the value of the bias weight.

The bias displaces the range over which the output moves from 0 to 1. In actual fact, it moves the center of the range, and why it is important that it is the center that moves will be seen in a moment. The logistic curves have a central value of 0.5, and the bias weight positions this point along the *x*-axis.

The input weight has a very different effect. Figure 10.5 shows the effect of changing the input weight. For ease of illustration, the bias weight remains at 0. In this image the shape of the curve stretches over a larger range of values. The smaller the input weight, the more widely the translation range stretches. In fact, although not shown, for very large values the function is essentially a "step," suddenly switching from 0 to 1. For a value of 0, the function looks like a horizontal line at a value of 0.5.

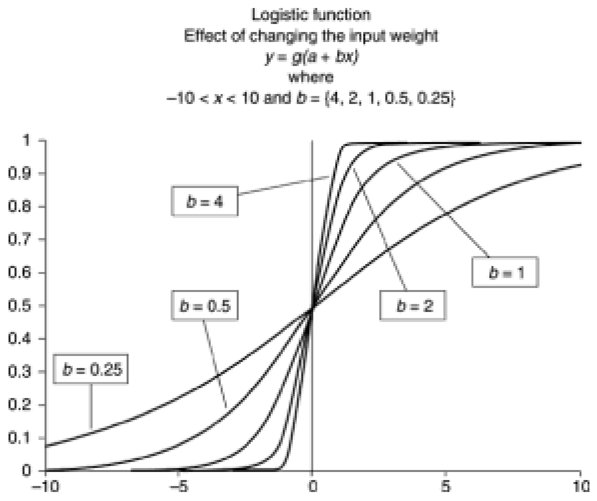


Figure 10.5 Holding the bias weight at 0 and changing the input weight *b*

changes the transition range of the logistic function.

Figure 10.6 has similar curves except that they all move in the opposite direction! This is the result of using a negative input weight. With positive weights, the output values translate from 0 to 1 as the input moves from negative to positive values of *x*. With negative input weights, the translation moves from 1 toward 0, but is otherwise completely adjustable exactly as for positive weights.

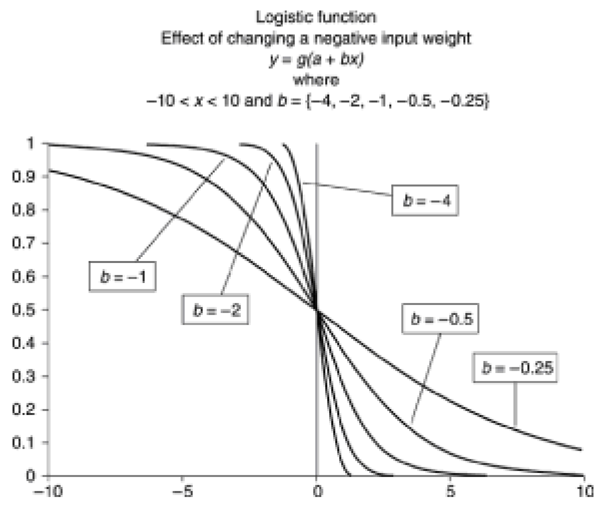


Figure 10.6 When the input weight is negative, the curve is identical in shape to a positively weighted curve, except that it moves in the opposite direction—positive to negative instead of negative to positive.

The logistic curve can be positioned and shaped as needed by the use of the bias and input weights. The range, slope, and center of the curve are fully adjustable. While the characteristic shape of the curve itself is not modified, weight modification positions the center and range of the curve wherever desired.

This is indeed what a neuron does. It moves its transfer function around so that whatever output it actually gives best matches the required output—which is found by back-propagating the errors.

Well, it can easily be seen that the logistic function is nonlinear, so a neuron can learn at least that much of a nonlinear function. But how does this become part of a complex nonlinear function?

10.3.5 Multiple-Input Neurons

So far, the neuron in the example has dealt with only one input. Whether the hidden layer neurons have multiple inputs or not, the output neuron of a multi-hidden-node network must deal with multiple inputs. How does a neuron weigh multiple inputs and pass them across its transfer function?

Figure 10.7 shows schematically a five-input neuron. Looking at this figure shows that the bias weight, *a0*, is common to all of the inputs. Every input into this neuron shares the effect of this common bias weight. The input weights, on the other hand, *bn*, are specific to each input. The input value itself is denoted by *xn*.

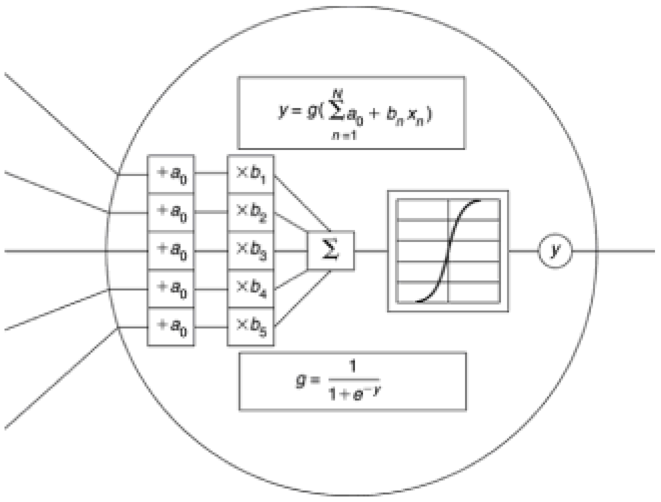


Figure 10.7 The “Secret Life of Neurons”! Inside a neuron, the common bias weight (*a*0®MDNM¯) is added to all inputs, but each separate input is multiplied by its own input weight (*bn*). The summed result is applied to the transfer function, which produces the neuron’s output (*y*).

There is an equation specific to each of the five inputs:

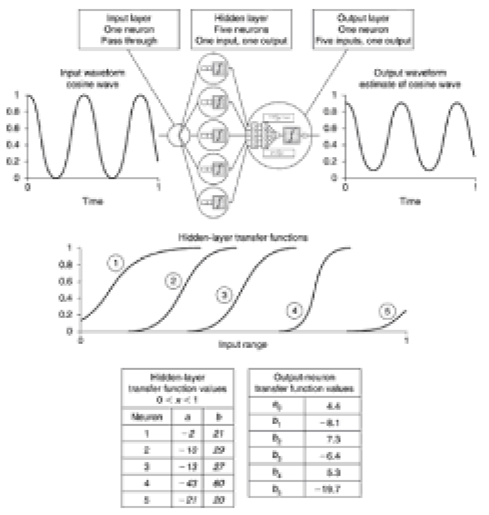
$$y_n = a_0 + b_n x_n$$

where *n* is the number of the input. In this example, *n* ranges from 1 to 5. The neuron code evaluates the equations for specific input values and sums the results. The expression in the top box inside the neuron indicates this operation. The logistic function (shown in the neuron’s lower box) transfers the sum, and the result is the neuron’s output value.

Because each input has a separate weight, the neuron can translate and move each input into the required position and direction of effect to approximate the actual output. This is critical to approximating a complex function. It allows the neuron to use each input to estimate part of the overall output and assembles the whole range of the output from these component parts.

10.3.6 Networking Neurons to Estimate a Function

Figure 10.8 shows a complete one-input, five-hidden-neuron, one-output neural network. There are seven neurons in all. The network has to learn to reproduce the 2 cycles of cosine wave shown as input to the network.



Hidden-layer transfer function values 0 < x < 1		
Neuron	a	b
1	−2	21
2	−10	29
3	−13	37
4	−43	60
5	−21	20

Output neuron transfer function values	
a_0	4.4
b_1	−8.1
b_2	7.3
b_3	−6.4
b_4	5.3
b_5	−19.7

Figure 10.8 A neural network learning the shape of a cosine waveform. The input neuron splits the input to the hidden neurons. Each hidden neuron learns part of the overall wave shape, which the output neuron reassembles when prediction is required.

The input neuron itself serves only as a placeholder. It has no internal structure, serving only to represent a single input point. Think of it as a “splitter” that takes the single input and splits it between all of the neurons in the hidden layer. Each hidden-layer neuron “sees” the whole input waveform, in this case the 2 1/4 cosine wave cycles. The amplitude of the cosine waveform is 1 unit, from 0 to 1, corresponding to the input range for the logistic transfer function neurons. The limit in output range of 0–1 requires that the input range be limited too. Since the neuron has to try to duplicate the input as its output, then the input has to be limited to the range the neuron actually can output. The “time” range for the waveform is also normalized to be across the range 0–1, again matching the neuron output requirements.

The reexpression of the time is necessary because the network has to learn to predict the value of the cosine wave at specific times. When predicting with this network, it will be asked, in suitably anthropomorphic form, “What is the value of the function at time *x*?” where *x* is a number between 0 and 1.

Each hidden-layer neuron will learn part of the overall waveform shape. Figure 10.9 shows why five neurons are needed. Each neuron can move and modify the exact shape of its logistic transfer function, but it is still limited to fitting the modified logistic shape to part of the pattern to be learned as well as it can. The cosine waveform has five roughly logistic-function-shaped pieces, and so needs five hidden-layer neurons to learn the five pieces.

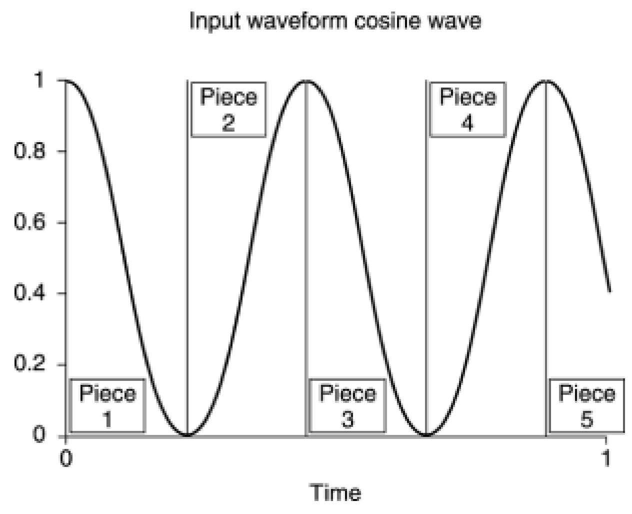


Figure 10.9 Learning this waveform needs at least five neurons. Each neuron can only learn an approximately logistic-function-shaped piece of the overall waveform. There are five such pieces in this wave shape.

10.3.7 Network Learning

During network setup, the network designer takes care to set all of the neuron weights at random. This is an important part of network learning. If the neuron weights are all set identically, for instance, each neuron tries to learn the same part of the input waveform as all of the other neurons. Since identical errors are then back-propagated to each, they all continue to be stuck looking at one small part of the input, and no overall learning takes place. Setting the weights at random ensures that, even if they all start trying to approximate the same part of the input, the errors will be different. One of the neurons predominates and the others wander off to look at approximating other parts of the curve. (The algorithm uses sophisticated methods of ensuring that the neurons do all wander to different parts of the overall curve, but they do not need to be explored here.)

Training the network requires presenting it with instances one after the other. These instances, of course, comprise the miner-selected training data set. For each instance of data presented, the network predicts the output based on the state of its neuron weights. At the output there is some error (difference between actual value and predicted value)—even if in a particular instance the error is 0. These errors are accumulated, not fed back on an instance-by-instance basis. A complete pass through the training data set is called an *epoch*. Adequately training a neural network usually requires many epochs. Back-propagation only happens at the end of each epoch. Then, each neuron adjusts its weights to better modify and fit the logistic curve to the shape of its input. This ensures that each neuron is trying to fit its own curve to some “average” shape of the overall input waveform.

Overall, each neuron tries to modify and fit its logistic function as well as possible to some part of the curve. It may succeed well, or it may do very poorly, but when training is complete, each approximates a part of the input as well as possible. The miner determines the criteria that determine training to be "complete." Usually, training stops either when the input wave shape can be re-created with less than some selected level of error, say, 10%, or when a selected number of epochs have passed without any improvement in the prediction.

It is usual to reserve a test data set for use during training. The network learns the function from the training set, but fitting the function to the test data determines that training is complete. As training begins, and the network better estimates the needed function in the training data set, the function improves its fit with the test data too. When the function learned in the training data begins to fit the test data less well, training is halted. This helps prevent learning noise. (Chapter 2 discusses sources of noise, Chapter 3 discusses noise and the need for multiple data sets when training, and Chapter 9 discusses noise in time series data, and waveforms.)

10.3.8 Network Prediction—Hidden Layer

So what has the network learned, and how can the cosine waveform be reproduced? Returning to Figure 10.8, after training, each hidden-layer neuron learned part of the waveform. The center graph shows the five transfer functions of the individual hidden-layer neurons. But looking at these transfer functions, it doesn't appear that putting them together will reproduce a cosine waveform!

Observe, however, that the transfer functions for each neuron are each in a separate position of the input range, shown on the (horizontal) *x*-axis. None of the transfer functions seems to be quite the same shape as any other, as well as being horizontally shifted. The actual weights learned for each hidden neuron are shown in the lower-left box. It is these weights that modify and shape the transfer function. For any given input value (between 0 and 1), the five neurons will be in some characteristic state.

Suppose the value 0.5 is input—what will be the state of the hidden-layer neurons? Hidden neurons 1 and 2 will both produce an output close in value to 1. Hidden neuron 3 is just about in the middle of its range and will produce an output close to 0.5. Hidden neurons 4 and 5 will produce an output close to 0. So it is for any specific input value—the hidden neurons will each produce a specific value.

But these outputs are not yet similar to the original cosine waveform. How can they be assembled to resemble the input cosine waveform?

10.3.9 Network Prediction—Output Layer

The task of the output neuron involves taking as input the various values output by the hidden layer and reproducing the input waveform from them. This, of course, is a multiple-input neuron. The lower-right box in Figure 10.8 shows the learned values for its inputs. The bias weight (a_0) is common, but the input weights are each separate. Careful inspection shows that some of them are negative. Negative input weights, recall, have the effect of "flipping" the direction in which the transfer function moves. In fact, the first, third, and fifth weights (b_1, b_3, b_5) are all negative. During the part of the input range when these hidden-layer neurons are changing value, their positive going change will be translated at the output neuron into a negative going change. It is these weights that change the direction of the hidden-layer transfer functions.

The output layer sums the inputs, transfers the resulting value across its own internal function, and produces the output shown. Clearly the network did not learn to reproduce the input perfectly. More training would improve the *shape* of the output. In fact, with enough training, it is possible to come as close as the miner desires to the original shape. But there is another distortion. The range of the original input was 0 to 1. The smallest input value was actually 0, while the largest was actually 1. The output seems to span a range of about 0.1 to 0.9. Is this an error? Can it be corrected?

Unfortunately, the logistic function cannot actually reach values of 0 or 1. Recall that it is this feature that makes it so useful as a squashing function (Chapter 7). To actually reach values of 0 or 1, the input to the logistic function has to be infinitely negative or infinitely positive. This allows neural networks to take input values of any size during modeling. However, the network will only actually "see" any very significant change over the linear part of the transfer function—and it will only produce output over the range it "sees" in the input.

10.3.10 Stochastic Network Performance

A neural network is a stochastic device. *Stochastic* comes from a Greek word meaning "to aim at a mark, to guess." Stochastic devices work by making guesses and improving their performance, often based on error feedback. Their strength is that they usually produce approximate answers very quickly. Approximate can mean quite close to the precise answer (should one exist) or having a reasonably high degree of confidence in the answer given. Actually producing exact answers requires unlimited repetitions of the feedback cycle—in other words, a 100% accurate answer (or 100% confidence in the answer) takes, essentially, forever.

This makes stochastic devices very useful for solving a huge class of real-world problems. There are an enormous number of problems that are extremely difficult, perhaps impossible, to solve exactly, but where a good enough answer, quickly, is far better than an exact answer at some very remote time.

Humans use stochastic techniques all the time. From grocery shopping to investment analysis, it is difficult, tedious, and time-consuming, and most likely impossible in practice, to get completely accurate answers. For instance, exactly—to the nearest whole molecule—how much coffee will you require next week? Who knows? Probably a quarter of a pound or so will do (give or take several trillions of molecules). Or again—compare two investments: one a stock mutual fund and the other T-bills. Precisely how much—to the exact penny, and including all transaction costs, reinvestments, bonuses, dividends, postage, and so on—will each return over the next 10 years (to the nearest nanosecond)? Again, who knows, but stocks typically do better over the long haul than T-bills. T-bills are safer. But only safer stochastically. If a precise prediction was available, there would be no uncertainty.

With both of these examples, more work will give more accurate results. But there comes a point at which good enough is good enough. More work is simply wasted. A fast, close enough answer is useable now. A comprehensive and accurate answer is not obtainable in a useful time frame.

Recall that at this stage in the data preparation process, all of the variables are fully prepared—normalized, redistributed, and no missing values—and all network input values are known. Because of this, the dimensionality collapse or reduction part of data preparation doesn’t use another enormously powerful aspect of stochastic techniques. Many of them are able to make estimates, inferences, and predictions when the input conditions are uncertain or unknown. Future stock market performance, for instance, is impossible to accurately predict—this is intrinsically unknowable information, not just unknown-but-in-principle-knowable information. Stochastic techniques can still estimate market performance even with inadequate, incomplete, or even inaccurate inputs.

The point here is that while it is not possible for a neural network to produce 100% accurate predictions in any realistic situation, it will quickly come to some estimate and converge, ever more slowly, never quite stopping, toward its final answer. The miner must always choose some acceptable level of accuracy or confidence as a stopping criterion. That accuracy or confidence must, of necessity, always be less than 100%.

10.3.11 Network Architecture 1—The Autoassociative Network

There are many varieties of neural networks. Many networks work on slightly different principles than the BP-ANN described here, and there are an infinite variety of possible network architectures. The architecture, in part, defines the number, layout, and connectivity of neurons within a network. Data preparation uses a class of architectures called *autoassociative networks*.

One of the most common neural network architectures is some variant of that previously shown in Figure 10.3. This type of network uses input neurons, some lesser number of hidden neurons fully connected to the input layer, and one, or at most a few, output neurons fully connected to the hidden layer. Such networks are typically used to predict the value of an output(s) when the output value(s) is not included in the input variables—for example, predicting the level of tomorrow's stock market index given selected current information.

An autoassociative network has a very different architecture, as shown in Figure 10.10. A key point is that the number of inputs and outputs are identical. Not only is the number identical, but all of the values put into the network are also applied as outputs! This network is simply learning to predict the value of its own inputs. This would ordinarily be a trivial task, and as far as predicting values for the outputs goes, it is trivial. After all, the value of the inputs is known, so why predict them?

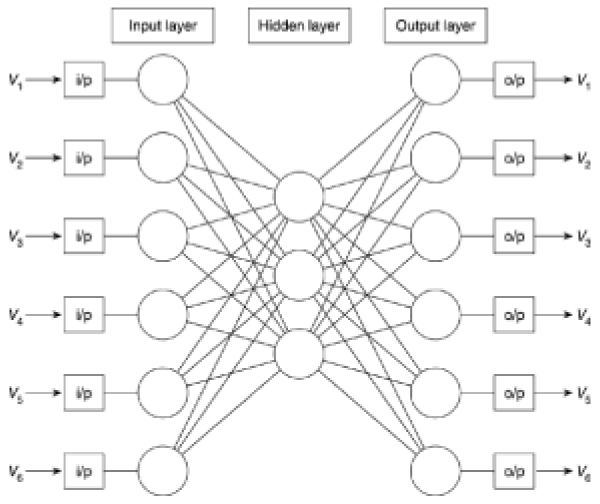


Figure 10.10 An autoassociative neural network has the same number of inputs and outputs. Each value applied as an input is also an output. The network learns to duplicate all of the inputs as outputs.

The key to this answer lies in the hidden layer. The hidden layer has less neurons than either input or output. The importance of this is discussed in a moment.

10.3.12 Network Architecture 2—The Sparsely Connected Network

The networks discussed so far were all fully connected. That is to say, every neuron in the input layer connects to every neuron in the hidden layer, and every neuron in the hidden layer connects to every neuron in the output layer. For a fully connected autoassociative neural network, the number of interconnections rises quite steeply as the dimensionality of the data set to be modeled increases. The actual number of interconnections is twice the number of dimensions multiplied by the number of neurons in the hidden layer:

$c = 2(n_i \times n_h)$

where

c	is number of interconnections
n_i	is number of inputs
n_h	is number of hidden neurons

For the example shown in Figure 10.10, that number is 2(6 x 3)= 36. For a 100-input, 50-hidden neuron network, the number is already 10,000. With 1000 inputs (not untypical with data compression problems), a 50% reduction in the size of the hidden layer requires 1,000,000 connections in a fully interconnected neural network. Each interconnection requires a small calculation for each instance value. When multiplied by the number of instances in a representative high-dimensionality data set, multiplied again by the number of training epochs, the number of calculations required to converge a large autoassociative network becomes truly vast.

Fortunately, there is an easy way around the problem. Any inputs to a neuron that are not being used to influence the transfer function can be ignored. A neuron with 1000 inputs may be using only a small fraction of them, and the others need not be calculated. A caveat to such a connection reduction method requires each unused interconnection to be sampled occasionally to see if it makes a difference if used. As the neuron moves its transfer function, it may be able to use other information from additional inputs. Dropping internal connections in this way can lead to a 90% reduction in interconnections—with a concomitant increase in training speed. When so configured, this is called a *sparsely connected network*. Although the predictive power of the network does degrade as the interconnectivity level falls, it is a graceful degradation. Such networks frequently retain 90% of their power with only 10% of the fully interconnected connections. (The exact performance depends very highly on the complexity of the function required, and so on.) A sparsely connected network that is making a good estimation of the required output values also strongly indicates that the required function can be approximated well with less hidden neurons.

10.4 Compressing Variables

The basic data compression tool, and the one included with the demonstration software on the accompanying CD-ROM, is the sparsely connected autoassociative neural network (SCANN). Its one task is to learn to predict its own inputs to some selected level of confidence—but with less neurons in the hidden layer than there are in the input and output layers. What does this achieve?

If the connections were “straight through,” that is, input connected directly to output, the task would be easy. Each input would perfectly predict the output, since the required prediction for the output value is the input value! But the fact is that there is not a direct connection. Whatever information is contained in the input has to be compressed into the hidden neurons. These neurons seek a relationship such that a few of them can predict, as accurately as needed, many outputs. And here, when the hidden neurons number less than the inputs, the information is squeezed into the hidden neurons. This is compressed information. Inasmuch as the hidden neurons can, at the outputs, re-create all of the values at the input, so they hold all of the information at the input, but in less neurons.

Suppose that a trained SCANN network is split between the hidden layer and the output layer. As instance values of a data set are applied to the input, the hidden-layer outputs are recorded in a file—one variable per hidden neuron. This “hidden-layer” file has less variables than the original file, yet holds information sufficient to re-create the input file. Applying the captured data to the output neurons re-creates the original file. This combination of outputs from the hidden neurons, together with the transform in the output neurons, shows that the information from many variables is compressed into fewer.

Compression serves to remove much of the redundancy embedded in the data. If several variables can be predicted from the same embedded information, that information only needs to be recognized once by the hidden neurons. Once the relationship is captured, the output neurons can use the single relationship recognized by the hidden neurons to re-create the several variables.

10.4.1 Using Compressed Dimensionality Data

When the miner and domain expert have a high degree of confidence that the training data set for the compression model is fully representative of the execution data, compression works well. A problem with compression is that the algorithm, at execution time, cannot know if the data being compressed can be recovered. If the execution data moves outside of the training sample parameters, the compression will not work, and the original values cannot be recovered. There are ways to establish a confidence level that the execution data originates from the same population distribution as the training sample (discussed in Chapter 11; see “Novelty Detection”), and these need to be used along with compression techniques.

Compressing data can offer great benefits. Compression is used because the dimensionality of the execution data set is too high for any modeling method to actually model the uncompressed data. Since a representative subset of the whole data set is used to build the compression model (part of the PIE described in Chapter 2), the compression model can be built relatively quickly. Using that compression model, the information in the full data set is quickly compressed for modeling. Compression, if practicable, reduces an intractable data set and puts it into tractable form. The compressed data can be modeled using any of the usual mining tools available to the miner, whereas the original data set cannot.

Creating a predictive model requires that the variable(s) to be predicted are kept out of the compressed data set. Because at execution time the prediction variable's value is unknown, it cannot be included for compression. If there are a very large number of prediction variables, they could be compressed separately and predicted in their compressed form. The decompression algorithm included in the PIE-O (see Chapter 2) will recover the actual predicted values.

Such compression models have been used successfully with both physical and behavioral data. Monitoring large industrial processes, for example, may produce data streams from high hundreds to thousands of instrumented monitoring points throughout the process. Since many instrumentation points very often turn out to be correlated (carry similar information), such as flow rates, temperature, and pressure, from many points, it is possible to compress such data for modeling very effectively. Compression models of the process require data from both normal and abnormal operations, as well as careful automated monitoring of the data stream to ensure that the data remains with the model's limits. This allows successful, real-time modeling and optimization of vast industrial processes.

Compressing large telecommunications data sets has also been successful for problem domains in which customer behavior changes relatively slowly over time. Huge behavioral data sets can be made tractable to model, particularly as many of the features are correlated, although often not highly. In very high dimensionality data sets there is enough redundancy in the information representation to facilitate good compression ratios.

10.5 Removing Variables

This is a last-ditch, when-all-else-fails option for the miner. However, sometimes there is nothing else to do if the dimensionality exceeds the limits of the compression or modeling tool and computer resources available.

The ideal solution requires removing redundant variables only. Redundant variables show a high degree of possibly nonlinear correlation. The solution, then, seems to be to remove variables that are most highly nonlinearly correlated. (Note that linearly correlated variables are also correlated when using nonlinear estimates. Linear correlation is, in a sense, just a special case of nonlinear correlation.) Unfortunately, there are many practical problems with discovering high-dimensionality nonlinear correlations. One of the problems is that there are an unlimited number of degrees of nonlinearity. However deep a so-far-fruitless search, it's always possible some yet greater degree of nonlinearity will show a high correlation. Another problem is deciding whether to compare single correlation (one variable with another one at a time) or multiple correlation (one variable against several at a time in combinations). Conducting such a search for massive dimensionality input spaces can be just as large a problem as trying to model the data—which couldn't be done, hence the search for variables to remove in the first place!

Once again the SCANN steps forward. Due to the sparse interconnections within the SCANN, it will build models of very high dimensionality data sets. The models will almost certainly be poor, but perhaps surprisingly, it is not the models themselves that are of interest when removing variables.

Recall that the weights of a SCANN are assigned random values when the network is set up. Even identically configured networks, training on identical data sets, are initiated with different neuron weights from training run to training run. Indeed, which neuron learns which piece of the overall function will very likely change from run to run. The precise way that two architecturally identical networks will represent a function internally may be totally different even though they are identically trained—same data, same number of epochs, and so on. Random initialization of weights ensures that the starting conditions are different.

Caution: Do not confuse a training cycle with an epoch. An epoch is a single pass through the training set. A training cycle is start-to-finish training to some degree of network convergence selected by the miner, usually consisting of many epochs.

However, variables that are important to the network will very likely remain important from training session to training session, random initialization or not. More significant from the standpoint of removing variables is that variables that remain unimportant from session to session are indeed unimportant. Whether highly correlated with other variables or not, the network does not use them much to create the input-to-output relationship. If these unimportant variables can be detected, they can be removed. The question is how to detect them.

10.5.1 Estimating Variable Importance 1: What Doesn’t Work

There is great danger in talking about using neuron weights to estimate variable importance because the values of the weights themselves are not a measure of the variable’s importance. This is a very important point and bears repeating: The importance of a variable to estimating the input-to-output relationship cannot be determined from inspecting the value of the weights. This may seem counterintuitive, and before looking at what can help, it is easier to frame the problem by looking at why inspecting weight values does not work.

The problem with looking at weight values is that the effects of different weights can be indirect and subtle. Suppose an input weight is very heavy—does it have a large effect on the output function? Not if the output neurons it connects to ignore it with a low weight. Or again, suppose that an input is very lightly weighted—is it unimportant? Not if all of the output neurons weight it heavily, thus amplifying its effect since it participates in every part of the function estimation.

With highly correlated inputs, just the sort that we are looking for, it may well be that the weights for two of them are very high—even if together they make no effect on the output! It could be that one of them has a large weight for the sole and exclusive purpose of nullifying the effect of the other large weight. The net result on the input-to-output function is nil.

If trying to untangle the interactions between input and hidden layer is hard, determining the effect over the whole input range of the network at the output neuron(s) is all but impossible.

There is no doubt that the values of the weights do have great significance, but only in terms of the entire structure of interactions within the network. And that is notoriously impenetrable to human understanding. The problem of explaining what a particular network configuration "means" is all but impossible for a network of any complexity.

10.5.2 Estimating Variable Importance 2: Clues

So if looking at the neuron weights doesn't help, what's left? Well, actually, it's the neuron weights. But it is not the level of the weights, it's the change in the weight from training cyle to training cycle.

Recall that, even for high-dimensionality systems, a SCANN trains relatively quickly. This is particularly so if the hidden layer is small relative to the input count. Speedy training allows many networks to be trained in a reasonable time. It is the "many networks" that is the key here. The automatic algorithm estimating variable importance starts by capturing the state of the initialized weights before each training cycle. After each training cycle is completed, the algorithm captures the adjusted state of the weights. It determines the difference between initial setting and trained setting, and accumulates that difference for every input weight. (The input neurons don't have any internal structure; the algorithm captures the input weights at the hidden layer.)

Looking at the total distance that any input weight has moved over the total number of training cycles gives an indication of importance. If the network always finds it necessary to move a weight (unless by happenstance it was set near to the correct weight when set up), it is clearly using the input for something. If consistently used, the input may well be important. More significantly, when looking for unimportant variables, if the network doesn't move the weight for an input much from cycle to cycle, it is definitely not using it for much. Since each weight is initially set at random and therefore takes on a range of values, if nonmoving during training, it clearly does not really matter what weighting the input is given. And if that is true, it won't matter much if the weighting is 0—which definitely means it isn't used.

Here is where to find clues to which variables are important. It is very unlikely that any variable has weights that are never moved by the network. It is the nature of the network to at least try many different configurations, so all weights will almost certainly be moved somewhat over a number of training cycles. However, when data set dimensionality is high, removing variables with small weight movement at least removes the variables that the network uses the least during many training cycles.

10.5.3 Estimating Variable Importance 3: Configuring and Training the Network

Configuring the training parameters for assessing variable importance differs from other training regimes. The quality of the final model is not important. It is important that the network does begin to converge. It is only when converging that the network demonstrates that it is learning. Network learning is crucial, since only when learning (improving performance) are the input variables accurately assessed for their importance. However, it is only important for variable importance assessment that significant convergence occurs.

Some rules of thumb. These are purely empirical, and the miner should adjust them in the light of experience. Actual performance depends heavily on the complexity of the relationship actually enfolded within the data set and the type and nature of the distributions of the variables, to mention only two factors. These are clearly not hard-and-fast rules, but they are a good place to start:

1. Initially configure the SCANN to have hidden-layer neurons number between 5% and 10% of the number of input variables.

2. Test train the network and watch for convergence. Convergence can be measured many ways. One measure is to create a pseudo-correlation coefficient looking simultaneously at network prediction error for all of the input/outputs from epoch to epoch. Ensure that the pseudo-r^2 (pr^2) measure improves by at least 35–50%. (See Chapter 9 for a description of r^2. Since this is a pseudo-correlation, it has a pseudo-r^2.) Failing that, use any measure of convergence that is convenient, but make sure the thing converges!

3. If the network will not converge, increase the size of the hidden layer until it does.

4. With a network configuration that converges during training, begin the Importance Detection Training (IDT) cycles.

5. Complete at least as many IDT cycles as the 0.6th power of the number of inputs (IDT cycles = $\text{Inputs}^{0.6}$).

6. Complete each IDT cycle when convergence reaches a 35–50% pr^2 improvement.

7. Cut as many input variables as needed up to 33% of the total in use.

8. If the previous cut did not reduce the dimensionality sufficiently for compression, go round again from the start.

9. Don’t necessarily cut down to modeling size; compress when possible.

10. Do it all again on the rejected variables. If true redundancy was eliminated from the first set, both sets of variables should produce comparable models.

Following this method, in one application the automated data reduction schedule was approximately as follows:

1. Starting with the 7000+-variable, multiterabyte data set mentioned earlier, a representative sample was extracted. About 2000 variables were collapsed into 5 as highly sparse variables. Roughly 5000 remained. The SCANN was initially configured with 500 hidden neurons, which was raised to 650 to aid convergence. 35% pr^2 convergence was required as an IDT terminating criterion. (Twenty-five epochs with less than 1% improvement was also a terminating criterion, but was never triggered.) After about 170 IDT cycles ($5000^{0.6} = 166$), 1500 variables were discarded.

2. Restarting with 3500 variables, 400 hidden neurons, 35% pr^2 convergence, and 140 IDT cycles, 1000 variables were discarded.

3. Restarting with 2500 variables, 300 hidden neurons, 35% pr^2 convergence, and 120 IDT cycles, 1000 variables were discarded.

4. The remaining 1500 variables were compressed into 700 with a minimum 90% confidence for data retention.

5. For the discarded 3500 variables left after the previous extraction, 2000 were eliminated using a similar method as above. This produced two data sets comprised of different sets of variables.

6. From both extracted variable data sets, separate predictive models were constructed and compared. Both models produced essentially equivalent results.

This data reduction methodology was largely automated, requiring little miner intervention once established. Running on parallel systems, the data reduction phase took about six days before modeling began. In a manual data reduction run at the same time, domain experts selected the variables considered significant or important, and extracted a data set for modeling. For this particular project, performance was measured in terms of “lift”—how much better the model did than random selection. The top 20% of selections for the domain expert model achieved a lift approaching three times. The extracted, compressed model for the top 20% of its selections produced a lift of better than four times.

The problem with this example is that it is impossible to separate the effect of the data reduction from the effect of data compression. As an academic exercise, the project suffers from several other shortcomings since much was not controlled, and much data about performance was not collected. However, as a real-world, automated solution to a particularly intractable problem requiring effective models to be mined from a massive data set to meet business needs, this data reduction method proved invaluable.

In another example, a data set was constructed from many millions of transaction records by reverse pivoting. The miners, together with domain experts, devised a number of features hypothesized to be predictive of consumer response to construct from the transaction data. The resulting reverse pivot produced a source data set for mining with more than 1200 variables and over 6,000,000 records. This data set, although not enormous by many standards (totaling something less than half a terabyte), was nonetheless too large for the mining tool the customer had selected, causing repeated mining software failures and system crashes during mining.

