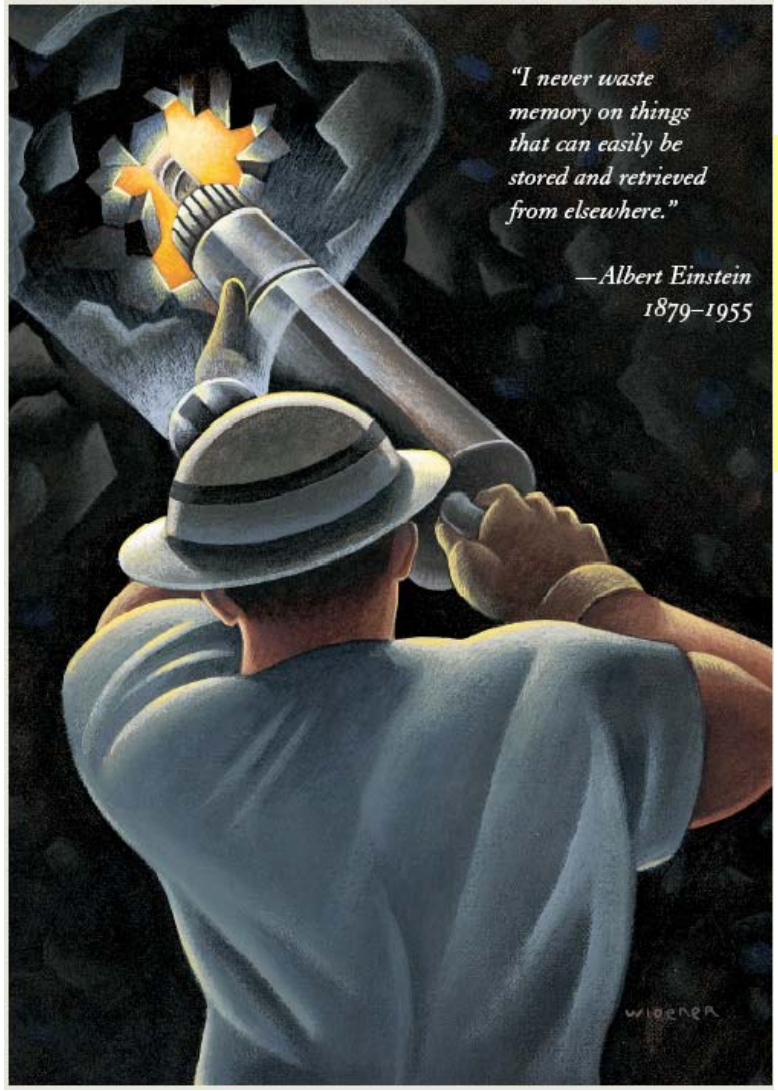
The University of Reading



Spring Term 2016

Data Mining (SE3DM11)

Dr. Giuseppe Di Fatta
Associate Professor of Computer Science
Director MSc Advanced Computer Science
G.DiFatta@reading.ac.uk
http://www.personal.reading.ac.uk/~sis06gd/

About the Module

- 20 hours contact time:
  - 2h lecture/week: Monday 16:00 – 18:00 (HUMSS 127)
  - no lecture in week 6
  - no practical sessions
- Office hours (room 144)
  - Preferably send an email to arrange an appointment
- Course Assessment:
  - **50%** coursework (1 major assignment)
    - deadline in week 11: Wednesday 23 March 2016 10:30am
  - **50%** final exam
    - typically in May and resit in Aug./Sept.

Aims & Objectives

- **Aims**:

  Automated data collection tools and mature database technology lead to tremendous amounts of data stored in databases, data warehouses and other information repositories. In this context automated data analysis techniques (Data Mining) are becoming essential components to any information system. Application areas of these techniques include scientific computing, intelligent business, direct marketing, customer relationship management, market segmentation, store shelf management, data warehouse management, fraud detection in e-commerce and in credit card transactions, etc.

  The module focuses on concepts, techniques and algorithms for the extraction of interesting knowledge (rules, regularities, patterns, constraints) from large data sets. The techniques span from statistics to machine learning and information science.

- **Intended learning outcomes**:

  Assessable outcomes

  Students are expected to understand the general Data Mining principles and techniques and to be able to apply them in different contexts. As final project they will implement and demonstrate a data mining technique among the ones presented in the module.

  Additional outcomes

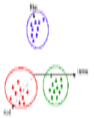
  The students will become familiar with the potential applications of data mining techniques in different domains. They will also learn how to carry out experimental tests for algorithm performance evaluations.

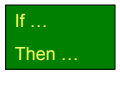
Course Material

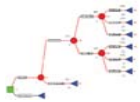
- **Lecture notes (slides)**
  - Edited and modified from slides of: Giuseppe Di Fatta, Jiawei Han, Vipin Kumar, Ad Feelders, Zdravko Markov, Slobodan Vucetic
- **Textbook:**
  1. "Introduction to Data Mining",
     Pang-Ning Tan, Michael Steinbach, Vipin Kumar
     Pearson International Edition, ISBN-10: 0321420527, ISBN-13: 9780321420527
  2. "Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)"
     Ian H. Witten, Eibe Frank
     Morgan Kaufmann, ISBN 0-12-088407-0
  3. "Data Mining, Concepts and Techniques, Second Edition"
     Jiawei Han, Micheline Kamber
     Morgan Kaufmann Publishers, March 2006, ISBN: 978-1-55860-901-3
- **Other material on Blackboard**
  - Please check it out

Outline

- Introduction to Data Mining
- Workflow Management Systems
- Classification and Regression
- Clustering
- Decision Trees
- Association Rules and Classification Rules







The University of Reading

SE3DM11- Data Mining

Module Overview

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Motivation

- Data explosion problem
  - Automated data collection tools and mature database technology lead to tremendous amounts of data stored in databases, data warehouses and other information repositories
- We are drowning in data, but starving for knowledge!
- Solution: Data warehousing and data mining
  - Data warehousing and on-line analytical processing
  - Extraction of interesting knowledge (rules, regularities, patterns, constraints) from data in large databases

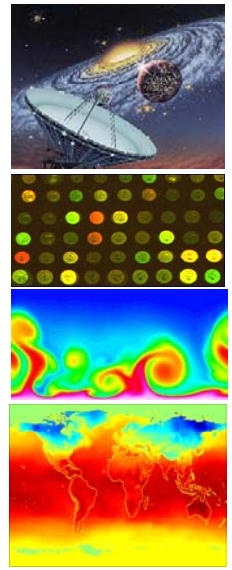
Why Mine Data? Commercial Viewpoint

- Lots of data is being collected and warehoused
  - Web data, e-commerce
  - purchases at department/ grocery stores
  - Bank/Credit Card transactions

- Computers have become cheaper and more powerful
- Competitive Pressure is Strong
  - Provide better, customized services for an *edge* (e.g. in Customer Relationship Management)

Why Mine Data? Scientific Viewpoint

- Data collected and stored at enormous speeds (GB/hour)
  - remote sensors on a satellite
  - telescopes scanning the skies
  - microarrays generating gene expression data
  - scientific simulations generating terabytes of data
- Traditional techniques infeasible for raw data
- Data mining may help scientists
  - in classifying and segmenting data
  - in Hypothesis Formation



Why Mine Data? Looking ahead…

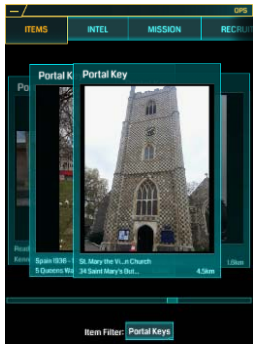
More and more data…

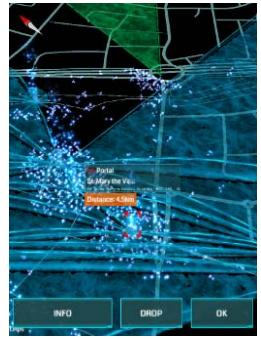
- RF Tag
  - Radiofrequency tags require no battery to read and operate, are cost-effective, and will read and write multiple tags in an RF field.
- Smart Dust
  - Miniature machines, each the size of a dust mote, may eventually saturate the environment, invisibly performing countless tasks.

Why Mine Data? Looking ahead…

More and more data…

- Augmented Reality MMOG
  - Ingress is an augmented reality massively multiplayer online video game created by NianticLabs@Google, released for Android devices. The game has been called "**a data gold mine**" for Google.





What Is Data Mining?

- Data mining (knowledge discovery in databases):
  - Extraction of interesting (non-trivial, implicit, previously unknown and potentially useful) information or patterns from data in large databases

- Learning and describing concepts from data
- Alternative names:
  - Data mining: a misnomer?
  - Knowledge discovery(mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, business intelligence, etc.

Examples: What is (not) Data Mining?

- **What is not Data Mining?**
  - Look up phone number in phone directory
  - Query a Web search engine for information about "Amazon"

- **What is Data Mining?**
  - Certain names are more prevalent in certain US locations (O'Brien, O'Rurke, O'Reilly… in Boston area)
  - Group together similar documents returned by search engines according to their context
    - try searching for ambiguous words at www.kartoo.com

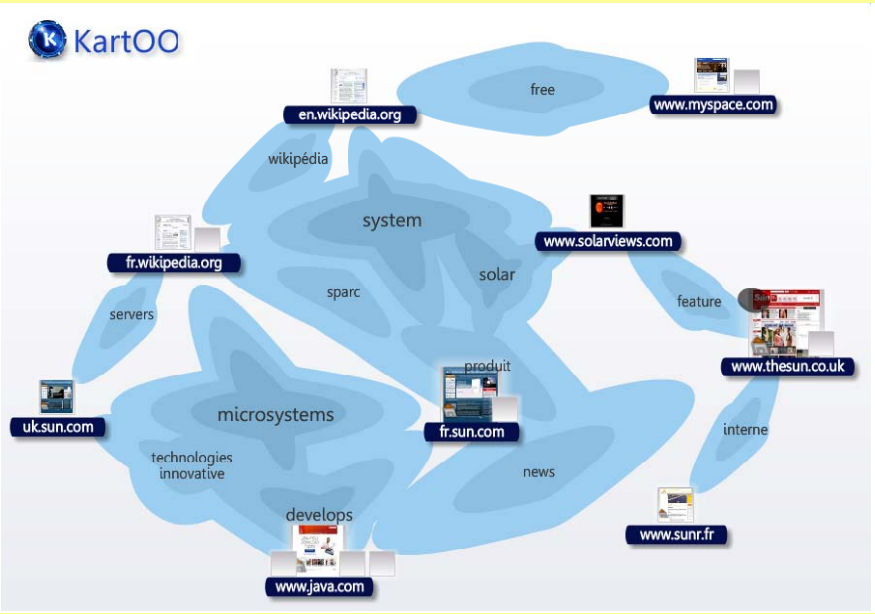
Data Mining is not…

- Data warehousing
- SQL / Ad Hoc Queries / Reporting
- Software Agents
- Online Analytical Processing (OLAP)
- Data Visualization

Data Mining is… about

- describing and understanding data in order to extract "hidden" knowledge, i.e. concepts.
  - analyze the data
  - find relations in the data
  - describe the relations

Searching for “sun”



Data Mining: Classification Schemes

- Decisions in data mining
  - Kinds of databases to be mined
  - Kinds of knowledge to be discovered
  - Kinds of techniques utilized
  - Kinds of applications
- Data mining tasks
  - Descriptive data mining
  - Predictive data mining

Decisions in Data Mining

- **Databases to be mined**
  - Relational, transactional, object-oriented, object-relational, active, spatial, time-series, text, multi-media, heterogeneous, legacy, WWW, etc.
- **Knowledge to be mined**
  - Characterization, discrimination, association, classification, clustering, trend, deviation and outlier analysis, etc.
  - Multiple/integrated functions and mining at multiple levels
- **Techniques utilized**
  - Database-oriented, data warehouse (OLAP), machine learning, statistics, visualization, neural network, etc.
- **Applications**
  - Retail, telecommunication, banking, fraud analysis, DNA mining, stock market analysis, Web mining, Weblog analysis, etc.

Data Mining Tasks

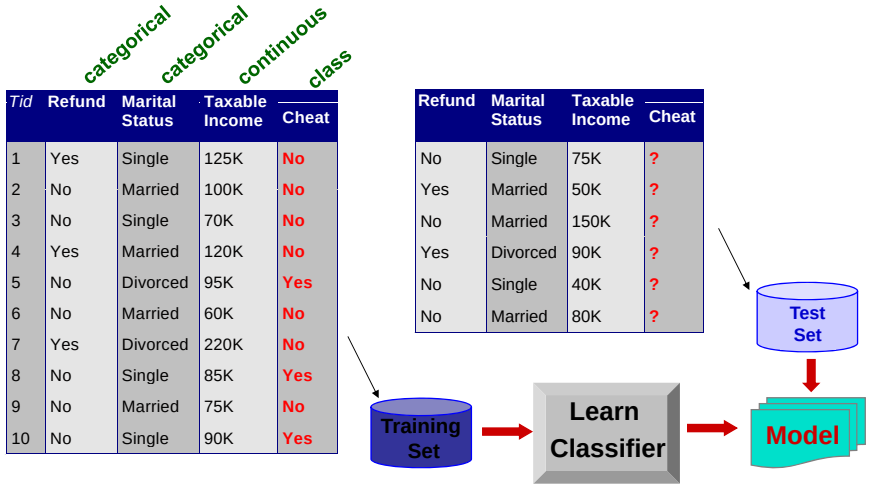
- **Predictive Tasks**
  - Use some variables to predict unknown or future values of other variables
- **Descriptive Tasks**
  - Find human-interpretable patterns that describe the data.

Common data mining tasks
- Classification [Predictive]
- Clustering [Descriptive]
- Association Rule Discovery [Descriptive]
- Sequential Pattern Discovery [Descriptive]
- Regression [Predictive]
- Deviation Detection [Predictive]

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for the class attribute as a function of the values of other attributes.
- Goal: <u>previously unseen</u> records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Classification Example



Classification: Application 1

- Direct Marketing
  - Goal: Reduce cost of mailing by *targeting* a set of consumers likely to buy a new cell-phone product.
  - Approach:
    - Use the data for a similar product introduced before.
    - We know which customers decided to buy and which decided otherwise. This *{buy, don't buy}* decision forms the *class attribute*.
    - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
      - Type of business, where they stay, how much they earn, etc.
    - Use this information as input attributes to learn a classifier model.

Classification: Application 2

- Fraud Detection
  - Goal: Predict fraudulent cases in credit card transactions.
  - Approach:
    - Use credit card transactions and the information on its account-holder as attributes.
      - When does a customer buy, what does he buy, how often he pays on time, etc
    - Label past transactions as fraud or fair transactions. This forms the class attribute.
    - Learn a model for the class of the transactions.
    - Use this model to detect fraud by observing credit card transactions on an account.

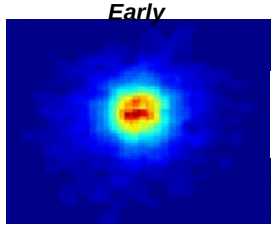
Classification: Application 3

- Customer Attrition/Churn:
  - Goal: To predict whether a customer is likely to be lost to a competitor.
  - Approach:
    - Use detailed record of transactions with each of the past and present customers, to find attributes.
      - How often the customer calls, where he calls, what time-of-the day he calls most, his financial status, marital status, etc.
    - Label the customers as loyal or disloyal.
    - Find a model for loyalty.

Classification: Application 4

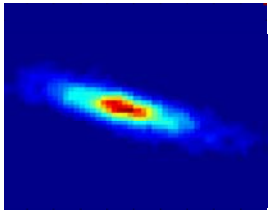
- Sky Survey Cataloging
  - Goal: To predict class (star or galaxy) of sky objects, especially visually faint ones, based on the telescopic survey images (from Palomar Observatory).
    - 3000 images with 23,040 x 23,040 pixels per image.
  - Approach:
    - Segment the image.
    - Measure image attributes (features) - 40 of them per object.
    - Model the class based on these features.
    - Success Story: Could find 16 new high red-shift quasars, some of the farthest objects that are difficult to find!

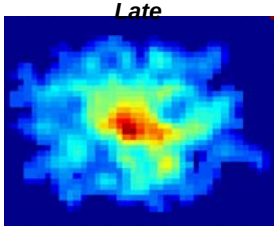
Classifying Galaxies



Class:
- **Stages of Formation**

Attributes:
- **Image features,**
- **Characteristics of light waves received, etc.**





Data Size:
- **72 million stars, 20 million galaxies**
- **Object Catalog: 9 GB**
- **Image Database: 150 GB**

Clustering Definition

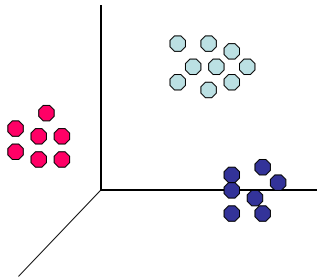
- Given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that
  - Data points in one cluster are more similar to one another.
  - Data points in separate clusters are less similar to one another.
- Similarity Measures:
  - Euclidean distance if attributes are continuous.
  - Other problem-specific measures.

Illustrating Clustering

- Euclidean Distance Based Clustering in 3 D space.

Intracluster distances are minimized

Intercluster distances are maximized



Clustering: Application 1

- Market Segmentation:
  - Goal: subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix.
  - Approach:
    - Collect different attributes of customers based on their geographical and lifestyle related information.
    - Find clusters of similar customers.
    - Measure the clustering quality by observing buying patterns of customers in same cluster vs. those from different clusters.

Clustering: Application 2

- Document Clustering:
  - Goal: To find groups of documents that are similar to each other based on the important terms appearing in them.
  - Approach: To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.
  - Gain: Information Retrieval can utilize the clusters to relate a new document or search term to clustered documents.