The data reduction methodology described above reduced the data set (no compression) to 35 variables in less than 12 hours total elapsed time. The mining tool was able to digest the reduced data set, producing an effective and robust model. Several other methods of selecting variables were tried on the same data set as a validation, but no combination produced a model as effective as the original combination of automatically selected variables.

Further investigation revealed that five of the reduction system selected variables carried unique information that, if not used, prevented any other model not using them from being as effective. The automatic reduction technique clearly identified these variables as of highest importance, and they were included in the primary model. Since they had been selected by the SCANN, they weren’t used in the check data set made exclusively from the discards. Without these key variables, no check model performed as well. However, SCANN importance ranking singled them out, and further investigation revealed these five as the key variables. In this case, the discarded information was indeed redundant, but was not sufficient to duplicate the information retained.

10.6 How Much Data Is Enough?

Chapter 5 examined the question of how much data was enough in terms of determining how much data was needed to prepare individual variables. Variability turned out to be a part of the answer, as did establishing suitable confidence that the variability was captured. When considering the data set, the miner ideally needs to capture the variability of the whole data set to some acceptable degree of confidence, not just the variability of each variable individually. Are these the same thing?

There is a difference between individual variability and joint variability. Joint variability was discussed previously—for instance, in Chapter 6, maintaining joint variability between variables was used as a way of determining a suitable replacement value for missing values. Recall that joint variability is a measure of how the values of one (or more) variable's values change as another (or several) variable's values also change. The joint variability of a data set is a measure of how all of the values change based on all of the other variables in the data set.

Chapter 5 showed that because of the close link between distribution and variability, assuring that the distribution of a variable was captured to a selected degree of confidence assured that the variability of the variable was captured. By extension of the same argument, the problem of capturing enough data for modeling can be described in terms of capturing, to some degree of confidence, the joint distribution of the data set.

10.6.1 Joint Distribution

Joint distributions extend the ideas of individual distribution. Figure 10.11 shows a two-dimensional state space (Chapter 6 discusses state space) showing the positions of 100 data points. The distribution histograms for the two variables are shown above and to the right of the graphical depiction of state space. Each histogram is built from bars that cover part of the range. The height of each bar indicates the number of instance values for the variable in the range the bar covers. The continuous curved lines on the histograms trace a normal distribution curve for the range of each variable. The normal curve measures nothing about the variable—it is there only for comparison purposes, so that it is easier to judge how close the histogram distribution is to a normal distribution. Comparing each histogram with the normal curve shows that the variables are at least approximately normally distributed, since the bar heights roughly approximate the height of the normal curve. Looking at the state space graph shows, exactly as expected with such individual distributions, that the density of the data points is higher around the center of this state space than at its borders. Each individual variable distribution has maximum density at its center, so the joint density of the state space too is greatest at its center.

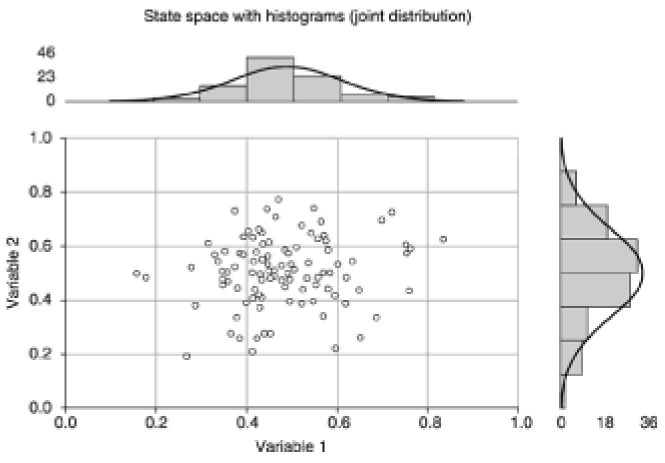


Figure 10.11 One hundred data points of two variables mapped into a 2D state

space with histograms showing the distribution of the individual variables compared with the normal curve.

The density of a one-dimensional distribution can be shown by the height of a continuous line, rather than histogram bars. This is exactly what the normal curve shows. Such a continuous line is, of course, a manifold. By extension, for two dimensions the density-mapping manifold is a surface.

Figure 10.12 shows the density manifold for the state space shown in Figure 10.11. The features in the manifold, shown by the unevenness (lumps and bumps), are easy to see. Every useful distribution has such features that represent variance in the density, regardless of the number of dimensions. (Distributions that have no variance also carry no useable information.) By extension from the single variable case (Chapter 5), as instance count in a sample increases, density structures begin to emerge. The larger the sample, the better defined and more stable become the features. With samples that are not yet representative of the population's distribution, adding more samples redefines (moves or changes) the features. When the sample is representative, adding more data randomly selected from the population essentially reinforces, but does not change, the features of the multidimensional distribution.

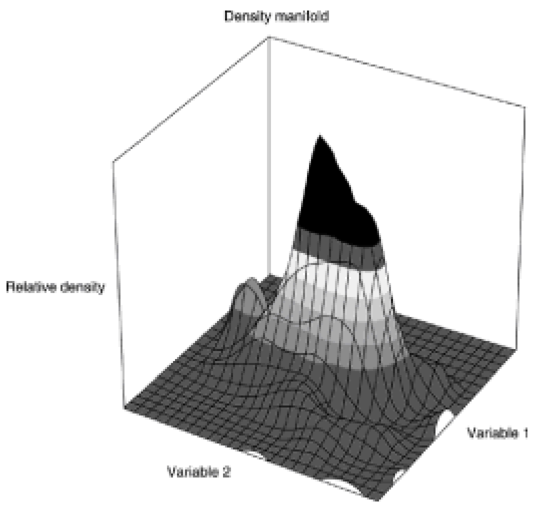


Figure 10.12 Density manifold for data points in a 2D state space. The manifold clearly shows density features ("lumps and bumps" in the manifold), as the density does not vary uniformly across state space.

It might seem that fully capturing the variability of individual variables is enough to capture the variability of the population. This, unfortunately, is not the case. A simple example shows that capturing individual variable variability is not necessarily sufficient to capture joint variability.

Suppose that two variables each have three possible values: 1, 2, and 3. The proportions in which each value occurs in the population of each variable are shown in Table 10.1.

TABLE 10.1 Value frequency for two variables.TABLE 10.1

Variable A values	Variable A proportion	Variable B values	Variable B proportion
1	10%	1	10%
2	20%	2	40%
3	70%	3	50%

Table 10.1 shows, for example, that 10% of the values of variable A are 1, 40% of the values of variable B are 2, and so on. It is very important to note that the table does not imply which values of variable A occur with which values of variable B. For instance, the 10% of variable A instances that have the value 1 may be paired with any value of variable B—1, 2, or 3. The table does not imply that values are in any way matched with each other.

Figure 10.13 shows distributions in samples of each variable. Recall that since each of the histograms represents the distribution of a sample of the population, each distribution approximates the distribution of the population and is not expected to exactly match the population distribution. Two separate samples of the population are shown, for convenience labeled “sample 1” and “sample 2.” The figure clearly shows that the sample 1 and sample 2 distributions for variable A are very similar to each other, as are those for variable B.

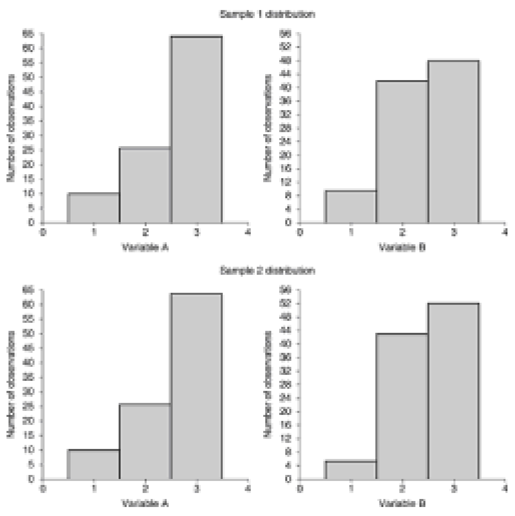


Figure 10.13 Individual distributions of two variables (A and B) from two representative samples (1 and 2) show very similar individual distributions in each sample.

If it is true that capturing the individual variability of each variable also captures the joint variability of both, then the joint variability of both samples should be very similar. Is it? Figure 10.14 shows very clearly that they are totally different. The histograms above and to the right of each joint distribution plot show the individual variable distributions. These distributions for both samples are exactly as shown in the previous figure. Yet the joint distribution, shown by the positions of the small circles, are completely different. The circles show where joint values occur—for instance, the circle at variable A = 3 and variable B = 3 shows that there are instances where the values of both variables are 3. It is abundantly clear that while the individual distributions of the variables in the two samples are essentially identical, the joint distributions are totally different.

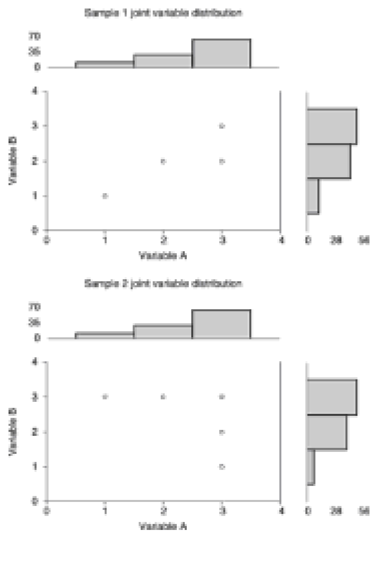


Figure 10.14 Although the individual variables’ distributions are effectively

identical in each sample, the joint distribution, shown by the position of the circles, is markedly different between the two samples.

Figure 10.15 shows the joint histogram distributions for the two samples. The height of the columns is proportional to the number of instances having the A and B values shown. Looking at these images, the difference between the joint distributions is plain to see, although it is not so obvious that the individual distributions of the two variables are almost unchanged in the two samples. The column layout for the two samples is the same as the point layout in the previous figure. The joint histogram, Figure 10.15, shows the relative differences in joint values by column height. Figure 10.14 showed which joint values occurred, but without indicating how many of each there were.

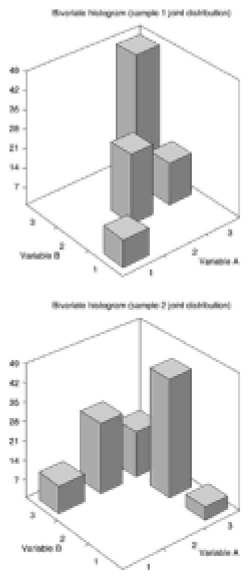


Figure 10.15 Joint distribution histograms for variables A and B are clearly seen as different between the two samples. The height of the column represents the number of instances having a particular joint combination of values.

The key point to understand from this example is that capturing individual variable variability to any degree of confidence does not in itself provide any degree of confidence that joint variability is captured.

10.6.2 Capturing Joint Variability

The clue to capturing joint variability has already been shown in Figure 10.12. This figure showed a relative-density plot. It mapped the density of a two-dimensional distribution. Very clearly it showed peaks and valleys—high points and low—representing the density of the distribution at any point on the map. Clearly, the exact shape of the surface reflects the underlying joint distribution of the sample, since the distribution is only an expression of the density of instance values in state space.

Just as a population has a specific, characteristic distribution for each individual variable (discussed in Chapter 5), so too the population has some particular characteristic joint distribution. As a sample gets larger, it better reflects the actual population distribution. At first, as instance values are added to the sample, the density manifold will change its shape—the peaks and valleys will move in state space. New ones may appear, and old ones disappear. When the sample is representative of the joint distribution, then the shape of the surface will change little as more data from the same population is added. This is no more than a multidimensional description of the way that a single variable was sampled in Chapter 5. By extension of the same discussion, multidimensional variability can be captured, to any selected degree of confidence, using density manifold stability.

But here is where data preparation steps into the data survey. The data survey (Chapter 11) examines the data set as a whole from many different points of view. Two of the questions a miner may need answers to are

1. Given a data set, what is the justifiable level of confidence that the joint variability has been captured?

2. Given a data set, how much more data is estimated to be needed to yield a given level of confidence that the sample is representative of the population?

Both of these questions are addressed during the survey. Both questions directly address the level of confidence justifiable for particular inferential or predictive models. The answers, of course, depend entirely on estimating the confidence for capturing joint variability.

Why is estimating joint variability confidence part of the survey, rather than data preparation? Data preparation concentrates on transforming and adjusting variables' values to ensure maximum information exposure. Data surveying concentrates on examining a prepared data set to glean information that is useful to the miner. Preparation manipulates values; surveying answers questions.

In general, a miner has limited data. When the data is prepared, the miner needs to know what level of confidence is justified that the sample data set is representative. If more data is available, the miner may ask how much more is needed to establish some greater required degree of confidence in the predictions or inferences. But both are questions about a prepared data set; this is not preparation of the data. Thus the answers fall under the bailiwick of data survey.

10.6.3 Degrees of Freedom

Degrees of freedom measure, approximately, how many things there are to change in a system that can affect the outcome. The more of them that there are, the more likely it is that, purely by happenstance, some particular, but actually meaningless, pattern will show up. The number of variables in a data set, or the number of weights in a neural network, all represent things that can change. So, yet again, high-dimensionality problems turn up, this time expressed as degrees of freedom. Fortunately for the purposes of data preparation, a definition of degrees of freedom is not needed as, in any case, this is a problem previously encountered in many guises. Much discussion, particularly in this chapter, has been about reducing the dimensionality/combinatorial explosion problem (which is degrees of freedom in disguise) by reducing dimensionality. Nonetheless, a data set always has some dimensionality, for if it does not, there is no data set! And having some particular dimensionality, or number of degrees of freedom, implies some particular chance that spurious patterns will turn up. It also has implications about how much data is needed to ensure that any spurious patterns are swamped by valid, real-world patterns. The difficulty is that the calculations are not exact because several needed measures, such as the number of significant system states, while definable in theory, seem impossible to pin down in practice. Also, each modeling tool introduces its own degrees of freedom (weights in a neural network, for example), which may be unknown to the minere .mi..

The ideal, if the miner has access to software that can make the measurements (such as data surveying software), requires use of a multivariable sample determined to be representative to a suitable degree of confidence. Failing that, as a rule of thumb for the minimum amount of data to accept, for mining (as opposed to data preparation), use *at least* twice the number of instances required for a data preparation representative sample. The key is to have enough representative instances of data to swamp the spurious patterns. Each significant system state needs sufficient representation, and having a truly representative sample of data is the best way to assure that.

10.7 Beyond Joint Distribution

So far, so good. Capturing the multidimensional distribution captures a representative sample of data. What more is needed? On to modeling!

Unfortunately, things are not always quite so easy. Having a representative sample in hand is a really good start, but it does not assure that the data set is modelable! Capturing a representative sample is an essential minimum—that, and knowing what degree of confidence is justified in believing the sample to be representative. However, the miner needs a modelable representative sample, and the sample simply being representative of the population may not be enough. How so?

Actually, there are any number of reasons, all of them domain specific, why the minimum representative sample may not suffice—or indeed, why a nonrepresentative sample is needed. (Heresy! All this trouble to ensure that a fully representative sample is collected, and now we are off after a nonrepresentative sample. What goes on here?)

Suppose a marketing department needs to improve a direct-mail marketing campaign. The normal response rate for the random mailings so far is 1.5%. Mailing rolls out, results trickle in. A (neophyte) data miner is asked to improve response. “Aha!,” says the miner, “I have just the thing. I’ll whip up a quick response model, infer who’s responding, and redirect the mail to similar likely responders. All I need is a genuinely representative sample, and I’ll be all set!” With this terrific idea, the miner applies the modeling tools, and after furiously mining, the best prediction is that no one at all will respond! Panic sets in; staring failure in the face, the neophyte miner begins the balding process by tearing out hair in chunks while wondering what to do next.

Fleeing the direct marketers with a modicum of hair, the miner tries an industrial chemical manufacturer. Some problem in the process occasionally curdles a production batch. The exact nature of the process failure is not well understood, but the COO just read a business magazine article extolling the miraculous virtues of data mining. Impressed by the freshly minted data miner (who has a beautiful certificate attesting to skill in mining), the COO decides that this is a solution to the problem. Copious quantities of data are available, and plenty more if needed. The process is well instrumented, and continuous chemical batches are being processed daily. Oodles of data representative of the process are on hand. Wielding mining tools furiously, the miner conducts an onslaught designed to wring every last confession of failure from the recalcitrant data. Using every art and artifice, the miner furiously pursues the problem until, staring again at failure and with desperation setting in, the miner is forced to fly from the scene, yet more tufts of hair flying.

Why has the now mainly hairless miner been so frustrated? The short answer is that while the data is representative of the population, it isn’t representative of the *problem*. Consider the direct marketing problem. With a response rate of 1.5%, any predictive system has an accuracy of 98.5% if it uniformly predicts “No response here!” Same thing with the chemical batch processing—lots of data in general, little data about the failure conditions.

Both of these examples are based on real applications, and in spite of the light manner of introducing the issue, the problem is difficult to solve. The feature to be modeled is insufficiently represented for modeling in a data set that is representative of the population. Yet, if the mining results are to be valid, the data set mined must be representative of the population or the results will be biased, and may well be useless in practice. What to do?

10.7.1 Enhancing the Data Set

When the density of the feature to be modeled is very low, clearly the density of that feature needs to be increased—but in a way that does least violence to the distribution of the population as a whole. Using the direct marketing response model as an example, simply increasing the proportion of responders in the sample may not help. It's assumed that there are some other features in the sample that actually do vary as response varies. It's just that they're swamped by spurious patterns, but only because of their low density in the sample. Enhancing the density of responders is intended to enhance the variability of connected features. The hope is that when enhanced, these other features become visible to the predictive mining tool and, thus, are useful in predicting likely responders.

These assumptions are to some extent true. Some performance improvement may be obtained this way, usually more by happenstance than design, however. The problem is that low-density features have more than just low-level interactions with other, potentially predictive features. The instances with the low-level feature represent some small proportion of the whole sample and form a subsample—the subsample containing only those instances that have the required feature. Considered alone, because it is so small, the subsample almost certainly does not represent the sample as a whole—let alone the population. There is, therefore, a very high probability that the subsample contains much noise and bias that are in fact totally unrelated to the feature itself, but are simply concomitant to it in the sample taken for modeling.

Simply increasing the desired feature density also increases the noise and bias patterns that the subsample carries with it—and those noise and bias patterns will then appear to be predictive of the desired feature. Worse, the enhanced noise and bias patterns may swamp any genuinely predictive feature that is present.

This is a tough nut to crack. It is very similar to any problem of extracting information from noise, and that is the province of information theory, discussed briefly in Chapter 11 in the context of the data survey. One of the purposes of the data survey is to understand the informational structure of the data set, particularly in terms of any identified predictive variables. However, a practical approach to solving the problem does not depend on the insights of the data survey, helpful though they might be. The problem is to construct a sample data set that represents the population as much as possible while enhancing some particular feature.

Feature Enhancement with Plentiful Data

If there is plenty of data to draw upon, instances of data with the desired feature may also be plentiful. This is the case in the first example above. The mailing campaign produces many responses. The problem is their low density as a proportion of the sample. There may be thousands or tens of thousands of responses, even though the response rate is only 1.5%.

In such a circumstance, the shortage of instances with the desired feature is not the problem, only their relative density in the mining sample. With plenty of data available, the miner constructs two data sets, both fully internally representative of the population—except for the desired feature. To do this, divide the source data set into two subsets such that one subset has only instances that contain the feature of interest and the other subset has no instances that contain the feature of interest. Use the already described techniques (Chapter 5) to extract a representative sample from each subset, ignoring the effect of the key feature. This results in two separate subsets, both similar to each other and representative of the population as a whole when ignoring the effect of the key feature. They are effectively identical except that one has the key feature and the other does not.

Any difference in distribution between the two subsets is due either to noise, bias, or the effect of the key feature. Whatever differences there are should be investigated and validated whatever else is done, but this procedure minimizes noise and bias since both data sets are representative of the population, save for the effect of the key feature. Adding the two subsets together gives a composite data set that has an enhanced presence of the desired feature, yet is as free from other bias and noise as possible.

Feature Enhancement with Limited Data

Feature enhancement is more difficult when there is only limited data available. This is the case in the second example of the chemical processor. The production staff bends every effort to prevent the production batch from curdling, which only happens very infrequently. The reasons for the batch failure are not well understood anyway (that is what is to be investigated), so may not be reliably reproducible. Whether possible or not, batch failure is a highly expensive event, hitting directly at the bottom line, so deliberately introducing failure is simply not an option management will countenance. The miner was constrained to work with the small amount of failure data already collected.

Where data is plentiful, small subsamples that have the feature of interest are very likely to also carry much noise and bias. Since more data with the key feature is unavailable, the miner is constrained to work with the data at hand. There are several modeling techniques that are used to extract the maximum information from small subsamples, such as multiway cross-validation on the small feature sample itself, and intersampling and resampling techniques. These techniques do not affect data preparation since they are only properly applied to already prepared data. However, there is one data preparation technique used when data instances with a key feature are particularly low in density: data multiplication.

The problem with low feature-containing instance counts is that the mining tool might learn the specific pattern in each instance and take those specific patterns as predictive. In other words, low key feature counts prevent some mining tools from generalizing from the few instances available. Instead of generalizing, the mining tool learns the particular instance configurations—which is particularizing rather than generalizing. *Data multiplication* is the process of creating additional data instances that appear to have the feature of interest. White (or colorless) noise is added to the key feature subset, producing a second data subset. (See Chapter 9 for a discussion of noise and colored noise.) The interesting thing about the second subset is that its variables all have the same mean values, distributions and so on, as the original data set—yet no two instance values, except by some small chance, are identical. Of course, the noise-added data set can be made as large as the miner needs. If duplicates do exist, they should be removed.

When added to the original data set, these now appear as more instances with the feature, increasing the apparent count and increasing the feature density in the overall data set. The added density means that mining tools will generalize their predictions from the multiplied data set. A problem is that any noise or bias present will be multiplied too. Can this be reduced? Maybe.

A technique called color matching helps. Adding white noise multiplies everything exactly as it is, warts and all. Instead of white noise, specially constructed colored noise can be added. The multidimensional distribution of a data sample representative of the population determines the precise color. *Color matching* adds noise that matches the multivariable distribution found in the representative sample (i.e., it is the same color, or has the same spectrum). Any noise or bias present in the original key feature subsample is still present, but color matching attempts to avoid duplicating the effect of the original bias, even diluting it somewhat in the multiplication.

As always, whenever adding bias to a data set, the miner should put up mental warning flags. Data multiplication and color matching adds features to, or changes features of, the data set that simply are not present in the real world—or if present, not at the density found after modification. Sometimes there is no choice but to modify the data set, and frequently the results are excellent, robust, and applicable. Sometimes even good results are achieved where none at all were possible without making modifications. Nonetheless, biasing data calls for extreme caution, with much validation and verification of the results before applying them.

10.7.2 Data Sets in Perspective

Constructing a composite data set enhances the visibility of some pertinent feature in the data set that is of interest to the miner. Such a data set is no longer an unbiased sample, even if the original source data allowed a truly unbiased sample to be taken in the first place. Enhancing data makes it useful only from one particular point of view, or from a particular perspective. While more useful in particular circumstances, it is nonetheless not so useful in general. It has been biased, but with a purposeful bias deliberately introduced. Such data has a *perspective*.

When mining perspectival data sets, it is very important to use nonperspectival test and evaluation sets. With the best of intentions, the mining data has been distorted and, to at least that extent, no longer accurately represents the population. The only place that the inferences or predictions can be examined to ensure that they do not carry an unacceptable distortion through into the real world is to test them against data that is as undistorted—that is, as representative of the real world—as possible.

10.8 Implementation Notes

Of the four topics covered in this chapter, the demonstration code implements algorithms for the problems that can be automatically adjusted without high risk of unintended data set damage. Some of the problems discussed are only very rarely encountered or could cause more damage than benefit to the data if applied without care. Where no preparation code is available, this section includes pointers to procedures the miner can follow to perform the particular preparation activity.

10.8.1 Collapsing Extremely Sparsely Populated Variables

The demonstration code has no explicit support for collapsing extremely sparsely populated variables. It is usual to ignore such variables, and only in special circumstances do they need to be collapsed. Recall that these variables are usually populated at levels of small fractions of 1%, so a much larger proportion than 99% of the values are missing (or empty).

While the full tool from which the demonstration code was drawn will fully collapse such variables if needed, it is easy to collapse them manually using the statistics file and the complete-content file produced by the demonstration code, along with a commercial data manipulation tool, say, an implementation of SQL. Most commercial statistical packages also provide all of the necessary tools to discover the problem, manipulate the data, and create the derived variables.

1. If using the demonstration code, start with the “stat” file.

2. Identify the population density for each variable.

3. Check the number of discrete values for each candidate sparse variable.

4. Look in the complete-content file, which lists all of the values for all of the variables.

5. Extract the lists for the sparse variables.

6. Access the sample data set with your tool of choice and search for, and list, those cases where the sparse variables simultaneously have values. (This won’t happen often, even in sparse data sets.)

7. Create unique labels for each specific present-value pattern (PVP).

8. Numerate the PVPs.

Now comes the only tricky part. Recall that the PVPs were built from the representative *sample*. (It's representative only to some selected degree of confidence.) The execution data set may, and if large enough almost certainly will, contain a PVP that was not in the sample data set. If important, and only the domain of the problem provides that answer, create labels for all of the *possible* PVPs, and assign them appropriate values. That is a judgment call. It may be that you can ignore any unrecognized PVPs, or more likely, flag them if they are found.

10.8.2 Reducing Excessive Dimensionality

Neural networks comprise a vast topic on their own. The brief introduction in this chapter only touched the surface. In keeping with all of the other demonstration code segments, the neural network design is intended mainly for humans to read and understand. Obviously, it also has to be read (and executed) by computer systems, but the primary focus is that the internal working of the code be as clearly readable as possible. Of all the demonstration code, this requirement for clarity most affects the network code. The network is not optimized for speed, performance, or efficiency. The sparsity mechanism is modified random assignment without any dynamic interconnection. Compression factor (hidden-node count) is discovered by random search.

The included code demonstrates the key principles involved and compresses information. Code for a fully optimized autoassociative neural network, including dynamic connection search with modified cascade hidden-layer optimization, is an impenetrable beast! The full version, from which the demonstration is drawn, also includes many other obfuscating (as far as clarity of reading goes) “bells and whistles.” For instance, it includes modifications to allow maximum compression of information into the hidden layer, rather than spreading it between hidden and output layers, as well as modifications to remove linear relationships and represent those separately. While improving performance and compression, such features completely obscure the underlying principles.

10.8.3 Measuring Variable Importance

Everything just said about neural networks for data compression applies when using the demonstration code to measure variable importance. For explanatory ease, both data compression and variable importance estimation use the same code segment. A network optimized for importance search can, once again, improve performance, but the principles are as well demonstrated by any SCANN-type BP-ANN.

10.8.4 Feature Enhancement

To enhance features in a data set, build multiple representative data subsets, as described, and merge them.

Describing construction of colored noise for color matching, if needed, is unfortunately outside the scope of the present book. It involves significant multivariable frequency modeling to reproduce a characteristic noise pattern emulating the sample multivariable distribution. Many statistical analysis software packages provide the basic tools for the miner to characterize the distribution and develop the necessary noise generation function.

10.9 Where Next?

A pause at this point. Data preparation, the focus of this book, is now complete. By applying all of the insights and techniques so far covered, raw data in almost any form is turned into clean prepared data ready for modeling. Many of the techniques are illustrated with computer code on the accompanying CD-ROM, and so far as data preparation for data mining is concerned, the journey ends here.

However, the data is still unmined. The ultimate purpose of preparing data is to gain understanding of what the data “means” or predicts. The prepared data set still has to be used. How is this data used? The last two chapters look not at preparing data, but at surveying and using prepared data.

Chapter 11: The Data Survey

Overview

Suppose that three separate families are planning a vacation. The Abbott family really enjoys lake sailing. Their ideal vacation includes an idyllic mountain lake, surrounded by trees, with plenty of wildlife and perhaps a small town or two nearby in case supplies are needed. They need only a place to park their car and boat trailer, a place to launch the boat, and they are happy.

The Bennigans are amateur archeologists. There is nothing they like better than to find an ancient encampment, or other site, and spend their time exploring for artifacts. Their four-wheel-drive cruiser can manage most terrain and haul all they need to be entirely self-sufficient for a couple of weeks exploring—and the farther from civilization, the better they like it.

The Calloways like to stay in touch with their business, even while on vacation. Their ideal is to find a luxury hotel in the sun, preferably near the beach but with nightlife. Not just any nightlife; they really enjoy cabaret, and would like to find museums to explore and other places of interest to fill their days.

These three families all have very different interests and desires for their perfect vacation. Can they all be satisfied? Of course. The locations that each family would like to find and enjoy exist in many places; their only problem is to find them and narrow down the possibilities to a final choice. The obvious starting point is with a map. Any map of the whole country indicates broad features—mountains, forests, deserts, lakes, cities, and probably roads. The Abbotts will find, perhaps, the Finger Lakes in upstate New York a place to focus their attention. The Bennigans may look at the deserts of the Southwest, while the Calloways look to Florida. Given their different interests, each family starts by narrowing down the area of search for their ideal vacation to those general areas of the country that seem likely to meet their needs and interests.

Once they have selected a general area, a more detailed map of the particular territory lets each family focus in more closely. Eventually, each family will decide on the best choice they can find and leave for their various vacations. Each family explores its own vacation site in detail. While the explorations do not seem to produce maps, they reveal small details—the very details that the vacations are aimed at. The Abbotts find particular lake coves, see particular trees, and watch specific birds and deer. The Bennigans find individual artifacts in specific places. The Calloways enjoy particular cabaret performers and see specific exhibits at particular museums. It is these detailed explorations that each family feels to be the whole purpose for their vacations.

Each family started with a general search to find places likely to be of interest. Their initial search was easy. The U.S. Geological Survey has already done the hard work for them. Other organizations, some private survey companies, have embellished maps in particular ways and for particular purposes—road maps, archeological surveys, sailing maps (called “charts”), and so on. Eventually, the level of detail that each family needed was more than a general map could provide. Then the families constructed their own maps through detailed exploration.

What does this have to do with data mining? The whole purpose of the data survey is to help the miner draw a high-level map of the territory. With this map, a data miner discovers the general shape of the data, as well as areas of danger, of limitation, and of usefulness. With a map, the Abbotts avoided having to explore Arizona to see if any lakes suitable for sailing were there. With a data survey, a miner can avoid trying to predict the stock market from meteorological data. “Everybody knows” that there are no lakes in Arizona. “Everybody knows” that the weather doesn’t predict the stock market. But these “everybodies” only know that through experience—mainly the experience of others who have been there first. Every territory needed exploring by pioneers—people who entered the territory first to find out what there was in general—blazing the trail for the detailed explorations to follow. The data survey provides a miner with a map of the territory that guides further exploration and locates the areas of particular interest, the areas suitable for mining. On the other hand, just as with looking for lakes in Arizona, if there is no value to be found, that is well to know as early as possible.

11.1 Introduction to the Data Survey

This chapter deals entirely with the data survey, a topic at least as large as data preparation. The introduction to the use, purposes, and methods of data surveying in this chapter discusses how prepared data is used during the survey. Most, if not all, of the surveying techniques can be automated. Indeed, the full suite of programs from which the data preparation demonstration code is drawn is a full data preparation and survey tool set. This chapter touches only on the main topics of data surveying. It is an introduction to the territory itself. The introduction starts with understanding the concept of “information.”

This book mentions “information” in several places. “Information is embedded in a data set.” “The purpose of data preparation is to best expose information to a mining tool.” “Information is contained in variability.” Information, information, information. Clearly, “information” is a key feature of data preparation. In fact, information—its discovery, exposure, and understanding—is what the whole preparation-survey-mining endeavor is about. A data set may represent information in a form that is not easily, or even at all, understandable by humans. When the data set is large, understanding significant and salient points becomes even more difficult. Data mining is devised as a tool to transform the impenetrable information embedded in a data set into understandable relationships or predictions.

However, it is important to keep in mind that mining is not designed to extract information. Data, or the data set, enfolds information. This information describes many and various relationships that exist enfolded in the data. When mining, the information is being mined for what it contains—an explanation or prediction based on the embedded relationships. It is almost always an explanation or prediction of specific details that solves a problem, or answers a question, within the domain of inquiry—very often a business problem. What is required as the end result is *human understanding* (enabling, if necessary, some action). Examining the nature of, and the relationships in, the information content of a data set is a part of the task of the data survey. It prepares the path for the mining that follows.

Some information is always present in the data—understandable or not. Mining finds relationships or predictions embedded in the information inherent in a data set. With luck, they are not just the obvious relationships. With more luck, they are also useful. In discovering and clarifying some novel and useful relationship embedded in data, data mining has its greatest success. Nonetheless, the information exists prior to mining. The data set enfolds it. It has a shape, a substance, a structure. In some places it is not well defined; in others it is bright and clear. It addresses some topics well; others poorly. In some places, the relationships are to be relied on; in others not. Finding the places, defining the limits, and understanding the structures is the purpose of data surveying.

The fundamental question posed by the data survey is, "Just what information is in here anyway?"

11.2 Information and Communication

Everything begins with information. The data set embeds it. The data survey surveys it. Data mining translates it. But what exactly is *information*? The *Oxford English Dictionary* begins its definition with "The act of informing, . . ." and continues in the same definition a little later, "Communication of instructive knowledge." The act referred to is clearly one where this thing, "information," is passed from one person to another. The latter part of the definition explicates this by saying it is "communication." It is in this sense of communicating intelligence—transferring insight and understanding—that the term "information" is used in data mining. Data possesses information only in its latent form. Mining provides the mechanism by which any insight potentially present is explicated. Since information is so important to this discussion, it is necessary to try to clarify, and if possible quantify, the concept.

Because information enables the transferring of insight and understanding, there is a sense in which quantity of information relates to the amount of insight and understanding generated; that is, more information produces greater insight. But what is it that creates greater insight?

A good mystery novel—say, a detective story—sets up a situation. The situation described includes all of the necessary pieces to solve the mystery, but in a nonobvious way. Insight comes when, at the end of the story, some key information throws all of the established structure into a suddenly revealed, surprising new relationship. The larger and more complex the situation that the author can create, the greater the insight when the true situation is revealed. But in addition to the complexity of the situation, it seems to be true that the more surprising or unexpected the solution, the greater the insight.