Association Rule Discovery: Definition

- Given a set of records each of which contain some number of items from a given collection;
  - Produce dependency rules which will predict occurrence of an item based on occurrences of other items.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:
{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Association Rule Discovery: Application 1

- Marketing and Sales Promotion:
  - Let the rule discovered be

    {Bagels, … } --> {Potato Chips}
  - Potato Chips as consequent => Can be used to determine what should be done to boost its sales.
  - Bagels in the antecedent => Can be used to see which products would be affected if the store discontinues selling bagels.
  - Bagels in antecedent *and* Potato chips in consequent => Can be used to see what products should be sold with Bagels to promote sale of Potato chips!

Association Rule Discovery: Application 2

- Supermarket shelf management.
  - Goal: To identify items that are bought together by sufficiently many customers.
  - Approach: Process the point-of-sale data collected with barcode scanners to find dependencies among items.
  - A classic rule
    - If a customer buys diaper and milk, then he is very likely to buy beer:

$$Diapers \rightarrow Beer,\ support = 20\%,\ confidence = 85\%$$

Sequential Pattern Discovery: Definition

- Given is a set of *objects*, with each object associated with its own *timeline of events*, find rules that predict strong sequential dependencies among different events:
  - In telecommunications alarm logs,
    - (Inverter_Problem Excessive_Line_Current) (Rectifier_Alarm) --> (Fire_Alarm)
  - In point-of-sale transaction sequences,
    - Computer Bookstore:
      (Intro_To_Visual_C) (C++_Primer) --> (Perl_for_dummies,Tcl_Tk)
    - Athletic Apparel Store:
      (Shoes) (Racket, Racketball) --> (Sports_Jacket)

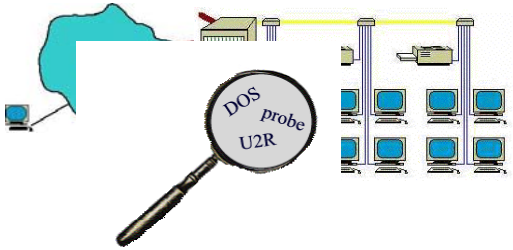
Regression

- Predict a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency.
- Greatly studied in statistics, neural network fields.
- Examples:
  - Predicting sales amounts of new product based on advertising expenditure.
  - Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
  - Time series prediction of stock market indices.

Deviation/Anomaly Detection

- Detect significant deviations from normal behavior
- Applications:
  - Credit Card Fraud Detection

  - Network Intrusion Detection



Data Mining and Induction Principle

Induction vs. Deduction

- Deductive reasoning is truth-preserving:
  1. All horses are mammals
  2. All mammals have lungs
  3. Therefore, all horses have lungs
- Induction reasoning adds information:
  1. All horses observed so far have lungs.
  2. Therefore, all horses have lungs.

The Problems with Induction

From true facts, we may induce false models.

Prototypical example:
- European swans are all white.
- Induce: ”Swans are white” as a general rule.
- 1606: discover Australia and black swans...
- Problem: the set of examples is not random and representative

Another example: distinguish US tanks from Iraqi tanks
- Method: Database of pictures split in train set and test set; Classification model built on train set
- Result: Good predictive accuracy on test set; bad score on independent pictures
- Why did it go wrong: other distinguishing features in the pictures (hangar versus desert)

Hypothesis-Based vs. Exploratory-Based

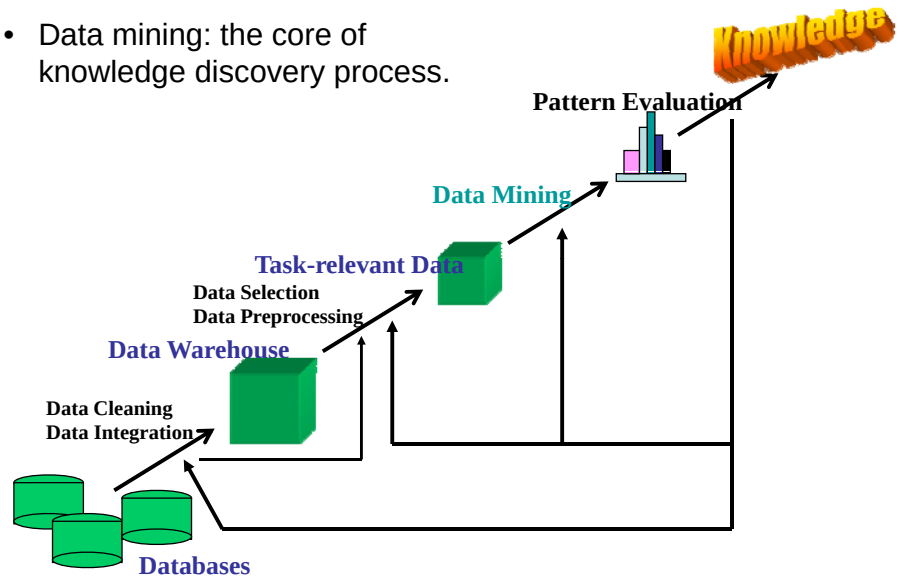
- The hypothesis-based method:
  - Formulate a hypothesis of interest.
  - Design an experiment that will yield data to test this hypothesis.
  - Accept or reject hypothesis depending on the outcome.
- Exploratory-based method:
  - Try to make sense of a bunch of data without an a priori hypothesis!
  - The only prevention against false results is significance:
    - ensure statistical significance (using train and test etc.)
    - ensure domain significance (i.e., make sure that the results make sense to a domain expert)

Hypothesis-Based vs. Exploratory-Based

- Experimental Scientist:
  - Assign level of fertilizer randomly to plot of land.
  - Control for: quality of soil, amount of sunlight,...
  - Compare mean yield of fertilized and unfertilized plots.
- Data Miner:
  - Notices that the yield is somewhat higher under trees where birds roost.
  - Conclusion: droppings increase yield.
  - Alternative conclusion: moderate amount of shade increases yield. ("Identification Problem")

Data Mining: A KDD Process

- Data mining: the core of knowledge discovery process.



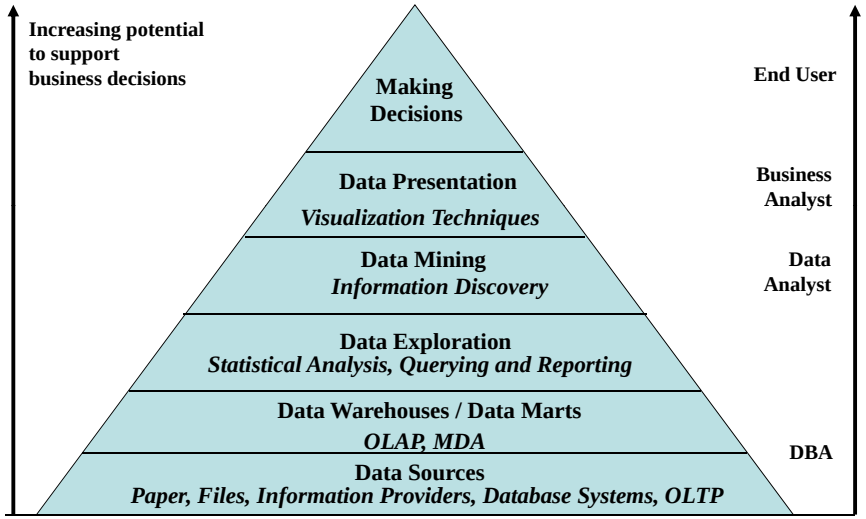
Steps of a KDD Process

1. data gathering
2. data cleansing
3. data transformation
4. selecting techniques
5. applying data mining
6. processing results

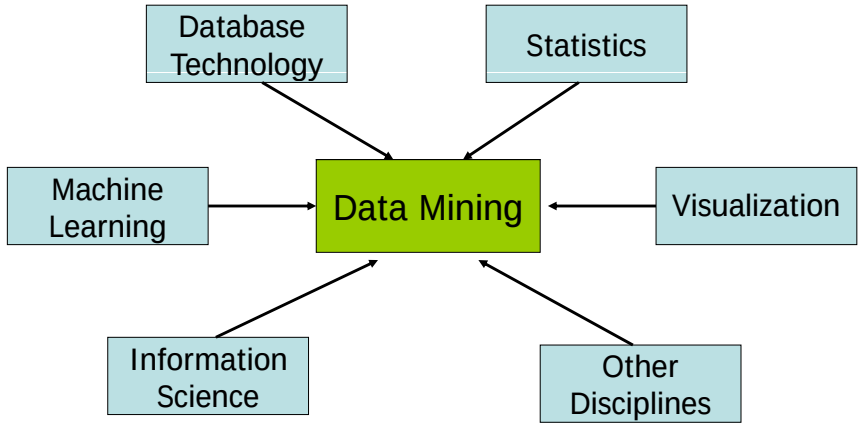
Steps of a KDD Process

- Learning the application domain:
  - relevant prior knowledge and goals of application
- Creating a target data set: data gathering, data selection
- **Data cleaning** and preprocessing: (may take 60% of the effort!)
- **Data reduction and transformation**:
  - Find useful features, dimensionality/variable reduction, invariant representation.
- Choosing functions of data mining
  - summarization, classification, regression, association, clustering, etc.
- Choosing the specific data mining algorithm(s)
- **Data mining**: search for patterns of interest, models, etc.
- **Pattern/model evaluation and knowledge presentation**
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

Data Mining and Business Intelligence



Data Mining: Confluence of Multiple Disciplines



The University of Reading

SE3DM11- Data Mining

Input Data

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

What is "Data"?

Input data: a set of instances
- instances, aka: records, objects, points, cases, samples, entities, etc.
- Individual, independent examples of the concept to be learned.
- Single relation DB, flat file.
- A collection of data objects and their attributes

An **attribute** is a property or characteristic of an object
- Examples: eye color of a person, temperature, etc.
- Attribute is also known as variable, field, characteristic, or feature

Attributes

Objects

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Attribute Values

- Attribute **values** are numbers or symbols assigned to an attribute
- Distinction between **attributes** and **attribute values**
  - Same attribute can be mapped to different attribute values
    - Example: height can be measured in feet, meters, or categorical values (tall, medium, short)
  - Different attributes can be mapped to the same set of values
    - Example: Attribute values for ID and age are integers
    - But properties of attribute values can be different
      - ID has no limit but age has a maximum and minimum value

Types of Attributes

There are different types of attributes:

- Nominal
  - Examples: ID numbers, eye color, zip codes
- Ordinal
  - Examples: rankings (e.g., taste of potato chips on a scale from 1-10), grades, height in {tall, medium, short}, ranking in a marathon
- Interval
  - Examples: calendar dates, temperatures in Celsius or Fahrenheit, level of happiness (e.g. rated from 1 to 10)
- Ratio
  - Examples: temperature in Kelvin, length, time, counts

Properties of Attribute Values

The type of an attribute depends on which of the following properties it possesses:

- Distinctness: = ≠
- Order: < >
- Addition: + -
- Multiplication: * /

- Nominal attribute: distinctness
- Ordinal attribute: distinctness & order
- Interval attribute: distinctness, order & addition
- Ratio attribute: all 4 properties

Attribute Type Description

Attribute Type	Description	Examples	Operations
Nominal	The values of a nominal attribute are just different names, i.e., nominal attributes provide only enough information to distinguish one object from another. (=, ≠)	zip codes, employee ID numbers, eye color, sex: {*male*, *female*}	mode, entropy, contingency correlation, χ^2 test
Ordinal	The values of an ordinal attribute provide enough information to order objects. (<, >)	hardness of minerals, {*good*, *better*, *best*}, grades, street numbers	median, percentiles, rank correlation, run tests, sign tests
Interval	For interval attributes, the differences between values are meaningful, i.e., a unit of measurement exists. (+, -)	calendar dates, temperature in Celsius or Fahrenheit	mean, standard deviation, Pearson's correlation, *t* and *F* tests
Ratio	For ratio variables, both differences and ratios are meaningful. (*, /)	temperature in Kelvin, monetary quantities, counts, age, mass, length, electrical current	geometric mean, harmonic mean, percent variation

Attribute Value Transformation

Attribute Type	Transformation	Comments
Nominal	Any permutation of values	If all employee ID numbers were reassigned, would it make any difference?
Ordinal	An order preserving change of values, i.e., *new_value = f(old_value)* where *f* is a monotonic function.	An attribute encompassing the notion of good, better best can be represented equally well by the values {1, 2, 3} or by { 0.5, 1, 10}.
Interval	*new_value =a * old_value + b* where a and b are constants	Thus, the Fahrenheit and Celsius temperature scales differ in terms of where their zero value is and the size of a unit (degree).
Ratio	*new_value = a * old_value*	Length can be measured in meters or feet.

Discrete and Continuous Attributes

- Discrete Attribute
  - Has only a finite or countably infinite set of values
  - Examples: zip codes, counts, or the set of words in a collection of documents
  - Often represented as integer variables.
  - Note: binary attributes are a special case of discrete attributes
- Continuous Attribute
  - Has real numbers as attribute values
  - Examples: temperature, height, or weight.
  - Practically, real values can only be measured and represented using a finite number of digits.
  - Continuous attributes are typically represented as floating-point variables.

Types of Data Sets

- Record
  - Data Matrix
  - Document Data
  - Transaction Data
- Multi-Relational
  - Star or snowflake schema
- Graph
  - World Wide Web
  - Molecular Structures
- Ordered
  - Spatial Data
  - Temporal Data
  - Sequential Data

Important Characteristics of Structured Data

- **Dimensionality**
  - Number of attributes each object is described with
  - Challenge: high dimensionality (curse of dimensionality)
- **Sparsity**
  - Sparse data: values of most attributes are zero
  - Challenge: sparse data call for special handling
- **Resolution**
  - Data properties often could be measured with different resolutions
  - Challenge: decide on the most appropriate resolution (e.g. “Can’t See the Forest for the Trees”)

Record Data

- Data that consists of a collection of records, each of which consists of a fixed set of attributes

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Data Matrix

- If data objects have the same fixed set of numeric attributes, then the data objects can be thought of as points in a multi-dimensional space, where each dimension represents a distinct attribute
- Such data set can be represented by an m by n matrix, where there are m rows, one for each object, and n columns, one for each attribute

Projection of x Load	Projection of y load	Distance	Load	Thickness
10.23	5.27	15.22	2.7	1.2
12.65	6.25	16.22	2.2	1.1

Document Data

- Each document becomes a 'term' vector,
  - each term is a component (attribute) of the vector,
  - the value of each component is the number of times the corresponding term occurs in the document.

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

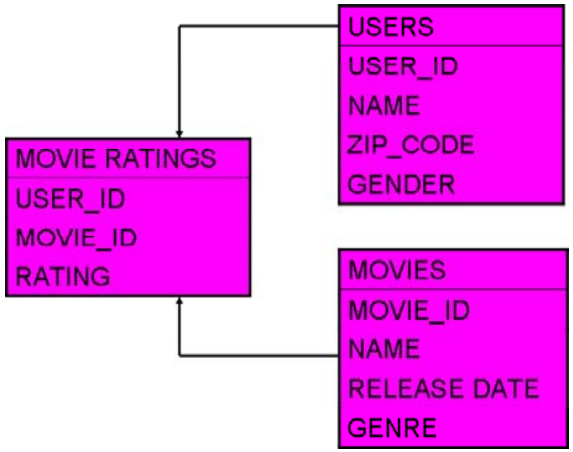
Transaction Data

- A special type of record data, where
  - each record (transaction) involves a set of items.
  - E.g., consider a grocery store. The set of products purchased by a customer during one shopping trip constitute a transaction, while the individual products that were purchased are the items.

TID	*Items*
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

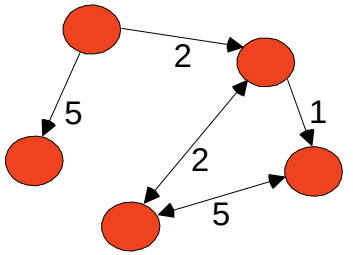
Multi-Relational Data

- Attributes are objects themselves



Graph Data

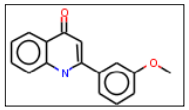
- Examples: Generic graph and HTML Links



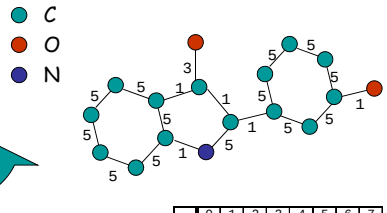

```
<a href="papers/papers.html#bbbb">
Data Mining </a>
<li>
<a href="papers/papers.html#aaaa">
Graph Partitioning </a>
<li>
<a href="papers/papers.html#aaaa">
Parallel Solution of Sparse Linear System of Equations </a>
<li>
<a href="papers/papers.html#ffff">
N-Body Computation and Dense Linear System Solvers
```

Chemical Data

Chemical compound



Graph



Graph representations:

- adjacency matrix
- edges list
  - Source atom type
  - Source atom
  - Bond type
  - Dest. Atom type
  - Dest. atom

	0	1	2	3	4	5	6	7
0		1			1			1
1	1		3				3	
2		3		5				
3			5		5			
4	1			5		5		1
5					5		5	
6		3				5		1
7	1				1		1	

```
(C,0,1,C,1)
(C,0,5,N,4)
(C,0,1,O,7)
...
```


Ordered Data

❑ Sequences of transactions

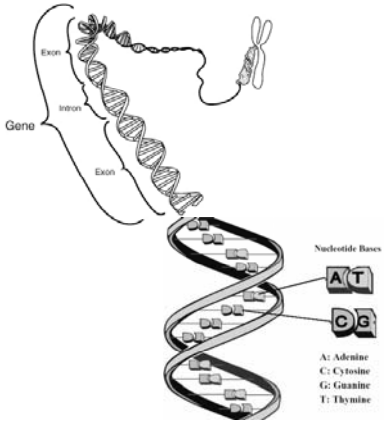
Items/Events

(A B) (D) (C E)
(B D) (C) (E)
(C D) (B) (A E)

An element of the sequence

Ordered Data

- Genomic sequence data

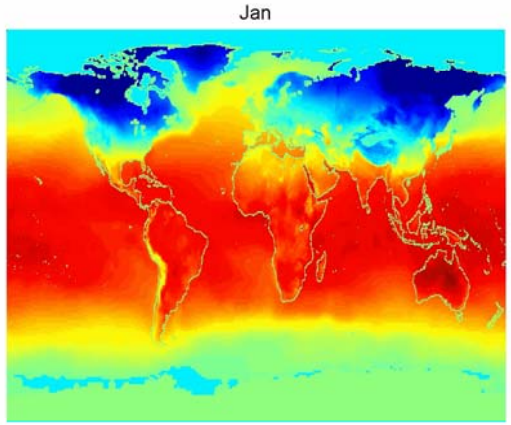


GGTTCCGCCTTCAGCCCCGCGCC
CGCAGGGCCCGCCCCGCGCCGTC
GAGAAGGGCCCGCCTGGCGGGCG
GGGGGAGGCGGGGCCGCCCGAGC
CCAACCGAGTCCGACCAGGTGCC
CCCTCTGCTCGGCCTAGACCTGA
GCTCATTAGGCGGCAGCGGACAG
GCCAAGTAGAACACGCGAAGCGC
TGGGCTGCCTGCTGCGACCAGGG

Ordered Data

- Spatial-Temporal Data

Average Monthly Temperature of land and ocean



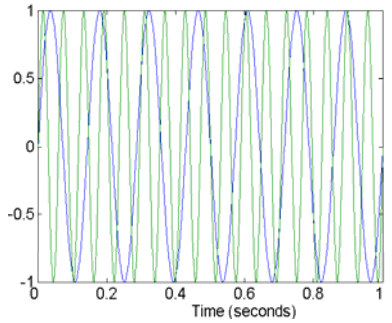
Data Quality

- ✓ What kinds of data-quality problems?
- ✓ How can we detect problems with the data?
- ✓ What can we do about these problems?

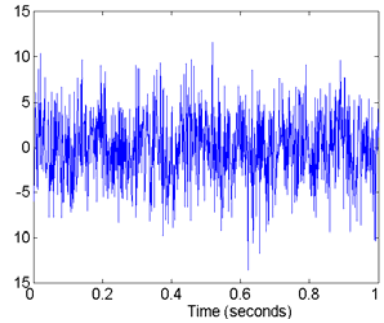
- Examples of data quality problems:
  - Noise and outliers
  - missing values
  - duplicate data

Noise

- Noise refers to modification of original values
  - Examples: distortion of a person's voice when talking on a poor phone and "snow" on television screen



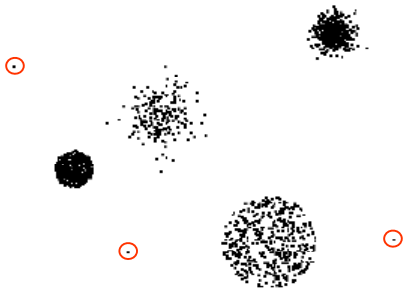
Two Sine Waves



Two Sine Waves + Noise

Outliers

- Outliers are data objects with characteristics that are considerably different than most of the other data objects in the data set



Missing Values

- Reasons for missing values
  - Information is not collected (e.g., people decline to give their age and weight)
  - Attributes may not be applicable to all cases (e.g., annual income is not applicable to children)
- Handling missing values
  - Eliminate Data Objects
  - Estimate Missing Values
  - Ignore the Missing Value During Analysis

Duplicate Data

- Data set may include data objects that are duplicates, or almost duplicates of one another
  - Major issue when merging data from heterogeneous sources
- Examples:
  - Same person with multiple email addresses
- Data cleaning
  - Process of dealing with duplicate data issues

The University of Reading

SE3DM11- Data Mining

Data Preprocessing

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Data Preprocessing:

- Aggregation
- Sampling
- Dimensionality Reduction
- Feature subset selection
- Feature creation
- Discretization and Binarization
- Attribute Transformation

Aggregation

- Combining two or more attributes (or objects) into a single attribute (or object)
- Purpose
  - Data reduction
    - Reduce the number of attributes or objects
  - Change of scale
    - Cities aggregated into regions, states, countries, etc
  - More “stable” data
    - Aggregated data tends to have less variability

Sampling

- Sampling is the main technique employed for data selection.
  - It is often used for both the preliminary investigation of the data and the final data analysis.
- Statisticians sample because obtaining the entire set of data of interest is too expensive or time consuming.
- Sampling is used in data mining because processing the entire set of data of interest is too expensive or time consuming.

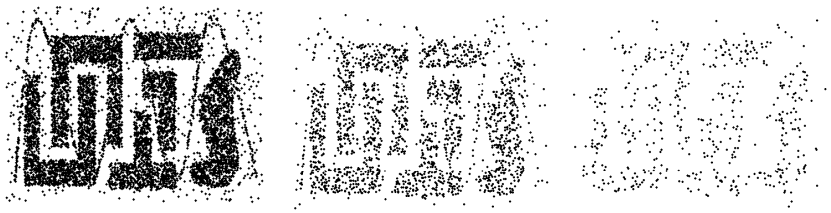
Sampling

- The key principle for effective sampling is the following:
  - using a sample will work almost as well as using the entire data sets, if the sample is representative.
  - A sample is representative if it has approximately the same property (of interest) as the original set of data

Types of Sampling

- Simple Random Sampling
  - There is an equal probability of selecting any particular item
- Sampling without replacement
  - As each item is selected, it is removed from the population
  - Each outcome depends on all previous outcomes
- Sampling with replacement
  - Objects are not removed from the population as they are selected for the sample.
    - In sampling with replacement, the same object can be picked up more than once
    - One outcome does not affect the other outcomes
- Stratified sampling
  - Split the data into several partitions; then draw random samples from each partition

Sample Size



8000 points **2000 Points** **500 Points**

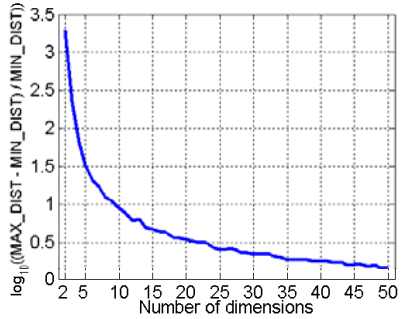
Curse of Dimensionality

- The 'curse of dimensionality' (Bellmann, 1961) refers to phenomena that arise when analysing data with a 'large' number of attributes. When dimensionality increases, data becomes increasingly sparse in the space that it occupies.
- Definitions of *density* and *distance* between points, which is critical, e.g., for clustering and outlier detection become less meaningful.

Example

- **Randomly generate 500 points in** $\Re^n$
- **Compute difference between max and min distance between any pair of points**

$$f(n) = \log_{10}\left(\frac{\max(dist(v_i, v_j)) - \min(dist(v_i, v_j))}{\min(dist(v_i, v_j))}\right)$$

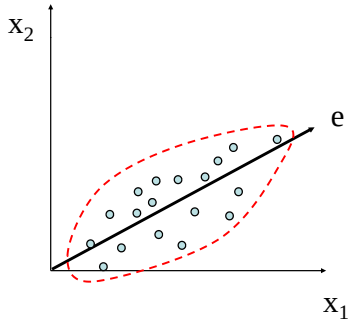


Dimensionality Reduction

- Purpose:
  - Avoid curse of dimensionality
  - Reduce amount of time and memory required by data mining algorithms
  - Allow data to be more easily visualized
  - May help to eliminate irrelevant features or reduce noise
- Techniques
  - Principle Component Analysis
  - Singular Value Decomposition
  - Others: supervised and non-linear techniques

Dimensionality Reduction: PCA

- Goal is to find a projection that captures the largest amount of variation in data



Feature Subset Selection

- Main benefits when learning models from data:
  - redundant and irrelevant features are identified and discarded
  - may solve or at least alleviate the curse of dimensionality
  - model interpretability is improved
  - generalisation is improved (by reducing overfitting)
  - training times are shortened
- Redundant features
  - duplicate much or all of the information contained in one or more other attributes
  - Example: purchase price of a product and the amount of sales tax paid
- Irrelevant features
  - contain no information that is useful for the data mining task at hand
  - Example: students' ID is often irrelevant to the task of predicting students' Grade Point Average (GPA)

Feature Subset Selection

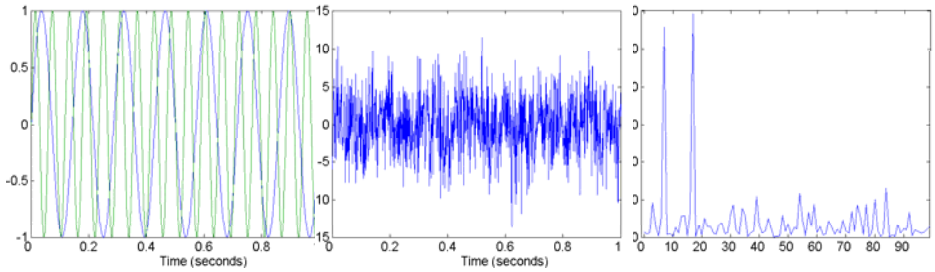
- Techniques:
  - Brute-force approach:
    - Try all possible feature subsets as input to data mining algorithm
  - Embedded approaches:
    - Feature selection occurs naturally as part of the data mining algorithm
  - Filter approaches:
    - Features are selected before data mining algorithm is run
  - Wrapper approaches:
    - Use the data mining algorithm as a black box to find best subset of attributes

Feature Creation

- Create new attributes that can capture the important information in a data set much more efficiently than the original attributes
- Three general methodologies:
  - Feature Extraction
    - domain-specific
  - Mapping Data to New Space
  - Feature Construction
    - combining features

Example: Mapping Data to a New Space

- **Fourier transform**
- **Wavelet transform**

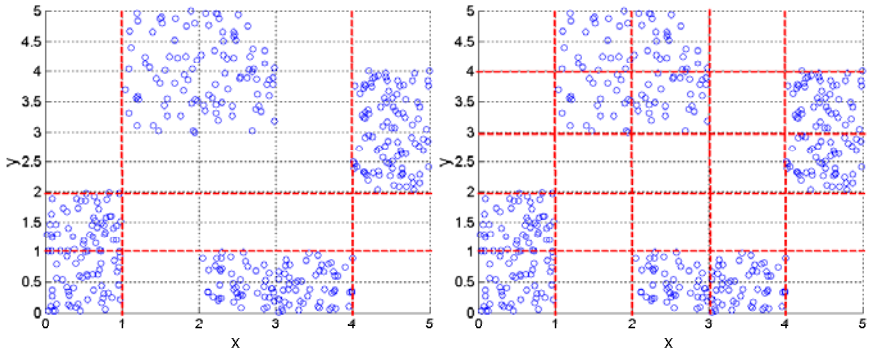


Two Sine Waves **Two Sine Waves + Noise** **Frequency**

Discretization and Binarization

- Different data mining applications require specific data formats
  - Categorical only (discretization)
  - Binary only (binarization)
  - Interval/Ratio only (binarization)
- Discretization: transforming interval attribute into categorical
- Binarization: transforming non-binary attribute into a set of binary attributes

Discretization Using Class Labels



3 categories for both x and y **5 categories for both x and y**

Attribute Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values.
  - E.g., simple functions: x^k, log(x), e^x, |x|
  - Normalization and Standardization
    - *Normalization* is the transformation of the variable vectors into vectors of unit length.
    - *Standardization* transforms the variable vector into a vector of unit length, with a mean of zero, and a standard deviation of one.

$$x' = \frac{(x - \mu_x)}{\sigma_x}$$

The University of Reading

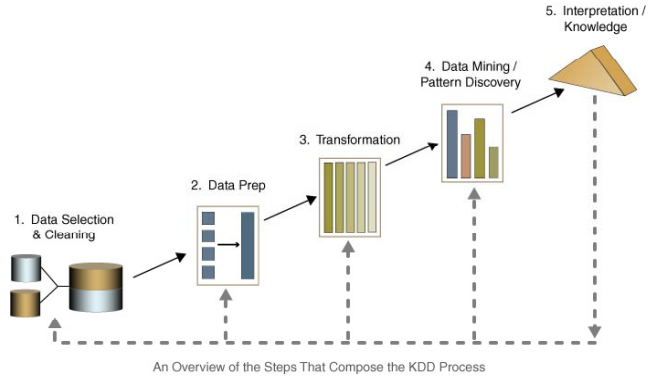
SE3DM11 - Data Mining

KDD Development Environments and KNIME

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

KDD Development Environments

- Increasing demand for integrated environments to facilitate the KDD process
- **Data mining workflow systems** that integrates analytical data mining methods for prediction, discovery, classification, etc., with data management and information visualization.



An Overview of the Steps That Compose the KDD Process

KDD Development Environments – Open Source

- **Weka** 3, Data Mining Software in Java
- **Orange**, a component-based data mining software (C++)
- **MLC++** is a library of C++ classes for supervised machine learning
- **D2K**, Data to Knowledge (Java)
- **KNIME**, Konstanz Information Miner (Java)

KDD Development Environments - Commercial

- **RapidMiner** (formerly **YALE**, Yet Another Learning Environment) (Java) – free trial version available
- Pentaho – free trial/light version available
  Also free community edition: http://community.pentaho.com/
  (Note: Pentaho Data Mining used Weka)
- IBM SPSS (Statistical Package for Social Science)
- SAS

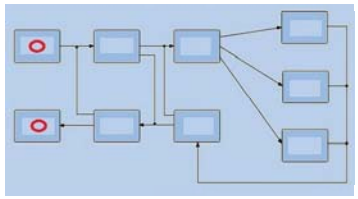
Flow-based Programming

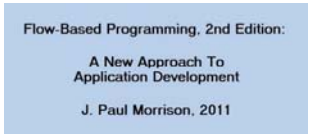
- Invented by JPR Morrison in the 1970s
- Flow-based Programming is a programming paradigm that defines applications as networks of "black box" processes.
- Processes exchange data across predefined connections by message passing.
- Flow-based Programming is intrinsically component-oriented.
  - The processes can be reused and reconnected endlessly to form different applications without having to be modified internally.

Πάντα ῥεῖ

Panta rhei

Everything flows





Introducing KNIME

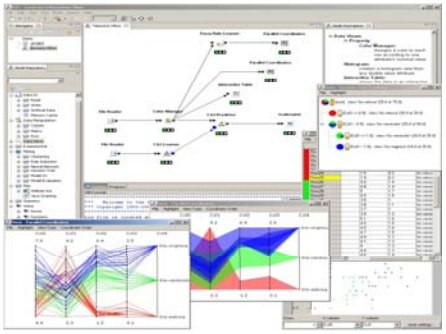
- Developed at the **ALTANA-Chair for Bioinformatics and Information Mining**, Department of Computer and Information Science, University of Konstanz, Germany
- 1st release: ~140 MM effort in two years
- Under continuous evolution and extension
  - April 2006: 1st release
  - Dec. 2008: version 2.0.0 released
  - Current release: version 2.9.1
    - KNIME Desktop and KNIME SDK
    - Available for Windows, Linux, and Mac OS X.

My presentation of the first release of KNIME given in 2006:

M. Berthold, N. Cebron, F. Dill, G. Di Fatta, T. Gabriel, F. Georg, T. Meinl, P. Ohl, C. Sieb, B. Wiswedel, "KNIME: the Konstanz Information Miner", Proc. of Workshop on Multi-Agent Systems and Simulation (MAS&S), 4th Annual Industrial Simulation Conference (ISC), Palermo, Italy, June 5-7, 2006, pp.58-61.