The detective story illustrates the two key ingredients for insight. The first is what for a detective story is described as “the situation.” The situation comprises a number of individual components and the relationship between the components. For a detective story, these components are typically the characters, the attributes of characters, their relationship to one another, and the revealed actions taken by each during the course of the narrative. These various components, together with their relationships, form a knowledge structure. The second ingredient is the communication of a key insight that readjusts the knowledge structure, changing the relationship between the components. The amount of insight seems intuitively related to how much readjustment of the knowledge structure is needed to include the new insight, and the degree to which the new information is unexpected.

As an example, would you be surprised if you learned that to the best of modern scientific knowledge, the moon really is made of green cheese? Why? For a start, it is completely unexpected. Can you honestly say that you have ever given the remotest credence to the possibility that the moon might really be made of green cheese? If true, such a simple communication carries an enormous amount of information. It would probably require you to reconfigure a great deal of your knowledge of the world. After all, what sort of possible rational explanation could be constructed to explain the existence of such a phenomenon? In fact, it is so unlikely that it would almost certainly take much repetition of the information in different contexts (more evidence) before you would accept this as valid. (Speaking personally, it would take an enormous readjustment of my world view to accept any rational explanation that includes several trillion tons of curdled milk products hanging in the sky a quarter of a million miles distant!)

These two very fundamental points about information—how surprising the communication is, and how much existing knowledge requires revision—both indicate something about how much information is communicated. But these seem very subjective measures, and indeed they are, which is partly why defining information is so difficult to come to grips with.

Claude E. Shannon did come to grips with the problem in 1948. In what has turned out to be one of the seminal scientific papers of the twentieth century, “A Mathematical Theory of Communication,” he grappled directly with the problem. This was published the next year as a book and established a whole field of endeavor, now called “information theory.” Shannon himself referred to it as “communication theory,” but its effects and applicability have reached out into a vast number of areas, far beyond communications. In at least one sense it is only about communications, because unless information is communicated, it informs nothing and no one. Nonetheless, information theory has come to describe information as if it were an object rather than a process. A more detailed look at information will assume where needed, at least for the sake of explanation, that it exists as a thing in itself.

11.2.1 Measuring Information: Signals and Dictionaries

Information comes in two pieces: 1) an informing communication and 2) a framework in which to interpret the information. For instance, in order to understand that the moon is made of green cheese, you have to know what “green cheese” is, what “the moon” is, what “is made of” means, and so on. So the first piece of information is a signal of some sort that indicates the informing communication, and the second is a dictionary that defines the interpretation of the signaled communication. It is the dictionary that allows the signaled information to be placed into context within a framework of existing knowledge.

Paul Revere, in his famous ride, exemplified all of the basic principles with the “One if by land, two if by sea” dictionary. Implicit in this is “None if not coming.” This number of lamps shown—0, 1, or 2 in the Old North Church tower in Boston, indicating the direction of British advance—formed the dictionary for the communication system. The actual signal consisted of 0 or 1 or 2 lamps showing in the tower window.

11.2.2 Measuring Information: Signals

A *signal* is a system state that indicates a defined communication. A system can have any number of signals. The English language has many thousands—each word carrying, or signaling, a unique meaning. Paul Revere came close to using the least possible signal. The least possible signal is a system state that is either present or not present. Any light in the Old North Church signaled that the British were coming—no light, no British coming. This minimal amount of signaled information can be indicated by any two-state arrangement: on and off, 1 and 0, up and down, present and absent. It is from this two-state system of signal information that we get the now ubiquitous *bi*nary digi*t*, or *bit* of information. Modern computer systems are all built from many millions of two-state switches, each of which can represent this minimal signal.

Back to the Old North Church tower. How many bits did Paul Revere’s signal need? Well, there are three defined system states: 0 lamps = no sign of the British, 1 lamp = British coming by land, 2 lamps = British coming by sea. One bit can carry only two system states. State 0 = (say) no British coming, state 1 = (say) land advance. (Note that there is no necessary connection between the number of lamps showing and the number of bits.) There is no more room in one bit to define more than two system states. So in addition to one bit signaling two states—no advance or land advance—at least one more bit is needed to indicate a sea advance. With two bits, up to four system states can be encoded, as shown in Table 11.1.

TABLE 11.1 Only three system states are needed to carry Paul Revere's message (using two bits leaves one state undefined).

Bit 1 state	Bit 2 state	Tower lights	Meaning
0	0	None	No sign of the British
0	1	1	Land advance
1	0	2	Sea advance
1	1	Undefined	Undefined

But two bits seems to be too many as this communication system has only three states. There is an undefined system state when, in this example, both bits are in their "1" state. It looks like 1 1/2 bits is enough to carry the message—and indeed it is. Fractional bits may seem odd. It may seem that fractional bits can't exist, which is true. But the measurement here is only of how many bits are needed to signal the information, and for that about 1 1/2 bits will do the job.

When Paul Revere constructed his signaling system, he directly faced the problem that, in practice, two bits are needed. When the signals were devised, Paul used one lighted lantern to indicate the state of one bit. He needed only 1 1/2 lights, but what does it mean to show 1/2 a light? His solution introduced a redundant system state, as shown in Table 11.2

TABLE 11.2 Paul Revere's signaling system used redundancy in having two states carry the same message.

Bit 1 state	Bit 2 state	Tower lights	Meaning
0	0	None	No sign of the British
0	1	1	Land advance

1	0	1	Land advance
1	1	2	Sea advance

With this signaling system, land advance is indicated by two separate system states. Each state could have been used to carry a separate message, but instead of having an undefined system state, an identical meaning was assigned to multiple system states. Since the entire information content of the communication system could be carried by about 1/2 bits, there is roughly 1/2 a bit of redundancy in this system.

Redundancy measures duplicate information in system states. Most information-carrying systems have redundancy—the English language is estimated to be approximately 50% redundant. Tht is why yu cn undrstnd ths sntnce, evn thgh mst f th vwls are mssng! It is also what allows data set compression—squeezing out some of the redundancy.

There are many measures that can be used to measure information content, but the use of bits has gained wide currency and is one of the most common. It is also convenient because one bit carries the least possible amount of information. So the information content of a data set is conveniently measured as the number of bits of information it carries. But given a data set, how can we discover how many bits of information it does carry?

11.2.3 Measuring Information: Bits of Information

When starting out to measure the information content of a data set, what can be easily discovered within a data set is its number of system states—not (at least directly) the number of bits needed to carry the information. As an understandable example, however, imagine two data sets. The first, set A, is a two-bit data set. It comprises two variables each of which can take values of 0 or 1. The second data set, set B, comprises one one-bit variable, which can take on values of 0 or 1. If these two data sets are merged to form a joint data set, the resulting data set must carry three bits of information.

To see that this is so, consider that set A has four possible system states, as shown in Table 11.3. Set B, on the other hand, has two possible system states, as shown in Table 11.4.

TABLE 11.3 Data set A, using two bits, has four discrete states.

Set A variable 1	Set A variable 2	System state

0	0	1
0	1	2
1	0	3
1	1	4

TABLE 11.4 Data set B, using one bit, has two discrete states.

Set B variable 1	**System state**
0	1
1	2

Clearly, combining the two data sets must result in an information-carrying measurement for the combined data set of three bits of information total. However, usually the information content in bits is unknown; only the numbers of system states in each data set is known. But adding the two data sets requires multiplying the number of possible system states in the combined data set, as Table 11.5 shows.

TABLE 11.5 Combining data sets A and B results in a composite data set with 2 x 4 = 8 states, not 2 + 4 = 6 states.

Set A variable 1	**Set A variable 2**	**Set B variable 1**	**Set A system state**	**Set B system state**	**Composite system state**

0	0	0	1	1	1
0	0	1	1	2	2
0	1	0	2	1	3
0	1	1	2	2	4
1	0	0	3	1	5
1	0	1	3	2	6
1	1	0	4	1	7
1	1	1	4	2	8

There are *not* 4 + 2 = 6 combined states, but 4 x 2 = 8 combined states. Adding data sets requires multiplying the number of system states. However, although the number of system states is multiplicative, the number of bits of information is additive—two bits in data set A and one bit in data set B gives 2 + 1 = 3 bits of information. This relationship allows a very convenient measure of information that is based on the properties of logarithms.

Adding the logarithms of numbers is an easy way to multiply the numbers themselves. The logarithm of a number simply consists of whatever power some base number has to be raised to so as to yield the required number. The relationship is

$\text{base}^{\text{logarithm}} = \text{number}$

So, for instance, since 62 = 36, and 63 = 216, the logarithm of 36, to the base of 6, is 2, and the logarithm of 216, to the base of 6, is 3. But

36 x 216 = 7776.

Since $6^2 = 36$, and $6^3 = 216$, then substituting gives

62 x 63 = 7776

but

$6^{2\,+\,3} = 6^5 = 7776.$

So

$\log_6(36) + \log_6(216) = 36 \times 216 = 7776$.

So it is easy to see that adding the logarithms of numbers is equivalent to multiplying the numbers themselves.

When measuring information content, the bit has already been seen to be a useful unit of measurement. The bit represents the minimum amount of information possible. But a bit has only two states. When using logarithms for determining information content, because a bit has only two states, the logarithm base used is 2. The "number" in the formula "$\text{base}^{\text{logarithm}}$ = number" is the number of system states, so that the logarithm of the number of system states, to the base 2, is a measure of the information content of the data set in bits. So

$\log_2(\text{system states})$ = information content in bits

Using logarithms to the base of 2 gives the information bit measure needed to carry a specified number of system states. Returning to the example of data sets A and B, for data set A with four system states that number is $\log_2(4) = 2$ bits. For data set B with two system states the information content is $\log_2(2) = 1$ bit. So for four system states, 2 bits of information are needed, and for two system states, 1 bit is needed, which is exactly as it appears in the example. When adding the two data sets, the combined data set has eight system states, which requires $\log_2(8) = 3$ bits.

(Note, however, that since $\log_2(3) = 1.58$, Paul Revere's system, while still needing fractional bits to carry the information, seems to need a little more than just 1 bits. More on this shortly.)

So far as this example goes, it is fine. But the example assumes that each of the system states counts equally, that is to say, each outcome is equally likely. Yet, unless a data set has a uniform distribution, the frequency with which each system state turns up is not uniform. To put it another way, unless equally distributed, not all system states are equally likely. Is this important?

11.2.4 Measuring Information: Surprise

As already discussed above, information content of a particular signal depends to some extent on how unexpected, or surprising, it is. In a fixed data set, the signals correspond to system states. Any specific data sample contains a fixed amount of information. However, the various signals, or system states, in the data set do not all carry an identical amount of information. The least surprising, or most common, signals carry less information than the most surprising, or least common, signals.

Surprise may seem like a subjective factor to measure. However, so far as signal information is concerned, surprise only measures the “unexpectedness” of a particular signal. The more likely a signal, the less surprising it is when it actually turns up. Contrarily, the less likely a signal, the more surprising it is. The degree of surprise, then, can be measured by how likely it is that a particular signal will occur. But in a data set, a signal is just a system state, and how likely it is that a particular system state occurs is measured by its joint probability, discussed in Chapter 10. Surprise can be quantified in terms of the relative frequency of a particular signal, or system state, in a representative sample. This is something that, if not always exactly easy to measure, can certainly be captured in principle. Surprisingness, then, turns out to be a measure of the probability of a particular system state in a representative sample. This, of course, is just its relative frequency in the data set.

As an example, suppose that you enter a lottery. It is a very small lottery and only 10 tickets are sold. As it happens, you bought 5 tickets. Clearly, in a fair drawing you stand a 50% chance of winning. After the drawing, you receive a signal (message) that you have won. Are you very surprised? How much information does the system contain, and how much is contained in the signal?

This system has two states as far as you are concerned, each 50% likely. Either you win or you lose. System information content: $\log_2(2) = 1$ bit of information. Each state, win or lose, has a probability of 1/2 = 0.5, or 50%. To discover how much information each signal carries, we must find how many bits of information the individual signal carries, weighted (multiplied) by the chance of that particular signal occurring. For this lottery the chance of a win is 0.5 (or 50%) and the unweighted win information is $\log_2(0.5)$, so the weighted win information is

chance of winning x unweighted win information in bits of winning x $\log_2$(chance of winning)

50% x $\log_2(0.5)$

50% x 1 = 0.5 bits of information

This system carries half a bit of information in the win signal, and by similar reasoning, half a bit in the lose signal. But suppose you had only bought one ticket? Now the win state only occurs (for you) with a 10% probability. Obviously you have a 90% probability of losing. If you win under these circumstances, you should be more surprised than before since it is much less likely that you will win. If a win signal does arrive, it should carry more information for you since it is unexpected—thus more surprising. Also, the more highly expected lose signal carries little information, since that is the one you most expect anyway. But what does this do to the total amount of information in the system?

The total amount of information in this system is lower than when the outcomes were equally likely—adding the weighted information (shown in Table 11.6 by P $\log_2(P)$) for win signal (0.332) and lose signal (0.137) gives an information content for the system of 0.469 bits. Yet the win signal, if it occurs, carries more than 0.3 bits of information—over 70% of the information available in the system!

TABLE 11.6 Information content in the lottery system when the win/lose signals aren't equal.

	Win	Lose
Probability (P)	0.1	0.9
$\log_2(P)$	3.32	0.152
P $\log_2(P)$	0.332	0.137

11.2.5 Measuring Information: Entropy

The information measure that has been developed actually measures a feature of the information content of a data set called *entropy*. Much confusion has been provoked by the use of this term, since it is identical to one used in physics to describe the capability of a system to perform work. Increased confusion is caused by the fact that the mathematical underpinnings of both types of entropy appear almost identical, and much discussion has ensued about similarities and differences between them. Data miners can leave this confusion aside and focus on the fact that *entropy measures uncertainty in a system*. When, for instance, the lottery had a 50/50 outcome, both the information measure and the uncertainty of the outcome were at their greatest for the system (1). It was impossible to say that any one outcome was more likely than any other—thus maximum uncertainty. This is a state of *maximum entropy*. As the system states move away from maximum uncertainty, so too does entropy decrease. Low entropy, as compared to the theoretical maximum for a system, suggests that the data set is possibly biased and that some system states may not be well represented. This measure of entropy is an important tool in general, and one that is particularly important in parts of the data survey.

11.2.6 Measuring Information: Dictionaries

Thus far in measuring the information content of a data set, looking at the data set alone seems sufficient. However, recall that communicated information comes in two parts—the signal and the dictionary. The dictionary places the signal in its context within a knowledge structure. It's a fine thing to know that you have won the lottery, but winning $1 is different from winning $1,000,000—even if the signal and its chance of occurring are identical. Sometimes one bit of information can carry an enormous amount of dictionary information. (When I say "go," play Beethoven's Fifth Symphony!) Creating knowledge structures is not yet within the realm of data mining (but see Chapter 12).

Measuring total information content requires evaluating the significance of the signal in context—and that is beyond the scope of the data survey! However, it is important to the miner since the data survey measures and uses the information content of the data set in many different ways. It still remains entirely up to the miner and domain experts to decide on significance, and that requires access to the system dictionaries. That may range (typically) from difficult to impossible. What this means for the miner is that while there are some objective measures that can be applied to the data set, and while valid measurements about the signal information content of the data set can be made, no blind reliance can be placed on these measures. As ever, they require care and interpretation.

To see the effect of changing even a simple dictionary, return to the example of the 10-ticket lottery. Previously we examined it from your point of view as a potential winner. We assumed that your only concern was for your personal win outcome. But suppose that you knew the other entrants and cared about the outcomes for them. This changes your dictionary, that is, the context in which you place the messages. Suppose that, as in the original case, you bought five of the 10 tickets. You still have a 50% chance of winning. But now five of your family members (say) each bought one of the other five tickets: Anne, Bill, Catherine, David, and Liz. Obviously they each have a 10% chance of winning. Does this change the information content of the system *as far as you are concerne*d?

There are now six outcomes of concern to you, as shown in Table 11.7. Well, the information in the "you win" signal hasn't changed, but the information content of the other five signals now totals 1.66. When previously you didn't care who else won, the signals totaled 0.5. As expected, when you care about the other signals, and place them in a more meaningful framework of reference, their information content changes—for you.

TABLE 11.7 Information content in the signals when outcome meaning changes.

Winner	**You**	**Anne**	**Bill**	**Catherine**	**David**	**Liz**
Probability (P)	0.5	0.1	0.1	0.1	0.1	0.1

$\log_2(P)$	1.0	3.32	3.32	3.32	3.32	3.32
$P \log_2(P)$	0.5	0.332	0.332	0.332	0.332	0.332

You might observe that in the previous example, there were in any case five other outcomes. It was just that the example lumped them all together for convenience. Suddenly, saying that the extra outcomes have meaning is “cheating.” What changed, however, is your knowledge of what the outcomes mean. It is this change in your state of knowledge of the signals’ meaning—in your dictionary—that is the crux of the matter. The knowledge you bring can materially affect the information content of the system. Outcomes and probabilities are unaffected, but information content may be dramatically affected.

As already pointed out, access to the dictionary may be impossible. So far as surveying data is concerned, it means that signal information enfolded into a data set can be assessed, but it may—probably will—be impossible to fully evaluate the information available. Fortunately, using entropic and other information measures on the signal information can be immensely useful. It is always important for the miner to keep in mind that, as powerful a tool as information analysis is, there is no point-and-shoot method of definitively capturing the information enfolded into a data set. The dictionary, or at least large parts of it, will almost always be unavailable and its content unknown.

However, what this example does show clearly is that knowledge of meaningful outcomes can materially change the information carried in a data set. It is well worth the miner’s time and effort to try to establish what constitutes meaningful outcomes for a particular project.

For instance, consider the by now almost classic example of Barbie dolls and chocolate bars. Allegedly, a major retailer discovered in a large data set that Barbie dolls and chocolate bars sell together. Assume, for the sake of the example, that this is indeed true. What can be done with the information? What, in other words, does it mean? The answer is, Not much! Consider the possibilities. Does the information indicate that Barbie dolls made out of chocolate would sell particularly well? Does it mean that Barbie dolls should be placed on the display shelf next to chocolate bars? Does it mean that a chocolate bar should be packaged with a Barbie doll? And what implications do these possibilities have for targeting the promotion? What, in short, can usefully be done with this signal? The signal carries essentially meaningless information. There is no way to act on the insight, to actually use the information communicated in context. The signal carries measurable information, but the dictionary translates it as not meaningful. The difference between system states and meaningful system states lies in the dictionary.

11.3 Mapping Using Entropy

Entropic measurements are the foundation for evaluating and comparing information content in various aspects of a data set. Recall that entropy measures levels of certainty and uncertainty. Every data set has some theoretical maximum entropy when it is in a state of maximum uncertainty. That is when all of the meaningful outcomes are equally likely. The survey uses this measure to examine several aspects of the data set.

As an example of what entropy measurements can tell the miner about a data set, the following discussion uses two data sets similar to those first described in Chapter 6 as an example. Chapter 6 discussed, in part, relationships that were, and those that were not, describable by a function. Two data sets were illustrated (Figures 6.1 and 6.2). The graphs in these two figures show similar manifolds except that, due to their orientation, one can be described by a function and the other cannot. Almost all real-world data sets are noisy, so the example data sets used here comprise noisy data approximating the two curves. Data set F (functional) contains the data for the curve that can be described by a function, and data set NF (nonfunctional) contains data for the curve not describable by a function.

While simplified for the example, both of these data sets contain all the elements of real-world data sets and illustrate what entropy can tell the miner. The example data is both simplified and small. In practice, the mined data sets are too complex or too large to image graphically, but the entropic measures yield the same useful information for large data sets as for small ones. Given a large data set, the miner must, in part, detect any areas in the data that have problems similar to those in data set NF—and, if possible, determine the scale and limits of the problem area. There are many modeling techniques for building robust models of nonfunctionally describable relationships—but only if the miner knows to use them! Very briefly, the essential problem presented in data set NF is that of the “one-to-many problem” in which one value (signal or system state) of the input variable (*x*) is associated validly with several, or many, output (*y*) values (signals or system states) that are not caused by noise.

Figure 11.1 illustrates data set F and shows the predicted values of a function fitted to the noisy data. The function fits the data quite well. Figure 11.2 shows data set NF. This figure shows three things. First, the squares plot the noisy data. Second, the continuous line shows the modeled function, which fits the data very badly. Third, the circles show the underlying *x*-*y* relationship.

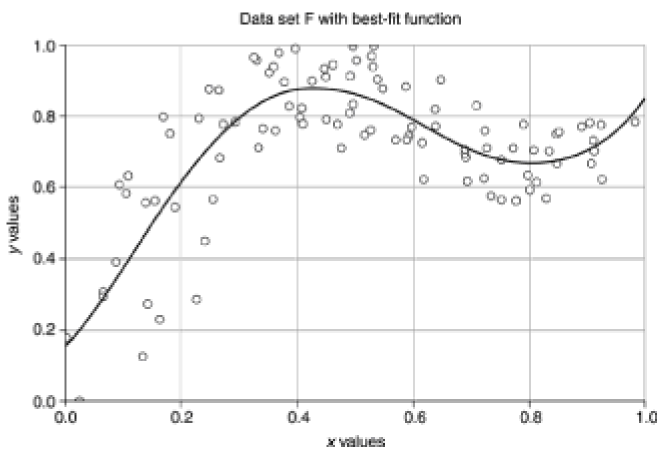


Figure 11.1 A noisy data set for which the underlying relationship can be described by a function. The *x* values are input and the *y* values are to be predicted. The continuous line shows a modeled estimate of the underlying function.

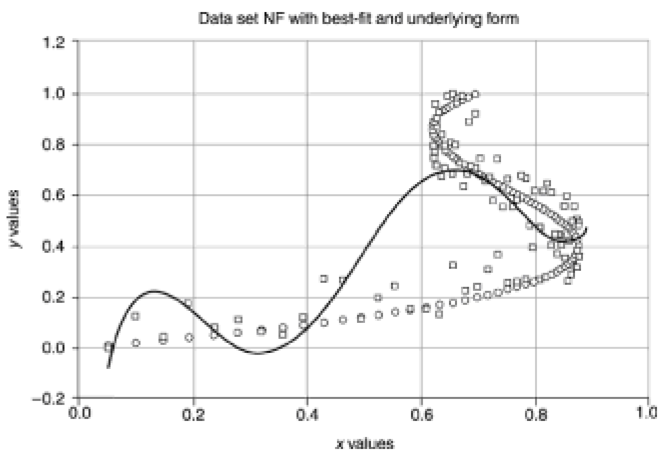


Figure 11.2 This illustrates data set NF. The squares plot the data set, and the continuous line shows the values of a modeled function to fit the data, which it does very poorly. The circles plot the actual underlying *x*-*y* relationship.

The following discussion uses six different information measures. Since the actual values vary from data set to data set, the values are normalized across the range 0–1 for convenience. The actual values for this data set, along with their normalized values, are shown in Table 11.8.

TABLE 11.8 Information measure values for the example data set NF.

Measure	Actual	Norm
Max entropy	3.459	
Entropy X	3.347	0.968
Entropy Y	3.044	0.880
Entropy (Y|X)	1.975	0.649
Entropy (X|Y)	2.278	0.681
Mutual info (X;Y)	1.069	0.351

The Supplemental Material section at the end of this chapter further explains the meaning and interpretation of these measures and their values. And because information measures are very important to the data survey, you'll also find information on deriving these measures from the data in the same section.

11.3.1 Whole Data Set Entropy

An ideal data set for modeling has an entropy that is close to the theoretical maximum. Since the modeling data set should represent all of its meaningful states, entropy will be high in such a data set. For modeling, the higher, the better. But a single data set for modeling usually comes in two pieces—the "input" and the "output." The input piece comprises all of the variables that, statistically, are described as the independent variables. These are the variables that contain the "evidence" or information that is to be mapped. The output variables contain the information to be predicted, or for which, or about which, inferences will be extracted from the input variables. Although usually collected as a single entity, both input and output data subsets should be independently examined for their entropy levels.

In the example, the input is a single variable, as is the output. Since both input and output are single variables, and have the same possible number of signals—that is, they cover the same range of 0–1—the maximum possible entropy for both variables *x* and *y* is identical. In this case, that maximum possible entropy for both variables is 3.459. This is the entropy measure if the variables' signals were completely evenly balanced.

The actual measured entropy for each variable is close to the maximum, which is a desirable characteristic for modeling. It also means that not much can be estimated about the values of either *x* or *y* before modeling begins.

11.3.2 Conditional Entropy between Inputs and Outputs

Conditional probabilities are in the form "What is the probability of B occurring, given that A has already happened?" For example, it is clear that the answer to "What is the chance of rain tomorrow?" may be very different from the answer to the question "What is the chance of rain tomorrow, given that it is pouring with rain today?" Having a knowledge of existing evidence allows a (usually) more accurate estimate of the conditional probability.

Conditional entropy is very similar. It results in a measure of mutual information. The amount of *mutual information* is given by the answer to the question "How much information is gained about the output by knowing the input?" In other words, given the best estimate of the output before any input is known, how much better is the estimated output value when a specific input *is* known? Ideally, one specific output signal would have a conditional probability of 1, given a particular input. All other outputs would have a probability of 0. That means that as a prediction, any input signal (value) implies one specific output signal (value) with complete certainty. Usually the probabilities are not so tightly focused and are spread across several outputs. Mutual information measures exactly how much information is available to determine a specific output value, given an input value.

Every different input/output value combination may have a different mutual information value. Taking the weighted sum value of all the individual mutual information values results in a general idea of how well the inputs (*x* values) predict the outputs (*y* values). The symbol used for this measure is "Entropy(Y|X)" which is read as "The entropy of the Y variable given the values of the X variable." In the example the Entropy(Y|X) is 1.975. This means that the entropy reduces from its starting state of 3.347 to 1.975, on average, when the *x* value is specified. The level of uncertainty is reduced to 1.975/3.347 = about 59% of what it was, or by about 41%.

This 41% reduction in uncertainty is very important. Contrast it with the reduction in uncertainty in the value of *x* when *y* is known. Entropy(X|Y)/ Entropy(Y) = 75%, a reduction of only 25% in the uncertainty of *x* given the *y* value. This says

knowing the value of *x*

when predicting the value of *y*

reduces the amount of uncertainty by 41%

and

knowing the value of *y*

when predicting the value of *x*

reduces the amount of uncertainty by 25%.

It is clear that *x* predicts *y* much better than *y* predicts *x*, so these values are clearly not symmetrical. This is exactly as expected given that in the example F data set, a single value of *x* is associated with a single value of *y* plus noise. However, the reverse is not true. A single value of *y* is associated with more than one value of *x*, plus noise. The entropy measure points out this area. It also measures the amount of noise remaining in the system that is not reduced by knowing an *x* value. This is a measure of ambient noise in the system that places theoretical limits on how good any model can be.

These measures are very important to any miner. They point to potential problem areas and indicate how well any model will at best perform—before any model is attempted. They measure limits set by the nature of the data.

As well as looking at the overall mutual information, it can be worth looking at the individual signal entropy plots for the input and output data. Figure 11.3 shows the entropy plots for the range of the input variable *x* predicting variable *y* in both data sets. The upper image shows the entropy for data set F, which is functionally describable. The plot shows the level of uncertainty in the *y* value, given the *x* value, for all given *x* values. This is important information for the miner. It shows that, in general, low values of *x* have more uncertainty than higher values, although overall the entropy is fairly low. Probably a reasonably good model can be constructed so long as the accuracy required of the model does not exceed the noise in the system. (It is quite possible to specify a required minimum model accuracy, and work back to the maximum allowable noise level that the required accuracy implies, although a description of how to do this is beyond the scope of this brief introduction to the data survey.)

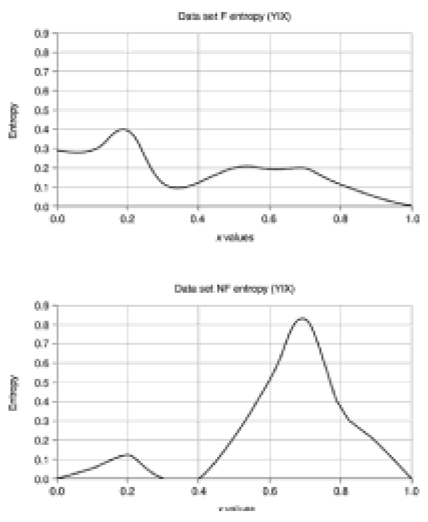


Figure 11.3 Entropy for the range of input values of *x* in the F data set (top) and in the NF data set (bottom).

A glance at the lower image tells a very different story. In the NF data set, values between about 0.6 and 0.8 are very uncertain, not well predicted at all. A look at the data shows that this is exactly the range of values where the *y* value has multiple values for a single *x* value. This may seem obvious in a small demonstration data set, but can be difficult or impossible to find in a real-world data set—unless found using mutual information.

11.3.3 Mutual Information

In an ideal data set, input and output signals all occur with equal probability—maximum entropy. In practice, such a state is very unlikely. Most of the signals will occur with some bias such that some are more likely to occur than others. Because of this bias, before modeling starts, the system is already known to have preferred states (carrying less information) and nonpreferred states (carrying more information). The purpose of any model is to use the input information to indicate the appropriate output with as high a degree of confidence as possible—or with a very low uncertainty. But the amount of uncertainty about an output depends entirely on how much information the input carries about the output. If this information can be measured, the miner can estimate the overall accuracy of any model prediction of that output—without actually building the model. In other words, the miner can estimate if any worthwhile model of specific outputs can, even theoretically, be built. Mutual information provides this estimate.

Mutual information is always balanced. The information contained in *x* and *y* is the same as that contained in *y* and *x*. For the data set F, the mutual information measure is 1.069, or has a ratio of about 0.35. The higher this number, the closer the ratio is to 1, the more information is available, so the better the model. In this case, there is a fair amount of noise in the data preventing an accurate model, as well as possible distortions that may be correctable. As it stands, the fairly low degree of information limits the maximum possible accuracy of a model based on this data set. Once again, discovering the actual limits is fairly straightforward, but beyond the scope of this introductory tour.

11.3.4 Other Survey Uses for Entropy and Information

In general, the miner greatly benefits from the power of entropic analysis when surveying data. The applications discussed so far reveal a tremendous amount about any prepared data set. Information theory can be used to look at data sets in several ways, all of which reveal useful information. Also, information theory can be used to examine data as instances, data as variables, the data set as a whole, and various parts of a data set. Entropy and mutual information are used to evaluate data in many ways. Some of these include

To evaluate the quality and problem areas of the input data set as a whole

- To evaluate the quality and problem areas of the output data set as a whole
- To evaluate the quality and problem areas of other data sets (e.g., test and verification) as a whole
- To evaluate the quality of individual variables over their range of values
- To estimate the independence of variables (measuring the entropy between inputs) both on average and in different parts of their range
- To select the input variables most independent from each other that carry maximum predictive or inferential information about the output(s)
- To estimate the maximum possible quality of a model over its range of inputs and outputs
- To identify problem areas, problem signals, and poorly defined areas of a model

11.3.5 Looking for Information

Making a comprehensive calculation of all possible variable and signal combinations is almost always impossible. The number of combinations is usually too high as the combinatorial explosion defeats a comprehensive search. In practice, data surveying searches some portion of the possibilities at a high level, looking for potential problems. This is a form of what is known as *attention processin*g—taking a high-level look at an area, and only looking more closely at any potentially interesting or difficult areas.

Attention processing simply describes a familiar method of searching for information or problems. Readers, hunting for information in, say, an encyclopedia, usually start with a high-level overview such as the index. At another level, they may skim articles for features of interest. The actual interesting features may well be scrutinized in detail. This is attention processing—more attention paid to areas in proportion to their degree of interest.

Detecting addresses on envelopes provides a practical computational example. Figure 11.4 illustrates the process. The first part of the problem is to determine a ZIP code. Rather than building a model to scrutinize the whole surface of an envelope, which would be very difficult, models with various degrees of attention are used. A top-level model determines if there is an address at all. (If the envelope is face down, for instance, there is no address.) When an address is likely to be present, another model determines if it is right side up. When right side up, the next model identifies a likely location for the ZIP code. The next model focuses only on the location of the ZIP code to extract the actual digits. Finally, the extracted digits are individually identified. Of course, then the ZIP code itself has to be identified, and so on. By using attention processing, a large task was divided into smaller tasks, each of which searched for a particular feature, allowing the next feature to receive more detailed attention.

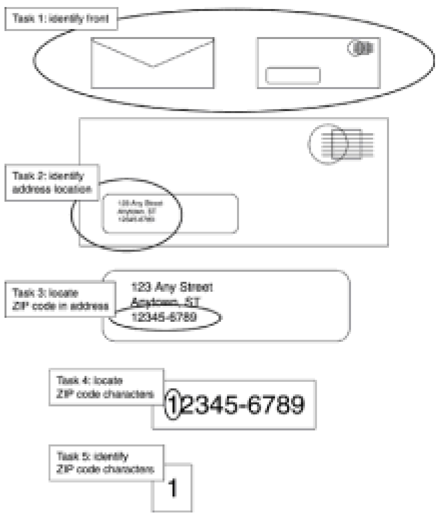

Task 1: identify front
Task 2: identify address location
Task 3: locate ZIP code in address
123 Any Street
Anytown, ST
12345-6789
Task 4: locate ZIP code characters
12345-6789
Task 5: identify ZIP code characters
1

Figure 11.4 Attention processing separates a task into parts and attends to separate parts of the task rather than trying to perform the whole task in one step.

So it is with the data survey. The survey starts with the general entropic calculations. This results in maps of particularly troublesome input/output signal interactions and establishes a level of expectation for the accuracy of the model. From here, the miner can explore either more broadly, or more deeply, into various problem areas, or areas that seem particularly promising.

The information analysis part of the survey provides a very good idea of the overall quality of the data set, and also identifies potential problem areas. However, while it identifies *where* the problems are, it says little about *what* they are.

11.4 Identifying Problems with a Data Survey

There is fundamentally one reason and three problems that can reduce or prevent mining tools from identifying a good relationship between input and output data sets. They are

Reason: The data set simply does not enfold sufficient information to define the relationship between input and output with the accuracy required.

- Problem 1: The relationship between input and output is very complex.

- Problem 2: Part(s) of the input/output relationship are not well defined by the available data.

- Problem 3: High variance or noise obscures the underlying relationship between input and output.

Turning first to the reason: The data set simply does not contain sufficient information to define the relationship to the accuracy required. This is not essentially a problem with the data sets, input and output. It may be a problem for the miner, but if sufficient data exists to form a multivariably representative sample, there is nothing that can be done to “fix” such data. If the data on hand simply does not define the relationship as needed, the only possible answer is to get other data that does. A miner always needs to keep clearly in mind that the solution to a problem lies in the problem domain, not in the data. In other words, a business may need more profit, more customers, less overhead, or some other business solution. The business does not need a better model, except as a means to an end. There is no reason to think that the answer has to be wrung from the data at hand. If the answer isn’t there, look elsewhere. The survey helps the miner produce the best possible model from the data that is on hand, and to know how good a model is possible from that data before modeling starts.

But perhaps there are problems with the data itself. Possible problems mainly stem from three sources: one, the relationship between input and output is very complex; two, data describing some part of the range of the relationship is sparse; three, variance is very high, leading to poor definition of the manifold. The information analytic part of the survey will point to parts of the multivariable manifold, to variables and/or subranges of variables where entropy (uncertainty) is high, but does not identify the exact problem in that area.

Remedying and alleviating the three basic problems has been thoroughly discussed throughout the previous chapters. For example, if sparsity of some particular system state is a problem, Chapter 10, in part, discusses ways of multiplying or enhancing particular features of a data set. But unless the miner knows that some particular area of the data set has a problem, and that the problem is sparsity, it is impossible to fix. So in addition to indicating overall information content and possible problem areas, the survey needs to suggest the nature of the problem, if possible.

The survey looks to identify problems within a specific framework of assumptions. It assumes that the miner has a multivariably representative sample of the population, to some acceptable level of confidence. It also assumes that in general the information content of the input data set is sufficient to adequately define the output. If this is not the case, get better data. The survey looks for local problem areas within a data set that overall meet the miners needs. The survey, as just described, measures the general information content of the data set, but it is specific, identified problems that the survey assesses for the possible causes. Nonetheless, in spite of these assumptions, the survey estimates the confidence level that the miner has sufficient data.

11.4.1 Confidence and Sufficient Data

A data set may be inadequate for mining purposes simply because it does not truly represent the population. If a data set doesn't represent the population from which it is drawn, no amount of other checking, surveying, and measuring will produce a valid model. Even if entropic analysis indicated that it is possible to produce a valid, robust model, that is still a mistake. Entropy measures what is present, and if what is present is not truly representative, the entropic measures cannot be relied upon either. The whole foundation of mining rests on an adequate data set. But what constitutes an adequate data set?

Chapter 5 addressed the issue of capturing a representative sample of a variable, while Chapter 10 extended the discussion to the multivariable distribution and capturing a multivariably representative sample. Of course, any data set can only be captured to some degree of confidence selected by the miner. But the miner may face the problem in two guises, both of which are addressed by the survey.

First, the miner may have a particular data set of a fixed size. The question then is, "Just how multivariably representative is this data set?" The answer determines the reliability of any model made, or inferences drawn, from the data set. Regardless of the entropic measurements, or how apparently robust the model built, if the sample data set has a very low confidence of being representative, so too must the model extracted, or inferences drawn, have a low confidence of being representative. The whole issue hinges on the fact that if the sample does not represent the population, nothing drawn from such a sample can be considered representative either.

The second situation arises when plenty of data is available, perhaps far more than can possibly be mined. The question then is, "How much data captures the multivariable variability of the population?" The data survey looks at any existing sample of data, estimates its probability of capturing the multivariable variability, and also estimates how much more data is required to capture some specified level of confidence. This seems straightforward enough. With plenty of data available, get a big enough sample to meet some degree of confidence, whatever that turns out to be, and build models. But, strange as it may seem, and for all the insistence that a representative sample is completely essential, a full multivariable representative sample may not be needed!

It is not that the sample need not be representative, but that perhaps all of the variables may not be needed. Adding variables to a data set may enormously expand the number of instances needed to capture the multivariable variability. This is particularly true if the added variable is not correlated with existing variables. It is absolutely true that to capture a representative sample with the additional variable, the miner needs the very large number of instances. But what if the additional variable is not correlated (contains little information about) the predictions or relationships of interest? If the variable carries little information of use or interest, then the size of the sample to be mined was expanded for little or no useful gain in information. So here is another very good reason for removing variables that are not of value.

Chapter 10 described a variable reduction method that is implemented in the demonstration software. It works and is reasonably fast, particularly when the miner has not specifically segregated the input and output data sets. Information theory allows a different approach to removing variables. It requires identifying the input and output data sets, but that is needed to complete the survey anyway. The miner selects the single input variable that carries most of the information about the output data set. Then the miner selects the variable carrying the next most information about the output, such that it also carries the least information in common (mutual information content) with the previously selected variable(s). This selection continues until the information content of the derived input data set sufficiently defines the model with the needed confidence. Automating this selection is possible. Whatever variable is chosen first, or whichever variables have already been chosen, can enormously affect the order in which the following variables are chosen. Variable order can be very sensitive to initial choice, and any domain knowledge contributed by the miner (or domain expert) should be used where possible.

If the miner adopts such a data reduction system, it is important to choose carefully the variables intended for removal. It may be that a particular variable carries, in general, little information about the output signals, but for some particular subrange it might be critically important. The data survey maps all of the individual variables' entropy, and these entropy maps need to be considered before making any final discard decision.

However, note that this data reduction activity is not properly part of the data survey. The survey only looks at and measures the data set presented. While it provides information about the data set, it does not manipulate the data in any way, exactly as a map makes no changes to the territory, but simply represents the relationship of the features surveyed for the map. When looking at multivariate distribution, the survey presents only two pieces of information: the estimated confidence that the multivariable variability is captured, and, if required, an estimate of how many instances are needed to capture some other selected level of confidence. The miner may thus learn, say, that the input data set captured the multivariable variability of the population with a 95% confidence level, and that an estimated 100,000 more records are needed to capture the multivariable variability to a 98% confidence level.

11.4.2 Detecting Sparsity

Overall, of course, the data points in state space (Chapter 6) vary in density from place to place. This is not necessarily any problem in itself. Indeed, it is a positive necessity as this variation in density carries much of the information in the data set! A problem only arises if the sparsity of data points in some local area falls to such a level that it no longer carries sufficient information to define the relationship to the required degree. Since each area of state space represents a particular system state, this means only that some system states are insufficiently represented.

This is the same problem discussed in several places in this book. For instance, the last chapter described a direct-mail effort's very low response rate, which meant that a naturally representative sample had relatively very few samples of responders. The number of responses had to be artificially augmented—thus populating that particular area of state space more fully.

However, possibly there is a different problem here too. Entropy measures, in part, how well some particular input state (signal or value) defines another particular output state. If the number of states is low, entropy too may be low, since the number of states to choose from is small and there is little uncertainty about which state to choose. But the number of states to choose from may be low simply because the sample populates state space sparsely in that area. So low entropy in a sparsely populated part of the output data set may be a warning sign in itself! This may well be indicated by the forward and reverse entropy measures (Entropy(X|Y) and Entropy(Y|X)), which, you will recall, are not necessarily the same. When different in the forward and reverse directions, it may indicate the "one-to-many problem," which could be caused by a sparsely populated area in one data set pointing to a more densely populated area in the other data set.

The survey makes a comprehensive map of state space density—both of the input data set and the output data set. This map presents generally useful information to the miner, some of which is covered later in this chapter in the discussion of clustering. Comparing density and entropy in problematic parts of state space points to possible problems if the map shows that the areas are sparse relative to the mean density.

11.4.3 Manifold Definition

Imagine the manifold as a state space representation of the underlying structure of the data, less the noise. Remember that this is an imaginary construct since, among other ideas, it supposes that there is some "underlying mechanism" responsible for producing the structure. This is a sort of causal explanation that may or may not hold up in the real world. For the purposes of the data survey, the manifold represents the configuration of estimated values that a good model would produce. In other words, the best model should fill state space with its estimated values exactly on the manifold. What is left over—the difference between the manifold and the actual data points—is referred to as *error* or *noise*. But the character of this noise can vary from place to place on the manifold, and may even leave the "correct" position of the manifold in doubt. (And go back to the discussion in Chapter 2 about how the states map to the world to realize that any idea of a "correct" position of a manifold is almost certainly a convenient fiction.) All of these factors add up to some level of uncertainty in the prediction from place to place across the manifold, and it is this uncertainty that, in part, entropy measures. However, while measuring uncertainty, entropy does not actually characterize the exact nature of the uncertainty, for which there are several possible causes. This section considers problems with variance. Although this is a very large topic, and a comprehensive discussion is far beyond the scope of this section, a brief introduction to some of the main issues is very helpful in understanding limits to a model's applicability.

Much has been written elsewhere about analyzing variability. Recall that the purpose of the data survey is not to analyze problems. The data survey only points to possible problem areas, delivered by an automated sweep of the data set that quickly delivers clues to possible problems for a miner to investigate and analyze more fully if needed. In this vein, the manifold survey is intended to be quick rather than thorough, providing clues to where the miner might usefully focus attention.

Skewness

Variance was previously considered in producing the distribution of variables (Chapter 5) or in the multivariable distribution of the data set as a whole (Chapter 10). In this case, the data survey examines the variance of the data points in state space as they surround the manifold. In a totally noise-free state space, the data points are all located exactly on (or in) the manifold. Such perfect correspondence is almost unheard of in practice, and the data points hover around the manifold like a swarm of bees. All of the points in state space affect the shape of every part of the manifold, but the effect of any particular data point diminishes with distance. This is analogous to the gravity of Pluto—a remote and small body in the solar system—that does have an effect on the Earth, but as it is so far away, it is almost unnoticeable. The Moon, on the other hand, although not a particularly massive body as solar system bodies go, is so close that it has an enormous effect (on the tides, for instance).

Figure 11.5 shows a very simplified state space with 10 data points. The data points form two columns, and the straight line represents a manifold to fit these points. Although the two columns cover the same range of values, it's easy to see that the left column's values cluster around the lower values, while the right column has its values clustered around the higher values. The manifold fits the data in a way that is sensitive to the clustering, as is entirely to be expected. But the nature of the clustering has a different pattern in different parts of the state space. Knowing that this pattern exists, and that it varies, can be of great interest to a miner, particularly where entropy indicates possible problems. It is often the case that by knowing patterns exist, the miner can use them, since pattern implies some sort of order.

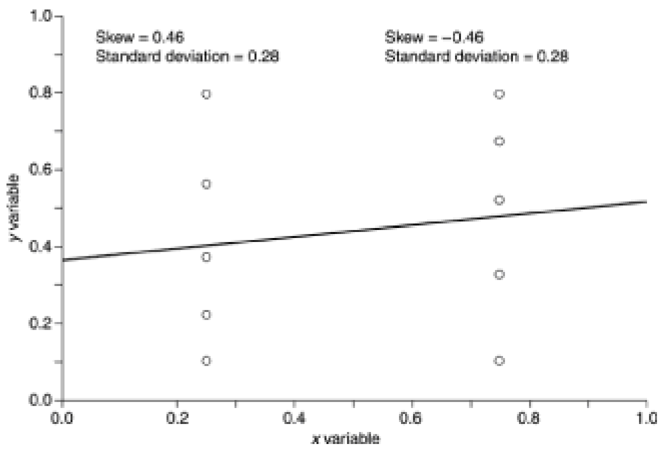


Figure 11.5 A simplified state space with 10 data points.

The survey looks at the local data affecting the position of the manifold and maps the data distribution around the manifold. The survey reports the standard deviation (see Chapter 5 for a description of this concept) and skew of the data points around the manifold. *Skewness* measures exactly what the term seems to imply—the degree of asymmetry, or lopsidedness, of a distribution about its mean. In this example the number is the same, but the sign is different. Zero skewness indicates an evenly balanced distribution. Positive skew indicates that the distribution is lighter in its values on the positive side of the mean. Negative skew indicates that the distribution is lighter in the more negative values of its range. Although not shown in the figure, the survey also measures how close the distribution is to being multivariably normal.

Why choose these measures? Recall that although the individual variables have been redistributed, the multivariable data points have not. The data set can suffer from outliers, clusters, and so on. All of the problems already mentioned for individual variable distributions are possible in multivariable data distributions too. Multivariable redistribution is not possible since doing so removes all of the information embedded in the data. (If the data is completely homogenous, there is no density variation—no way to decide how to fit a manifold—since regardless of how the manifold is fitted to the data, the uniform density of state space would make any one place and orientation as good as any other.) These particular measures give a good clue to the fact that, in some particular area, the data has an odd pattern.

Manifold Thickness

So far, the description of the manifold has not addressed any implications of its thickness. In two or three dimensions, the manifold is an imaginary line or a sheet, neither of which have any thickness. Indeed, for any particular data set there is always some specific best way to fit a manifold to that data. There are various ways of defining how to make the manifold fit the data—or, in other words, what actually constitutes a best fit. But it always results in some particular way of fitting the manifold to the data.

However, in spite of the fact that there is always a best fit, that does not mean that the manifold always represents the data over all parts of state space equally well. A glance at Figure 11.6 shows the problem. The manifold itself is not actually shown in this illustration, but the mean value of the *x* variable across the whole range of the *y* variable is 0.5. This is where the manifold would be fitted to this data by many best-fit metrics. What the illustration does show are the data points and envelopes estimating the maximum and minimum values across the *y* dimension. It is clear that where the envelope is widely spaced, the values of *x* are much less certain than where the envelope is narrower. The variability of *x* changes across the range of *y*. Assuming that this distribution represents the population, uncertainty here is not caused by a lack of data, but by an increase in variability. It is true that in this illustration density has fallen in the balloon part of the envelope. However, even if more data were added over the appropriate range of *y*, variability of *x* would still be high, so this is not a problem of lack of data in terms of *x* and *y*.

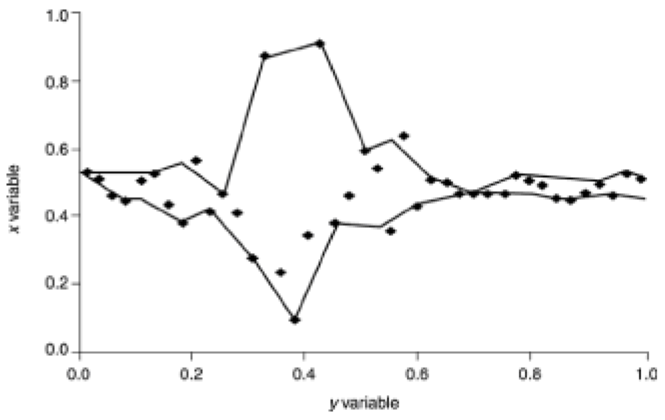


Figure 11.6 State space with a nonuniform variance. This envelope represents uncertainty due to local variance changes across the manifold.

Of course, adding data in the form of another variable might help the situation, but in terms of *x* and *y* the manifold's position is hard to determine. This increase in the variability leaves the exact position of the manifold in the "balloon" area uncertain and ill defined. More data still leaves predicting values in this area uncertain as the uncertainty is inherent in the data—not caused by, say, lack of data. Figure 11.7 illustrates the variability of *x* across *y*.

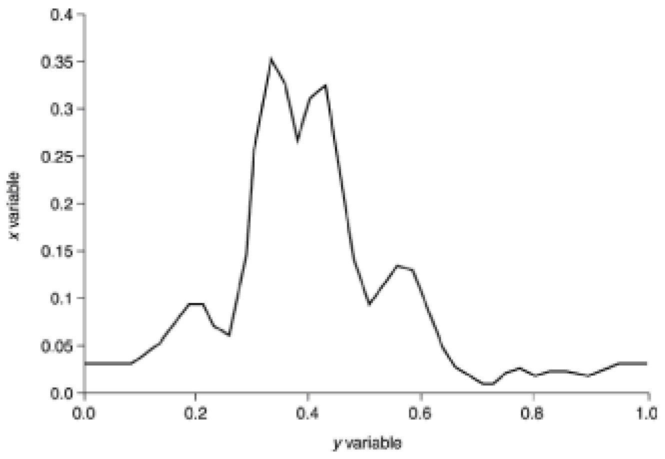


Figure 11.7 The variability in *x* is shown across the range of the variable *y*. Where variability is high, the manifold's position and shape are less certain.

The caveat with these illustrations is that in multidimensional state space, the situation is much more complex indeed than can be illustrated in two dimensions. It may be, and in practice it usually is, that some restricted part of state space has particular problems. In any case, recall that the individual variable values have been carefully redistributed and normalized, so that state space is filled in a very different way than illustrated in these examples. It is this difficulty in visualizing problem areas that, in part, makes the data survey so useful. A computer has no difficulty in making the multidimensional survey and pointing to problem areas. The computer can easily, if sometimes seemingly slowly, perform the enormous number of calculations required to identify which variables, and over which parts of their ranges, potential problems lurk. "Eyeballing" the data would be more effective at detecting the problems—if it were possible to look at all of the possible combinations. Humans are the most formidable pattern detectors known. However, for just one large data set, eyeballing all of the combinations might take longer than a long lifetime. It's certainly quicker, if not as thorough, to let the computer crunch the numbers to make the survey.

Very Complex Relationships

Relationships between input and output can be complex in a number of different ways. Recall that the relationship described here is represented by a manifold. The required values that the model will ideally predict fall exactly on the manifold. This means that describing the shape of the manifold necessarily has implications for a predictive model that has to re-create the shape of the manifold later. So, for the sake of discussion, it is easy to consider the problem as being with the shape of the manifold. This is simpler for descriptive purposes than looking at the underlying model. In fact, the problem is for the model to capture the shape of the manifold.

Where the manifold has sharp creases, or where it changes direction abruptly, many modeling tools have great difficulty in accurately following the change in contour. There are a number of reasons for this, but essentially, abrupt change is difficult to follow. This phenomenon is encountered even in everyday life—when things are changing rapidly, and going off in a different direction, it is hard to follow along, let alone predict what is going to happen next! Modeling tools suffer from exactly this problem too.

The problem is easy to show—dealing with it is somewhat harder! Figure 11.8 shows a manifold that is noise free and well defined, together with one modeling tool's estimate of the manifold shape. It is easy to see that the "point" at the top of the manifold is not well modeled at all. The modeled function simply smoothes the point into a rounded hump. As it happens, the "sides" of the manifold are slightly concave too—that is, they are curves bending in toward the center. Because of this concavity, which is in the opposite direction to the flexure of the point, the modeled manifold misses the actual manifold too. Learning this function requires a more complex model than might be first imagined.

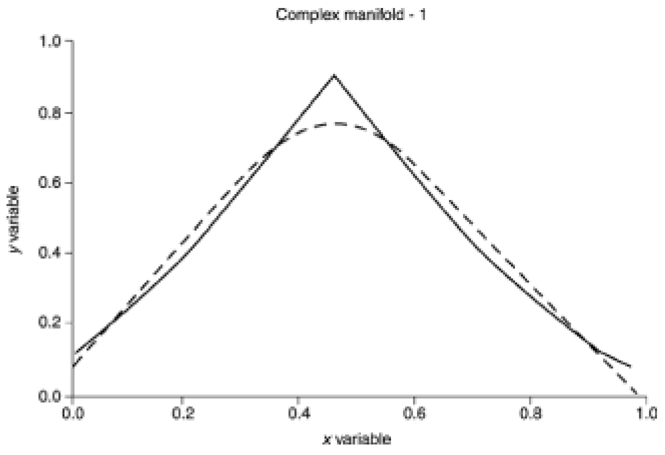


Figure 11.8 The solid arch defines the data points of the actual manifold and the dotted line represents one model's best attempt to represent the actual manifold.

However, the relative complexity of the manifold in Figure 11.9 is far higher. This manifold has two "points" and a sudden transition in the middle of an otherwise fairly sedate curve. The modeled estimate does a very poor job indeed. It is the "points" and sudden transitions that make for complexity. If the discontinuity is important to the model, and it is likely to be, this mining technique needs considerable augmentation to better capture the actual shape of the relationship.

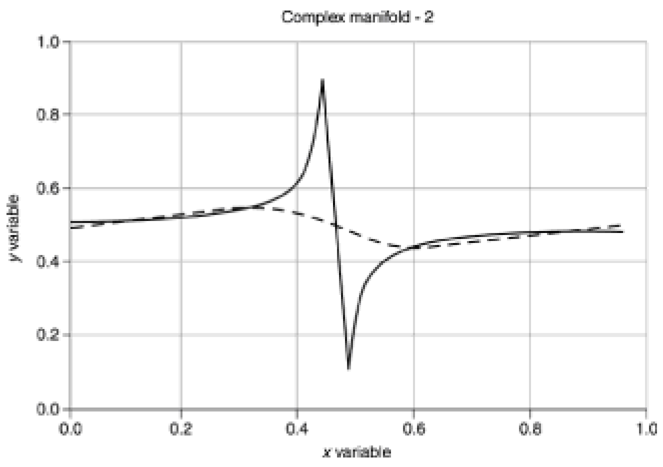


Figure 11.9 This manifold is fairly smooth except around the middle. The model (dotted line) entirely misses the sharp discontinuity in the center of the manifold—even though the manifold is completely noise-free and well-defined.

Curves such as this are more common than first glance might suggest. The curve in Figure 11.9, for instance, could represent the value of a box of seats during baseball season. For much of the season, the value of the box increases as the team keeps winning. Immediately before the World Series, the value rises sharply indeed since this is the most desirable time to have a seat. The value peaks at the beginning of the last game of the series. It then drops precipitously until, when the game is over, the value is low—but starts to rise again at the start of a new season. There are many such similar phenomena in many areas. But accurately modeling such transitions is difficult.

There is plenty of information in these examples, and the manifolds for the examples are perfectly defined, yet still a modeling tool struggles. So complexity of the manifold presents the miner with a problem. What can the survey do about detecting this?

In truth, the answer is that the survey does little. The survey is designed to make a "quick once over" pass of the data set looking, in this case, for obvious problem areas. Fitting a function to a data set—that is, estimating the shape of the manifold—is the province of modeling, not surveying. Determining the shape of the manifold and measuring its complexity are computationally intensive, and no survey technique can do this short of building different models.

However, all is not completely lost. The output from a model is itself a data set, and it should estimate the shape of the manifold. Most modeling techniques indicate some measure of "goodness of fit" of the manifold to the data, but this is a general, overall measure. It is well worth the miner's time to exercise the model over a representative range of inputs, thus deriving a data set that should describe the manifold. Surveying this *derived* (or predicted) data set will produce a survey map that looks at the predicted manifold shape and points to potential problem areas. Such a survey reveals exactly how much information was captured across the surface of the manifold. Where particularly problematic areas show up, building smaller models of the restricted, troublesome area very often produces better results in the restricted area than the general model. As a result, some models are used in some areas, while other models are used on other parts of the input space. But this is a modeling technique, rather than a surveying technique. Nonetheless, a sort of “post-survey survey” can point to problem areas with any model.

11.5 Clusters

Earlier, this chapter used the term “meaningful system states.” What exactly is a meaningful system state? The answer varies, and the question can only be answered within the framework of the problem domain. It might be that some sort of binning (described in Chapter 10) assigns continuous measurements to more meaningful labels. At other times, the measurements are meaningfully continuous, limited only by the granularity of the measurement (to the nearest penny, say, or the nearest degree). However, the system may inherently contain some system states that appear, from wholly internal evidence, to be meaningful within the system of variables. (This does not imply that they are necessarily meaningful in the real world.) The system “prefers” such internally meaningful states.

Recall that at this stage the data set is assumed to represent the population. Chapter 6 discussed the possibility that apparently preferred system states result from sampling bias preferentially sampling some system states over others. The miner needs to take care to eliminate such bias wherever possible. Those preferred system states that remain should tell something about the “natural” state of the system. But how does the miner find and identify any such states?

Chapter 6 discussed the idea that density of data points across state space varies. If areas that are more dense than average are imagined as points lower than average, and less dense points imagined to be higher, the density manifold can be conceived of as peaks and valleys. Each peak (the locally highest point) is surrounded by lower points. Each valley is surrounded by peaks and ridges. The ridges surrounding a particular valley actually are defined by a contour running through the lowest density surrounding a higher-density cluster. The valley bottoms actually describe the middle of higher-than-the-mean density clusters. These clusters represent the preferred states of the system of variables describing state space.

Such clusters, of course, represent likely system states. The survey identifies the borders and centers of these clusters, together with their probability. But more than that, it is often useful to aggregate these clusters as meaningful system states. The survey also makes an entropy map from all of the input clusters to all of the identified output clusters. This discovers if knowing which cluster an input falls into helps define an output.

For many states this is very useful information. Many models, both physical and behavioral, can make great use of such state models, even when precise models are not available. For instance, it may be enough to know for expensive and complex process machinery that it is “ok” or “needs maintenance” or is “about to fail.” If the output states fall naturally into one of these categories and the input states map well to the output states, a useful model may result even when precise predictions are not available from the model. Knowing what works allows the miner to concentrate on the borderline areas. Again, from behavioral data, it may be enough to map input and output states reliably to such categories as “unhappy customer warning,” “likely to churn,” and “candidate for cross-sell product X.”

Clustering is also useful when the miner is trying to decide if the data is biased.

11.6 Sampling Bias

Sampling bias is a major bugaboo and very hard to detect, but it’s easy to describe. When a sampling method repeatedly takes samples of data from a population that differ from the true population measures in the same way and in the same direction, then that method is introducing *sampling bias*. It is a distortion of the true values in the sample from those in the population that is introduced by the selection method itself, independent of other factors biasing the data. It is difficult to avoid since it may be quite unconsciously introduced. Since miners often work with data collected for purposes uncertain, by methods unknown, and with measurements obscure, after the fact detection of sampling bias may be all but impossible. Yet if the data does not reflect the real world, neither will any model mined, regardless of how assiduously it is checked against test and evaluation sample data sets.

The best that can be had from internal evaluation of a data set are clues that perhaps the data is biased. The only real answer lies in comparing the data with the world! However, that said, what can be done? There are two main types of sampling bias: errors of omission and errors of commission.

Errors of omission, of course, involve leaving out data that should be put in, whereas errors of commission involve putting in what should be left out. For instance, many interest groups seem to be able to prove a point completely at odds with the point proved by interest groups opposing them. Both sets of conclusions are solidly based on the data collected by each group, but, unconsciously or not, if the data is carefully selected to support desired conclusions, it can only tell a partial story. This may or may not be deliberately introduced bias. If an honest attempt to collect all the relevant data was made, but it still leads to dispute, it may be the result of sampling bias, either omission or commission. In spite of all the heat and argument, the only real answer is to collect all relevant data and look hard for possible bias.

As an example of the problem, an automobile manufacturer wanted to model vehicle reliability. A lot of data was available from the dealer network service records. But here was a huge problem. Quite aside from trying to decide what constitutes “reliability,” the data was very troublesome. For instance, those people who regularly used the dealer for service tended to be those people who took care of their vehicle and thus had reliable vehicles. On the other hand, repair work was often done for people who had no maintenance record with the dealer network. Conclusion: maintenance enhances reliability? Perhaps. But surely some people had maintenance outside of the dealer network. Some, perhaps, undertook their own maintenance and minor repairs, only having major work done at a dealer. There are any number of other possible biasing factors. Regardless of the possibilities, this was a very selective sample, almost certainly not representative of the population. So biased was this data that it was hard to build models of reliability even for those people who visited dealers, let alone the population at large!

Detecting such bias from the internal structure of the data is not possible. Any data set is what it is, and whether or not it accurately reflects the worldly phenomenon, it can never for sure be known just by looking at the data. But there might be clues.

The input data set covers a particular area. (A reminder that the term “area” is really applicable to a two-dimensional state space only, but it is convenient to use this term in general for the *n*-dimensional analog of area in other than two dimensions.) The output data set similarly covers its area. Any space in the input area maps, or points to, some particular space in the output area. This is illustrated in Figure 11.10. Exactly which part of the input space points to which part of the output space is defined by the relationship between them. The relevant spaces may be patches of different sizes and shapes from place to place, but the input points to some part of the output space, therefore being identified with some particular subsample, or patch, of the output sample.

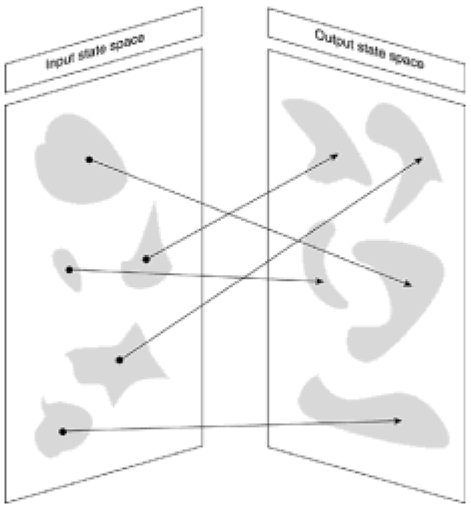


Figure 11.10 At least two data sets are used when modeling: an input data set (left) and an output data set (right).

It is often found in practice that for unbiased data sets, while the values of specific output variables change as the values of an input variable change, the distribution of data points at the different output values is fairly constant. For example, suppose, as illustrated in Figure 11.11, that the output patch of data points is normally distributed for some specific input value. As the input value changes, the output values will be expected to change (the patch moves through the output space), and the number of points in the output patch too is expected to change. However, if this assumption holds, wherever the output patch is located, the distribution of the points in it is expected to remain normally distributed.

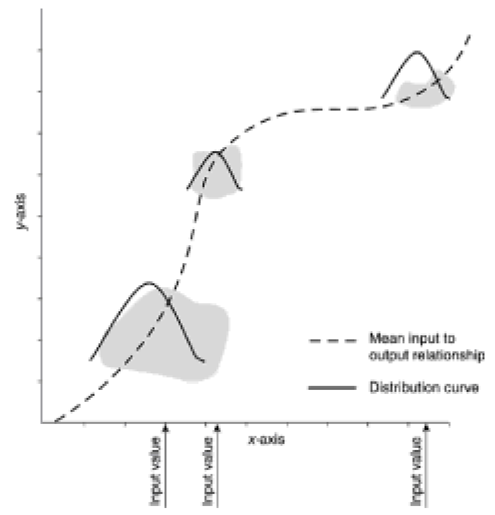


Figure 11.11 The output state space (made up of the *x* and *y* variables) has a manifold representing the input to output relationship. Any specific input value maps to some area in the output space, forming a "patch" (gray areas).

Figure 11.12 illustrates the idea that the distribution doesn't change as the value of a variable changes. The effect of changing an input variable's value is expected to change the output value and the number of instances in the subsample, but other factors are expected to remain the same so the shape of the distribution isn't changed. Given that this often is true, what does it suggest when it is not true? Figure 11.13 illustrates a change in distribution as the *x* value changes. This distribution shift indicates that something other than just the *y* value has changed about the way the data responds to a change in the *x* value. Some other factor has certainly affected the way the data behaves at the two *x* values, and it is something external to the system of variables. This change may be caused by sampling bias or some other bias, but whatever the cause, the miner should account for the otherwise unaccounted change in system behavior between the two *x* values.

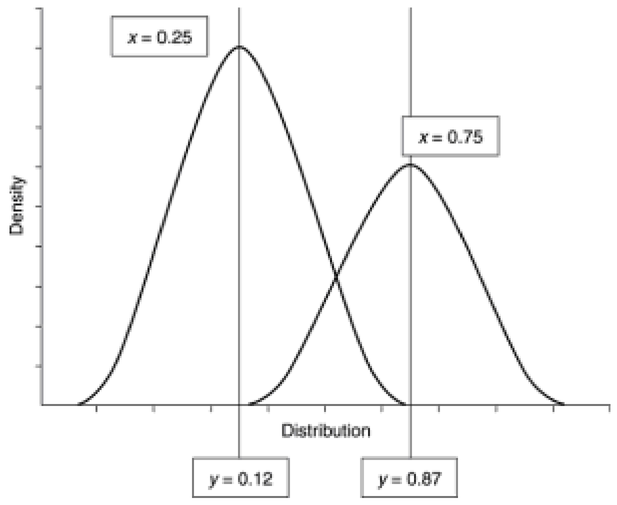


Figure 11.12 Distribution curves for the *x* values change the *y* values, but the curve remains similar in shape and not in size.

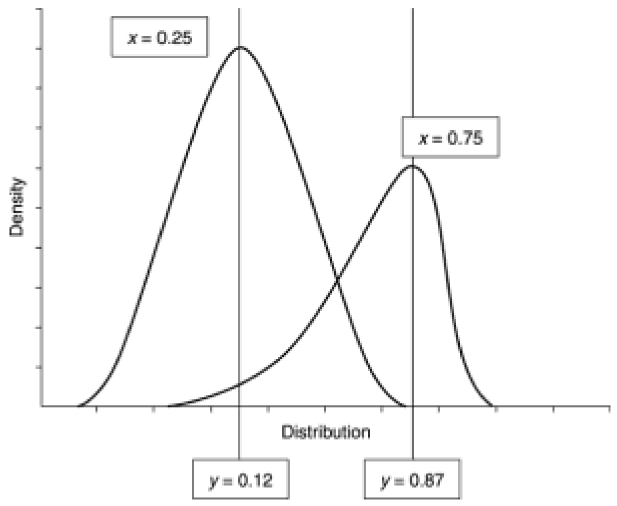


Figure 11.13 The change in *x* values is accompanied by a change in distribution shape as well as size. The tail is longer toward the low values for an *x* value of 0.75 than it is for an *x* value of 0.25.

The data survey samples the distributions, moving the input variables across their ranges of values. It makes a measurement of how much the distribution of output variables changes as inputs are moved, which is based on changes in both variability and skew.

11.7 Making the Data Survey

The components so far discussed form the basic backbone of the data survey. The survey's purpose is a quick look at the data. While modeling is a time-consuming process and focuses on detail, the survey deliberately focuses on the broad picture. The idea is not to resolve problems discovered, but to discover in broad terms the use and limits to use of the data and to be forewarned of any possible problem areas before modeling. The survey will not, and is not intended to, discover all of the problems that may lurk in data, and does little, if anything, toward fixing them. That is for the miner. But the miner can only fix known problems, and knowing comes from looking at the survey results. Survey software automatically makes these measurements so the miner can focus on problem areas by producing a unified numerical and graphical output. Even without such a tool, many statistical and data analysis tools have functions that allow the data to be surveyed manually. In spite of the brevity of the look at surveying in this chapter, time spent—manually if necessary—surveying data will repay the miner dividends.

Briefly, the steps for surveying data are

1. Sampling confidence—estimate level of multivariable distribution capture. (This confidence puts all of the other measures into perspective, whatever they are. If you're only 50% confident that the variability has been captured, it hardly matters what the other survey measurements are!)