KNIME

Knime: Interactive Data Exploration



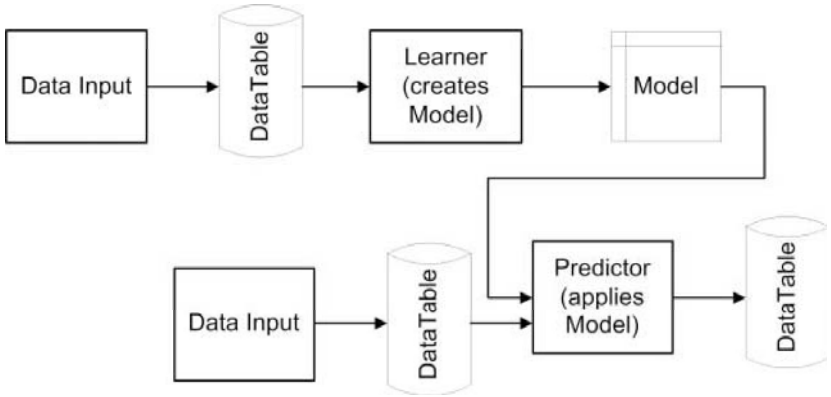
Features:

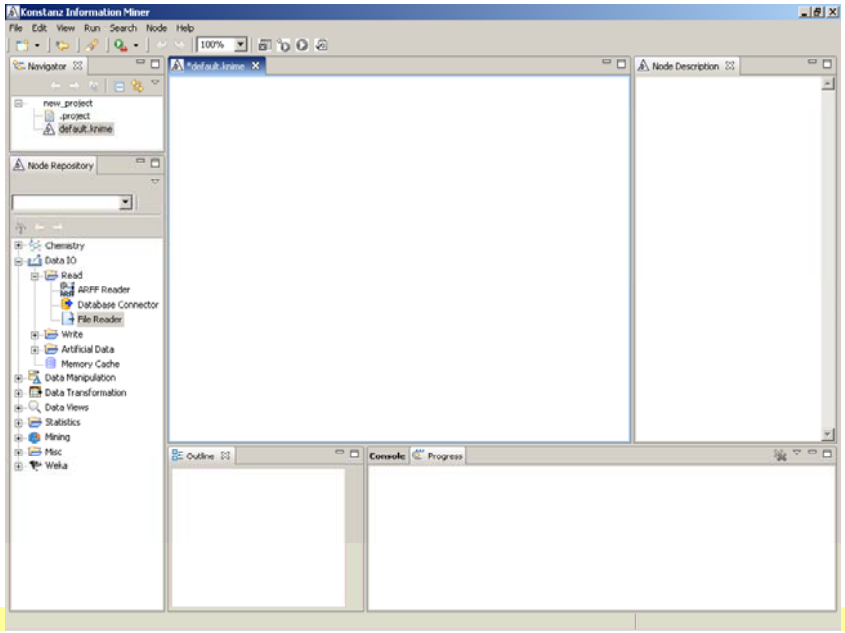
- Modular Data Pipeline Environment
- Large collection of Data Mining techniques
- Data and Model Visualizations
- Interactive Views on Data and Models
- Java Code Base as Open Source Project
- Seamless Integration: R Library, Weka, etc.
- Based on the Eclipse Plug-in technology

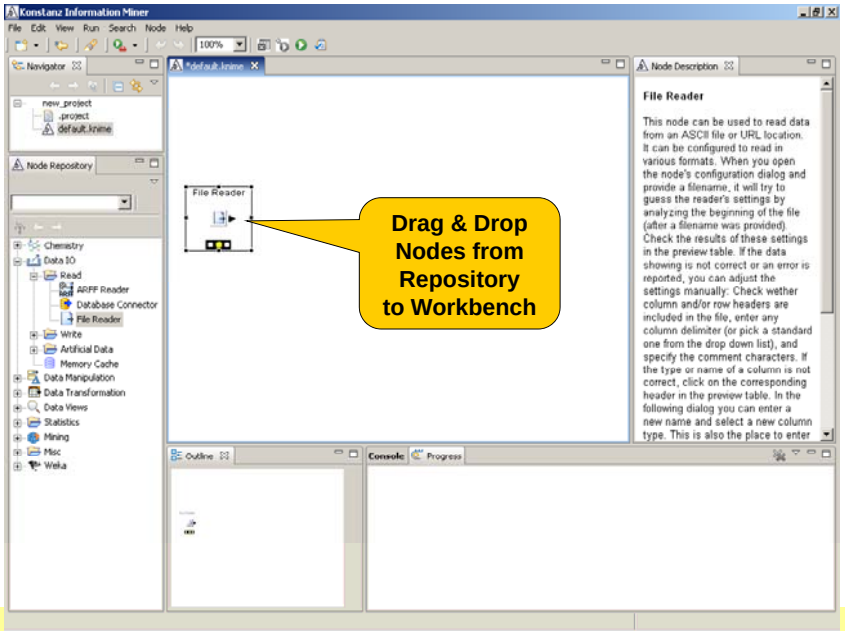
Easy extendibility

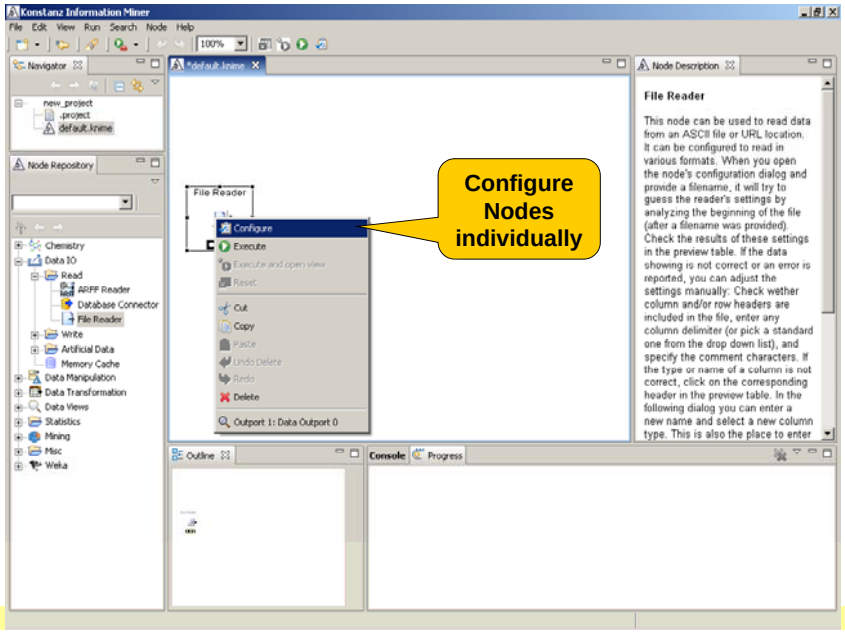
New nodes via open API and integrated wizard

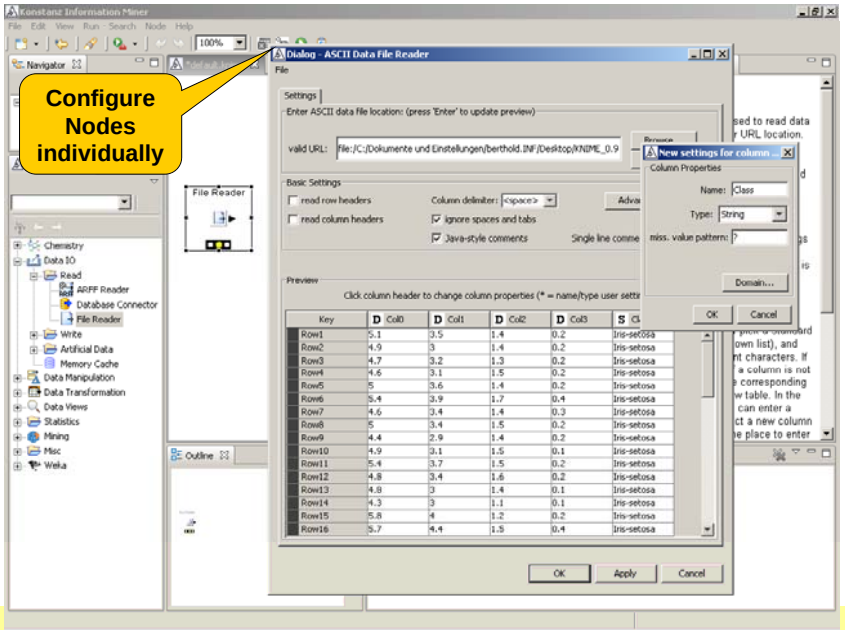
Data Pipeline



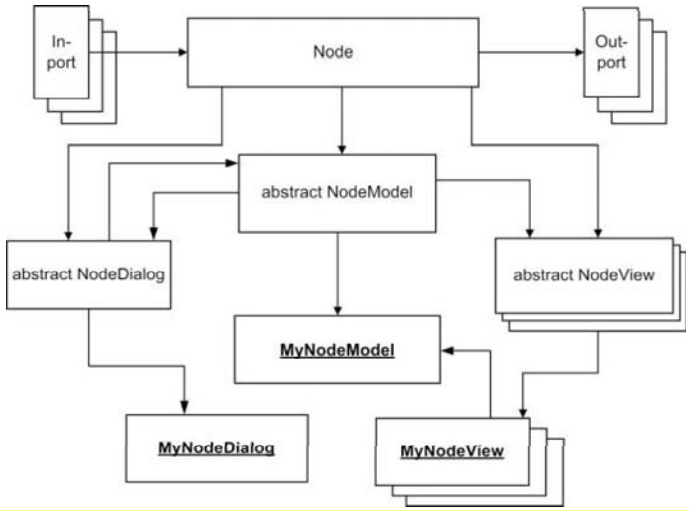

Konstanz Information Miner
File Edit View Run Search Node Help
Navigator
Node Repository
Node Description
Outline
Console
Progress

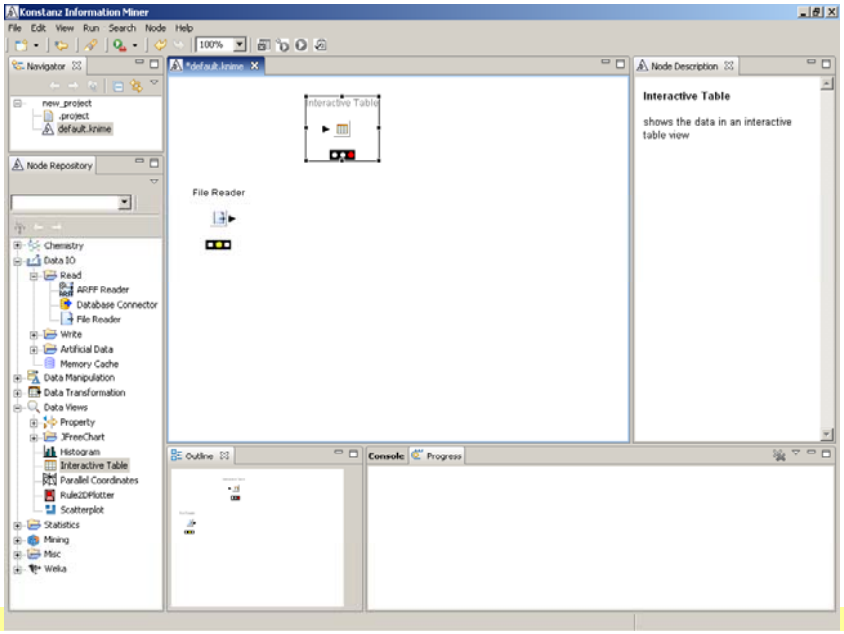

Konstanz Information Miner
File Edit View Run Search Node Help
Navigator
Node Repository
Node Description
File Reader
Drag & Drop Nodes from Repository to Workbench
Outline
Console
Progress

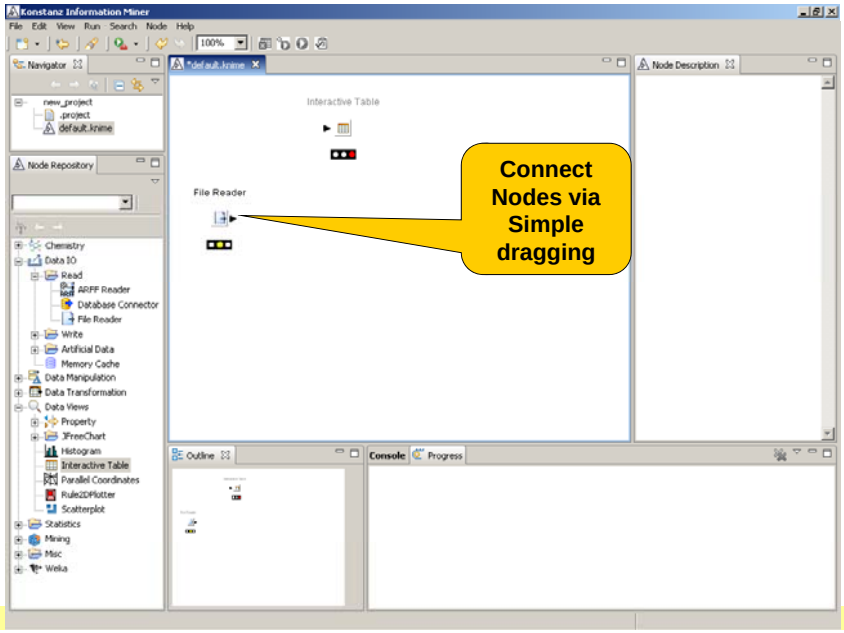

Konstanz Information Miner
File Edit View Run Search Node Help
Navigator
Node Repository
Node Description
File Reader
Configure
Execute
Cut
Copy
Delete
Configure Nodes individually
Outline
Console
Progress

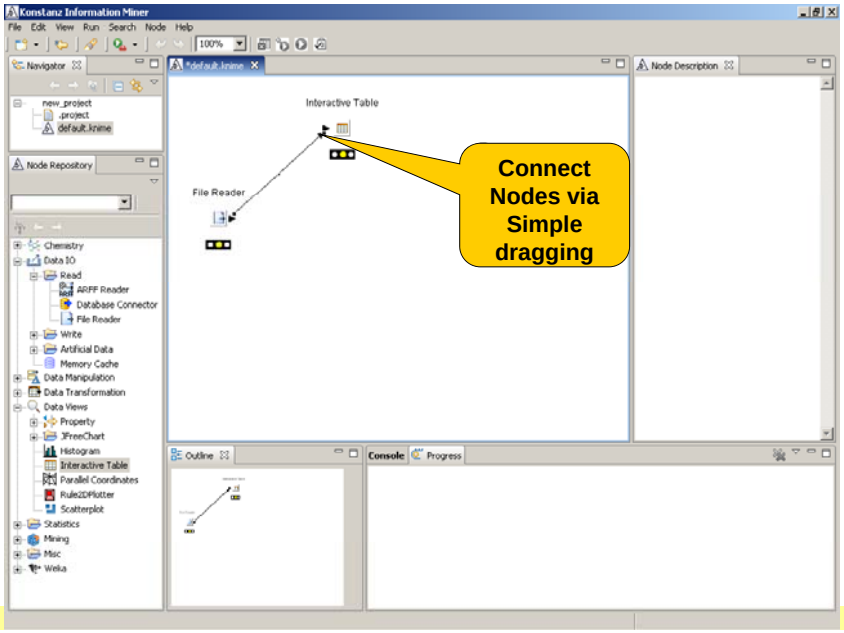

Konstanz Information Miner
Dialog - ASCII Data File Reader
Configure Nodes individually
Navigator
Node Repository
File Reader
Outline
Console
Progress

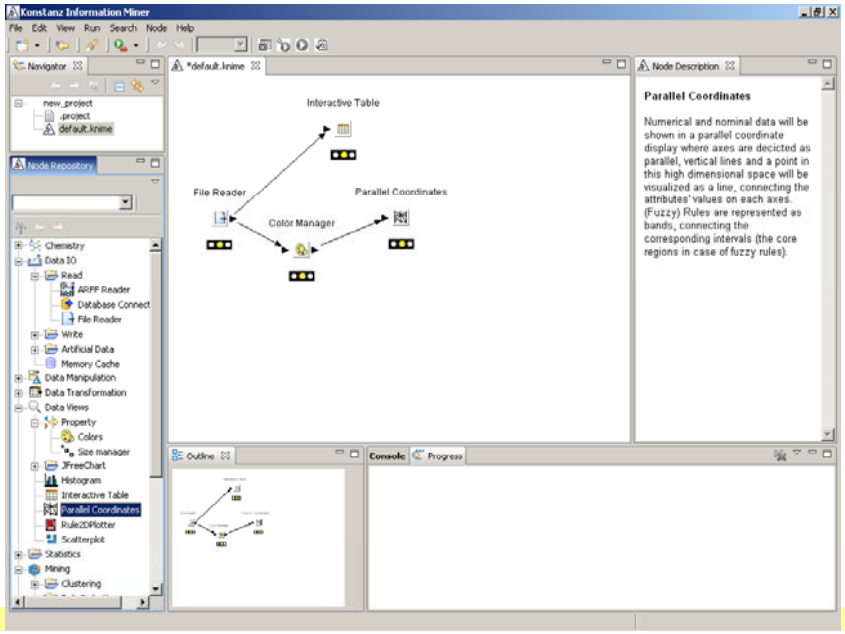
Node Model


In-port
Node
Out-port
abstract NodeModel
abstract NodeDialog
abstract NodeView
MyNodeModel
MyNodeDialog
MyNodeView

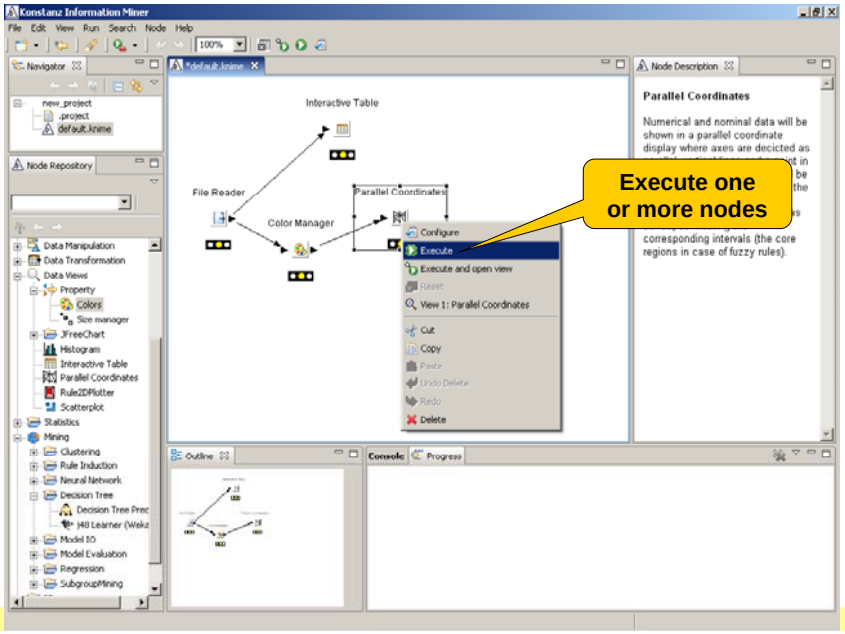

Konstanz Information Miner
Interactive Table
shows the data in an interactive table view
File Reader


Konstanz Information Miner
Interactive Table
File Reader
Connect Nodes via Simple dragging

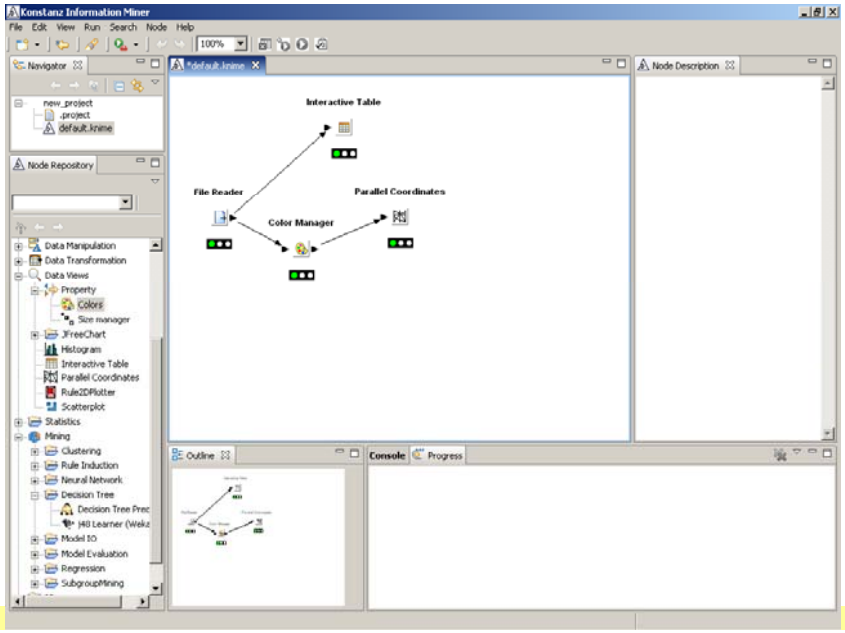

Konstanz Information Miner
Interactive Table
File Reader
Connect Nodes via Simple dragging


Konstanz Information Miner
Interactive Table
File Reader
Parallel Coordinates
Color Manager
Parallel Coordinates
Numerical and nominal data will be shown in a parallel coordinate display where axes are decicted as parallel, vertical lines and a point in this high dimensional space will be visualized as a line, connecting the attributes' values on each axes. (Fuzzy) Rules are represented as bands, connecting the corresponding intervals (the core regions in case of fuzzy rules).

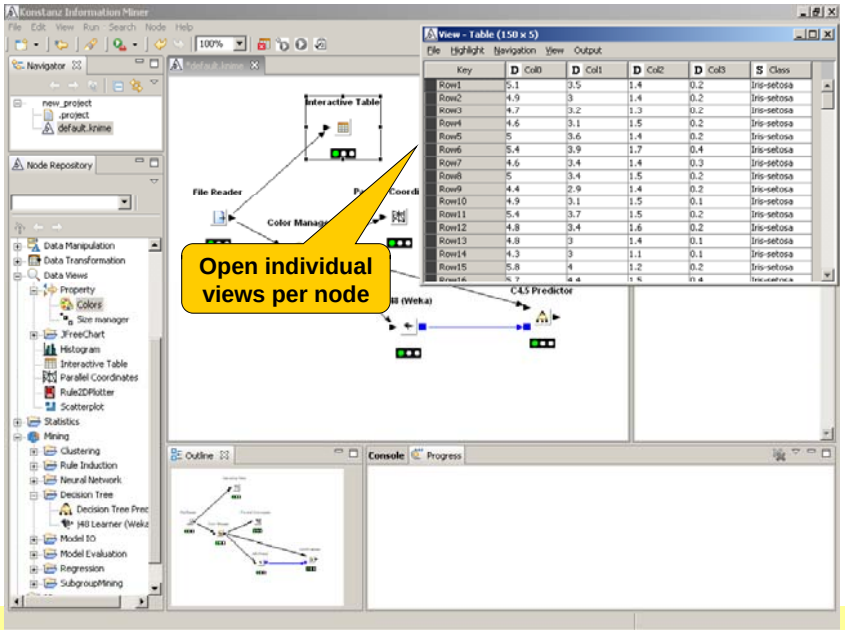
#17


Konstanz Information Miner
Interactive Table
File Reader
Parallel Coordinates
Color Manager
Execute one or more nodes
Configure
Execute
Execute and open view
Reset
View 1: Parallel Coordinates
Cut
Copy
Paste
Undo Delete
Redo
Delete
Parallel Coordinates
Numerical and nominal data will be shown in a parallel coordinate display where axes are decicted as
corresponding intervals (the core regions in case of fuzzy rules).

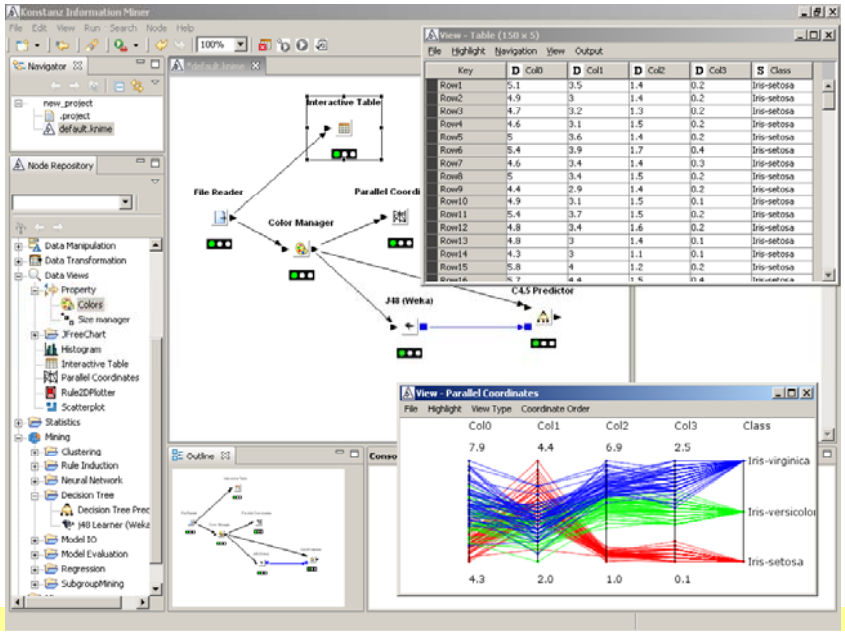
#18


Konstanz Information Miner
Interactive Table
File Reader
Parallel Coordinates
Color Manager

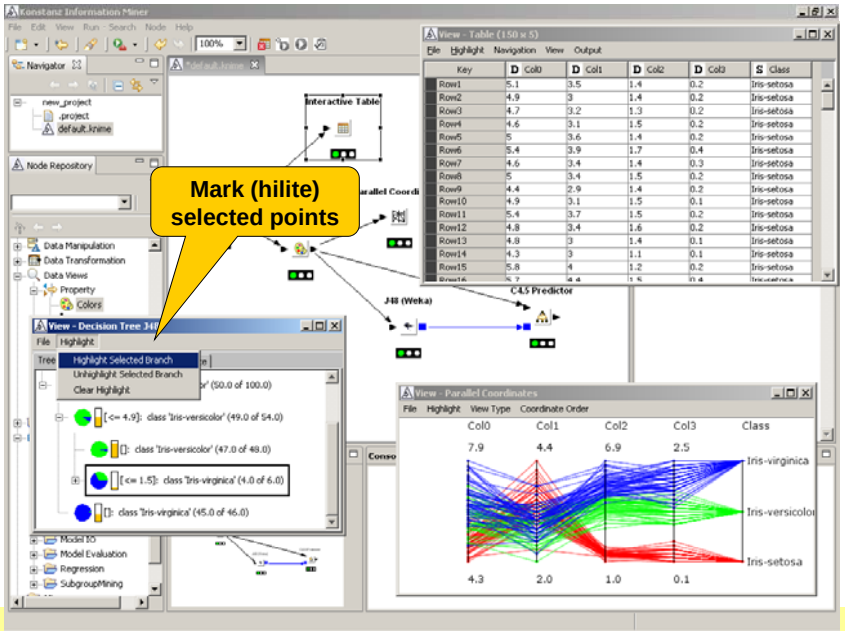
#19


Konstanz Information Miner
View - Table (150 x 5)
Interactive Table
File Reader
Color Manager
Open individual views per node
C4.5 Predictor

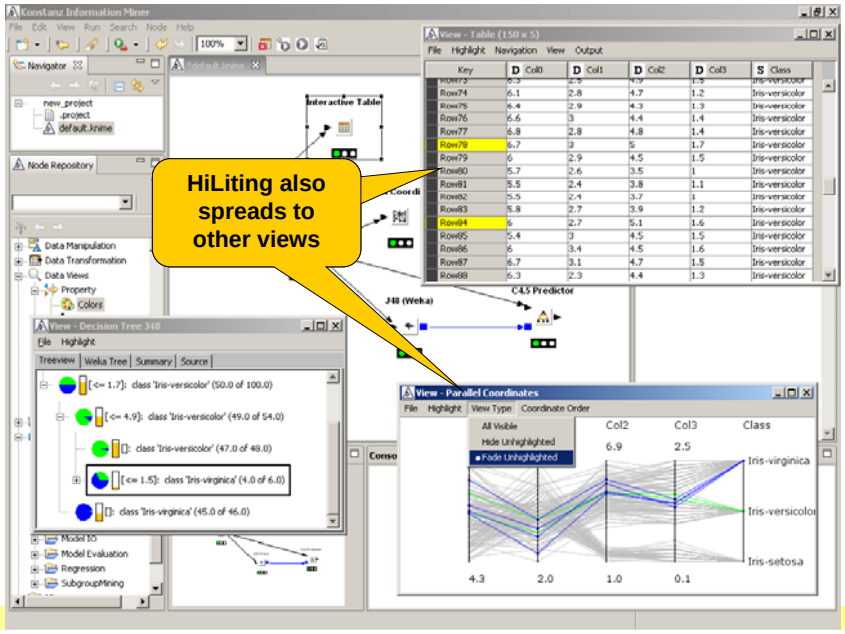
#20


Interactive Table
File Reader
Color Manager
Parallel Coordi
C4.5 Predictor
J48 (Weka)
View - Parallel Coordinates
Col0
Col1
Col2
Col3
Class
Iris-virginica
Iris-versicolor
Iris-setosa

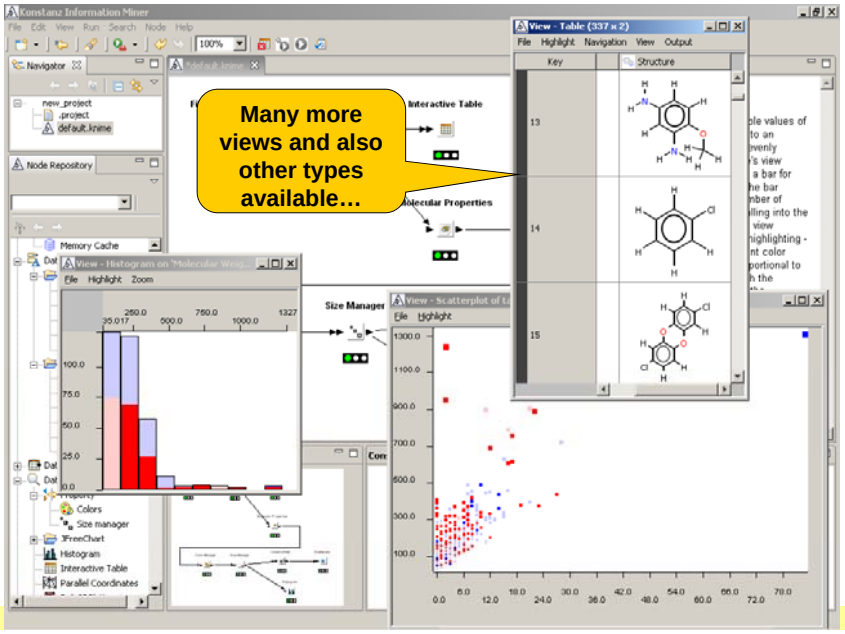
#21


Mark (hilite) selected points
Interactive Table
C4.5 Predictor
J48 (Weka)
View - Decision Tree J48
Highlight Selected Branch
Unhighlight Selected Branch
Clear Highlight
View - Parallel Coordinates
Col0
Col1
Col2
Col3
Class
Iris-virginica
Iris-versicolor
Iris-setosa

#22


HiLiting also spreads to other views
Interactive Table
C4.5 Predictor
J48 (Weka)
View - Decision Tree J48
View - Parallel Coordinates
All Visible
Hide Unhighlighted
Fade Unhighlighted
Col2
Col3
Class
Iris-virginica
Iris-versicolor
Iris-setosa

#23


Many more views and also other types available…
Interactive Table
Size Manager

#24

KNIME: useful resources for first steps

KNIME User → desktop version:

- http://www.knime.org/knime-desktop
- http://www.knime.org/downloads/datasets
- http://www.knime.org/introduction/examples

Workflow examples:

- parallel coordinates on Iris data
- PCA on Iris data
- decision tree on Iris data

KNIME Developer → SDK version (Eclipse):

- http://tech.knime.org/developer-guide
- http://tech.knime.org/developer/example
- API: for example see the DataTable interface in http://tech.knime.org/docs/api/org/knime/core/data/package-summary.html
- Simple exercise: create a new KNIME node to compute column stats

Rosaria's blog with lots of resources:

- http://www.dataminingreporting.com/

Conclusions on KNIME

- Modularity and extendibility
  - General and extendible data structure (DataTable and DataCell)
  - Nodes encapsulate computational processing tasks (algorithms)
- A workflow management system
  - directed edges connects nodes to create data pipelines
  - a workflow is, in general, a directed acyclic graph
  - multi-threading
  - Meta-nodes (nested workflows)
- New release
  - Enhanced GUI
  - Contains many more modules and features

The University of Reading

SE3DM11 - Data Mining

Proximity Measures

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Overview

- Proximity measures: similarity and dissimilarity
- Transformations
- Proximity between objects with a single attribute
- Proximity between objects with multiple attributes
- Useful proximity measurements
  - Dense data: correlation, Euclidian distance
  - Sparse data: Jaccard and cosine similarity measures

Slides modified from the original slides of "Tan,Steinbach, Kumar, Introduction to Data Mining, Pearson Int. Ed."

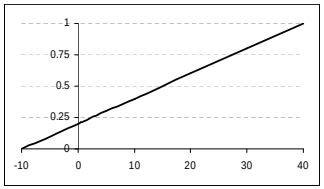
Similarity and Dissimilarity

- Similarity
  - Numerical measure of how alike two data objects are.
  - Is higher when objects are more alike.
  - Often falls in the range [0,1]
- Dissimilarity
  - Numerical measure of how different are two data objects
  - Lower when objects are more alike
  - Minimum dissimilarity is often 0
  - Upper limit varies (usually 1 or ∞)
- Proximity refers to a similarity or dissimilarity

Transformations

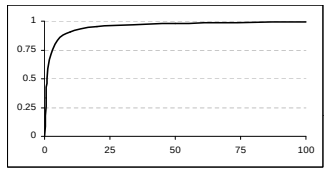
- Convert similarity to a dissimilarity, or vice versa. If both in the interval [0,1] it is straightforward: s=1-d
- Transform a proximity measure to fall within a particular range. For example [0,1]:

$$m' = \frac{(m - m_{\min})}{(m_{\max} - m_{\min})}$$



- If the proximity measure is defined in [0, ∞] then a non-linear transformation is needed. For example, for dissimilarity d:

$$d' = \frac{d}{1+d}$$



Similarity/Dissimilarity

Similarity/Dissimilarity for objects with a single attribute

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$
Ordinal	$d = \frac{\lvert p-q \rvert}{n-1}$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$d = \lvert p - q \rvert$	$s = -d$, $s = \frac{1}{1+d}$ or $s = 1 - \frac{d - min\_d}{max\_d - min\_d}$

p and *q* are the attribute values for two data objects.

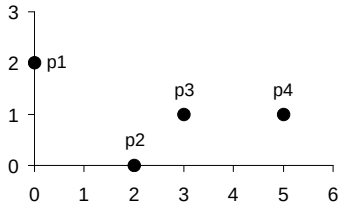
Distance as Dissimilarity Between Objects

- Distances are normally used to measure the dissimilarity (similarity) between two data objects.
- A distance that satisfies specific properties is a metric.
- A Metric is a function d(x,y) on a pair of points (x,y) with the following properties:
  1. $d(x,y) \geq 0$, $d(x,y) = 0$ if x=y (positive definiteness)
  2. $d(x,y) = d(y,x)$ *(symmetry)*
  3. $d(x,y) \leq d(x,k) + d(k,y)$ *(triangular inequality)*
- For example, the well known Euclidian distance is defined as

- Standardization is necessary, if scales differ.

$$d_E(x,y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

Euclidean Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distance Matrix

Minkowski Distance

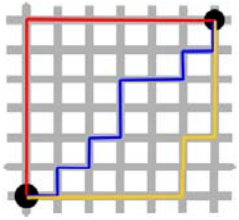
- The Minkowski Distance is a generalization of the Euclidean Distance

$$dist = \left(\sum_{k=1}^{n} \lvert p_k - q_k \rvert^r\right)^{\frac{1}{r}}$$

where *r* is a parameter, *n* is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects *p* and *q*.

Minkowski Distance: Examples

- r = 1. City block (Manhattan, taxicab, L_1 norm) distance.
  - A common example of this is the Hamming distance, which is just the number of bits that are different between two binary vectors



- r = 2. Euclidean distance (L_2 norm)
- $r \to \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component of the vectors
- Do not confuse r with n.

Minkowski Distance

point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

L_∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix

Standardization and Correlation

- Issues with distance measures:
  - Attributes may not have same range of values: "the variables have different scales". In this case, attributes can be/need to be standardized.
  - Some of the attributed may also be correlated.
- If the data distribution are approximately Gaussian, then the Mahalanobis distance can be used. It is a generalization of the Euclidian distance which takes these issues into account.

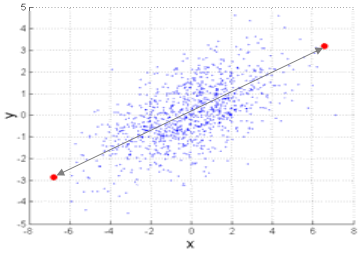
Mahalanobis Distance

The Statistical, or Mahalanobis, Distance: $d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma^{-1} (\vec{x}-\vec{y})}$

- It is the distance between two multi-dimensional points scaled by the statistical variation in each component.
- Useful for comparing feature vectors whose elements are quantities having different ranges and amounts of variation.
- It also takes into account the correlation among components.