2. Entropic analysis (normalized ranges)

    a. Input data set entropy—should be as high as possible

    b. Output data set entropy—should be as high as possible

    c. Other data set entropy—should be as high as possible, and similar among all of the data sets. (They should all be representative of the same population. Differences in these metrics mean something is fishy with the data!)

    d. Conditional entropy of outputs given inputs—should be as low as possible. If it is high, there is a problem! Either the data is snarled in some way, or the input simply doesn't contain sufficient information to predict the output. (Then try conditional entropy of outputs to inputs for comparison. If that's low, suspect a problem, not lack of information content. If it's high also, suspect insufficient information content.)

    e. Mutual information inputs to outputs—should be as high as possible

    f. Individual variable entropy input to output

    g. Individual between-variable entropy of the input (measures independence—may be useful too for data reduction)

3. Cluster analysis

    a. Plot peak, valley, and contour positions ("natural" clusters—do these make sense?

Why are they where they are? Could this be bias?)

   b. Entropy of input clusters to output clusters—should be low. If not, there is a problem.

   c. Cluster overlays—do input clusters map to output clusters, or do input clusters map across output clusters? (Overlaying each other is generally best, with small overlap.)

4. Manifold (maps only potential problem areas)

   a. Sparsity—do sparse areas map to important output states? (If they do, it's a problem.)

   b. Variability map (High variability will match areas of high uncertainty, but additional information given in distribution measures may help identify a problem.)

5. Sampling bias

   a. Individual variable distribution input-driven output mapping—flag areas of high distribution drift. If there are many, is it sampling bias? In any case, why?

11.8 Novelty Detection

A novelty detector is not strictly part of the data survey, but is easily built from some of the various components that are discovered during the survey. The novelty detector is mainly used during the execution stage of using a model. Novelty detectors address the problem of ensuring that the execution data continues to resemble the training and test data sets.

Given a data set, it is moderately easy to survey it, or even to simply take basic statistics about the individual variables and the joint distribution, and from them to determine if the two data sets are drawn from the same population (to some chosen degree of confidence, of course). A more difficult problem arises when as assembled data set is not available, but the data to be modeled is presented on an instance-by-instance basis. Of course, each of the instances can be assembled into a data set, and that data set examined for similarity to the training data set, but that only tells you that the data set now assembled was or wasn't drawn from the same population. To use such a method requires waiting until sufficient instances become available to form a representative sample. It doesn't tell you if the instances arriving *now* are from the training population or not. And knowing if the current instance is drawn from the same population can be very important indeed, for reasons discussed in several places. In any case, if the distribution is not stationary (see Chapter 9 for a brief discussion of stationarity), no representative sample of instances is going to be assembled! So with a nonstationary distribution, collecting instances to form a representative sample to measure against the training sample presents problems.

Novelty detectors can also be used with enormously large data sets to help extract a more representative sample than the techniques alluded to in Chapter 10. Many large data sets contain very low-level fluctuations that are nonetheless important in a commercial sense, although insignificant in a statistical sense. The credit card issuer example in Chapter 12 demonstrates just such a case. When a representative sample is taken to some degree of confidence, it is the low-level fluctuations that are the most likely to be underrepresented in the sample. It is these low-level fluctuations that fall below the confidence threshold most readily. Finding some way to include these low-level fluctuations without bloating the sample can have great business value. It is in this role that a novelty detector can also contribute much.

So what exactly is a novelty detector? While there is no room here to go into the details of how a novelty detector is constructed, the principle is easy to see. Essentially a novelty detector is a device that estimates the probability that any particular instance value comes from the training population. The data survey has estimated the multidimensional distribution, and from that it is possible to estimate how likely any given instance value is to be drawn from that population. For a single variable, if normally distributed, such an estimate is provided by the standard deviation, or *z* value. So here, a novelty detector can be seen for what it really is—no more than the nonnormal-distribution, multidimensional equivalent of standard deviation.

Naturally, with a multidimensional distribution, and one that is most likely multidimensionally nonnormal at that, constructing such a measure is not exactly straightforward, but in principle there is little difficulty in constructing such a measure. It is convenient to construct such a measure to return a value that resembles the *z* score (mentioned in Chapter 5), and such measures are sometimes called pseudo-*z* scores (*pz*). It is convenient to embed a novelty detector generating a *pz* score into the PIE-I, although it is not a necessary part of the PIE as it plays no role in actually preparing data for the model. However, with such a *pz* score available from the PIE-I, it is relatively easy to monitor any “drift” in the execution values that might indicate that, at the least, some recalibration of the model is needed.

11.9 Other Directions

This whistle-stop tour of the territory covered by the data survey has only touched briefly on a number of useful topics. A survey does much more, providing the miner with a considerable amount of information prior to mining. The survey looks at data sets from several other perspectives.

For instance, individual variable distributions are mapped and compared. Recall from Chapter 7 that much useful information can be discovered from the actual shape of the distribution—some idea of underlying generating processes or biases. The survey maps similarities and differences. Sensitivity analysis is another area surveyed—areas where the manifold is most sensitive to changes in the data.

The survey also uses three metaphors for examining data. Two have been used in the chapters on data preparation—“manifolds” and “shapes.” A manifold is a flexible structure fitted in some way to the data set. The metaphor of a shape regards all of the data points as “corners” and regards the data set as creating some multidimensional structure.

The other useful metaphor used in the survey is that of a *tensegrity structure*. Tensegrity structures are sometimes seen as sculpture. The tensegrity structure is made of beams and wires. The beams are rigid and resist compression. The wires are in tension. This “string and poles” structure, as a sculpture, forms a three-dimensional object that is self-supporting. The compression in the beams is offset by the tension in the wires. (As a matter of interest, tensegrity structures occur naturally as, for instance, the internal “scaffolding” of body cells.) A data set can be imagined as a structure of points being pulled toward each other by some forces, and pushed apart by others, so that it reaches some equilibrium state. This is a sort of multidimensional tensegrity data structure. The data survey uses techniques to estimate the strength of the tension and compression forces, the “natural” shape of the data set, how much “energy” it contains, and how much “effort” it takes to move the shape into some other configuration. All of these measures relate to real-world phenomena and prove very useful in assessing reliability and applicability of models. It is also a useful analogy for finding “missing” variables. For instance, if the tensegrity structure is in some way distorted (not in equlibrium), there must be some missing part of the structure that, if present, holds the tensegrity shape in balance. Knowing what the missing information looks like can sometimes give a clue as to what additional data is needed to balance the structure. On the other hand, it might indicate sampling bias.

Being very careful not to confuse correlation with causality, the survey also looks at the “direction” of the influencing forces between variables and variable groupings. This uses techniques that measures the “friction” between variables or states. As one variable or state moves (changes state or value), so others move, but there is some loss (friction) or gain (amplification) in the interaction between all of the components of the variable system. Measuring this gives “directional” information about which component is “driving” which. There is much that can be done with such information, which is sometimes also called “influence.”

Other, new methods of looking at data are coming to the fore that offer, or promise to offer, useful survey information. The problem with many of them is that they are computationally intensive—which defeats the whole purpose of surveying. The whole idea is a quick once-over, searching deeper only where it looks like there might be particular problems. But new techniques from such fields as fractal analysis, chaos theory, and complexity theory hold much promise.

Fractals use a measurement of self-similarity. So far we have assumed that a modeler is looking at a relationship in data to model. The model pushes and pulls (usually mathematically) at the manifold in order to best fit it to the data set. Essentially, the manifold is a simple thing, manipulated into a complex shape. The model is complex; the manifold is simple. Fractals take a different approach. They assume that many areas of the manifold are indeed complex in shape, but similar to each other. It may then be enough to simply model a little bit of the complex manifold, and for the rest, simply point to where it fits, and how big it is in that location. Fractals, then, take a complex manifold and fit it together in simple ways, stretching and shrinking the shape as necessary. With fractals, the manifold is complex; the model is simple.

When it works, this is an enormously powerful technique. If a manifold does exhibit self-similarity, that alone is powerful knowledge. A couple of particularly useful fractal measures are the cluster correlation dimension and the Korack patchiness exponent. The problem with these techniques is, especially for high dimensionalities, they become computationally intensive—too much so, very often, for the data survey.

Chaos theory allows a search for attractors—system states that never exactly repeat, but around which the system orbits. Looking in data sets for attractors can be very useful, but again, these tend to be too computationally expensive for the survey, at least at present. However, computers are becoming faster and faster. (Indeed, as I write, there is commentary in the computer press that modern microprocessors are “too” fast and that their power is not needed by modern software! Use it for better surveying and mining!) Additional speed is making it possible to use these new techniques for modeling and surveying. Indeed, the gains in speed allow the automated directed search that modern surveying accomplishes. Very soon, as computer power increases survey techniques, new areas will provide practical and useful results.

Supplemental Material

Entropic Analysis—Example

After determining the confidence that the multivariable variability of a data set is captured, entropic analysis forms the main tool for surveying data. The other tools are useful, but used largely for exploring only where entropic or information analysis points to potential problems. Since entropic analysis is so important to the survey, this section shows the way in which the entropy measures were derived for the example in this chapter. Working through this section is not necessary to understand the topics covered here.

Calculating Basic Entropy

The example used two variables: the input variable and the output variable. The full range of calculations for forward and reverse entropy, signal entropy and mutual information, even for this simplified example, are quite extensive. For instance, determining the entropy of each of these two variables requires a fair number of calculations.

All probability-based statistics is based on counting the frequency of occurrence of values and joint combinations of values. Where truly continuously valued variables are used, this requires limiting the number of discrete values in the continuous range in some way, perhaps by binning the values. Bins are used in this example. Figure 11.14 has a reprise of Figure 11.1 in the upper image for comparison with the lower image. The upper image shows the original data set together with the manifold of a fitted function. The lower image shows the binned values for this data set. For comparison, a modeled manifold has been fitted to the binned data too, and although not identical in shape to that for the unbinned data, it is very similar.

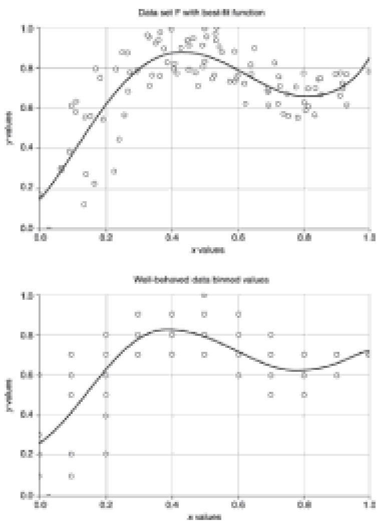


Figure 11.14 Test data set together with a manifold fitted by a modeling tool (top). The effect of binning the data (bottom); the circles show the center of the "full" bin positions.

Binning divides the state space into equally sized areas. If a data point falls into the area of the bin, that bin is denoted as "full." In an area with no data points, the appropriate bin is denoted as "empty." A circle shown on the lower image indicates a full bin. For simplicity, every bin is considered equally significant regardless of how many data points fall into it. This slightly simplifies the calculations and is reasonably valid so long as the bins are relatively small and each contains approximately the same number of points. (There are several other ways of binning such data. A more accurate method might weight the bins according to the number of data points in each. Another method might use the local density in each bin, rather than a count weighting.)

To make the calculation, first determine the frequency of the bins for X and Y. Simply count how many bins there are for each binned value of X, and then of Y, as shown in Table 11.9.

TABLE 11.9 Bin frequencies.

Bin	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
X	5	5	6	3	3	4	4	4	3	2	1
Y	1	2	3	1	1	4	7	10	6	4	1

This table indicates, for instance, that for the X value of 0, there are five bins. To discover this, look at the X value of 0, and count the bins stacked above that value.

From this, determine the relative frequency. (For instance, there are 40 bins altogether. For X bin value 0.0, there are five occurrences, and 5/40 = 0.125, which is the relative frequency.) This gives a relative frequency for the example data shown in Table 11.10.

TABLE 11.10   Bin relative frequencies.

Bin	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
X	0.125	0.125	0.150	0.075	0.075	0.100	0.100	0.100	0.075	0.050	0.025
Y	0.025	0.050	0.075	0.025	0.025	0.100	0.175	0.250	0.150	0.100	0.025

The reasoning behind the entropy calculations is already covered in the early part of this chapter and is not reiterated here. The relative frequency represents the probability of occurrence that directly allows the entropy determination as shown in Table 11.11.

TABLE 11.11 Entropy determination.

Log2(Px)	3.00	3.00	2.74	3.74	3.74	3.32	3.32	3.32	3.74	4.32	5.32

Log₂(P*y*)	5.32	4.32	3.74	5.32	5.32	3.32	2.52	2.00	2.74	3.32	5.32
P log₂(P*x*)	0.38	0.38	0.41	0.28	0.28	0.33	0.33	0.33	0.28	0.22	0.13
P log₂(P*y*)	0.13	0.22	0.28	0.13	0.13	0.33	0.44	0.50	0.41	0.33	0.13

The theoretical maximum entropy is P log₂(40), since there are 40 bins. The calculated entropy in this data set is, for X, <F128P9.5M%6>SP log₂(P*x*), and for Y, <F128P9.5M%6>SP log₂(P*y*):

Maximum entropy	3.459
Entropy X	3.347
Entropy Y	3.044

Clearly, there is not much to be estimated from simply knowing the values of X or Y as the entropy values are near to the maximum. Obviously, absolute entropy values will change with the number of bins. However, entropy can never be less than 0, not greater than the maximum entropy, so it is always possible to normalize these values across the range of 0–1.

Information-Driven Binning Strategies

Since bin count changes entropy measurements, some bin counts result in better or worse information measures than others. For any data set there is some optimum number of bins that best preserves the data set's information content. Discovering this optimum bin size/count requires a search across several to many bin sizes. Any binning strategy loses some information, but some modeling tools require binning and others are enormously faster with binned data rather than continuous data. The performance and training trade-offs versus the information lost (usually small, particularly if an optimal binning strategy is used) frequently favor binning as a worthwhile practical strategy. When the optimal bin count is used, it is described as *least information loss binning*.

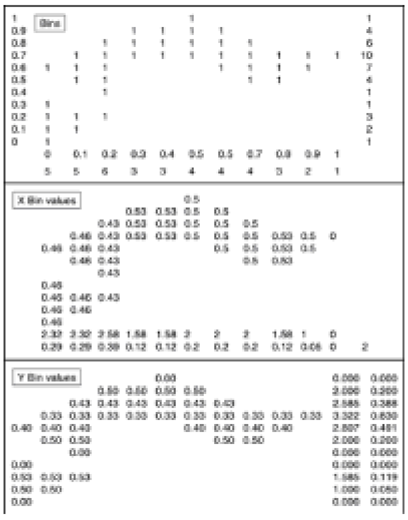
To complicate matters further, it is possible to use different bin sizes to best preserve information content. This is called *information-retentive adaptive binning* since the bin size adapts to optimize the information structure in the data.

Although the data survey derives these information-driven binning optimums, space constraints prevent a fuller discussion of the topic here.

Conditional Entropy and Mutual Information

Recall that mutual information between two data sets (individual variables in this example) is the entropy of one variable, less the entropy of the second, given the first. The first step is to find the entropy of all values of "the entropy of the second, given the first" for each discrete value of the first. This produces measures of conditional entropy for all values of one variable. Since both forward and reverse conditional entropy are surveyed, we must find both the conditional entropy of X given Y, and of Y given X for all values of X and Y.

Figure 11.15 shows the results of all the entropy calculations. The upper box labeled "Bins" reproduces in the pattern of ones the layout of bins shown in the lower part of Figure 11.14. For reference, to the left and immediately below the pattern of ones are shown the appropriate bin values to which each one corresponds. On the extreme right are the Y value bin counts, while on the extreme bottom are shown the X value bin counts. So, for instance, looking at the X value of 0, there are 5 ones in the vertical column above the X = 0 value, and so the bin count, shown below the X = 0 value, is 5. Similarly, for Y = 0.7 the horizontal bin count contains 10 ones, so the bin count is 10, as shown on the right.



```
1     Bins                        1                                 1
0.9                   1     1     1     1                           4
0.8             1     1     1     1     1     1                     6
0.7       1     1     1     1     1     1     1     1     1     1   10
0.6 1     1     1                       1     1     1     1         7
0.5       1     1                             1     1               4
0.4             1                                                   1
0.3 1                                                               1
0.2 1     1     1                                                   3
0.1 1     1                                                         2
0   1                                                               1
    0     0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8   0.9   1
    5     5     6     3     3     4     4     4     3     2     1

X Bin values
                                  0.5
                      0.53  0.53  0.5   0.5
                0.43  0.53  0.53  0.5   0.5   0.5
          0.46  0.43  0.53  0.53  0.5   0.5   0.5   0.53  0.5   0
    0.46  0.46  0.43                    0.5   0.5   0.53  0.5
          0.46  0.43                          0.5   0.53
                0.43
    0.46
    0.46  0.46  0.43
    0.46  0.46
    0.46
    2.32  2.32  2.58  1.58  1.58  2     2     2     1.58  1     0
    0.29  0.29  0.39  0.12  0.12  0.2   0.2   0.2   0.12  0.08  0      2

Y Bin values
                              0.00                                  0.000  0.000
                  0.50  0.50  0.50  0.50                            2.000  0.200
            0.43  0.43  0.43  0.43  0.43  0.43                      2.585  0.388
      0.33  0.33  0.33  0.33  0.33  0.33  0.33  0.33  0.33  0.33    3.322  0.830
0.40  0.40  0.40                    0.40  0.40  0.40  0.40          2.807  0.491
      0.50  0.50                          0.50  0.50                2.000  0.200
            0.00                                                    0.000  0.000
0.00                                                                0.000  0.000
0.53  0.53  0.53                                                    1.585  0.119
0.50  0.50                                                          1.000  0.050
0.00                                                                0.000  0.000
```

Figure 11.15 Calculating mutual entropy.

The two boxes "X Bin values" and "Y Bin values" maintain the same pattern, but show the entropy values for each state and bin. As an example, continue to look at the X bins for the value X = 0. There are five bins that match X = 0. These bins correspond to valid system states, or signals, and in this example we assume that they are each equally likely. Each of these five states has a probability of 1/5, or 0.2. The P $\log_2(P)$ for each of these five equally likely states is therefore 0.46. Thus the ones in the “Bins” box in the figure are replaced with 0.46s in the “X Bin values” box for the value of X = 0. This replacement is continued for all of the X and Y bin values in the appropriate boxes and with the appropriate values for each.

For all of the X bins, their values are summed and shown below the appropriate column. Thus, continuing to look at the X = 0 bins in the “X Bin values” box, the sum of these five values of 0.46 is 2.32, which is shown immediately below the bin column. (Rounding errors to two decimal places means that the figures shown seem slightly off. In fact, of course, the P $\log_2(P)$ is slightly greater than 0.46, so that five of them sum to 2.32, not 2.30!) For the Y bins the sum is shown to the immediate right of the bin pattern in the “Y Bin values” box, as these are summarized horizontally.

Recall that altogether there are 40 signals (bins). For the value of X = 0, the probability of the 5 bins occurring is 5/40 = 0.125. So the value X = 0 occurs with probability 0.125. This 0.125 is the probability weighting for the system state X = 0 and is applied to the total bin sum of 2.32, giving an entropy measure for X = 0 of 0.29, shown on the lowest line below the X = 0 value in the “X Bin values” box. Similarly, the corresponding entropy values for all of the Y values are shown on the extreme right of the “Y Bin values” box. Summing the totals produces the measures shown in Table 11.12. Mapping the entropy values calculated here has already been shown in Figure 11.3.

TABLE 11.12 Example data set entropies.

Measure	**Actual**	**Norm**
Maximum entropy	3.459	
Entropy X	3.347	0.968
Entropy Y	3.044	0.880
Entropy (Y|X)	1.975	0.649
Entropy (X|Y)	2.278	0.681
Mutual info (X;Y)	1.069	0.351

Surveying Data Sets

As with so much else in life, there is a gap between theory and practical application with entropic analysis. Three sample data sets are included on the accompanying CD-ROM: CARS, SHOE, and CREDIT. This section looks at what useful insights the entropic analysis part of the data survey discovers before modeling this data. Just as there is not enough space in the chapter to make more than a brief introduction to some elements of the data survey, so too there is not space to look at, and discuss, more than a small portion of the entropic analysis of the data sets, let alone a full data survey. This section limits its attention to a small part of what the survey shows about the example data sets, specifically how entropic analysis can discover information about a data set, and how the miner can use the discovered information.

Introductory Note: Sequential Natural Clustering

Before looking at extracts of surveys of these data sets, the explanation needs a couple of introductory notes to give some perspective as to what these survey extracts reveal. Information analysis bases its measurements on features of system states. This means that some way of identifying system states has to be used to make the survey. There are many possible ways of identifying system states: several have to be included in any surveying software suite since different methods are appropriate for different circumstances. The method used in the following examples is that of *sequential natural clustering*.

Natural clusters form in the state space constructed from a representative sample. A group of points forms a natural cluster when a low-density boundary around the group separates those points inside the boundary from those points outside. The mean location of the center of such a cluster can itself be used as a representative point for the whole cluster. When this is done, it is possible then to move to another stage and cluster these representative points—a sort of cluster of clusters. Continuing this as far as possible eventually ends with a single cluster, usually centered in state space. However, since the clusters begin as very small aggregations, which lead to larger but still small aggregations, there can be many steps from start to finish. Each step has fewer clusters than the preceding step. At each step of clustering, the group of clusters that exist at that step are called a *layer*. Every input point maps into just one cluster at each layer. Each layer in the sequence is built from a natural clustering of the points at the previous layer—thus the name “sequential natural clustering.”

Both the input states and the output states are clustered. In the examples that follow, the output is limited to the states of a single variable. There is no reason to limit the output to a single variable save that it makes the explanation of these examples easier. In practice, miners often find that however large the input data set, the output states are represented by the states of a single variable. Sticking to a single variable as output for the examples here is not unrealistic. However, the tools used to make the survey are not limited to using only a single variable as output.

Sequential natural clustering has several advantages, one of which is that it allows the survey to estimate the complexity of the model required to explicate the information enfolded into the data set. There is no room here to look at the underlying explanation for why this is so, but since it is of particular interest to miners, it is shown in the survey extracts discussed.

A full survey digests a vast amount of metadata (data about the data set) and makes available an enormous amount of information about the entropic relationships between all of the variables, and between all of the layers. Unfortunately, a full discussion of a single survey is beyond the scope intended for this overview. Rather, we briefly examine the main points of what the entropic measures in a survey show, and why and how it is useful in practice to a miner.

The Survey Extract

The survey extracts used in the following examples report several measures. Each of the measures reveals useful information. However, before looking at what is reported about the data sets, here is a brief summary of the features and what they reveal.

Input layer 0 to output layer 0 In these extracts, the survey directly reports the input and output layer 0's entropic information. Layer 0 uses unclustered signals so that the entropies reported are of the raw input and output signal states. Using input layer 0 and output layer 0 measures the maximum possible information about the input and output. Thus, the layer 0 measures indicate the information content under the best possible circumstances—with the maximum amount of information exposed. It is possible, likely even, that modeling tools used to build the actual mined models cannot use all of the information that is exposed. As discussed in the examples, a miner may not even want to use all of this information. However, the layer 0 measures indicate the best that could possibly be done using a perfect modeling tool and using only the analyzed data set.

Any number of factors can intrude into the modeling process that prevent maximum information utilization, which is not necessarily a negative since trade-offs for modeling speed, say, may be preferable to maximum information extraction. For example, using a less complex neural network than is needed for maximum information extraction may train tens or hundreds of times faster than one that extracts almost all of the information. If the model is good enough for practical application, having it tens or hundreds of times earlier than otherwise may be more important than wringing all of the information out of a data set. This is always a decision for the miner, based, of course, on the business needs of the required model. However, the reason the extracts here show only the entropy measures for layer 0 is that this is the theoretical maximum that cannot be exceeded given the data at hand.

The complexity graph, mentioned below, uses information from other layers, as does the measurement of noise in the data set.

Signal H(X) Entropy can evaluate the relationship between input signals and output signals. However, it can also be used to evaluate the signals in a single variable. Recall that when the signals are evenly distributed, entropy is 1. The usual symbol for entropy is "H." "X" symbolizes the input. Signal H(X) is evaluating the entropy of the input signal. These signals originate not from a single variable but from the whole input data set. (Recall that "signal" is definitely not synonymous with "variable.") The measure indicates how much entropy there is in the data set input signal without regard to the output signal. It measures, among other things, how well "balanced" the input signal states are. An ideal data set needs each of the input signals to be equally represented, therefore equally uncertain. Thus Signal H(X) should be as high as possible, measured against the maximum possible entropy for the number of signal states. The ratio measurement makes this comparison of actual entropy against maximum possible entropy, and ideally it should be as close to 1 as possible.

The ratio is calculated as Signal H(X):$\log_2(^{n}$input states).

Signal H(Y) Whereas "X" indicates the input states, "Y" indicates the output states. Signal H(Y) measures the entropy of the output signal states, and again, its ratio should be as high as possible. In other respects it is similar to Signal H(X) in that it too measures the entropy of the output states without regard to the input states.

The ratio is Signal H(Y):$\log_2(^{n}$output states).

Channel H(X) The channel measurements are all taken about the data set as a whole. In all of the channel measurements the relationship between the input signals and the output signals is of paramount importance. It is called "channel entropy" because these measures regard the data set as a communication channel and they all indicate something about the fidelity of the communication channel—how well the input signals communicate information about the output, how much of all the information enfolded into the data set is used to specify the output, and how much information is lost in the communication process.

Channel H(X) is usually similar in its entropic measure to Signal H(X). The difference is that Signal H(X) measures the average information content in the signal without reference to anything else. Channel H(X) measures the average information per signal at the input of the communication channel. The output signals may interact with the input when the channel is considered. If, for instance, some of the discrete input signals all mean the same thing at the output, the information content of the input is reduced. (For instance, the words "yes," "aye," "positive," "affirmative," and "roger" may seem to be discrete signals. In some particular communication channel where all these signals indicate agreement and are completely synonymous with each other, they actually map to effectively the same input state. For these signals, signal entropy is based on four signals, whereas channel entropy is based on only one composite signal. For a more practical example, look back to Chapter 4 where "M-Benz," "Merc," and so on are all different signals for Signal H(X), but comprise a single signal for Channel H(X).)

Channel H(X) gives a measure of how well the input channel signals are balanced. If Signal H(X) and Channel H(X) differ considerably, it may be well worth the miner's time to reconfigure the data to make them more similar. Channel H(X) is almost always less than Signal H(X). If Channel H(X) is much less, it implies that the model may have to be far more complex than if the two entropy measures are more nearly equal. Once again, the ratio measure should be as large as possible.

The ratio is Channel H(X):$\log_2$(*n*input states).

Note: In order to differentiate Channel H(X) and SignalH(X), and other entropy measures where confusion may occur, the symbols are preceded by "s" or "c" as appropriate to indicate signal or channel measures. Thus sH(X) refers to a signal measure, and cH(X) refers to a channel measure.

Channel H(Y) Channel H(Y) is like cH(X) except, of course, that the "Y" indicates measures about the output states. Another difference from the input is that any reduction in entropy from signal to channel indicates a potential problem since it means that some of the output states simply cannot be distinguished from each other as separate states. Ratio measures should be as close to 1 as possible.

The ratio is cH(Y):$\log_2(^n$output states).

Channel H(X|Y) Although listed first in the survey report, cH(X|Y) represents reverse entropy—the entropy of the input given the output. If it is less than the forward entropy—cH(Y|X)—then the problem may well be ill formed. (The example shown in the first part of this section shows an ill-formed relationship.) If this is the case, the miner will need to take corrective action, or get additional data to fix the problem. Channel H(X|Y) measures how much information is known about the input, given that a specific output has occurred.

The ratio is cH(X|Y):$\log_2(^n$input states).

Channel H(Y|X) This is a measure of the forward entropy, that is, how much information is known about the state of the output, given that a particular input has occurred. The ratio of this number needs to be as near to 0 as possible. Remember that entropy is a measure of uncertainty, and ideally there should be no uncertainty on average about the state of the output signal given the input signals. When the ratio of this measure is high (close to 1), very little is known about the state of the output given the state of the input.

The ratio is cH(Y|X):$\log_2(^n$output states).

Channel H(X,Y) (Note the comma, not a vertical bar.) Channel H(X,Y) measures the average information for every pair of input and output signals and the average uncertainty over the data set as a whole. In a sense, it measures how efficiently the data set enfolds (carries or represents) its information content. Perhaps another way of thinking of it is as yielding a measure of how much information in the data set isn't being used to define the output signals. (See cI(X;Y) below.)

The ratio is cH(X,Y):cH(X) + cH(Y).

Channel I(X;Y) This measures the mutual information between input and output (also between output and input since mutual information is a completely reciprocal measure). When the output signals are perfectly predicted by the input signals, ratio cI(X;Y) = 1. Note also that when the data set perfectly transfers all of its information, then cI(X;Y) = cH(X,Y).

The ratio is cI(X;Y):cH(Y).

Variable entropy and information measures Following the data set entropy and information measures are ratio measures for the individual variables. Due to space limitations, these examples do not always show all of the variables in a data set. Each variable has four separate measures:

H(X)—signal entropy for the individual variable

- H(Y|X)—showing how much information each variable individually carries about the output states

- I(X;Y)—the mutual information content for each individual variable with the output

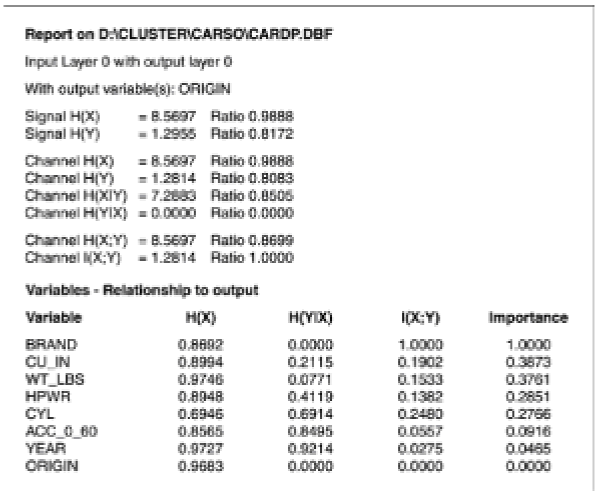
- Importance—an overall estimate of the uniqueness of the information contributed by each variable

The CARS Data Set

The CARS data set is fairly small, and although it is not likely to have to be mined to glean an understanding of what it contains, that property makes it a useful example! The data is intuitively understandable, making it easy to relate what the survey reveals about the relationships back to the data. For that reason, the examples here examine the CARS data set more extensively than is otherwise warranted. The meanings of the variable names are fairly self-evident, which makes interpretation straightforward. Also, this data set is close to actually being the population! Although the measure of possible sampling error (not shown) indicates the possible presence of sampling error, and although the "sample" is "small," the miner can establish that details of most of the car models available in the U.S. for the period covered are actually in the data set.

Predicting Origin

Information metrics Figure 11.16 shows an extract of the information provided by the survey. The cars in the data set may originate from Europe, Japan, or the U.S. Predicting the cars' origins should be relatively easy, particularly given the brand of each car. But what does the survey have to say about this data set for predicting a car's origin?



```
Report on D:\CLUSTER\CARSO\CARDP.DBF

Input Layer 0 with output layer 0

With output variable(s): ORIGIN

Signal H(X)       = 8.5697   Ratio 0.9888
Signal H(Y)       = 1.2955   Ratio 0.8172

Channel H(X)      = 8.5697   Ratio 0.9888
Channel H(Y)      = 1.2814   Ratio 0.8083
Channel H(X|Y)    = 7.2883   Ratio 0.8505
Channel H(Y|X)    = 0.0000   Ratio 0.0000

Channel H(X;Y)    = 8.5697   Ratio 0.8699
Channel I(X;Y)    = 1.2814   Ratio 1.0000

Variables - Relationship to output

Variable           H(X)       H(Y|X)      I(X;Y)      Importance

BRAND              0.8692     0.0000      1.0000      1.0000
CU_IN              0.8994     0.2115      0.1902      0.3873
WT_LBS             0.9746     0.0771      0.1533      0.3761
HPWR               0.8948     0.4119      0.1382      0.2851
CYL                0.6946     0.6914      0.2480      0.2766
ACC_0_60           0.8565     0.8495      0.0557      0.0916
YEAR               0.9727     0.9214      0.0275      0.0465
ORIGIN             0.9683     0.0000      0.0000      0.0000
```

Figure 11.16 Extract of the data survey report for the CARS data set when predicting the cars ORIGIN. Cars may originate from Japan, the U.S., or Europe.

First of all, sH(X) and sH(Y) are both fairly close to 1, showing that there is a reasonably good spread of signals in the input and output. The sH(Y) ratio is somewhat less than 1, and looking at the data itself will easily show that the numbers of cars from each of the originating areas is not exactly balanced. But it is very hard indeed for a miner to look at the actual input states to see if they are balanced—whereas the sH(X) entropy shows clearly that they are. This is a piece of very useful information that is not easily discovered by inspecting the data itself.

Looking at the channel measures is very instructive. The signal and channel H(X) are identical, and signal and channel H(Y) are close. All of the information present in the input, and most of the information present in the output, is actually applied across the channel.

cH(X|Y) is high, so that the output information poorly defines the state of the input, but that is of no moment. More importantly, cH(X|Y) is greater than cH(Y|X)—much greater in this case—so that this is not an ill-defined problem. Fine so far, but what does cH(Y|X) = 0 mean? That there is no uncertainty about the output signal given the input signal. No uncertainty is exactly what is needed! The input perfectly defines the output. Right here we immediately know that it is at least theoretically possible to perfectly predict the origin of a car, given the information in this data set.

Moving ahead to cI(X;Y) = 1 for a moment, this too indicates that the task is learnable, and that the information inside the channel (data set) is sufficient to completely define the output. cH(X;Y) shows that not all of the information in the data set is needed to define the output.

Let us turn now to the variables. (All the numbers shown for variables are ratios only.) These are listed with the most important first, and BRAND tells a story in itself! Its cH(Y|X) = 0 shows that simply knowing the brand of a vehicle is sufficient to determine its origin. The cH(Y|X) says that there is no uncertainty about the output given only brand as an input. Its cI(X;Y) tells the same story—the 1 means perfect mutual information. (This conclusion is not at all surprising in this case, but it's welcome to have the analysis confirm it!) It's not surprising also that its importance is 1. It's clear too that the other variables don't seem to have much to say individually about the origin of a car.