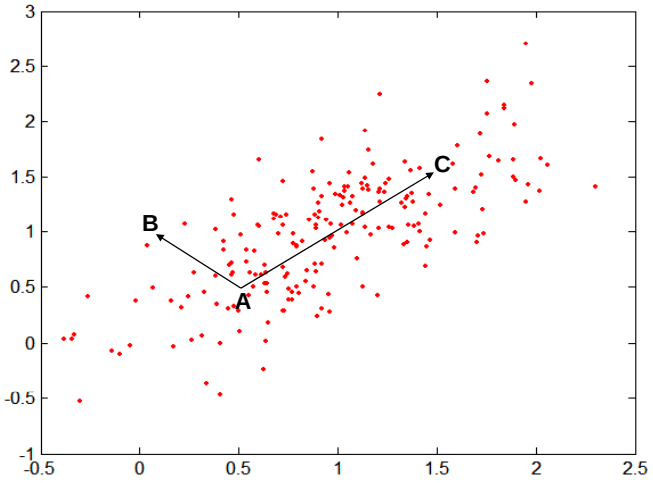
where Σ is the covariance matrix of the input data *X:*

$$\Sigma_{j,k} = \frac{1}{N-1}\sum_{i=1}^{N}(X_{ij}-\overline{X}_j)(X_{ik}-\overline{X}_k)$$



For red points, the Euclidean distance is 14.7, the Mahalanobis distance is 6.

Mahalanobis Distance



Covariance Matrix:

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

D_E(A,B) = 0.7

D_E(A,C) = 1.4

D_M(A,B) = 5

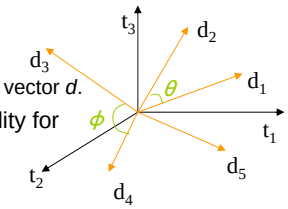
D_M(A,C) = 4

Cosine Similarity

- If d_1 and d_2 are two vectors, then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \, \|d_2\| ,$$

where • indicates vector dot product and $\|d\|$ is the length of vector d.



- Note: this is *similarity*, not distance. No triangle inequality for similarity.
- Example:

d_1 = **3 2 0 5 0 0 0 2 0 0**

d_2 = **1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2$= 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5

$\|d_1\|$ = (3*3+2*2+0*0+5*5+0*0+0*0+0*0+2*2+0*0+0*0)$^{0.5}$ = (42) $^{0.5}$ = 6.481

$\|d_2\|$ = (1*1+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2) $^{0.5}$ = (6) $^{0.5}$ = 2.245

$$\cos(d_1, d_2) = 0.3150$$

- Other (Dis-)Similarity measures: Pearson Correlation, Extended Jaccard Similarity, etc.

Common Properties of a Similarity

- Similarities, also have some well known properties.

1. *s(p, q) = 1* (or maximum similarity) only if *p = q*.
2. *s(p, q) = s(q, p)* for all *p* and *q*. (Symmetry)

where *s(p, q)* is the similarity between points (data objects), *p* and *q*.

Similarity Between Binary Vectors

- Common situation is that objects, *p* and *q*, have only binary attributes. Compute similarities using the following quantities:

M_{01} = the number of attributes where p was 0 and q was 1
M_{10} = the number of attributes where p was 1 and q was 0
M_{00} = the number of attributes where p was 0 and q was 0
M_{11} = the number of attributes where p was 1 and q was 1

- The Simple Matching Coefficient (SMC)

SMC = number of matches / number of attributes
= $(M_{11} + M_{00}) / (M_{01} + M_{10} + M_{11} + M_{00})$

- The Jaccard index, aka the Jaccard Similarity Coefficient

The Jaccard index (originally coined coefficient de communauté by Paul Jaccard) is a statistic used for comparing the similarity and diversity of sample sets.
The Jaccard coefficient measures similarity between sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

J = number of "1-1" matches / number of not-both-zero attributes values
= $(M_{11}) / (M_{01} + M_{10} + M_{11})$

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

SMC versus Jaccard: Example

p = 1 0 0 0 0 0 0 0 0 0
q = 0 0 0 0 0 0 1 0 0 1

M_{01} = 2 (the number of attributes where p was 0 and q was 1)
M_{10} = 1 (the number of attributes where p was 1 and q was 0)
M_{00} = 7 (the number of attributes where p was 0 and q was 0)
M_{11} = 0 (the number of attributes where p was 1 and q was 1)

SMC = $(M_{11} + M_{00})/(M_{01} + M_{10} + M_{11} + M_{00})$ = (0+7) / (2+1+0+7) = 0.7

J = (M_{11}) / $(M_{01} + M_{10} + M_{11})$ = 0 / (2 + 1 + 0) = 0

Hamming Distance

- In **Information Theory**, the Hamming distance between two strings of equal length is **the number of positions at which the corresponding symbols are different**.
- It measures the minimum number of substitutions required to change one string into the other, or the number of errors that transformed one string into the other.
- For binary data it corresponds to the L1 distance.
  - Hamming = $(M_{10} + M_{01})/(M_{01} + M_{10} + M_{11} + M_{00})$
- It's a distance, while SMC and Jaccard coefficients are similarities:
  - d = 1-s
  - In particular, Hamming = 1-SMC
- The Hamming distance is also used as a measure of genetic distance.

Cosine Similarity

- If d_1 and d_2 are two document vectors, then
  cos(d_1, d_2) = $(d_1 \bullet d_2) / ||d_1|| \, ||d_2||$,

where • indicates vector dot product and || *d* || is the length of vector *d*.

- Example:

  d_1 = **3 2 0 5 0 0 0 2 0 0**
  d_2 = **1 0 0 0 0 0 0 1 0 2**

$d_1 \bullet d_2$ = 3*1 + 2*0 + 0*0 + 5*0 + 0*0 + 0*0 + 0*0 + 0*0 + 2*1 + 0*0 + 0*2 = 5
$||d_1||$ = $(3*3+2*2+0*0+5*5+0*0+0*0+0*0+0*0+2*2+0*0+0*0)^{0.5}$ = $(42)^{0.5}$ = 6.481
$||d_2||$ = $(1*1+0*0+0*0+0*0+0*0+0*0+0*0+0*0+1*1+0*0+2*2)^{0.5}$ = $(6)^{0.5}$ = 2.245

cos(d_1, d_2) = 0.3150

Extended Jaccard Coefficient (Tanimoto)

- The cosine similarity metric may be modified such that it yields the Jaccard coefficient in the case of binary attributes.
- It is known as the **Tanimoto coefficient**: T(A, B)
- It is the extension of the Jaccard coefficient for continuous or count attributes (it reduces to Jaccard for binary attributes).

$$T(p,q) = \frac{p \bullet q}{\|p\|^2 + \|q\|^2 - p \bullet q}$$

Correlation

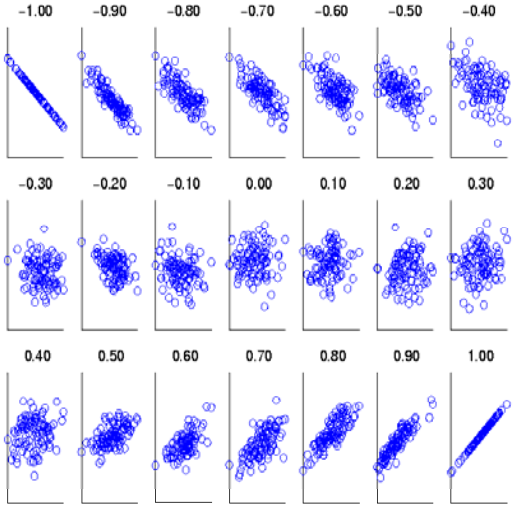
- Correlation measures the linear relationship between objects
- To compute correlation, we standardize data objects, p and q, and then take their dot product

$$p'_k = (p_k - mean(p)) / std(p)$$

$$q'_k = (q_k - mean(q)) / std(q)$$

$$correlation(p, q) = p' \bullet q'$$

Visually Evaluating Correlation



Scatter plots showing the similarity from –1 to 1.

General Approach for Combining Similarities

- Sometimes attributes are of many different types, but an overall similarity is needed.

1. For the k^{th} attribute, compute a similarity, s_k, in the range $[0, 1]$.
2. Define an indicator variable, δ_k, for the k_{th} attribute as follows:

$$\delta_k = \begin{cases} 0 & \text{if the } k^{th} \text{ attribute is a binary asymmetric attribute and both objects have a value of 0, or if one of the objects has a missing values for the } k^{th} \text{ attribute} \\ 1 & \text{otherwise} \end{cases}$$

3. Compute the overall similarity between the two objects using the following formula:

$$similarity(p, q) = \frac{\sum_{k=1}^{n} \delta_k s_k}{\sum_{k-1}^{n} \delta_k}$$

Using Weights to Combine Similarities

- May not want to treat all attributes the same.
  - Use weights w_k which are between 0 and 1 and sum to 1.

$$similarity(p, q) = \frac{\sum_{k=1}^{n} w_k \delta_k s_k}{\sum_{k=1}^{n} \delta_k}$$

$$distance(p, q) = \left(\sum_{k=1}^{n} w_k |p_k - q_k|^r \right)^{1/r}$$

- Careful consideration of both domain knowledge and purpose for selecting the appropriate proximity measure

Summary Table for Proximity Measures

Proximity measure	M_{00}	M_{11}	M_{10}	M_{01}	*Binary /continuous attributes*
SMC	x	x			b
Jaccard similarity coefficent		x			b
Hamming distance			x	x	b
Cosine similarity		x			c
Tanimoto coefficient		x			c

The University of Reading

SE3DM11 - Data Mining

Clustering

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Clustering

- Cluster Analysis
- Similarity
- Types of Clusters
- Clustering Approaches
  - Partitioning
  - Hierarchical
  - Density-based
  - Grid-based
  - Model-based
- Cluster validity

Definition of “Cluster”

Definition of “Cluster” in online dictionaries:

- a number of similar things growing together or of things or persons collected or grouped closely together: BUNCH.
- two or more consecutive consonants or vowels in a segment of speech.
- a group of buildings and esp. houses built close together on a sizable tract in order to preserve open spaces larger than the individual yard for common recreation.
- an aggregation of stars, galaxies, or super galaxies that appear close together in the sky and seem to have common properties (as distance).
- **A cluster is a closely-packed group (of people or things).**

Clustering in Data Mining

- **Cluster Analysis** is the process of partitioning a set of data (or objects) in a set of meaningful sub-classes, called **clusters**.
  - It helps users to understand the natural grouping or structure in a data set.
- Cluster analysis = Grouping a set of data objects into clusters
- Cluster: a collection of data objects
  - similar to one another within the same cluster
  - dissimilar to the objects in other clusters
- Clustering is unsupervised classification:
  - no predefined classes

General Applications of Clustering

- Pattern Recognition
- Spatial Data Analysis
- Image Processing
- Economic Science (especially market research)
- WWW
  - Document classification
  - Cluster Weblog data to discover groups of similar access patterns

Examples:

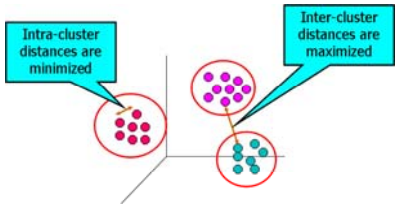
- Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
- Land use: Identification of areas of similar land use in an earth observation database
- Insurance: Identifying groups of motor insurance policy holders with a high average claim cost
- City-planning: Identifying groups of houses according to their house type, value, and geographical location
- Earth-quake studies: Observed earth quake epicenters should be clustered along continent faults

Requirements of Clustering in Data Mining

- Quality
- Scalability
- Ability to deal with different types of attributes
- Discovery of clusters with arbitrary shape
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- High dimensionality
- Incorporation of user-specified constraints
- Interpretability and usability

What Is Good Clustering?

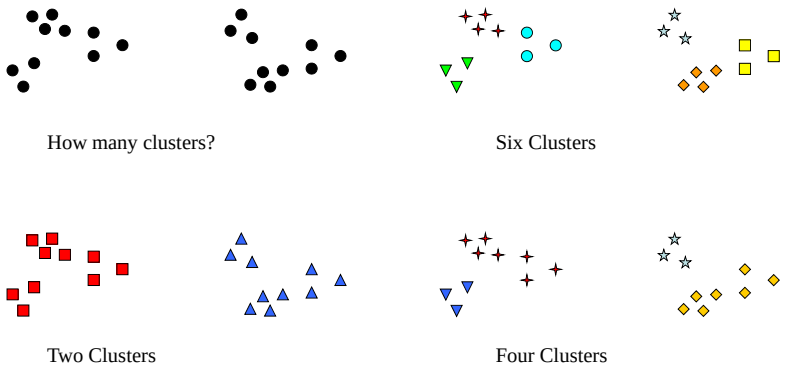
- A good clustering method will produce high quality clusters with
  - high intra-class similarity
  - low inter-class similarity
- The quality of a clustering result depends on both the similarity measure used by the method and its implementation.
- The quality of a clustering method is also measured by its ability to discover some or all of the hidden patterns.



Measuring Similarity

- Dissimilarity/Similarity metric: similarity is expressed in terms of a "**distance**" function, which is typically a Metric $d(x,y)$.
- There is a separate "**quality**" function that measures the "goodness" of a cluster.
- The definitions of distance functions are usually very different for boolean, nominal, ordinal, interval and ratio variables.
- Weights should be associated with different features based on applications and data semantics.
- It is hard to define "similar enough" or "good enough"
  - the answer is typically highly subjective.

The Notion of a Cluster can be Ambiguous



Other Distinctions Between Sets of Clusters

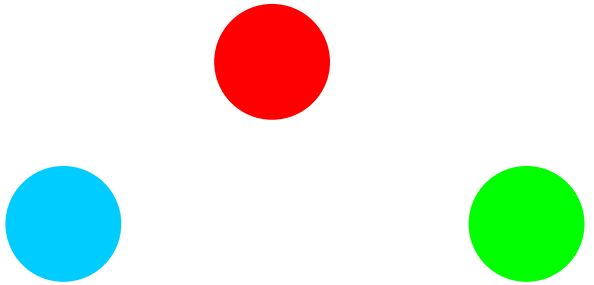
- Exclusive versus non-exclusive
  - In non-exclusive clusterings, points may belong to multiple clusters.
  - Can represent multiple classes or 'border' points
- Fuzzy versus non-fuzzy
  - In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1
  - Weights must sum to 1
  - Probabilistic clustering has similar characteristics
- Partial versus complete
  - In some cases, we only want to cluster some of the data
- Heterogeneous versus homogeneous
  - Cluster of widely different sizes, shapes, and densities

Types of Clusters

- Well-separated clusters
- Center-based clusters
- Contiguous clusters
- Density-based clusters
- Property or Conceptual
- Described by an Objective Function

Types of Clusters: Well-Separated

- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

Types of Clusters: Center-Based

- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the “center” of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most “representative” point of a cluster

4 center-based clusters

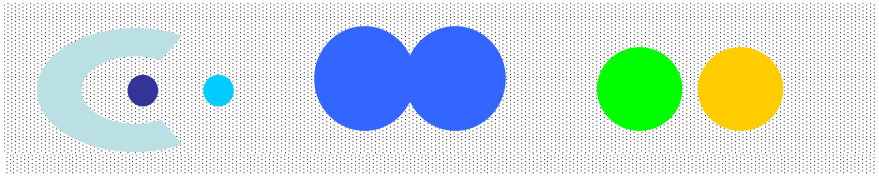
Types of Clusters: Contiguity-Based

- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.

8 contiguous clusters

Types of Clusters: Density-Based

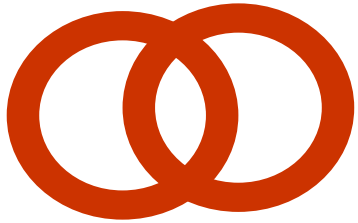
- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.



6 density-based clusters

Types of Clusters: Conceptual Clusters

- Shared Property or Conceptual Clusters
  - Finds clusters that share some common property or represent a particular concept.



2 Overlapping Circles

Types of Clusters: Objective Clusters

- Clusters Defined by an Objective Function
  - Finds clusters that minimize or maximize an objective function.
  - Enumerate all possible ways of dividing the points into clusters and evaluate the `goodness' of each potential set of clusters.
  - Can have global or local objectives.
    - Hierarchical clustering algorithms typically have local objectives
    - Partitional algorithms typically have global objectives

Clustering Approaches

1. **Partitioning algorithms**: Construct various partitions and then evaluate them by some criterion
2. **Hierarchy algorithms**: Create a hierarchical decomposition of the set of data (or objects) using some criterion
3. **Density-based**: based on connectivity and density functions
4. **Grid-based**: based on a multiple-level granularity structure
5. **Model-based**: A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other

Partitioning algorithms

- **Partitioning method:** construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters.
- Given a *k*, find a partition of *k clusters* that optimizes the chosen partitioning criterion.
  - Global optimal: exhaustively enumerate all partitions.
  - Heuristic methods: *k-means* and *k-medoids* algorithms.
    - *k-means* (MacQueen'67): each cluster is represented by the center of the cluster.
    - *k-medoids* or PAM (Partition Around Medoids) (Kaufman & Rousseeuw'87): each cluster is represented by one of the objects in the cluster.