This illustrates a phenomenon described as *coupling*. Simply expressed, coupling measures how well information used by a particular set of output signals connects to the data set as a whole. If the coupling is poor, regardless of how well or badly the output is defined by the input signals, very little of the total amount of information enfolded in the data set is used. The higher the coupling, the more the information contained in the data set is used.

Here the output signals seem only moderately coupled to the data set. Although a coupling ratio is not shown on this abbreviated survey, the idea can be seen here. The prediction of the states of ORIGIN depends very extensively on states of BRAND. The other variables do not seem to produce signal states that well define ORIGIN. So, superficially it seems that the prediction of ORIGIN requires the variable BRAND, and if that were removed, all might be lost. But what is not immediately apparent here (but is shown in the next example to some extent) is that BRAND couples to the data set as a whole quite well. (That is, BRAND is well integrated into the overall information system represented by the variables.) If BRAND information were removed, much of the information carried by this variable can be recovered from the signals created by the other variables. So while ORIGIN seems coupled only to BRAND, BRAND couples quite strongly to the information system as a whole. ORIGIN, then, is actually more closely coupled to this data set than simply looking at individual variables may indicate. Glancing at the variable's metrics may not show how well—or poorly—signal states are in fact coupled to a data set. The survey looks quite deeply into the information system to discover coupling ratios. In a full survey this coupling ratio can be very important, as is shown in a later example.

When thinking about coupling, it is important to remember that the variables defining the manifold in a state space are all interrelated. This is what is meant by the variables being part of a system of variables. Losing, or removing, any single variable usually does not remove all of the information carried by that variable since much, perhaps all, of the information carried by the variable may be duplicated by the other variables. In a sense, coupling measures the degree of the total interaction between the output signal states and all of the information enfolded in the data set, regardless of where it is carried.

Complexity map A *complexity map* (Figure 11.17) indicates highest complexity on the left, with lower complexity levels progressively further to the right. Information recovery indicates the amount of information a model could recover from the data set about the output signals: 1 means all of it, 0 means none of it. This one shows perfect predictability (information recovery = 1) for the most complex level (complexity level 1). The curve trends gently downward at first as complexity decreases, eventually flattening out and remaining almost constant as complexity reduces to a minimum.

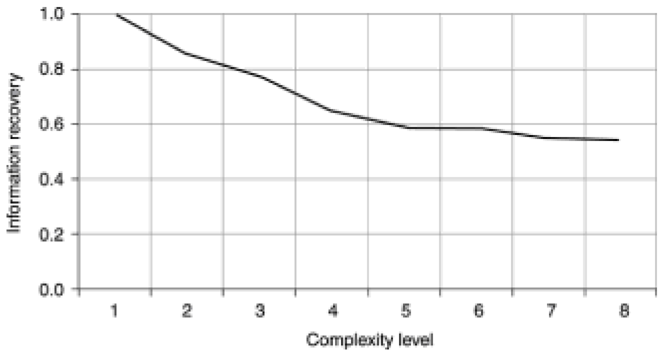


Figure 11.17 Complexity map for the CARS data set when predicting ORIGIN. Highest complexity is on the left, lowest complexity is on the right. (Higher numbers mean less complexity.)

In this case the data set represents the population. Also, a predictive model is not likely to be needed since any car can be looked up in the data. The chances are that a miner is looking to understand relationships that exist in this data. In this unusual situation where the whole population is present, noise is not really an issue. There may certainly be erroneous entries and other errors that constitute noise. The object is not to generalize relationships from this data that are then to be applied to other similar data. Whatever can be discovered in this data is sufficient, since it works in this data set, and there is no other data set to apply it to.

The shallow curve shows that the difficulty of recovering information increases little with increased complexity. Even the simplest models can recover most of the information. This complexity map promises that a fairly simple model will produce robust and effective predictions of origin using this data. (Hardly stunning news in this simple case!)

State entropy map A *state entropy map* (Figure 11.18) can be one of the most useful maps produced by the survey. This map shows how much information there is in the data set to define each state. Put another way, it shows how accurately, or confidently, each output state is defined (or can be predicted). There are three output signals shown, indicated as “1,” “2,” and “3” along the bottom of the map. These correspond to the output signal states, in this case “U.S.,” “Japan,” and “Europe.” For this brief look, the actual list of which number applies to which signal is not shown. The map shows a horizontal line that represents the average entropy of all of the outputs. The entropy of each output signal is shown by the curve. In this case the curve is very close to the average, although signal 1 has slightly less entropy than signal 2. Even though the output signals are perfectly identified by the input signals, there is still more uncertainty about the state of output signal 2 than of either signal 1 or signal 3.

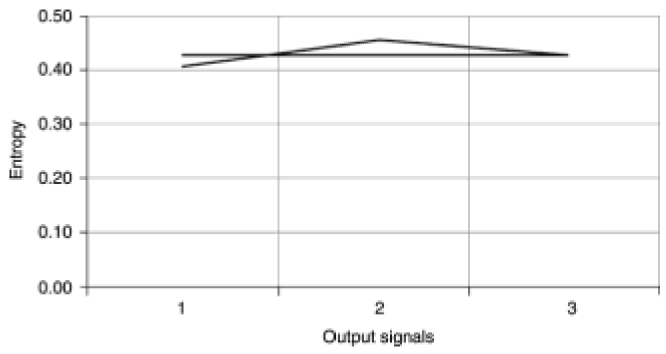


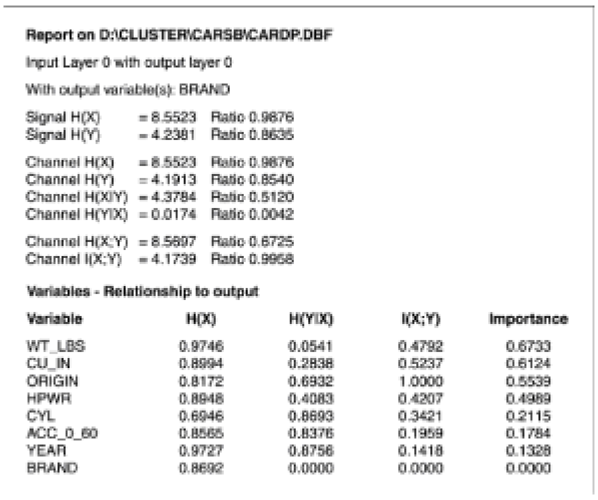
Figure 11.18 State entropy map for the CARS data set when predicting ORIGIN. The three states of ORIGIN are shown along the bottom of the graph (U.S., Japan, and Europe).

Summary No really startling conclusions jump out of the survey when investigating country of origin for American cars! Nevertheless, the entropic analysis confirmed a number of intuitions about the CARS data that would be difficult to obtain by any other means, particularly including building models.

This is an easy task, and only a simple model using a single-input variable, BRAND, is needed to make perfect predictions. However, no surprises were expected in this easy introduction to some small parts of the survey.

Predicting Brand

Information metrics Since predicting ORIGIN only needed information about the BRAND, what if we predict the BRAND? Would you expect the relationship to be reciprocal and have ORIGIN perfectly predict BRAND? (Hardly. There are only three sources of origin, but there are many brands.) Figure 11.19 shows the survey extract using the CARS data set to predict the BRAND.



```
Report on D:\CLUSTER\CARSB\CARDP.DBF

Input Layer 0 with output layer 0

With output variable(s): BRAND

Signal H(X)        = 8.5523   Ratio 0.9876
Signal H(Y)        = 4.2381   Ratio 0.8635

Channel H(X)       = 8.5523   Ratio 0.9876
Channel H(Y)       = 4.1913   Ratio 0.8540
Channel H(X|Y)     = 4.3784   Ratio 0.5120
Channel H(Y|X)     = 0.0174   Ratio 0.0042

Channel H(X;Y)     = 8.5697   Ratio 0.6725
Channel I(X;Y)     = 4.1739   Ratio 0.9958

Variables - Relationship to output

Variable          H(X)       H(Y|X)      I(X;Y)      Importance

WT_LBS            0.9746     0.0541      0.4792      0.6733
CU_IN             0.8994     0.2838      0.5237      0.6124
ORIGIN            0.8172     0.6932      1.0000      0.5539
HPWR              0.8948     0.4083      0.4207      0.4989
CYL               0.6946     0.8693      0.3421      0.2115
ACC_0_60          0.8565     0.8376      0.1959      0.1784
YEAR              0.9727     0.8756      0.1418      0.1328
BRAND             0.8692     0.0000      0.0000      0.0000
```

Figure 11.19 Part of the survey report for the CARS data set with output signals defined by the variable BRAND.

A quick glance shows that the input and output signals are reasonably well distributed (H(X) and H(Y)), the problem is not ill formed (H(X|Y) and H(Y|X)), and good but not perfect predictions of the brand of car can be made from this data (H(Y|X) and I(X;Y)).

BRAND is fairly well coupled to this data set with weight and cubic inch size of the engine carrying much information. ORIGIN appears third in the list with a cI(X;Y) = 1, which goes to show the shortcoming of relying on this as a measure of predictability! This is a completely reciprocal measure. It indicates complete information in one direction or the other, but without specifying direction, so which predicts what cannot be determined. Looking at the individual cH(Y|X)s for the variables, it seems that it carries less information than horsepower (HPWR), the next variable down the list.

Complexity map The diagonal line is a fairly common type of complexity map (Figure 11.20). Although the curve appears to reach 1, the cI(X;Y), for instance, shows that it must fall a minute amount short, since the prediction is not perfect, even with a highest degree of complexity model. There is simply insufficient information to completely define the output signals from the information enfolded into the data set.

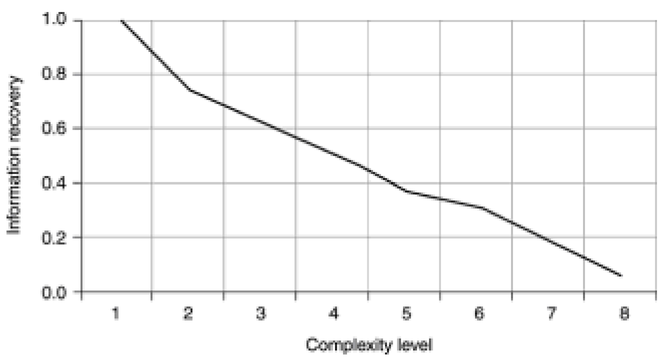


Figure 11.20 Complexity map for the CARS data set using output signals from the variable BRAND.

Once again, noise and sample size limitations can be ignored as the entire population is present. This type of map indicates that a complex model, capturing most of the complexity in the information, will be needed to build the model.

State entropy map Perhaps the most interesting feature of this survey is the *state entropy map* (Figure 11.21). The variable BRAND, of course, is a categorical variable. Prior to the survey it was numerated, and the survey uses the numerated information. Interestingly, since the survey looks at signals extracted from state space, the actual values assigned to BRAND are not important here, but the ordering reflected out of the data set is important. The selected ordering reflected from the data set shown here is clearly not a random choice, but has been somehow arranged in what turns out to be approximately increasing levels of certainty. In this example, the exact labels that apply to each of the output signals is not important, although they will be very interesting (maybe critically important, or may at least lend a considerable insight) in a practical project!

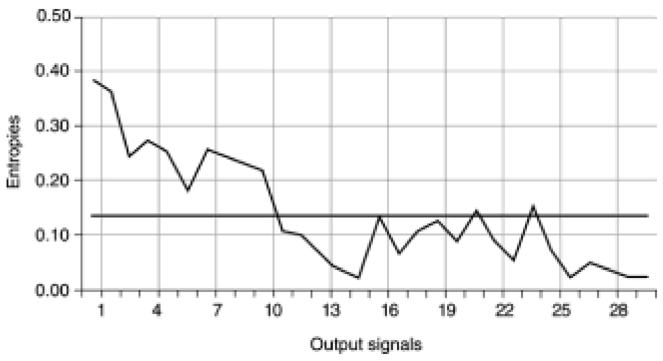


Figure 11.21 State entropy map for the CARS data set and BRAND output signals. The signals corresponding to positions on the left are less defined (have a higher entropy) than those on the right.

Once again, the horizontal line shows the mean level of entropy for all of the output signals. The entropy levels plotted for each of the output signals form the wavy curve. The numeration has ordered the vehicle brands so that those least well determined—that is, those with the highest level of entropy—are on the left of this map, while the best defined are on the right. From this map, not only can we find a definitive level of the exact confidence with which each particular brand can be predicted, but it is clear that there is some underlying phenomenon to be explained. Why is there this difference? What are the driving factors? How does this relate to other parts of the data set? Is it important? Is it meaningful?

This important point, although already noted, is worth repeating, since it forms a particularly useful part of the survey. The map indicates that there are about 30 different brands present in the data set. The information enfolded in the data set does, in general, a pretty good job of uniquely identifying a vehicle's brand. That is measured by the cH(Y|X). This measurement can be turned into a precise number specifying exactly how well—in general—it identifies a brand. However, much more can be gleaned from the survey. It is also possible to specify, for each individual brand, how well the information in the data specifies that a car is or is not that brand. That is what the state entropy map shows. It might, for instance, be possible to say that a prediction of "Ford" will be correct 999 times in 1000 (99.9% of the time), but "Toyota" can only be counted on to be correct 75 times in 100 (75% of the time).

Not shown, but also of considerable importance in many applications, it is possible to say which signals are likely to be confused with each other when they are not correctly specified. For example, perhaps when "Toyota" is incorrectly predicted, the true signal is far more likely to be "Honda" than "Nissan"—and whatever it is, it is very unlikely to be "Ford." Exact confidence levels can be found for confusion levels of all of the output signals. This is very useful and sometimes crucial information.

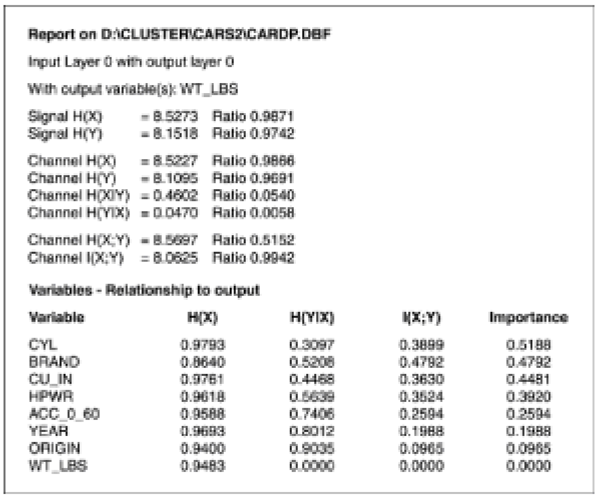
Recall also that this information is all coming out of the survey before any models have been built! The survey is not a model as it can make no predictions, nor actually identify the nature of the relationships to be discovered. The survey only points out potential—possibilities and limitations.

Summary Modeling vehicle brand requires a complex model to extract the maximum information from the data set. Brand cannot be predicted with complete certainty, but limits to accuracy for each brand, and confidence levels about confusion between brands, can be determined. The output states are fairly well coupled into the data set, so that any models are likely to be robust as this set of output signals is itself embedded and intertwined in the complexity of the system of variables as a whole. Predictions are not unduly influenced only by some limited part of the information enfolded in the data set.

There is clearly some phenomenon affecting the level of certainty across the ordering of brands that needs to be investigated. It may be spurious, evidence of bias, or a significant insight, but it should be explained, or at least examined. When a model is built, precise levels of certainty for the prediction of each specific brand are known, and precise estimates of which output signals are likely to be confused with which other output signals are also known.

Predicting Weight

Information metrics There seem to be no notable problems predicting vehicle weight (WT_LBS). In Figure 11.22, cH(X|Y) seems low—the input is well predicted by the output—but as we will see, that is because almost every vehicle has a unique weight. The output signals seem well coupled into the data set.



```
Report on D:\CLUSTER\CARS2\CARDP.DBF

Input Layer 0 with output layer 0

With output variable(s): WT_LBS

Signal H(X)        = 8.5273   Ratio 0.9871
Signal H(Y)        = 8.1518   Ratio 0.9742

Channel H(X)       = 8.5227   Ratio 0.9866
Channel H(Y)       = 8.1095   Ratio 0.9691
Channel H(X|Y)     = 0.4602   Ratio 0.0540
Channel H(Y|X)     = 0.0470   Ratio 0.0058

Channel H(X;Y)     = 8.5697   Ratio 0.5152
Channel I(X;Y)     = 8.0625   Ratio 0.9942

Variables - Relationship to output

Variable           H(X)       H(Y|X)      I(X;Y)      Importance

CYL                0.9793     0.3097      0.3899      0.5188
BRAND              0.8640     0.5208      0.4792      0.4792
CU_IN              0.9761     0.4468      0.3630      0.4481
HPWR               0.9618     0.5639      0.3524      0.3920
ACC_0_60           0.9588     0.7406      0.2594      0.2594
YEAR               0.9693     0.8012      0.1988      0.1988
ORIGIN             0.9400     0.9035      0.0965      0.0965
WT_LBS             0.9483     0.0000      0.0000      0.0000
```

Figure 11.22 Survey extract for the CARS data set predicting vehicle weight (WT_LBS).

There is a clue here in cH(Y|X) and cH(X|Y) that the data is overly specific, and that if generalized predictions were needed, a model built from this data set might well benefit from the use of a smoothing technique. In this case, but only because the whole population is present, that is not the case. This discussion continues with the explanation of the state entropy map for this data set and output.

Complexity map Figure 11.23 shows the complexity map. Once again, a diagonal line shows that a more complex model gives a better result.

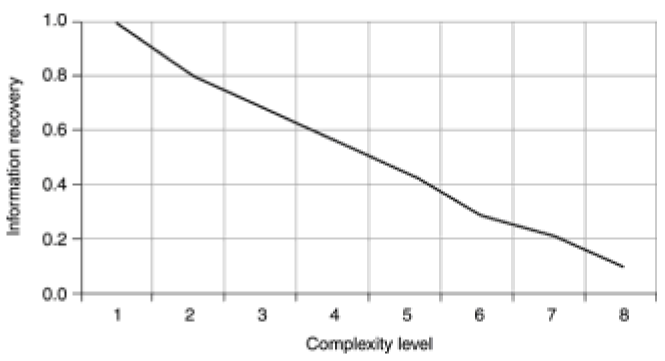


Figure 11.23 Complexity map for the CARS data set predicting vehicle weight.

State entropy map This state entropy map (Figure 11.24) shows many discrete values. In fact, as already noted, almost every vehicle has a unique weight. Since the map shows spikes—in spite of the generally low level of entropy of the output, which indicates that the output is generally well defined—the many spikes show that several, if not many, vehicles are not well defined by the information enfolded into the data set. There is no clear pattern revealed here, but it might still be interesting to ask why certain vehicles are (anomalously?) not well specified. It might also be interesting to turn the question around and ask what it is that allows certainty in some cases and not others. A complete survey provides the tools to explore such questions.

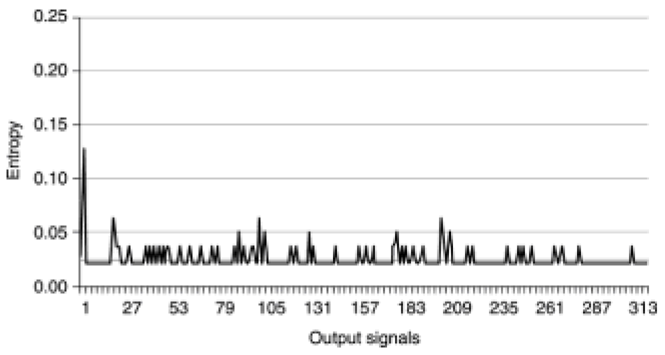


Figure 11.24 State entropy map for the CARS data set with output vehicle weight. The large number of output states reflects that almost every vehicle in the data set weighs a different amount than any of the other vehicles.

In this case, essentially the entire population is present. But if some generalization were needed for making predictions in other data sets, the spikes and high number of discrete values indicate that the data needs to be modified to improve the generalization. Perhaps least information loss binning, either contiguously or noncontiguously, might help. The clue that this data might benefit from some sort of generalization is that both cH(Y|X) and cH(X|Y) are so low. This can happen when, as in this case, there are a large number of discrete inputs and outputs. Each of the discrete inputs maps to a discrete output.

The problem for a model is that with such a high number of discrete values mapping almost directly one to the other, the model becomes little more than a lookup table. This works well only when every possible combination of inputs to outputs is included in the training data set—normally a rare occurrence. In this case, the rare occurrence has turned up and all possible combinations are in fact present. This is due entirely to the fact that this data set represents the population, rather than a sample. So here, it is perfectly valid to use the lookup table approach.

If this were instead a small but representative sample of a much larger data set, it is highly unlikely that all combinations of inputs and outputs would be present in the sample. As soon as a lookup-type model (known also as a *particularized mode*l) sees an input from a combination that was not in the training sample, it has no reference or mechanism for generalizing to the appropriate output. For such a case, a useful model generalizes rather than particularizes. There are many modeling techniques for building such generalized models, but they can only be used if the miner knows that such models are needed. That is not usually hard to tell. What is hard to tell (without a survey) is what level of generalization is appropriate.

Having established from the survey that a generalizing model is needed, what is the appropriate level of generalization? Answering that question in detail is beyond the scope of this introduction to a survey. However, the survey does provide an unambiguous answer to the appropriate level of generalization that results in least information loss for any specific required resolution in the output (or prediction).

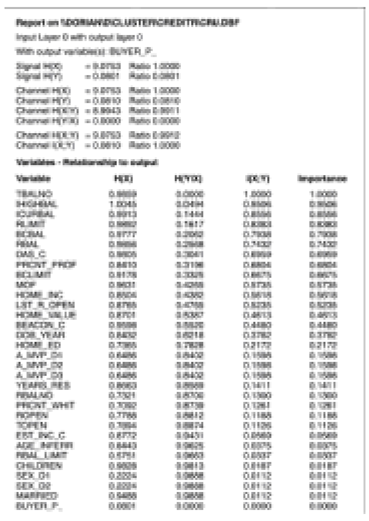
Summary Apart from the information discussed in the previous examples, looking at vehicle weight shows that some form of generalized model has to be built for the model to be useful in other data sets. A complete survey provides the miner with the needed information to be able to construct a generalized model and specifies the accuracy and confidence of the model's predictions for any selected level of generalization. Before modeling begins, the miner knows exactly what the trade-offs are between accuracy and generalization, and can determine if a suitable model can be built from the data on hand.

The CREDIT Data Set

The CREDIT data set represents a real-world data set, somewhat cleaned (it was assembled from several disparate sources) and now ready for preparation. The objective was to build an effective credit card solicitation program. This is data captured from a previous program that was not particularly successful (just under a 1% response rate) but yielded the data with which to model customer response. The next solicitation program, run using a model built from this data, generated a better than 3% response rate.

This data is slightly modified from the actual data. It is completely anonymized and, since the original file comprised 5 million records, it is highly reduced in size!

Information metrics Figure 11.25 shows the information metrics. The data set signals seem well distributed, sH(X) and cH(X), but there is something very odd about sH(Y) and cH(Y)—they are so very low. Since entropy measures, among other things, the level of uncertainty in the signals, there seems to be very little uncertainty about these signals, even before modeling starts! The whole purpose of predictive models is to reduce the level of uncertainty about the output signal given an input signal, but there isn't much uncertainty here to begin with! Why?



```
Report on \DORIAN\D\CLUSTER\CREDIT\CRU.DSF
Input Layer 0 with output layer 0
With output variable(s): BUYER_P_
Signal H(X)        = 9.0753   Ratio 1.0000
Signal H(Y)        = 0.0801   Ratio 0.0801
Channel H(X)       = 9.0753   Ratio 1.0000
Channel H(Y)       = 0.0810   Ratio 0.0810
Channel H(X|Y)     = 8.9943   Ratio 0.9911
Channel H(Y|X)     = 0.0000   Ratio 0.0000
Channel H(X,Y)     = 9.0753   Ratio 0.9912
Channel I(X;Y)     = 0.0810   Ratio 1.0000
Variables - Relationship to output
Variable            H(X)      H(Y|X)     I(X;Y)    Importance
TBALNO              0.9859    0.0000     1.0000    1.0000
IHGHBAL             1.0045    0.0494     0.8506    0.8506
ICURBAL             0.8913    0.1444     0.8556    0.8556
RLIMIT              0.9862    0.1617     0.8383    0.8383
BCBAL               0.8777    0.2062     0.7938    0.7938
RBAL                0.9896    0.2568     0.7432    0.7432
DAS_C               0.9805    0.3041     0.6959    0.6959
FRONT_PROF          0.8610    0.3196     0.6804    0.6804
BCLIMIT             0.9178    0.3325     0.6675    0.6675
MOF                 0.9631    0.4265     0.5735    0.5735
HOME_INC            0.8504    0.4382     0.5618    0.5618
LST_R_OPEN          0.8765    0.4765     0.5235    0.5235
HOME_VALUE          0.8701    0.5387     0.4613    0.4613
BEACON_C            0.9598    0.5520     0.4480    0.4480
DOB_YEAR            0.8432    0.6218     0.3782    0.3782
HOME_ED             0.7365    0.7828     0.2172    0.2172
A_MVP_D1            0.6486    0.8402     0.1598    0.1598
A_MVP_D2            0.6486    0.8402     0.1598    0.1598
A_MVP_D3            0.6486    0.8402     0.1598    0.1598
YEARS_RES           0.8663    0.8589     0.1411    0.1411
RBALNO              0.7321    0.8700     0.1300    0.1300
FRONT_WHIT          0.7092    0.8739     0.1261    0.1261
ROPEN               0.7768    0.8812     0.1188    0.1188
TOPEN               0.7894    0.8874     0.1126    0.1126
EST_INC_C           0.8772    0.9431     0.0569    0.0569
AGE_INFERR          0.8443    0.9625     0.0375    0.0375
RBAL_LIMIT          0.5751    0.9463     0.0537    0.0537
CHILDREN            0.9828    0.9813     0.0187    0.0187
SEX_D1              0.2224    0.9888     0.0112    0.0112
SEX_D2              0.2224    0.9888     0.0112    0.0112
MARRIED             0.5488    0.9888     0.0112    0.0112
BUYER_P_            0.0801    0.0000     0.0000    0.0000
```

Figure 11.25 Information metrics for the CREDIT data set.

The reason, it turns out, is because this is the unmodified response data set with a less than 1% response rate. The fact is that if you guessed the state of a randomly selected record, you would be right more than 99% of the time by guessing that record referred to a nonbuyer. Not really much uncertainty about the output at all!

Many modeling techniques—neural networks or regression, for example—cannot deal with such low levels of response. In fact, very many methods have trouble with such low levels of response as this unless especially tuned to deal with it. However, since information metrics measure the nature of the manifold in state space, they are remarkably resistant to any distortion due to very low-density responses. Continuing to look at this data set, and later comparing it with a balanced version, demonstrates the point nicely.

With a very large data set, such as is used here, and a very low response rate, the rounding to four places of decimals, as reported in the information metrics, makes the ratio of cH(Y|X) appear to equal 0, and cI(X;Y) appears to be equal to 1. However, the state entropy map shows a different picture, which we will look at in a moment.

Complexity map This is an unusual, and really a rather nasty-looking, complexity map seen in Figure 11.26. The concave-shaped curve indicates that adding additional complexity to the model (starting with the simplest model on the right) gains little in predictability. It takes a really complex model, focusing closely on the details of the signals, to extract any meaningful determination of the output signals.

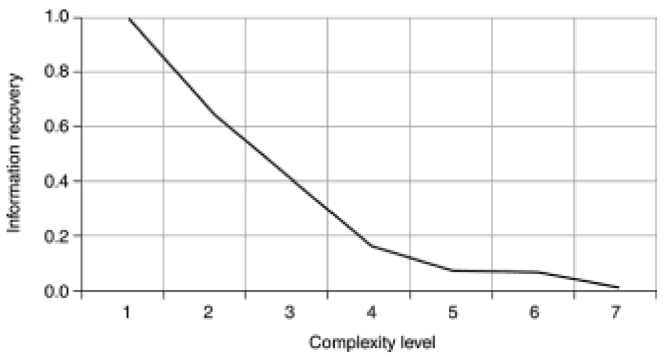


Figure 11.26 Complexity map for the CREDIT data set predicting BUYER. This curve indicated that the data set is likely to be very difficult to learn.

If this data set were the whole population, as with the CARS data set, there would be no problem. But here the situation is very different. As discussed in many places through the book (see, for example, Chapter 2), when a model becomes too complex or learns the structure of the data in too much detail, overtraining, or learning spurious patterns called noise, occurs. That is exactly the problem here. The steep curve on the left of the complexity map indicates that meaningful information is only captured with a high complexity model, and naturally, that is where the noise lies! The survey measures the amount of noise in a data set, and although a conceptual technical description cannot be covered here, it is worth looking at a noise map.

Noise Figure 11.27 shows the *information and noise map* for the CREDIT data set. The curve beginning at the top left (identical with that in Figure 11.26) shows how much information is recovered for a given level of complexity and is measured against the vertical scale shown on the left side of the map. The curve ending at the top right shows how much noise is captured for a given level of complexity and is measured against the vertical scale shown on the right side of the map.

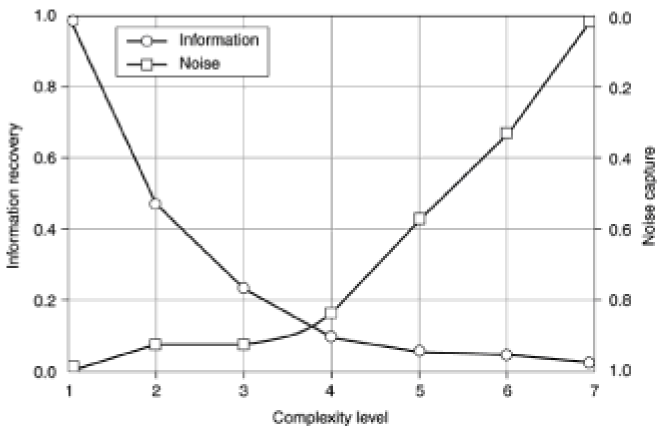


Figure 11.27 Information and noise map for the CREDIT data set.

The information capture curve and its interpretation are described above. Maximum complexity captures information uniquely defining each output state, so the curve starts at a level of 1 shown on the left scale. The noise curve starts at 1 too, but that is shown on the right scale. It indicates that the most complex model captures all of the noise present in the data set. This is very often the case for many data sets with the highest degree of complexity. Maximum complexity obviously captures all of the noise and often captures enough information to completely define the output signals within that specific data set.

At complexity level 2, the information capture curve has already fallen to about 0.5, showing that even a small step away from capturing all of the complexity in the data set loses much of the defining information about the output states. However, even though at this slightly reduced level of complexity much information about the output state is lost, the noise curve shows that any model still captures most of the noise! Noise capture falls from about 1.0 to about 0.95 (shown on the right scale). A model that captures most of the noise and little of the needed defining information is not going to be very accurate at predicting the output.

Complexity level 3 is even worse! The amount of noise captured is almost as much as before, which still amounts to almost all of the noise in the data set. While the amount of noise captured is still high, the amount of predictive information about the output has continued to fall precipitously! This is truly going to be one tough data set to get any decent model from!

By complexity level 4, the information capture curve shows that at this level of complexity, and on for all of the remaining levels too, there just isn’t much predictive information that can be extracted. The noise capture begins to diminish (the rising line indicates less noise), but even if there is less noise, there just isn’t much of the needed information that a relatively low-complexity model can capture.

By complexity level 7, although the noise capture is near 0 (right scale), the amount of information about the output is also near 0 (left scale).

No very accurate model is going to come of this. But if a model has to be built, what is the best level of complexity to use, and how good (or in this case, perhaps, bad) will that model be?

Optimal information capture points Given the noise and information characteristics at every complexity level shown in Figure 11.27, is it possible to determine how much noise-free information is actually available? Clearly the amount of noise-free information available isn't going to be much since the data set is so noisy. However, the curve in Figure 11.28 is interesting. At complexity level 1, to all intents and purposes, noise swamps any meaningful signal.

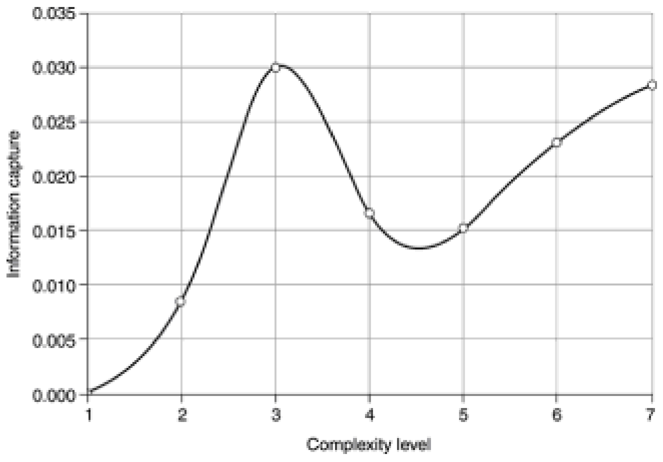


Figure 11.28 Information capture map showing the amount of noise-free information captured at different levels of complexity in the CREDIT data set.

Noise, of course, represents information patterns that are present in this specific data set, but not in any other data set or in the population. Since the noise map in the previous figure showed that perfect information about the output is available at level 1, for this specific data set the output can be perfectly learned. However, what the noise curve points out is that none, or essentially none, of the information relationships used to make these perfect predictions of the output state will be present in any other data set. So the noise map shows that there is almost no noise-free information available at level 1. (Although the graph does indeed appear to show 0 at level 1, it is in fact an infinitesimally small distance away from 0—so small that it is impossible to show graphically and is in any case of no practical use.)

By complexity level 3, the amount of noise-free information has risen to a maximum, although since the scale is in ratio entropy, it turns out to be precious little information! After that it falls a bit and rises back to nearly its previous level as the required model becomes less complex.

Unfortunately, Figure 11.28 has exaggerated the apparent amount of information capture by using a small scale to show the curve so that its features are more easily visible.

The maps shown and discussed so far were presented for ease of explanation. The most useful information map generally used in a survey combines the various maps just discussed into one composite map, as shown for this data set in Figure 11.29. This map and the state entropy map are the pair that a miner will mainly use to get an overview of the high-level information relationships in a data set. At first glance, Figure 11.29 may not appear so different from Figure 11.27, and indeed it is mainly the same. However, along the bottom is a low wavy line that represents the amount of available noise-free information. This is exactly the same curve that was examined in the last figure. Here it is shown to the same scale as the other curves. Clearly, there really isn't much noise-free information available in this data set! With so little information available, what should a miner do here? Give up? No, not at all!

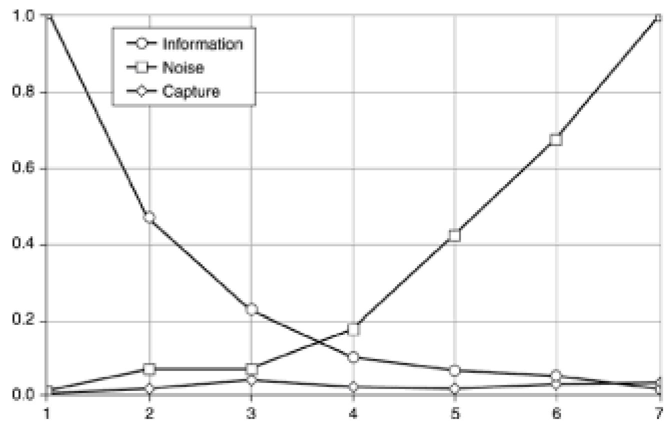


Figure 11.29 The information/noise/capture (INC) map is the easiest summary for a miner to work with. In this case it summarizes the information content, amount of noise captured, and noise-free information level into a single picture, and all at the same scale.

Recall that the original objective was to improve on a less than 1% response rate. The model doesn't seem to need much information to do that, and while there is little noise-free information available, perhaps noisy information will do. And in fact, of course, noisy information *will* do. Remember that the miner's job is *to solve a business problem*, not to build a perfect model! Using the survey and these maps allows a miner to (among other things) quickly estimate the chance that a model good enough to solve the business problem can actually be built. It may be surprising, but this map actually indicates that it very likely can be done.

Without going into the details, it is possible to estimate exactly how complex a model is needed to yield the best response for the problem at hand. It turns out that a model of complexity level 5.7 (approximately) is a good trade-off between speed, noise resistance, and improved accuracy for this application. Without regard to any other insights gained, or understanding of the data set that the survey yields, it can be determined that a model built to about a 5.7 level of complexity will capture enough information to make improved predictions of BUYER possible to a sufficient degree to have economic benefit.

When the model is built, it can be useful to see how much information has actually been captured. Surveying the modeled data *after* the model is built, together with the model predictions, can be used to measure how much information the model captured, if it has learned noise, and if so, how much noise—all useful information for the miner.

Not all information/noise/capture maps look like this one does. For comparison, Figure 11.30 shows a map for a different data set.

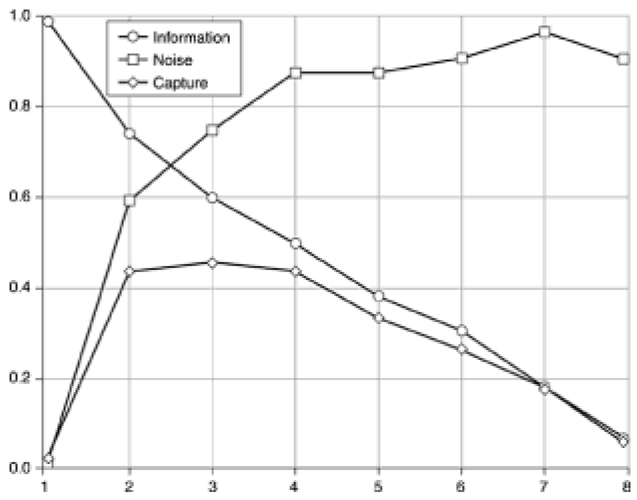


Figure 11.30 An INC map from a different data set. Far more noise-free information is available in this data set, and a complexity level 4 model looks to be a good choice. (Depending always, of course, on the exact nature of the business problem!)

State entropy map After having given so much attention to the complexity map, there is still the state entropy map for the CREDIT data set in Figure 11.31 that carries useful information. In spite of the apparent perfect predictions possible from the information enfolded in this data (shown in the information metrics **I**(X;Y) = 1 and cH(Y|X) = 0), the state entropy map tells a different tale. One of the two states has low entropy (uncertainty), the other high. It is only the minute proportion of uncertain states in the data set (less than 1%) that leads to the misleading entropic and mutual information measures shown in the information metrics. If the low cH(Y) isn't warning enough, the tilt shown in this map is a clear warning sign that further investigation is needed.

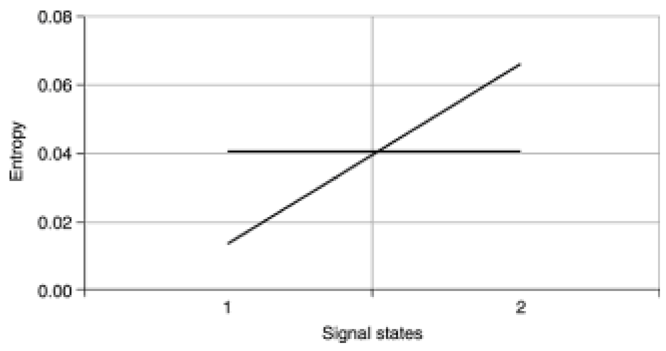
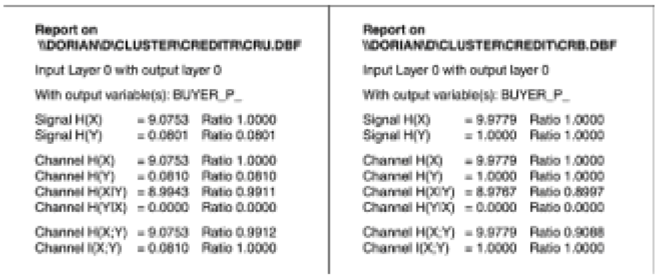


Figure 11.31 State entropy map for the CREDIT data set.

Entropy of the variables Refer back to Figure 11.25, which shows, in part, the entropy of the variables in the data set. Domain knowledge is really needed to produce a detailed interpretation of what this means, but it seems clear that the most important variables are those that deal with previous credit behavior. This reflects an insight, well known in the credit card industry, that people who already are users of credit are the most likely to take further credit. These variables are mainly shown for later comparison with the balanced data set metrics.

Interestingly, when this modeling project was actually in process, the information here was looked at in many different ways. One of the most revealing ways of examining the data in this case, so far as the success of the project went, was to look at the signals after removing the signal information about credit use. This is totally different from removing variables about credit activity and looking at what remains. It turns out that the variables that seem totally concerned exclusively with credit information also, in fact, carry other information. The advantage of working in the survey with signal states is that the systemic information (information describing the behavior of the system as a whole) can be manipulated—even when it comes from several variables. Even when spread across several variables, specific information can be disregarded, thus exposing the remaining interrelationships more clearly. This allowed a model to be built on the systemic information remaining after credit use information was disregarded. One of the useful features of this particular submodel was that it allowed the credit card company to target people who were migrating from low credit use to higher credit use. It is this ability to extract and manipulate information, not variables, that allows such powerful and effective models to be built. It also allows the survey to be used as a true information mining tool—digging into the information content to see what is there.

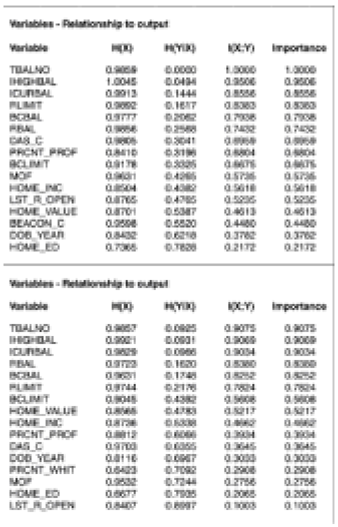
Balancing the data set Creating a balanced data set was discussed in Chapter 10. Figure 11.32 shows the information metrics side by side for the unbalanced (shown on the left) and balanced (shown on the right) data sets. The main difference appears to be only in sH(Y) and cH(Y), which is to be expected as that is what is being balanced.



```
Report on                                   Report on
\\DORIAN\D\CLUSTER\CREDITR\CRU.DBF          \\DORIAN\D\CLUSTER\CREDIT\CRB.DBF

Input Layer 0 with output layer 0           Input Layer 0 with output layer 0

With output variable(s): BUYER_P_           With output variable(s): BUYER_P_

Signal H(X)      = 9.0753  Ratio 1.0000     Signal H(X)      = 9.9779  Ratio 1.0000
Signal H(Y)      = 0.0801  Ratio 0.0801     Signal H(Y)      = 1.0000  Ratio 1.0000

Channel H(X)     = 9.0753  Ratio 1.0000     Channel H(X)     = 9.9779  Ratio 1.0000
Channel H(Y)     = 0.0810  Ratio 0.0810     Channel H(Y)     = 1.0000  Ratio 1.0000
Channel H(X|Y)   = 8.9943  Ratio 0.9911     Channel H(X|Y)   = 8.9767  Ratio 0.8997
Channel H(Y|X)   = 0.0000  Ratio 0.0000     Channel H(Y|X)   = 0.0000  Ratio 0.0000

Channel H(X;Y)   = 9.0753  Ratio 0.9912     Channel H(X;Y)   = 9.9779  Ratio 0.9088
Channel I(X;Y)   = 0.0810  Ratio 1.0000     Channel I(X;Y)   = 1.0000  Ratio 1.0000
```

Figure 11.32 Information metrics for the unbalanced CREDIT data set on the left, and the balanced CREDIT data set on the right. The unbalanced data set has less than 1% buyers, while the balanced data set has 50% buyers.

Figure 11.33 contrasts the entropy of the most important variables for the original unbalanced data set with a newly constructed, balanced data set. Briefly, the balanced data set was constructed from two data subsets, both representative in themselves, save only that one subset contained all buyers, the other all nonbuyers. When merged, these two subsets were as representative of the population as possible except that one contained the original density of buyers (1%) and the other was balanced to have (in this case) 50% buyers and 50% nonbuyers.



Variables - Relationship to output

Variable	H(X)	H(Y|X)	I(X;Y)	Importance
TBALNO	0.9859	0.0000	1.0000	1.0000
IHIGHBAL	1.0045	0.0494	0.9506	0.9506
ICURBAL	0.9913	0.1444	0.8556	0.8556
RLIMIT	0.9892	0.1617	0.8383	0.8383
BCBAL	0.9777	0.2062	0.7938	0.7938
RBAL	0.9856	0.2568	0.7432	0.7432
DAS_C	0.9805	0.3041	0.6959	0.6959
PRCNT_PROF	0.8410	0.3196	0.6804	0.6804
BCLIMIT	0.9178	0.3325	0.6675	0.6675
MOF	0.9631	0.4265	0.5735	0.5735
HOME_INC	0.8504	0.4382	0.5618	0.5618
LST_R_OPEN	0.8765	0.4765	0.5235	0.5235
HOME_VALUE	0.8701	0.5387	0.4613	0.4613
BEACON_C	0.9598	0.5520	0.4480	0.4480
DOB_YEAR	0.8432	0.6218	0.3782	0.3782
HOME_ED	0.7365	0.7828	0.2172	0.2172

Variables - Relationship to output

Variable	H(X)	H(Y|X)	I(X;Y)	Importance
TBALNO	0.9857	0.0925	0.9075	0.9075
IHIGHBAL	0.9921	0.0931	0.9069	0.9069
ICURBAL	0.9829	0.0966	0.9034	0.9034
RBAL	0.9723	0.1620	0.8380	0.8380
BCBAL	0.9631	0.1748	0.8252	0.8252
RLIMIT	0.9744	0.2176	0.7824	0.7824
BCLIMIT	0.9045	0.4392	0.5608	0.5608
HOME_VALUE	0.8565	0.4783	0.5217	0.5217
HOME_INC	0.8736	0.5338	0.4662	0.4662
PRCNT_PROF	0.8812	0.6066	0.3934	0.3934
DAS_C	0.9703	0.6355	0.3645	0.3645
DOB_YEAR	0.8116	0.6967	0.3033	0.3033
PRCNT_WHIT	0.6423	0.7092	0.2908	0.2908
MOF	0.9532	0.7244	0.2756	0.2756
HOME_ED	0.6677	0.7935	0.2065	0.2065
LST_R_OPEN	0.8407	0.8997	0.1003	0.1003

Figure 11.33 The top 16 variables for the CREDIT data set when unbalanced (top) and balanced (bottom). There is little effect from the balancing on the information content or the ordering of these variables between the two data sets.

These figures show that, while the ratio of buyers to nonbuyers in the two data sets is dramatically different, the channel entropy, and the information carried by the balanced data set about buyers, is not markedly affected. This illustrates just how resistant signal information analysis is to such changes as occurred between these two data sets.

Remarkably, it seems that buyers and nonbuyers can still be perfectly separated in both data sets (cH(Y|X), I(X;Y))!

Figure 11.34 contrasts the maps for the two data sets. Obviously, the state entropy map is totally different, since that is what was balanced in the second data set. It is so balanced that the two lines—average entropy and state entropy for the two states—both fall directly on top of each other. More unexpected, perhaps, is that the complexity curve is almost unaffected. In spite of the balancing, information content and complexity levels are almost untouched as shown by the two complexity maps. In fact, if anything, the balanced data set has an even more concave curve.

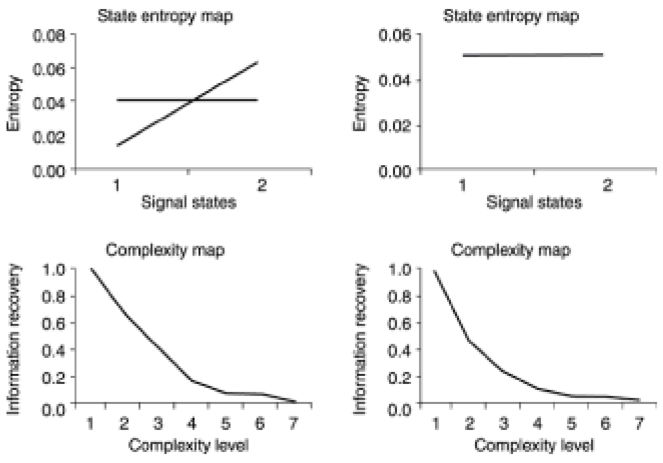


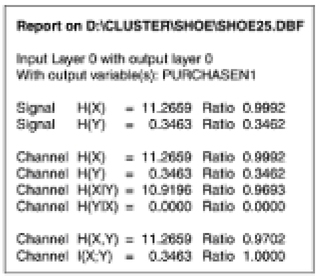
Figure 11.34 Comparing the state entropy and complexity maps for the unbalanced (left) and balanced (right) CREDIT data sets.

Noise level remains almost unchanged too (not actually shown here), so the information metrics of the data survey report that the information content is almost unchanged for the two data sets, even though the balance of the data is completely different between them. In other words, even though the balance of the data sets is changed, it is just as difficult to build a predictive model from either data set. This is a remarkable and powerful feature of the data survey. The information structure of the manifold is not unduly distorted by changing the balance of the data, so long as there is enough data to establish a representative manifold in state space in the first place. Balancing the data set, if indeed the modeler chooses to do so, is for the convenience of some modeling tools for which it makes a difference (say, neural nets). It makes little or no difference to the information content.

The SHOE Data Set

This data set was collected by a nationally known manufacturer of running shoes about the behavior of buyers of their shoes in various shoe stores around the country. They had in place a frequent buyer program that they hoped would spur sales. From the collected data they hoped to be able to predict and target those customers who fit the profile as potential members in their program. This survey looks at the data from the perspective of discovering if the desired prediction is likely to be of value.

General Comments Figure 11.35 shows many of the features discussed previously, and much should be recognizable. Rather than look at each item individually, a general look at this data set should suffice.



```
Report on D:\CLUSTER\SHOE\SHOE25.DBF

Input Layer 0 with output layer 0
With output variable(s): PURCHASEN1

Signal   H(X)   = 11.2659  Ratio 0.9992
Signal   H(Y)   =  0.3463  Ratio 0.3462

Channel  H(X)   = 11.2659  Ratio 0.9992
Channel  H(Y)   =  0.3463  Ratio 0.3462
Channel  H(X|Y) = 10.9196  Ratio 0.9693
Channel  H(Y|X) =  0.0000  Ratio 0.0000

Channel  H(X,Y) = 11.2659  Ratio 0.9702
Channel  I(X;Y) =  0.3463  Ratio 1.0000
```

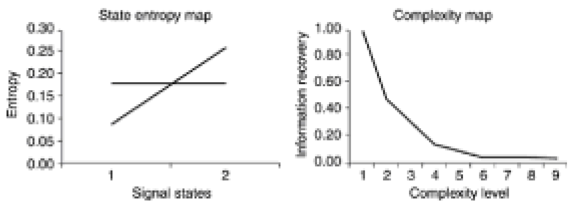
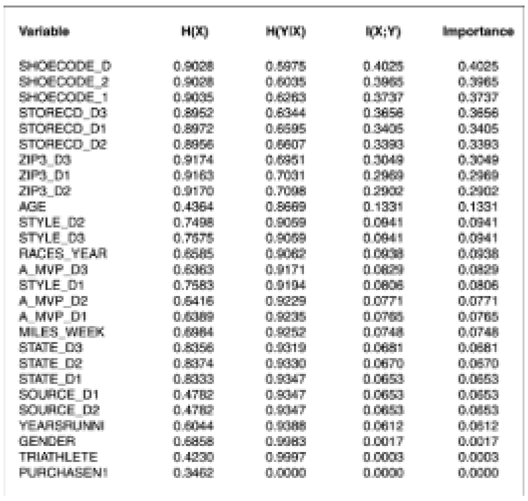


Figure 11.35 Information metrics, state entropy map, and complexity map for the SHOE data set.

The low, although not vanishingly low, entropy of cH(Y) is suspect. The state entropy map is another clue that something might be askew here. The complexity map has a very nasty curve. This is going to be a difficult data set from which to build a model, although what is required of the model depends on the business objective. Recall that although the CREDIT data set did not produce a good model, it generated a very high return.

Figure 11.36 reveals a very different problem, especially when considering the previous metrics. One problem here is that the information being used to predict the frequent buyer plan response is simply not well coupled to the data set as a whole. Although the coupling ratio itself is not shown, the information is clearly carried in few of the variables—namely shoecode and storecode. (There are several dimensions of these categorical variables, three for shoecode and two for storecode.) The problem is that these are not well coupled to any of the other variables.



Variable	H(X)	H(Y|X)	I(X;Y)	Importance
SHOECODE_D	0.9028	0.5975	0.4025	0.4025
SHOECODE_2	0.9028	0.6035	0.3965	0.3965
SHOECODE_1	0.9035	0.6263	0.3737	0.3737
STORECD_D3	0.8952	0.6344	0.3656	0.3656
STORECD_D1	0.8972	0.6595	0.3405	0.3405
STORECD_D2	0.8956	0.6607	0.3393	0.3393
ZIP3_D3	0.9174	0.6951	0.3049	0.3049
ZIP3_D1	0.9163	0.7031	0.2969	0.2969
ZIP3_D2	0.9170	0.7098	0.2902	0.2902
AGE	0.4364	0.8669	0.1331	0.1331
STYLE_D2	0.7498	0.9059	0.0941	0.0941
STYLE_D3	0.7575	0.9059	0.0941	0.0941
RACES_YEAR	0.6585	0.9062	0.0938	0.0938
A_MVP_D3	0.6363	0.9171	0.0829	0.0829
STYLE_D1	0.7583	0.9194	0.0806	0.0806
A_MVP_D2	0.6416	0.9229	0.0771	0.0771
A_MVP_D1	0.6389	0.9235	0.0765	0.0765
MILES_WEEK	0.6984	0.9252	0.0748	0.0748
STATE_D3	0.8356	0.9319	0.0681	0.0681
STATE_D2	0.8374	0.9330	0.0670	0.0670
STATE_D1	0.8333	0.9347	0.0653	0.0653
SOURCE_D1	0.4782	0.9347	0.0653	0.0653
SOURCE_D2	0.4782	0.9347	0.0653	0.0653
YEARSRUNNI	0.6044	0.9388	0.0612	0.0612
GENDER	0.6858	0.9983	0.0017	0.0017
TRIATHLETE	0.4230	0.9997	0.0003	0.0003
PURCHASEN1	0.3462	0.0000	0.0000	0.0000

Figure 11.36 Variable information metrics for the running SHOE data set.

In practice, when a model was built it seemed to be moderately effective for independent test and verification data sets. (Good "lift" was generated.) However, it simply was not applicable in the real world. Why? The answer is that shoecodes vary constantly because new shoe styles are introduced frequently. People who buy certain types of running shoes join the frequent buyer program, and that's about it. As new shoecodes are introduced, the model, never having seen those shoe codes before, failed to make valid predictions, and so failed in the real world. (It is for exactly this sort of situation that a type of model known as a self-adaptive model should be used.)

Chapter 12: Using Prepared Data

Overview

So what’s the benefit of using prepared data to build models? You get more effective models faster. Most of the book so far has described the role of data preparation and how to properly prepare data for modeling. This chapter takes a brief look at the effects and benefits of using prepared data for modeling.

Actually, to examine the preparation results a little more closely, quite often the prepared data models are both better and produced faster. However, sometimes the models are of better quality, but produced in the same time as with unprepared data. Sometimes they are no better but produced a lot faster. For a given data set, both the quality of the model and the speed of modeling depend on the algorithm used and the particular implementation of that algorithm. However, it is almost invariably the case that when data is prepared in the manner described in this book, using the prepared data results in either a better model, a faster model, or a better model faster than when using unprepared data—or than when using data inadequately or improperly prepared. Can this statement be justified?

One credit card company spent more than three months building a predictive model for an acquisition program that generated a response rate of 0.9% over a 2,000,000-piece mailing. Shortly thereafter, using essentially the same modeling tools and data, the company launched another campaign using a model constructed from prepared data. Response rate in this model was 1.23%—a more than 36% response improvement. Was this particularly significant to the credit card company? That 36% improvement comes straight off the cost of acquisition. In this case, data preparation translated into 6,522 more customers. Another way of looking at it is that in the first program, the company paid about $138 to acquire a new customer. Data preparation reduced this cost to about $101 per acquisition.

A large commercial baker servicing nearly 100 stores sought to increase profitability by focusing attention on under- or oversupply of products, that is, not having enough, or having too many, bagels, croissants, and other “morning goods” on the shelves. (Shelf life is the day of manufacture only.) Insufficient products mean lost sales; too many products means waste since they have to be discarded. The baker used both statistical models and data mining to estimate demand and appropriate inventory levels. Rebuilding existing models with prepared data produced models that were better than 8% more accurate than the previous models. This immediately translated into saving about $25 per store per day in shortage and overage losses. What seems like a tiny saving per store comes to nearly $1 million saved over all the stores in a year—and that was before improved models were built.

A financial institution modeled trading data over many markets and combinations of markets, all of which produced different types of data. Each required from two to four weeks to build an optimized model of each market. Each successful model contributed to trading success during the life of the model, which could be quite short (from days to weeks after deployment). By using automated data preparation techniques they produced models of equivalent quality in hours or days, thus getting more models of more markets deployed more quickly.

An industrial controls manufacturer developed a printing press color process optimizer. When a multicolor press starts its print run, and until the press is at operating temperature and speed (which can take a lot of time), the press is producing such inferior-quality print that it is waste. Paper and ink are very expensive, and reducing the waste is a high priority (and gives a high payback). The manufacturer’s control measured performance characteristics and implemented a model that controlled the ink flow rates, paper feed rate, and manipulated other press controls. The automated model, when built conventionally, cut the run-up time to half of what it was without the control. Using prepared data to build the model produced an improvement that reduced the waste run time by an additional 10%.

Data preparation techniques as described in this book have been used to improve modeling results in traffic pattern analysis in a major U.S. city, molecular structure analysis in the research department of a major pharmaceutical company, and in feature detection for medical image processing. They have been used to enhance severe weather detection in meteorological data and to reduce consumable product returns in the confectionery industry. In every case, data preparation alone—that is, just the application of the techniques described in this book—has yielded from small to moderate but in all cases a real and significant improvement. In one stunning case (see Chapter 10), a brokerage data set that was essentially useless before preparation was very profitably modeled after preparation.

Preparing data for mining uses techniques that produce better models faster. But the bottom line—the whole reason for using the best data preparation available—is that data preparation adds value to the business objective that the miner is addressing.

12.1 Modeling Data

Before examining the effect the data preparation techniques discussed in this book have on modeling, a quick look at how modeling tools operate to extract a model from data will provide a frame of reference.

12.1.1 Assumptions

Modeling of any data set is based on five key assumptions. They are worth reviewing since if any of them do not hold, no model will reflect the real world, except by luck! The key assumptions are

1. Measurements of features of the world represent something real about the world.

2. Some persistent relationship exists between the features measured and the measurements taken.

3. Relationships between real-world features are reflected as relationships between measurements.

4. Understanding relationships between measurements can be applied to understanding relationships between real-world features.

5. Understanding relationships between real-world features can be used to influence events.

In other words, data reflects and connects to the world so that understanding data and its relationships contributes to an understanding of the world. When building a model, the modeler must ask if, in this particular case, these assumptions hold. (In several places throughout the book, there are examples of failed models. In every case, if the modeling itself was valid, the failure was that the modeler failed to check one or all of these five assumptions.)

12.1.2 Models

It's fine to say that a modeler builds a model, but what actually is a model? A *model*, in a general sense, is a replica of some other object that duplicates selected features of that larger object, but in a more convenient form. A plastic model World War II battleship, for instance, models the external appearance of the original to some reduced scale and is far more convenient for displaying in a living room than the original! Small-scale aircraft, made from a material that is much too heavy to allow them to fly, are useful for studying airflow around the aircraft in a wind tunnel. In hydrographic research, model ships sail model seas, through model waves propelled by model winds.

In some way, all models replicate some useful features of the original so that those features themselves, singled out from all other features, can be studied and manipulated. The nonphysical models that the data miner deals with are still models in the sense that they reflect useful features of the original objects in some more-convenient-to-manipulate way. These are symbolic models in which the various features are represented by symbols. Each symbol has a specific set of rules that indicate how the symbol can be manipulated with reference to other symbols. The symbolic manipulators used today are usually digital computers. The symbols consist of a mixture of mathematical and procedural structures that describe the relationships between, and operations permitted on, the symbols that comprise the model itself.

Typically, data miners (and engineers, mathematicians, economists, and statisticians, too) construct models by using symbols and rules for manipulating those symbols. These models are active creations in the sense that they can be computationally manipulated to answer questions posed about the model's behavior—and thus, by extension, about the behavior of the real world.

But data miners and statisticians differ from economists, engineers, and scientists in the way that they construct their models. And indeed, statisticians and data miners differ from each other, too. Engineers, scientists, and economists tend to form theories about the behavior of objects in the world, and then use the language of symbols to express their appreciation of the interrelationships that they propose. Manipulating the model of the proposed behavior allows them to determine how well (or badly) the proposed explanation "works." So far as modeling goes, data miners and statisticians tend to start with fewer preconceived notions of how the world works, but to start instead with data and ask what phenomenon or phenomena the data might describe. However, even then, statisticians and data miners still have different philosophical approaches to modeling from data.

12.1.3 Data Mining vs. Exploratory Data Analysis

Exploratory data analysis (EDA) is a statistical practice that involves using a wide variety of single-variable and multivariable analysis techniques to search for the underlying systemic relationships between variables. Data mining shares much of the methodology and structure of EDA, including some EDA techniques. But EDA focuses on discovering the basic nature of the underlying phenomena; data mining focuses tightly on the practical application of results. This difference, while it may seem narrow, is actually very broad.

Data miners are perfectly happy to accept a proven working "black box" that is known to work under a specified range of conditions and circumstances. The essential nature of the underlying phenomenon is not of particular interest to a miner, so long as the results are reliable, robust, and applicable in the real world to solve an identified problem. Statisticians, using EDA, want to be able to "explain" the data (and the world). How does this difference in approach turn out in practice?

A major credit card issuer had a large staff of statistical modelers. Their job was to investigate data and build models to discover "interesting" interactions in the data. Their approach was to take statistically meaningful samples from the credit card issuer's voluminous data set and look for proposed interesting interactions between the variables in the data set. Some of the proposed interactions were found to be statistically justified, and were further proposed for possible marketing action. Other interactions were found to be statistically insignificant, and no further action was taken on these.

The models produced by the statisticians were working well, but the vice president of marketing wanted to try data mining, a new discipline to her, but one that had received good reviews in the business press. How did the data mining project differ from the statistical approach?

Data mining did not start with a sample of the data. It also did not start with proposed interactions that might or might not be supported by the data. Instead miners started with the business problem by identifying areas that the marketing VP defined as strategically important for the company.

Next, miners selected data from the credit card company's resources that seemed most likely to address issues of strategic importance, guided in this by the marketers as domain experts. The mining tools surveyed all of the selected data available, which amounted to many gigabytes, including many hundreds of variables. One data set was built from credit card transaction records that were fully assayed, reverse pivoted, and transformed into consumer activity by account. A data survey revealed many possibly interesting areas for exploration.

Instead of approaching the data with ideas to test, the miners used the data survey to extract from the data a moderately comprehensive set of all the interactions that the data could possibly support. Many of these interactions were, naturally, noise-level interactions, only fit to be discarded. However, a key point here is that the miners asked the data for a comprehensive list of interactions that could possibly be supported by that data. The statisticians asked the data what level of support was available for one, or a few, predefined ideas that they had developed.

With a list of candidate interactions generated, the miners reviewed the list with the domain experts (marketers) to determine which might, if they proved successful, be the best and most effective possibilities.

Given the reverse pivot, both the statistical staff and the data miners attacked the data set with their respective tools. The statisticians cut off their samples and identified broad market segments that differentiated credit card usage. The data miners set their tools to mining all the data, extracting both broad and narrow fluctuations. The main search criteria for the data miners was to find the "drivers" for particularly profitable groups and subgroups in the data (inferential modeling); they then had to design models that would predict who those people were (predictive modeling).

The statistical analysts built regression models mainly, carefully examining the "beta weights" for linear and nonlinear regressions; analyzing residuals, confidence intervals, and many other items in the models; and translating them back into English to explain what was happening in the data. Because they were working with samples, they were able to build only high-level models of general trends.

The data miners used several techniques, the three principal ones being rule induction, decision trees, and neural networks. The decision tree proved to be the most useful on this occasion. It has the ability, with or without guidance, to automatically find the gross structure in the data and extract increasingly finer structure as more and more data is examined. In this case, several very valuable insights were waiting in the fine structure.

As is so often the case, one insight was fairly obvious with hindsight. It turned out that 0.1% of the accounts showed a pattern of exceptionally high profitability. This subsegment was identified as about 30% of all people buying ski equipment valued at $3000 in a 30-day period. These people also went on to buy travel packages, presumably to ski resorts, well in excess of an additional $3000. High profitability indeed.

The insight was used to build a marketing offer rolling ski equipment purchases, travel packages, and other benefits and services into a discounted package that was triggered by any purchase of ski equipment of appropriate value. A separate program was also built on the same insight: a “lifestyle” credit card offering the cardholder permanent packages and point accumulation toward ski vacations.

When appropriately targeted, this structured offer produced spectacular returns. The figures look like this: the identified segment was 0.1% of 2.5 million people’s accounts generated from the reverse pivot, or about 2500 people. These 2500 people were about 30% of the subpopulation, so the whole subpopulation was about 8300 people. Roughly 40% of the remaining subpopulation responded to the marketing offer. Forty percent of 8300 people—that is, 3300 additional people—purchased travel packages worth $3000 or more. These 3300 customers, by increasing their indebtedness by $3000 each, produced an increase in loan inventory of about $10 million!

A significant point: in the example, a fluctuation of about 0.1% would almost certainly be missed by statistical sampling techniques. This figure is technically “statistically insignificant” when considered as a fluctuation in a population. But it is far from insignificant in a commercial sense. Thus it is that the philosophical approach taken by EDA and data miners is quite different.

12.2 Characterizing Data

Before looking at the effect of data preparation on modeling, we need to look at what modeling itself actually does with and to data. So far, modeling was described as finding relationships in data, and making predictions about unknown outcomes from data describing the situation at hand. The fundamental methods used by data miners are easily outlined, although the actual implementation of these simple outlines may require powerful, sophisticated, and complex algorithms to make them work effectively in practice.

The general idea of most data mining algorithms is to separate the instance values in state space into areas that share features in common. What does this look like? Figure

12.1 shows a sample two-dimensional state space containing a few instance values. Simply looking at the illustrated state space seems to reveal patterns. This is hardly surprising, as one of the most formidably powerful pattern recognition devices known is the human brain. So powerful is the brain that it tends to find patterns even where objectively none are known to exist. Thus, just looking at the illustration of state space seems to reveal groups of points representing the instance values in the state space. A modeling tool has the task of separating and grouping these points in meaningful ways. Each modeling algorithm takes a slightly different approach. With a pencil, or in some other way, you could quite easily mark what seem to be "natural" groups, clumps, or clusters. But how do data mining algorithms, which cannot look at state space, go about finding the associations and related points?

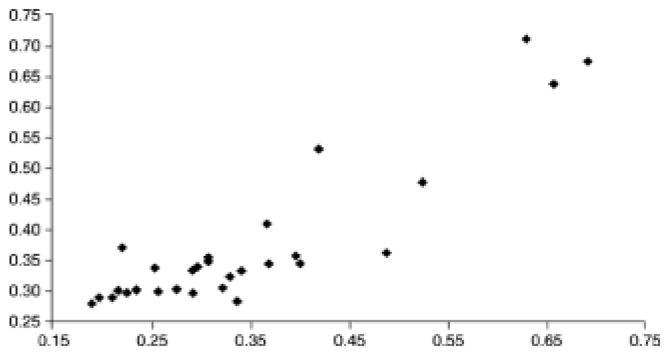


Figure 12.1 Instance values filling part of state space. Some patterns may be intuitively evident in the way the instance values fill the space. The axes show the range of values for each input variable.

12.2.1 Decision Trees

Decision trees partition state space such that the points in each separated partition have the maximum difference in some selected feature. Partitioning proceeds by selecting a partitioning point in the values of a single variable. Each of the partitions is then separately partitioned on the most appropriate partitioning variable and point for that partition. It continues until some stopping criterion is reached, or until each partition contains only a single point, or when it is no longer possible to separate the points.

The partition criteria can be expressed in the form of rules. In Figure 12.2, the partition shown in the upper-right partition is covered (explained) by a rule like "IF X > 0.5 AND Y > 0.51 THEN . . . ." The figure shows that the result of the partitioning is to create boundaries that are parallel to the dimensions of the space. Each axis is treated separately, resulting in a mosaic of boxes, each including some part of state space. Each partition covers some part of the whole space, and the whole of the space is covered by some partition. This comprehensive coverage is an important point.