K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, K, must be specified
- The simple basic algorithm:

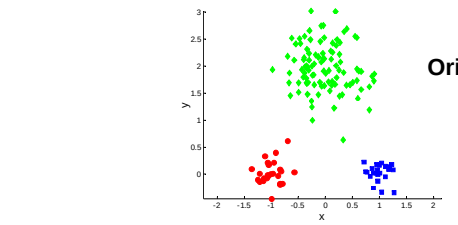
1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-means Clustering – Details

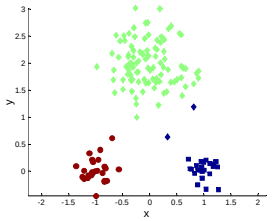
- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change cluster'
- Complexity is O(n * K * I * d)
  - n = number of points, K = number of clusters, I = number of iterations, d = number of attributes

Two different K-means Clusterings

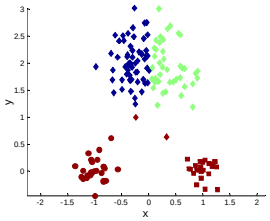
- Importance of choosing initial centroids



Original Points



Optimal Clustering



Sub-optimal Clustering

Evaluating K-means Clusters

- Most common measure is Sum of Squared Error (SSE)
  - For each point, the error is the distance to the nearest cluster
  - The objective function is:

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$$

  - x is a data point in cluster C_i and m_i is the representative point for cluster C_i
    - it can be shown that m_i corresponds to the center (mean) of the cluster
  - Given two clusters, we can choose the one with the smallest error
  - One easy way to reduce SSE is to increase K, the number of clusters
    - A good clustering with smaller K can have a greater SSE than a bad clustering with higher K.

Solutions to Initial Centroids Problem

- Multiple runs
  - Helps, but probability is not on your side
- Sample and use hierarchical clustering to determine initial centroids
- Select more than k initial centroids and then select among these initial centroids
  - Select most widely separated
- Post-processing attempts to "fix-up" the clustering
- Bisecting K-means
  - Not as susceptible to initialization issues

Handling Empty Clusters

- Basic K-means algorithm can yield empty clusters
- Several strategies
  - Choose the point that contributes most to SSE
  - Choose a point from the cluster with the highest SSE
- If there are several empty clusters, the above can be repeated several times.

Pre-processing and Post-processing

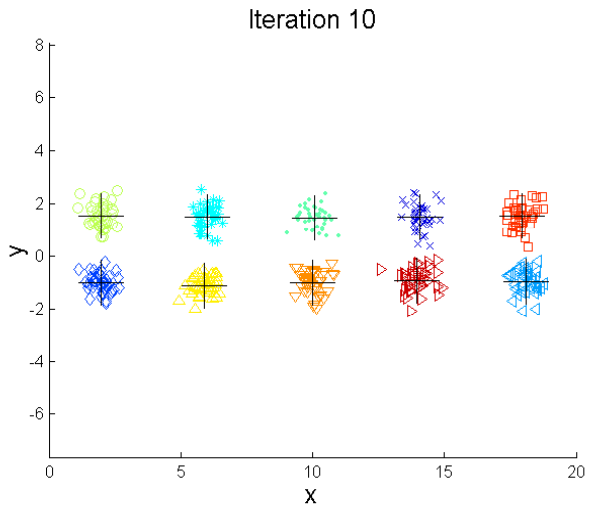
- Pre-processing
  - Normalize the data
  - Eliminate outliers
- Post-processing
  - eliminate “small” clusters since they may represent groups of outliers
  - split ‘loose’ clusters, i.e., clusters with relatively high SSE
  - merge clusters that are ‘close’ and that have relatively low SSE
  - can also use these steps during the clustering process
    - e.g., the ISODATA algorithm (Jensen, 1996)

Bisecting K-means

- Bisecting K-means algorithm
  - Variant of K-means that can produce a partitional or a hierarchical clustering

1: Initialize the list of clusters to contain the cluster containing all points.
2: **repeat**
3: Select a cluster from the list of clusters
4: **for** $i = 1$ to *number_of_iterations* **do**
5: Bisect the selected cluster using basic K-means
6: **end for**
7: Add the two clusters from the bisection with the lowest SSE to the list of clusters.
8: **until** Until the list of clusters contains K clusters

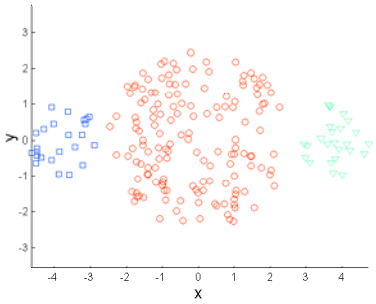
Bisecting K-means Example



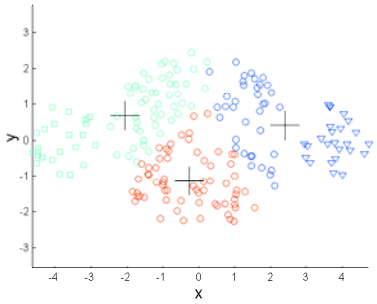
Other Limitations of K-means

- K-means has problems when clusters are of differing
  - sizes
  - densities
  - non-globular shapes
- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

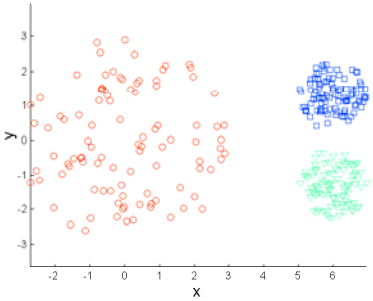


Original Points

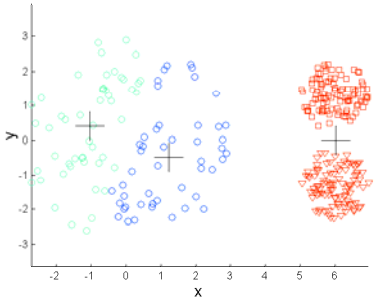


K-means (3 Clusters)

Limitations of K-means: Differing Density

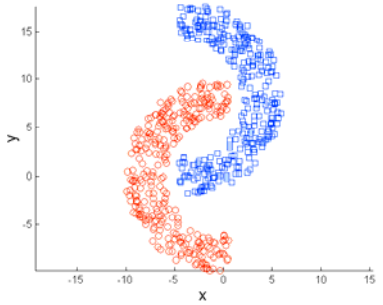


Original Points

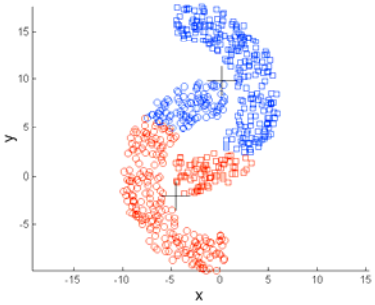


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes



Original Points

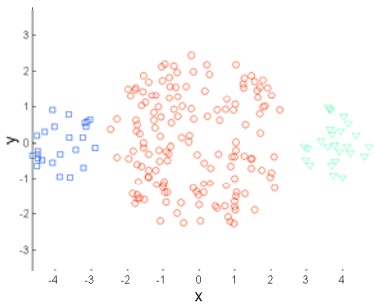


K-means (2 Clusters)

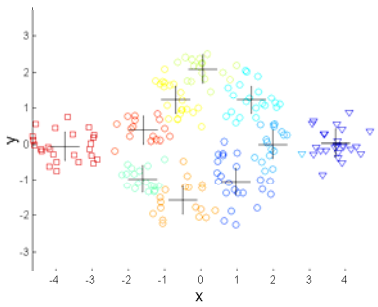
Overcoming K-means Limitations

One solution is to use many clusters.

- Find parts of clusters, but need to put together.

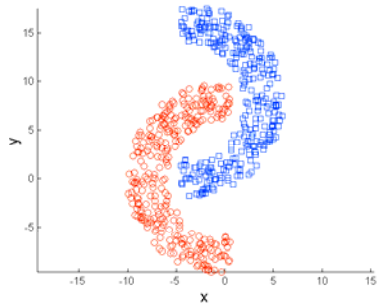


Original Points

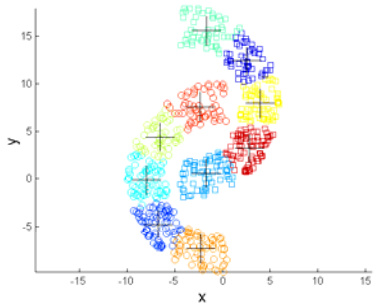


K-means Clusters

Overcoming K-means Limitations



Original Points



K-means Clusters

Variations of the *K-Means* Method

- A few variants of the *k-means* which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: ***k-modes*** (Huang'98)
  - Replacing means of clusters with modes
    - Assuming objects with *m* categorical attributes, the mode of a cluster is a vector $Q=\{q_1, q_2, \ldots, q_m\}$ where q_i is the most frequent value for the i^{th} attribute in the cluster of objects.
  - Using new dissimilarity measures to deal with categorical objects (e.g., Tanimoto)
  - Using a frequency-based method to update modes of clusters
- Handling a mixture of categorical and numerical data:
  - ***k-prototype*** method

The *K-Medoids* Clustering Method

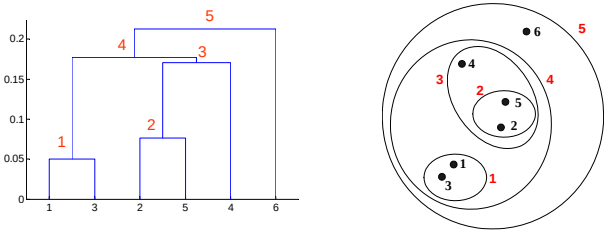
➢ Partitional clustering approach

- Find *representative* objects, called medoids, in clusters
  - Medoids are similar in concept to means or centroids, but medoids are always members of the data set.
- ***PAM*** (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- ***CLARA*** (Kaufmann & Rousseeuw, 1990)
  - draws *multiple samples* of the data set, applies *PAM* on each sample, and gives the best clustering as the output

Hierarchical Clustering

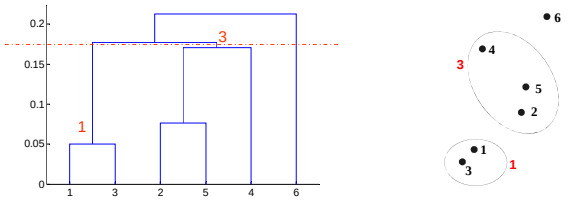
➢ Hierarchical clustering approach

- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)



Hierarchical Classification of Species

- Hierarchical clusters may correspond to meaningful taxonomies

Hierarchical Clustering

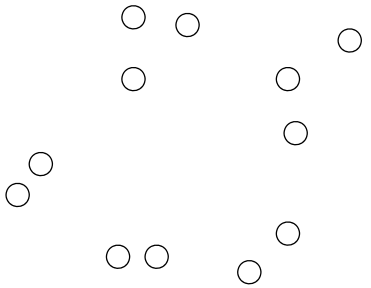
- Two main types of hierarchical clustering:
  - **Agglomerative**:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - **Divisive**:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity (proximity) or distance matrix
  - Merge or split one cluster at a time

Agglomerative Clustering Algorithm

- Popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

Starting Situation

- Start with clusters of individual points and a proximity matrix

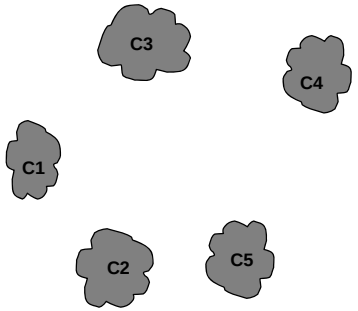


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

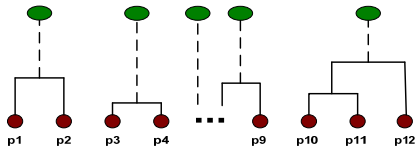
Intermediate Situation

- After some merging steps, we have some clusters



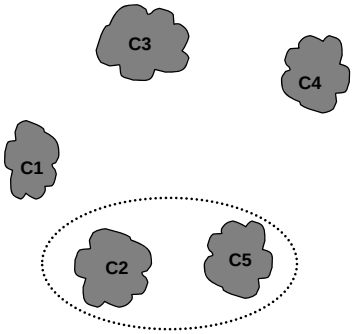
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



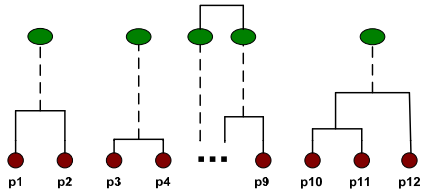
Intermediate Situation

- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.



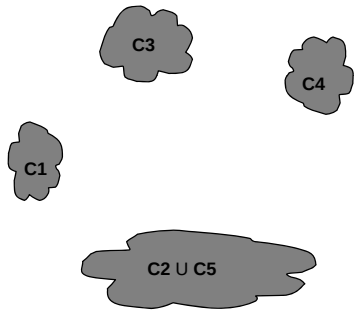
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



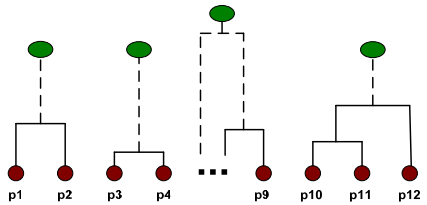
After Merging

- The question is “How do we update the proximity matrix?”

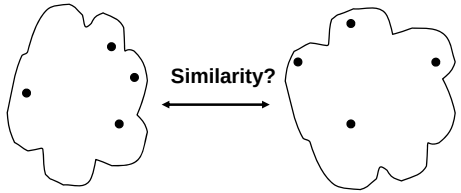


	C1	C2 U C5	C3	C4
C1		?		
C2 U C5	?	?	?	?
C3		?		
C4		?		

Proximity Matrix



How to Define Inter-Cluster Similarity

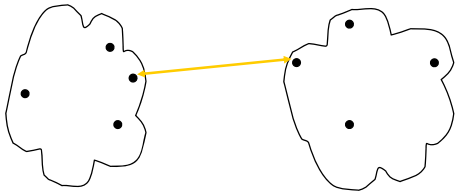


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward’s Method uses squared error

How to Define Inter-Cluster Similarity

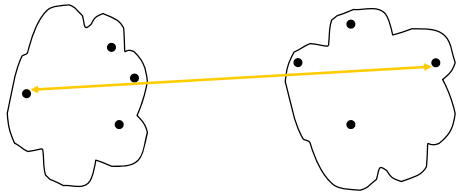


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward’s Method uses squared error

How to Define Inter-Cluster Similarity

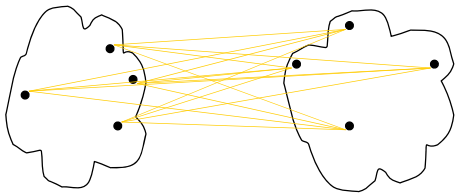


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward’s Method uses squared error

How to Define Inter-Cluster Similarity

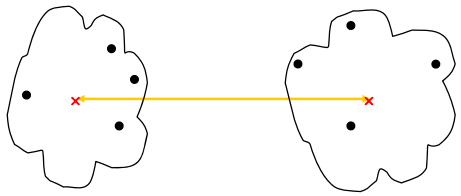


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

How to Define Inter-Cluster Similarity



	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

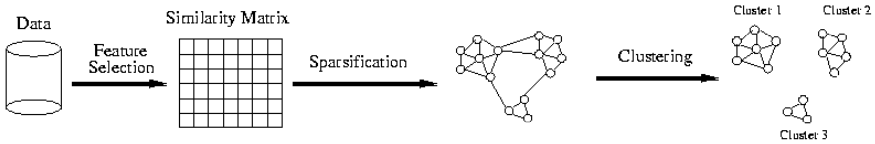
Graph-Based Clustering

- Hierarchical Clustering

- Graph-Based clustering uses the proximity graph
  - Start with the proximity matrix
  - Consider each point as a node in a graph
  - Each edge between two nodes has a weight which is the proximity between the two points
  - Initially the proximity graph is fully connected
  - MIN (single-link) and MAX (complete-link) can be viewed as starting with this graph
- In the simplest case, clusters are connected components in the graph.
- Examples:
  - ROCK (Robust Clustering using linKs)
  - CHAMELEON

Graph-based Clustering: Sparsification

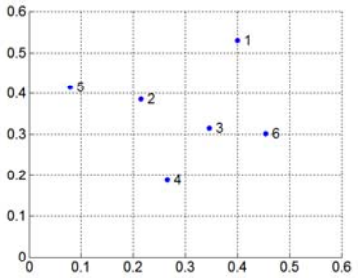
- Clustering may work better
  - Sparsification techniques keep the connections to the most similar (nearest) neighbors of a point while breaking the connections to less similar points.
  - The nearest neighbors of a point tend to belong to the same class as the point itself.
  - This reduces the impact of noise and outliers and sharpens the distinction between clusters.
- Sparsification facilitates the use of graph partitioning algorithms (or algorithms based on graph partitioning algorithms.
  - Chameleon and Hypergraph-based Clustering

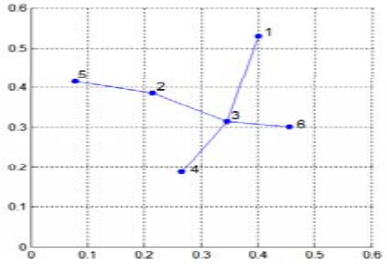


Minimum Spanning Tree Clustering

➢ Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q





Minimum Spanning Tree Clustering

- Use MST for constructing a hierarchy of clusters

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

1: Compute a minimum spanning tree for the proximity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: **until** Only singleton clusters remain

Density-based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - DBSCAN: Ester, et al. (KDD'96)
  - OPTICS: Ankerst, et al (SIGMOD'99).
  - DENCLUE: Hinneburg & D. Keim (KDD'98)

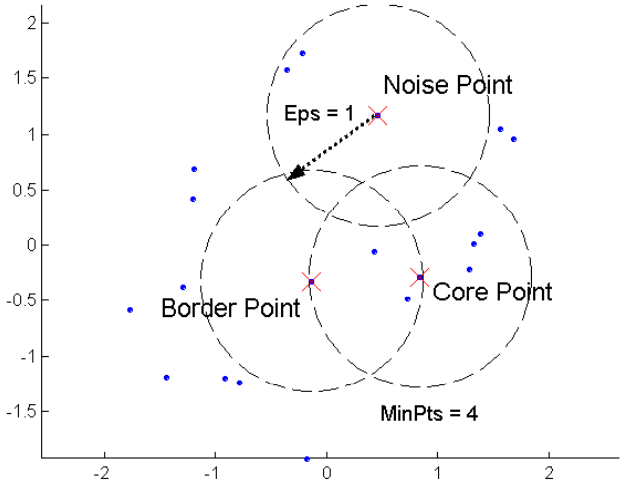
DBSCAN

➢ Density-based Clustering

DBSCAN algorithm

- Definitions:
  - Density = number of points within a specified radius (**Eps**)
  - A point is a **core point** if it has more than a specified number of points (**MinPts**) within Eps
    - These are points that are at the interior of a cluster
  - A **border point** has fewer than MinPts within Eps, but is in the neighborhood of a core point
  - A **noise point** is any point that is not a core point or a border point.