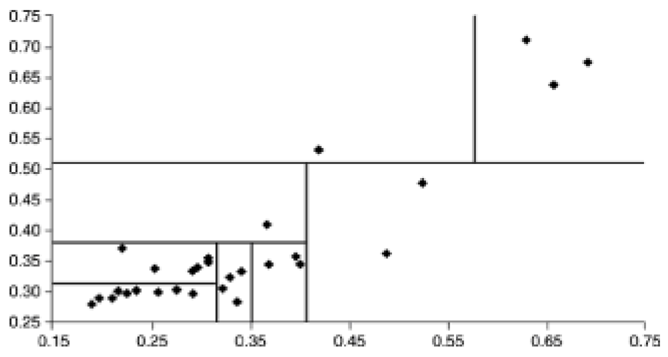


Figure 12.2 State space divided by a decision tree.

12.2.2 Clusters

Clustering also partitions state space, separating areas that have points sharing a common feature. This sounds similar to decision trees; however, there are marked differences. There are many different methods of clustering, but the end result of a clustering algorithm can be thought of as producing "clouds" in state space. The clouds, as shown in Figure 12.3, have marked differences from decision trees. One major difference is that they don't have linear boundaries parallel to the axes! The nonlinearity makes it very difficult to extract seemingly simple rules from clusters. However, expressing what clusters have in common—their ranges on various axes—can produce meaningful statements about the clusters.

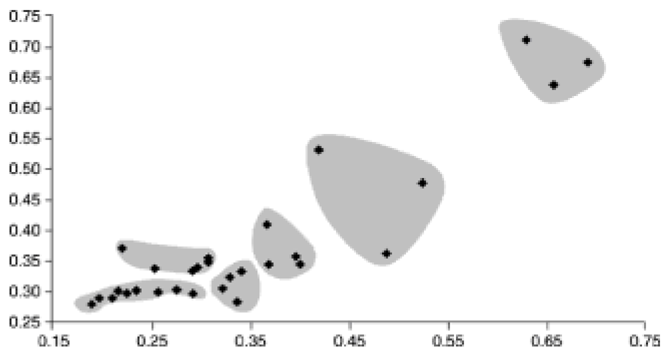


Figure 12.3 A clustering algorithm identifies groups, or clusters, of points that are associated in some way. The cluster areas cover all of the points in space, but do not necessarily cover the whole of the state space.

Another difference is that quite clearly the clustering algorithm used to produce the clusters shown does not cover the whole of state space. If the model is applied to a prediction data set that happens to have instance values defining a point outside a cluster boundary, the best that can be done is to estimate the nearest cluster. With such methods of clustering, the area outside the defined clusters is not itself a cluster, but more like undefined space. Other clustering methods, such as sequential natural clustering (briefly mentioned in Chapter 11), do produce clusters that have some cluster membership for every point in state space. In all cases, however, the clusters have irregular boundaries.

12.2.3 Nearest Neighbor

The way that nearest neighbors are used to describe interactions in state space is described in detail in Chapter 6. Very briefly, nearest-neighbor methods select some specific number of neighbors, and for every point use that number of nearest neighbors. Some averaging of the state of the neighbors for that point represents the point whose features are to be inferred or predicted. Figure 12.4 illustrates how neighbors might be selected. The figure illustrates the use of four nearest neighbors in any direction. For each of the selected points L, M, and N, the four closest points are used to estimate the characteristics of the selected point. Points L and M are characterized by four asymmetrically distributed points, which may not actually be the most representative points to choose.

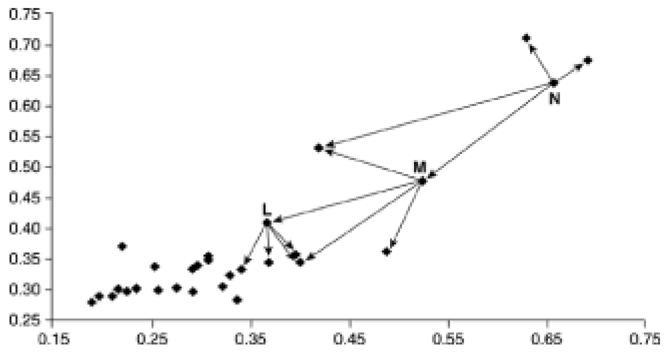


Figure 12.4 Principles of nearest-neighbor methods.

Nearest-neighbor methods have strengths and weaknesses. In the figure, some of the points use relatively distant nearest neighbors while others use closer neighbors. Another possible problem occurs when the nearest neighbors are not representatively distributed—points L and M in the figure show neighbors asymmetrically distributed. There are various methods for tuning nearest-neighbor algorithms to optimize for particular requirements, some of which are mentioned in Chapter 6.

12.2.4 Neural Networks and Regression

Figure 12.5 shows how manifold fitting methods characterize data. Neural networks, and how they characterize data, are discussed in Chapter 10. Linear regression (a basic statistical technique) fits a stiff, flat manifold to the data in some “best fit” way. In two dimensions a flat, stiff manifold is a straight line.

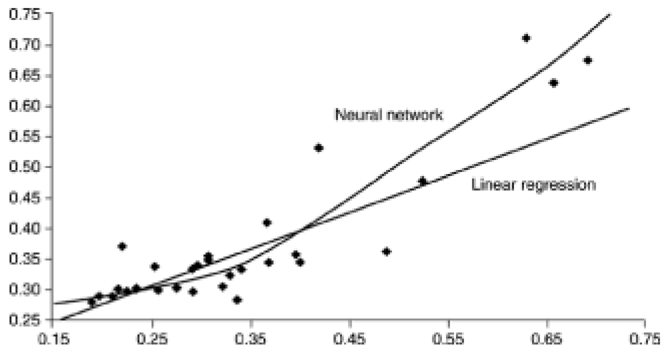


Figure 12.5 Fitting manifolds—either inflexible (linear regression) or flexible (neural network)—to the sample data results in a manifold that in some sense “best fits” the data.

These methods work by creating a mathematical expression that characterizes the state of the fitted line at any point along the line. Studying the nature of the manifold leads to inferences about the data. When predicting values for some particular point, linear regression uses the closest point on the manifold to the particular point to be predicted. The characteristics (value of the feature to predict) of the nearby point on the manifold are used as the desired prediction.

12.3 Prepared Data and Modeling Algorithms

These capsule descriptions review how some of the main modeling algorithms deal with data. The exact problems that working with unprepared data presents for modeling tools will not be reiterated here as they are covered extensively in almost every chapter in this book. The small, example data set has no missing values—if it had, they could not have been plotted. But how does data preparation change the nature of the data?

The whole idea, of course, is to give the modeling tools as easy a time as possible when working with the data. When the data is easy to model, better models come out faster, which is the technical purpose of data preparation. How does data preparation make the data easier to work with? Essentially, data preparation removes many of the problems. This brief look is not intended to catalog all of the features and benefits of correct data preparation, but to give a feel for how it affects modeling.

Consider the neural network—for example, as shown in Figure 12.5—fitting a flexible manifold to data. One of the problems is that the data points are closer together (higher density) in the lower-left part of illustrated state space, and far less dense in the upper right. Not only must a curve be fitted, but the flexibility of the manifold needs to be different in each part of the space. Or again, clustering has to fit cluster boundaries through the higher density, possibly being forced by proximity and the stiffness of the boundary, to include points that should otherwise be excluded. Or again, in the nearest-neighbor methods, neighborhoods were unbalanced.

How does preparation help? Figure 12.6 shows the data range normalized in state space on the left. The data with both range and distribution normalized is shown on the right. The range-normalized and redistributed space is a “toy” representation of what full data preparation accomplishes. This data is much easier to characterize—manifolds are more easily fitted, cluster boundaries are more easily found, neighbors are more neighborly. The data is simply easier to access and work with. But what real difference does it make?

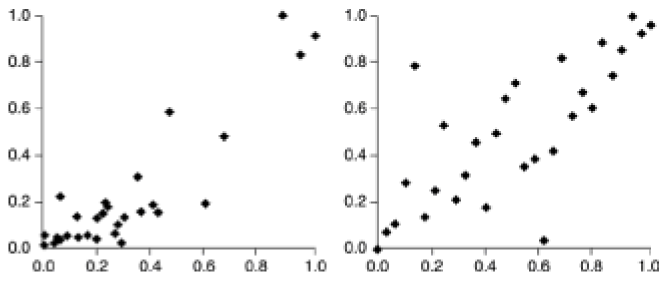


Figure 12.6 Some of the effects of data preparation: normalization of data range (left), and normalization and redistribution of data set (right).

12.3.1 Neural Networks and the CREDIT Data Set

The CREDIT data set is a derived extract from a real-world data set. Full data preparation and surveying enable the miner to build reasonable models—reasonable in terms of addressing the business objective. But what does data preparation alone achieve in this data set? In order to demonstrate that, we will look at two models of the data—one on prepared data, and the other on unprepared data.

Any difficulty in showing the effect of preparation alone is due to the fact that with ingenuity, much better models can be built with the prepared data in many circumstances than with the data unprepared. All this demonstrates, however, is the ingenuity of the miner! To try to “level the playing field,” as it were, for this example the neural network models will use all of the inputs, have the same number of nodes in the hidden layer, and will use no extracted features. There is no change in network architecture for the prepared and unprepared data sets. Thus, this uses no knowledge gleaned from the either the data assay or the data survey. Much, if not most, of the useful information discovered about the data set, and how to build better models, is simply discarded so that the effect of the automated techniques is most easily seen. The difference between the “unprepared” and “prepared” data sets is, as nearly as can be, only that provided by the automated preparation—accomplished by the demonstration code.

Now, it is true that a neural network cannot take the data from the CREDIT data set in its raw form, so some preparation must be done. Strictly speaking, then, there is no such thing—for a neural network—as modeling unprepared data. What then is a fair preparation method to compare with the method outlined in this book?

StatSoft is a leading maker of statistical analysis software. Their tools reflect statistical state-of-the art techniques. In addition to a statistical analysis package, StatSoft makes a neural network tool that uses statistical techniques to prepare data for the neural network. Their data preparation is automated and invisible to the modeler using their neural network package. So the "unprepared" data in this comparison is actually prepared by the statistical preparation techniques implemented by StatSoft. The "prepared" data set is prepared using the techniques discussed in this book. Naturally, a miner using all of the knowledge and insights gleaned from the data using the techniques described in the preceding chapters should—using either preparation technique—be able to make a far better model than that produced by this na‹ve approach. The object is to attempt a direct fair comparison to see the value of the automated data preparation techniques described here, if any.

As shown in Figure 12.7, the neural network architecture selected takes all of the inputs, passes them to six nodes in the hidden layer, and has one output to predict—BUYER. Both networks were trained for 250 epochs. Because this is a neural network, the data set was balanced to be a 50/50 mix of buyers and nonbuyers.

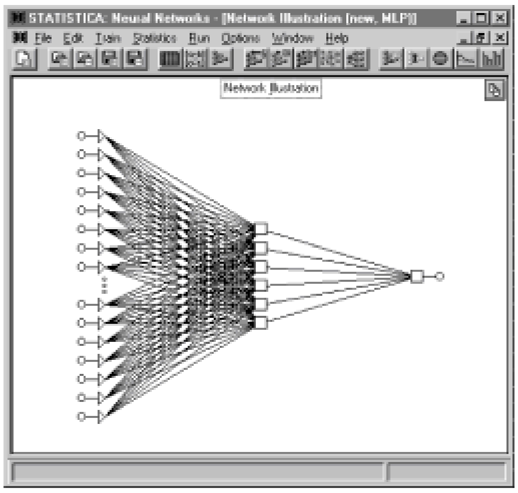

STATISTICA: Neural Networks - [Network Illustration (new, MLP)]
File Edit Train Statistics Run Options Window Help
Network Illustration

Figure 12.7 Architecture of the neural network used in modeling both the prepared and unprepared versions of the CREDIT data set predicting BUYER. It is an all-input, six-hidden-node, one-output, standard back-propagation neural network.

Figure 12.8 shows the result of training on the unprepared data. The figure shows a number of interesting features. To facilitate training, the instances were separated into training and verification (test) data sets. The network was trained on the training data set, and errors in both the training and verification data sets are shown in the “Training Error Graph” window. This graph shows the prediction errors made in the training set on which the network learned, and also shows the prediction errors made in the verification data set, which the network was not looking at, except to make this prediction. The lower, fairly smooth line is the training set error, while the upper jagged line shows the verification set error.

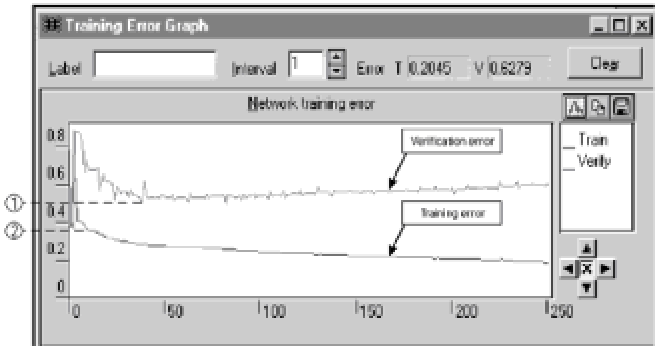


Figure 12.8 Errors in the training and verification data sets for 250 epochs of training on the unprepared CREDIT data set predicting BUYER. Before the network has learned anything, the error in the verification set is near its lowest at 2, while the error in the training set is at its highest. After about 45 epochs of training, the error in the training set is low and the error in the verification set is at its lowest—about 50% error—at 1.

As the training set was better learned, so the error rate in the training set declined. At first, the underlying relationship was truly being learned, so the error rate in the verification data set declined too. At some point, overtraining began, and the error in the training data set continued to decline but the error in the verification data set increased. At that point, the network was learning noise.

In this particular example, in the very early epochs—long before the network actually learned anything—the lowest error rate in the verification data set was discovered! This is happenstance due to the random nature of the network weights. At the same time, the error rate in the training set was at its highest, so nothing of value was learned by then. Looking at the graph shows that as learning continued, after some initial jumping about, the relationship in the verification data set was at its lowest after about 45 epochs. The error rate at that point was about 0.5. This is really a very poor performance, since 50% is exactly the same as random guessing! Recall that the balanced data set has 50% of buyers and nonbuyers, so flipping a fair coin provides a 50% accuracy rate. It is also notable that the error rate in the training data set continued to fall so that the network continued to learn noise. So much then for training on the “unprepared” data set.

The story shown for the prepared data set in Figure 12.9 is very different! Notice that the highest error level shown on the error graph here is about 0.55, or 55%. In the previous figure, the highest error shown was about 90%. (The StatSoft window scales automatically to accommodate the range of the graph.) In this graph, three things are very notable. First, the training and verification errors declined together at first, and are by no means as far separated as they were before. Second, error in the verification declined for more epochs than before, so learning of the underlying relationship continued longer. Third, the prediction error in the verification data set fell much lower than in the unprepared data set. After about 95 epochs, the verification error fell to 0.38, or a 38% error rate. In other words, with a 38% error rate, the network made a correct prediction 62% of the time, far better than random guessing!

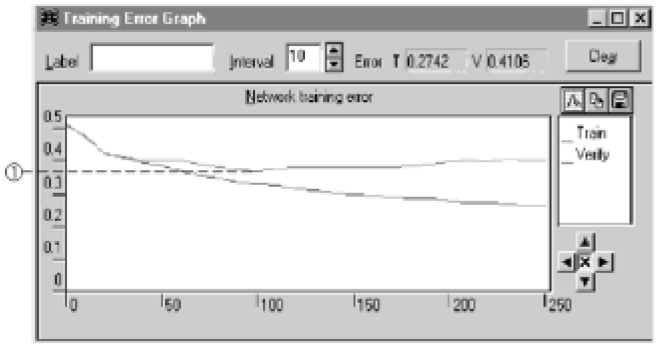


Figure 12.9 Training errors in the prepared data set for identical conditions as before. Minimum error is shown at 1.

Using the same network, on the same data set, and training under the same conditions, data prepared using the techniques described here performed 25% better than either random guessing or a network trained on data prepared using the StatSoft-provided, statistically based preparation techniques. A very considerable improvement!

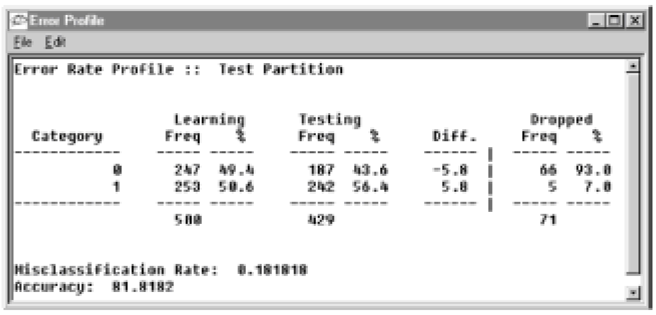
Also of note in comparing the performance of the two data sets is that the training set error in the prepared data did not fall as low as in the unprepared data. In fact, from the slope and level of the training set error graphs, it is easy to see that the network training in the prepared data resisted learning noise to a greater degree than in the unprepared data set.

12.3.2 Decision Trees and the CREDIT Data Set

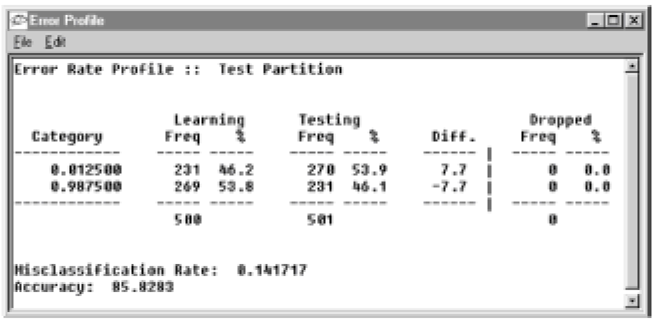
Exposing the information content seems to be effective for a neural network. But a decision tree uses a very different algorithm. It not only slices state space, rather than fitting a function, but it also handles the data in a very different way. A tree can digest unprepared data, and also is not as sensitive to balancing of the data set as a network. Does data preparation help improve performance for a decision tree? Once again, rather than extracting features or using any insights gleaned from the data survey, and taking the CREDIT data set as it comes, how does a decision tree perform?

Two trees were built on the CREDIT data set, one on prepared data, and one on unprepared data. The tree used was KnowledgeSEEKER from Angoss Software. All of the defaults were used in both trees, and no attempt was made to optimize either the model or the data. In both cases the trees were constructed automatically. Results?

The data was again divided into training and test partitions, and again BUYER was the prediction variable. The trees were built on the training partitions and tested on the test partitions. Figure 12.10 shows the results. The upper image shows the Error Profile window from KnowledgeSEEKER for the unprepared data set. In this case the accuracy of the model built on unprepared data is 81.8182%. With prepared data the accuracy rises to 85.8283%. This represents approximately a 5% improvement in accuracy. However, the misclassification rate improves from 0.181818 to 0.141717, which is an improvement of better than 20%. For decision trees, at least in this case, the quality of the model produced improves simply by preparing the data so that the information content is best exposed.



```
Error Rate Profile ::  Test Partition


                    Learning       Testing                 Dropped
  Category         Freq    %      Freq    %       Diff.    Freq    %
-------------     -----  -----   -----  -----   ------ |  -----  -----
            0       247   49.4     187   43.6     -5.8 |     66   93.0
            1       253   50.6     242   56.4      5.8 |      5    7.0
-------------     -----  -----   -----  -----   ------ |  -----  -----
                    500            429                       71


Misclassification Rate:  0.181818
Accuracy:  81.8182
```



```
Error Rate Profile ::  Test Partition


                    Learning       Testing                 Dropped
  Category         Freq    %      Freq    %       Diff.    Freq    %
-------------     -----  -----   -----  -----   ------ |  -----  -----
     0.012500       231   46.2     270   53.9      7.7 |      0    0.0
     0.987500       269   53.8     231   46.1     -7.7 |      0    0.0
-------------     -----  -----   -----  -----   ------ |  -----  -----
                    500            501                        0


Misclassification Rate:  0.141717
Accuracy:  85.8283
```

Figure 12.10 Training a tree with Angoss KnowledgeSEEKER on unprepared data shows an 81.8182% accuracy on the test data set (top) and an 85.8283% accuracy in the test data for the prepared data set (bottom).

12.4 Practical Use of Data Preparation and Prepared Data

How does a miner use data preparation in practice? There are three separate issues to address. The first part of data preparation is the assay, described in Chapter 4. Assaying the data to evaluate its suitability and quality usually reveals an enormous amount about the data. All of this knowledge and insight needs to be applied by the miner when constructing the model. The assay is an essential and inextricable part of the data preparation process for any miner. Although there are automated tools available to help reveal what is in the data (some of which are provided in the demonstration code), the assay requires a miner to apply insight and understanding, tempered with experience.

Modeling requires the selection of a tool appropriate for the job, based on the nature of the data available. If in doubt, try several! Fortunately, the prepared data is easy to work with and does not require any modification to the usual modeling techniques.

When applying constructed models, if an inferential model is needed, data extracts for training, test, and evaluation data sets can be prepared and models built on those data sets. For any continuously operating model, the Prepared Information Environment Input and Output (PIE-I and PIE-O) modules must be constructed to “envelop” the model so that live data is dynamically prepared, and the predicted results are converted back into real-world values.

All of these briefly touched-on points have been more fully discussed in earlier chapters. There are a wealth of practical modeling techniques available to any miner—far more than the number of tools available. Even a brief review of the main techniques for building effective models is beyond the scope of the present book. Fortunately, unlike data preparation and data surveying, much has been written about practical data modeling and model building. However, there are some interesting points to note about the state of current modeling tools.

12.5 Looking at Present Modeling Tools and Future Directions

In every case, modern data mining modeling tools are designed to attempt two tasks. The first is to extract interesting relationships from a data set. The second is to present the results in a form understandable to humans. Most tools are essentially extensions of statistical techniques. The underlying assumption is that it is sufficient to learn to characterize the joint frequencies of occurrence between variables. Given some characterization of the joint frequency of occurrence, it is possible to examine a multivariable input and estimate the probability of any particular output. Since full, multivariable joint frequency predictors are often large, unwieldy, and slow, the modeling tool provides some more compact, faster, or otherwise modified method for estimating the probability of an output. When it works, which is quite often, this is an effective method for producing predictions, and also for exploring the nature of the relationships between variables. However, no such methods directly try to characterize the underlying relationship driving the “process” that produces the values themselves.

For instance, consider a string of values produced from sequential calls to a pseudo-random number generator. There are many algorithms for producing pseudo-random numbers, some more effective at passing tests of randomness than others. However, no method of characterizing the joint frequency of the “next” number from the “previous” numbers in the pseudo-random series is going to do much toward producing worthwhile predictions of “next” values. If the pseudo-random number generator is doing its job, the “next” number is essentially unpredictable from a study of any amount of “previous” values. But since the “next” number is, in fact, being generated by a precisely determined algorithm, there is present a completely defined relationship between any “previous” number and the “next.” It is in this sense that joint frequency estimators are not looking for the underlying relationships. The best that they can accomplish is to characterize the results of some underlying process.

There are tools, although few commercially available, that actually seek to characterize the underlying relationships and that can actually make a start at estimating, for instance, a pseudo-random number generation function. However, this is a very difficult problem, and most of the recent results from such tools are little different in practical application than the results of the joint frequency estimators. However, tools that attempt to explore the actual underlying nature of the relationships, rather than to just predict their effects, hold much promise for deeper insights.

Another new direction for modeling tools is that of knowledge schema. Chapter 11 discussed the difference between the information embedded in a data set alone, and that embodied in the associated dictionary. A *knowledge schema* is a method of representing the dictionary information extracted and inferred from one or more data sets. Such knowledge schema represent a form of “understanding” of the data. Simple knowledge schema are currently being built from information extracted from data sets and offer powerful new insights into data—considerably beyond that of a mining or modeling tool that does not extract and apply such schema.

As powerful as they are, presently available modeling tools are at a first-generation level. Tools exploring and characterizing underlying relationships, and those building knowledge schema, for instance, represent the growth to a second generation of tools as astoundingly more powerful and capable as modern tools are than the statistical methods that preceded them. Data preparation too is at a first-generation level. The tools and techniques presented in this book for data preparation represent the best that can be done today with automated general-purpose data preparation and represent advanced first-generation tools. Just as there are exciting new directions and tremendous advances in modeling tools on the horizon, so too there are new advances and techniques on the horizon for automated data preparation. Such future directions will lead to powerful new insights and understanding of data—and ultimately will enhance our understanding of, and enable us to develop, practically applicable insights into the world.

12.5.1 Near Future

Data preparation looked at here has dealt with data in the form collected in mainly corporate databases. Clearly this is where the focus is today, and it is also the sort of data on which data mining tools and data modeling tools focus.

The near future will see the development of automated data preparation tools for series data. Approaches for automated series data preparation are already moving ahead and could even be available shortly after this book reaches publication.

Continuous text is a form of data that is not dealt with in this book and presents very different types of problems to those discussed here. Text can be characterized in various ways. There is writing of all types, from business plans to Shakespeare's plays. Preparation of text in general is well beyond the near future, but mining of one particular type of text, and preparation of that type of text for mining, is close. This is the type of text that can be described as thematic, discursive text.

Thematic means that it has a consistent thread to the subject matter that it addresses. Discursive means that it passes from premises to conclusions. Text is discursive as opposed to intuitive. This book is an example of thematic, discursive text, as are sales reports, crime reports, meeting summaries, many memoranda and e-mail messages, even newspaper and news magazine articles. Such text addresses a limited domain and presents facts, interpretations of facts, deductions from facts, and the relationships and inferences between facts. It also may express a knowledge structure. Looking at data of this sort in text form is called *text mining*. Much work is being done to prepare and mine such data, and there are already embryonic text mining tools available. Clearly, there is a huge range of types of even thematic, discursive text, and not all of it will soon be amenable to preparation and mining! But there are already tools, techniques, and approaches that promise success and are awaiting only the arrival of more computer power to become effective.

The near future for data preparation includes fully automated information exposure for the types of data discussed in this book, including series data, and adding thematic, discursive text preparation.

12.5.2 Farther Out

The Internet and World Wide Web present formidable challenges. So far there isn't even any beginning to classifying the types of data available on the Internet, let alone devising strategies for mining it! The Internet is not just a collection of text pages. It includes such a wealth and variety of information forms that they are almost impossible to list, let alone characterize—sound, audio-visual, HTML, ActiveX objects, and on and on. All of these represent a form of information communication. Computers "understand" them enough to represent them in human-understandable form—and humans can extract sense and meaning from them.

Automating the mining of such data presents a formidable challenge. Yet we will surely begin the attempt. A truly vast amount of information is enfolded in the Internet. It is arguably the repository for what we have learned about the nature of the world as well as a forum, meeting place, library, and market. It may turn out to be a development as revolutionary in terms of human development as the invention of agriculture. It is a new form of enterprise in human experience, and it is clearly beyond the scope of understanding by any individual human being. Automated discovery, cognition, and understanding are obviously going to be applied to this enormous resource. Already terms such as “web farming” are in current use as we try to come to grips with the problem.

There are many other forms of data that will benefit from automated “understanding” and so will need preparation to best expose its information content. These range from continuous speech to CAT (computerized axial tomography) scans. Just as the tools of the industrial revolution represent tools to enhance and extend our physical capabilities, so computers represent the beginnings of tools to extend the reach of our understanding. Whatever sort of data it is that we seem to understand, correct preparation of that data, as well as preparation of the miner, will always pay huge dividends.

Appendix A: Using the Demonstration Code on the CD-ROM

Data Format

The data must be in a comma-delimited ASCII file. The first line of the file contains the comma-delimited variable names. DP will automatically type the variables as numerical if the first value contains a number or categorical if the first value starts with an alphabetic character.

Various control flags can be added after the variable name in the form VarName<F>. The available flags are

N	Treat the variable as numerical. Values containing nonnumeric characters will be treated as nulls.
C	Treat the variable as categorical.
P	Treat the variable as the dependent or prediction variable.
X	Exclude the variable from analysis.
K	Key variable—exclude the variable from analysis but copy to the output file.

Control Variables

The operation of the DP program is controlled by an ASCII file containing names of control variables and their values: one control value per line separated by spaces from the value. Comments can be included by preceding them with a double slash (//).

IPFileName	Input file name
OPFileName	Output file name
ConfLevelSmpl	Confidence level to stop sampling
ConfLevelDrop	Confidence level to drop a variable
ConfLevelNum	Confidence level for numerating categoricals
CatCntDrop	Categorical count to Sample count drop ratio. That is, if

	CatCntDrop = 0.6, then if the number of unique categories in a variable is greater than 0.6 x number of sample rows, the variable will be dropped from further processing.
ReplaceMissing	1 if replace missing values
MaxNumDimension	Maximum dimension to use when numerating categoricals
MaxNumEstimates	Maximum number of numerical variables to use when numerating categoricals
OutputType	0 – raw, 1 – scaled, 2 – rank
Compress	1 if build compression model
ConfLevelCmp	Confidence level to use when searching for the compression model
ConnectPcnt	The percentage of paths to include when computing either the best compression model or the importance. Generally 10–20% works well.
Importance	1 if compute the neural network importance of the variables
ImportCnt	The number of complete cycles to use when computing importance

Usage

Suppose that the executable is located in c:\dos; and the control file, data file is located in c:\data; and the control file, ctrl, is also in the c:\data. To run dp10, make the c:\data directory the current directory and enter

```
c:\dos\dp10.exe ctrl
```

Appendix: Using the Demonstration Code on the CD-ROM

Sample Control File

`Actual control file comments in Letter Gothic font`

Comments that are not included in italics

```
IPFileName          cars.dat
```

OPFileName	carDP	
ConfLevelSmpl	.95	*Confidence level for sampling*
ConfLevelDrop	.70	*Confidence level for dropping variables*
ConfLevelNum	.95	*Confidence level for numerating*
CatCntDrop	0.8	*If > n x NoInstances drop categorical variable*
ReplaceMissing	1	*1 = Replace missing*
		0 = Don’t replace missing
MaxNumDimension	3	*Max dimensions when representing categoricals*
MaxNumEstimates	6	*Max start var count for numerating categoricals*
OutputType	2	*0 = Only numerate categoricals*
		1 = Numerate cat & norm all
		2 = Numerate, normalize, redistibute
Compress	0	*1 = Compress this file*
		0 = Don’t compress
		Then if compressing:
ConnectPcnt	50	*Connection percent*
HiddenNodes	0	*0 = Search*
		n = Use this number
ConfLevelCmp	.95	
Importance	1	*0 = No importance measures*
		1 = Measure variable importance
ImportCnt	10	*Number of test cycles*

Appendix B: Further Reading

- Abelson, Robert P. *Statistics as Principled Argument*. Hillsdale, NJ: Lawrence Erlbaum Associates, 1995.

  This book is about how to think about interpreting the results of investigations into data. It is statistically based, but the discussions apply equally well to the results of data mining. There is little mathematics in the discussion.

  Useful to business managers, analysts, modelers, and miners.

- Berry, Michael J. A., and Gordon Linoff. *Data Mining Techniques: For ~Marketing, Sales, and Customer Support.* New York: John Wiley & Sons, 1997.

  This book provides a conceptual overview of various data mining techniques, looking at how each actually works at a conceptual level. This is an almost entirely nonmathematical treatment. The examples discussed are mainly business oriented.

- Deming, William E. *Statistical Adjustment of Data*. New York: Dover Publications, 1984.

  Deming, William E. *Some Theory of Sampling*. New York: Dover Publications, 1984.

  These two books were originally written in 1938 and 1950, respectively. William Edwards Deming gave a great deal of thought to the problems of collecting data from, and using data in, the real world. These books are mathematical in their treatment of the problem and are statistically based. Many issues and problems are raised in these works that are not yet amenable to automated solutions.

  Useful to advanced modelers and technically oriented miners.

- Gleason, A. (translator). Transnational College of LEX. *Who is Fourier?: A Mathematical Adventure*. Cambridge, MA: Blackwell Science, 1995.

  A simply wonderful book explaining Fourier series. Although mathematical, the introduction to the concepts is so gentle that the mathematics is almost unnoticeable, even though it necessarily deals with trigonometry and elementary calculus.

  Useful to analysts, modelers, and miners.

- Grouse, Donald C., and Gerald M. Weinberg. *Exploring Requirements: Quality Before Design*. New York: Dorset House, 1989.

This is a book about exploring requirements for products and systems. Much of what they have to say is easily applicable to exploring requirements for business problems and solutions. Almost all of the points they raise, and the solutions and methods they propose to address those points, are applicable to general problem solving.

Useful to business managers, analysts, modelers, and miners.

- Jones, Morgan D. *The Thinker's Toolkit; Fourteen Skills for Making Smarter Decisions in Business and in Life*. New York: Times Business Books, 1995.

  The jacket description for this book introduces it as ". . . a unique collection of proven, practical methods for simplifying any problem and making faster, better decisions every time." All of these decision-making methods are applicable to problem and solution space exploration.

  Useful to business managers, analysts, modelers, and miners.

- Moore, David S. *Statistics: Concepts and Controversies*. New York: W. H. Freeman and Co., 1996.

  This is a nonmathematical treatment of an approach to thinking about the problems, use, and applicability of understanding information enfolded in data. It is not so much about statistics as about introducing the issues in thinking about how to understand data. Almost all of the issues raised and discussed are applicable to mining and modeling.

  Useful to business managers, analysts, modelers, and miners.

- Shannon, Claude E., and Warren Weaver. *The Mathematical Theory of Communication*. Urbana/Chicago: University of Illinois Press, 1998.

  This seminal work on communication (information) theory was originally published as a paper in 1948 and as a book in 1949. Dr. Warren Weaver's introduction is very readable. Claude Shannon's paper, although mathematical, is a model of clarity. A huge number of books have been written in the intervening 50 years, and the theory has been much extended, but this book still holds its own.

- Smith, Murray. *Neural Networks for Statistical Modeling*. New York: Van Nostrand Reinhold, 1993.

  A clear introduction to the concepts, construction, and operation of neural networks. It includes code in BASIC. Although it does not discuss sparsely connected networks or autoassociative neural networks, the general structure of back-propagation neural networks is well presented. The demonstration code for data preparation, while not

based on the code discussed in the book, uses the same constant and parameter names, and some of the same structure as the network in this book, so that the transition into further exploration of neural networks will be eased. The treatment is modestly mathematical.

Useful to modelers and technically oriented miners.