DBSCAN: Core, Border, and Noise Points



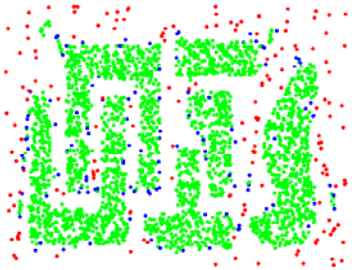
DBSCAN Algorithm

- Eliminate noise points
- Perform clustering on the remaining points

```
current_cluster_label ← 1
for all core points do
  if the core point has no cluster label then
    current_cluster_label ← current_cluster_label + 1
    Label the current core point with cluster label current_cluster_label
  end if
  for all points in the Eps-neighborhood, except i^th the point itself do
    if the point does not have a cluster label then
      Label the point with cluster label current_cluster_label
    end if
  end for
end for
```


DBSCAN: Core, Border and Noise Points

Original Points

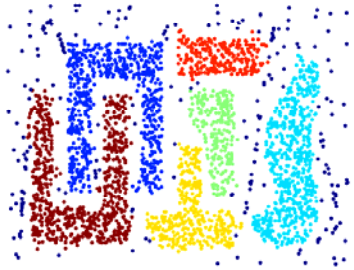


Point types:
core, border and noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

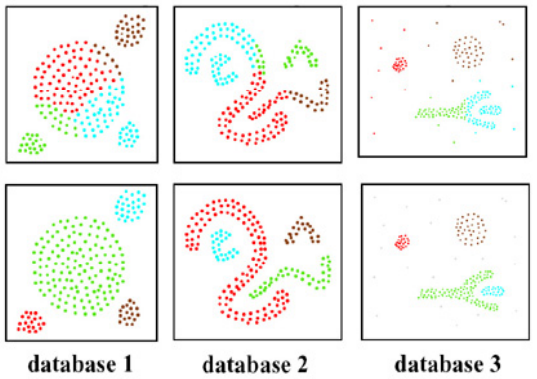
Original Points



Clusters

- **Resistant to Noise**
- **Can handle clusters of different shapes and sizes**

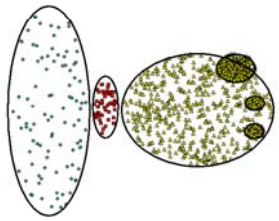
Partitioning vs. Density-based Approach



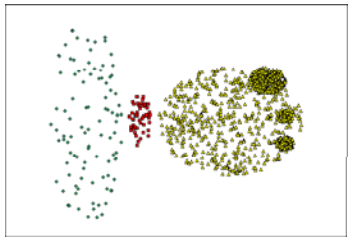
Partitioning approach:
- CLARANS

Density-based approach:
- DBSCAN

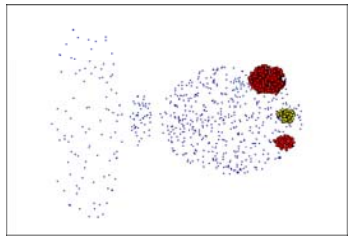
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).



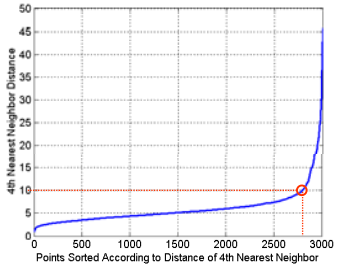
(MinPts=4, Eps=9.92)

- **Varying densities**
- **High-dimensional data**

DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor

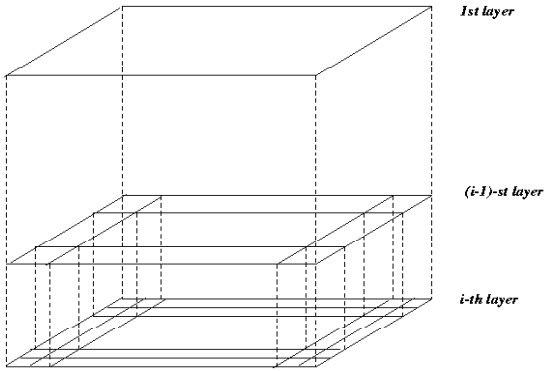
Sorted k-dist graph (k=4)



Grid-based Clustering

- Idea:
  - Using multi-resolution grid data structures
  - Use dense grid cells to form clusters.
- Several interesting methods
  - STING
  - WaveCluster
  - CLIQUE (also density-based)

- The spatial area is divided into rectangular cells
- There are several levels of cells corresponding to different levels of resolution



Model-Based Clustering Methods

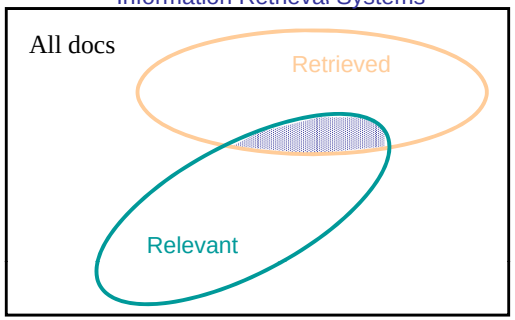
- Attempt to optimize the fit between the data and some mathematical model

- Statistical and AI approach
  - Conceptual clustering
    - A form of clustering in machine learning
    - Produces a classification scheme for a set of unlabeled objects
    - Finds characteristic description for each concept (class)
  - COBWEB (Fisher'87)
    - A popular a simple method of incremental conceptual learning
    - Creates a hierarchical clustering in the form of a classification tree
    - Each node refers to a concept and contains a probabilistic description of that concept

Evaluation of Learning Tasks

- For supervised classification we have a variety of measures to evaluate how good our model is
  - Classification in Machine Learning
    - **Accuracy**: the fraction of classifications that are correct
    - **Error rate** = 1 - accuracy
  - Information Retrieval
    - **Recall**: proportion of relevant material actually retrieved
    - **Precision**: proportion of retrieved material actually relevant

Information Retrieval Systems



$$\text{Precision} = \frac{|\text{Relevant Retrieved}|}{|\text{Retrieved}|}$$

$$\text{Recall} = \frac{|\text{Relevant Retrieved}|}{|\text{Relevant in Collection}|}$$

Cluster Validity

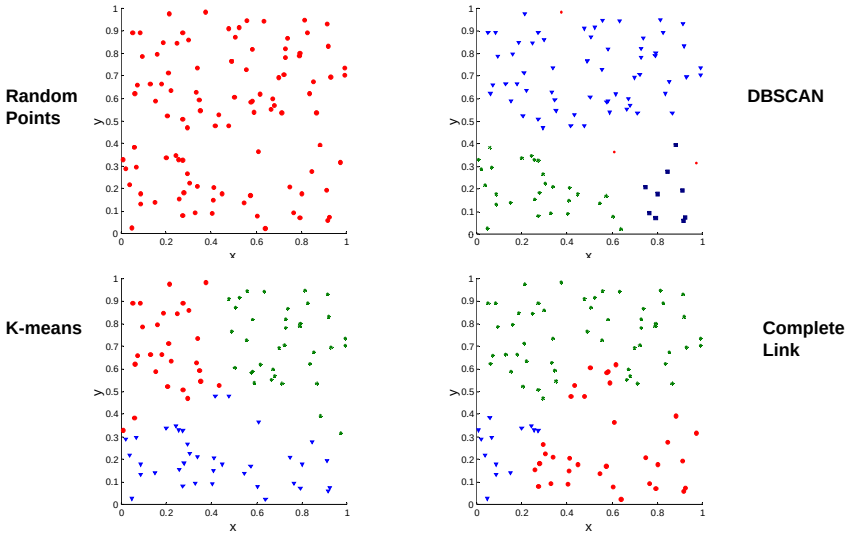
Clustering is an unsupervised learning task

- For cluster analysis, the analogous question is how to evaluate the "goodness" of the resulting clusters?

- But "clusters are in the eye of the beholder"!

- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two clusters
  - To compare two sets of clusters

Clusters found in Random Data



Measures of Cluster Validity

- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

Internal Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
  - Example: SSE
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the **W**ithin cluster **S**um of **S**quares (SSE)

$$WSS = \sum_i \sum_{x \in C_i} (x - m_i)^2$$

  - Separation is measured by the **B**etween cluster **S**um of **S**quares

$$BSS = \sum_i |C_i| (m - m_i)^2$$

    - Where $|C_i|$ is the size of cluster i, m_i is the cluster mean, m is the overall mean.

External Measures of Cluster Validity: Entropy and Purity

Entropy: $H = -\sum_i p_i \log(p_i)$

Table 5.9. K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203

entropy For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the 'probability' that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula $e_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$, where the L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^{K} \frac{m_i}{m} e_j$, where m_j is the size of cluster j, K is the number of clusters, and m is the total number of data points.

purity Using the terminology derived for entropy, the purity of cluster j, is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$.

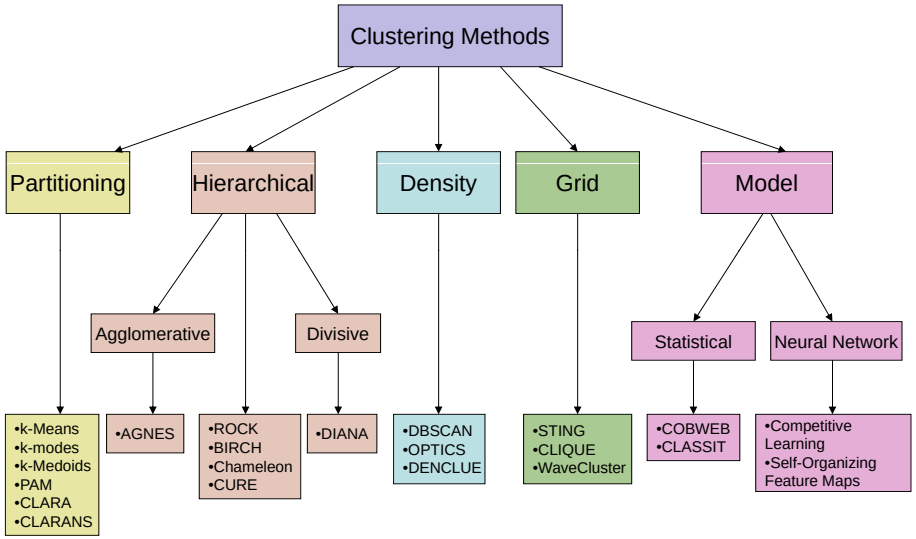
Final Comment on Cluster Validity

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

From "Algorithms for Clustering Data", Jain and Dubes

Taxonomy



Summary

- Cluster Analysis
- Similarity and Metrics
- 6 Types of Clusters
- 5 Clustering Approaches
- Cluster Validity
  - Cohesion and Separation
  - Entropy and Purity

The University of Reading

SE3DM11 - Data Mining

Regression and Classification

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Regression/Classification

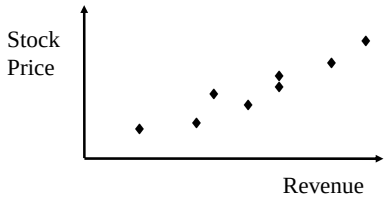
- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for the class attribute as a function of the values of other attributes.
- Goal: <u>previously unseen</u> records should be assigned a class as accurately as possible.
  - *training set* vs. *test set*

➢ Class attribute:
- Numerical → Regression
- Categorical → Classification

Regression

- Input: Table of paired data values (x, y)
  - Some connection between x and y.
  - Example: height ------ weight
  - Example: revenue ------ stock price
  - (x,y) data in a scatterplot

x	y
3.4	0.7
4.7	9.5
5.2	1.1
…	…



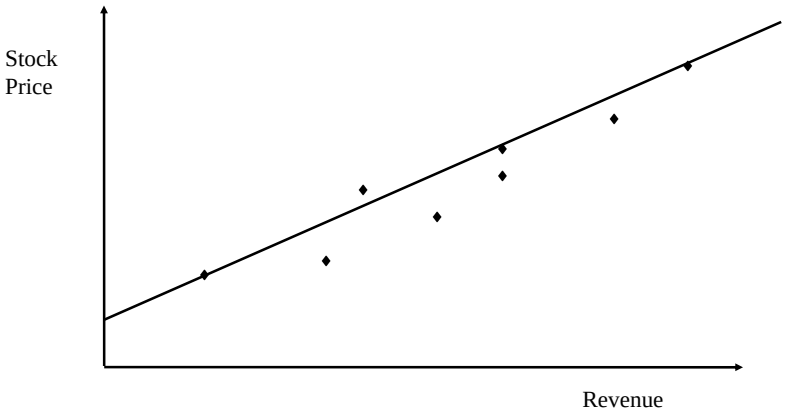
x	y
Independent	Dependent
Predictor	Predicted
Carrier	Response
Input	output

- Output: *a* model
  - to describe the data and
  - given a new instance x, to predict the outcome value y.

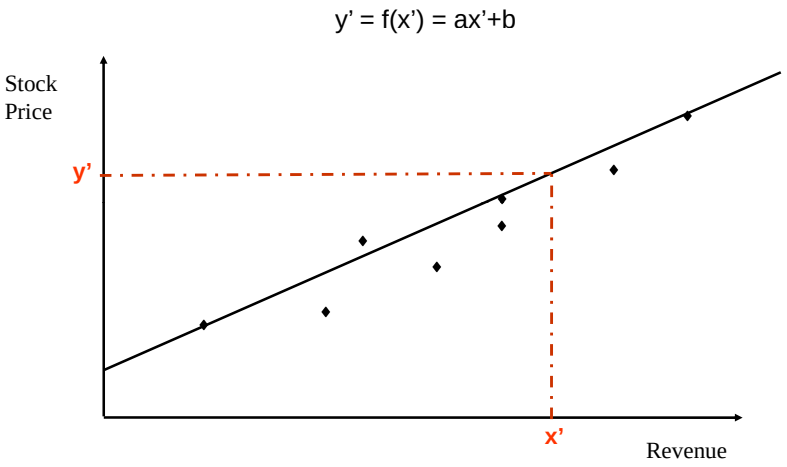
Linear Regression

- Output: *a* and *b* that best predict y from x.

$y = f(x) = ax + b$ *(the model)*

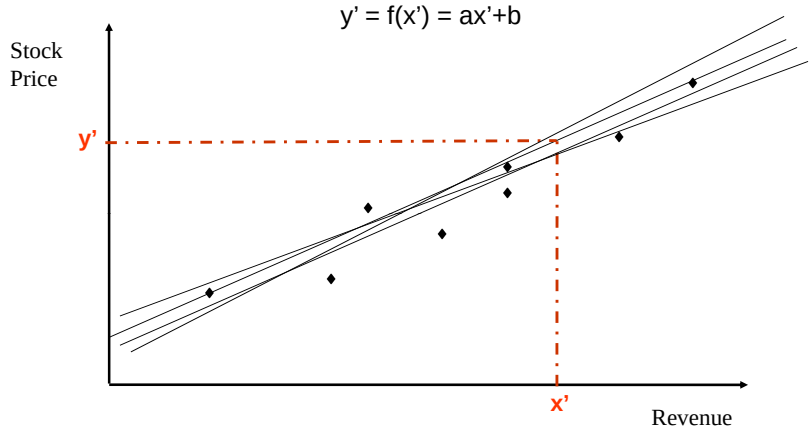


Predication with Regression Line



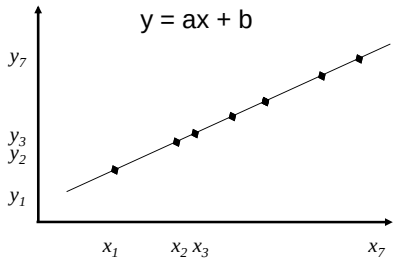
Which line is better?

i.e., how do we determine the parameters a and b?

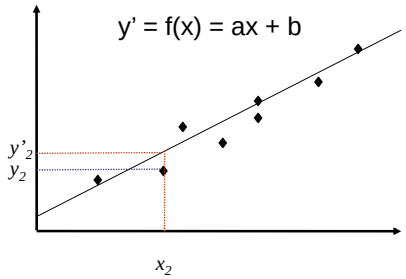


- Solution: reduce the "distance" between the observations and the model.

How to determine the parameters?



$$\begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \\ x_4 & 1 \\ x_5 & 1 \\ x_6 & 1 \\ x_7 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix}$$

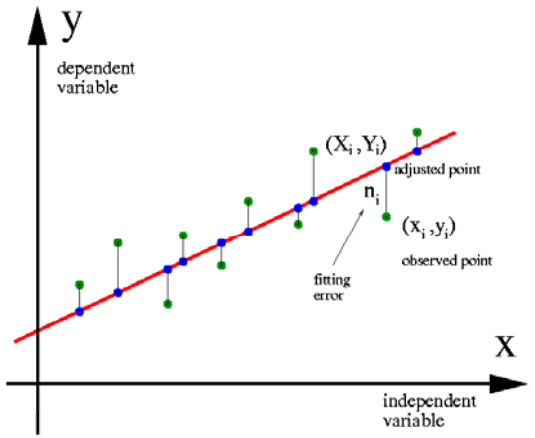


observed value

predicted value

$$e = y - y' = y - (ax + b)$$

Least Squares Regression



- No errors in *x*
- Errors in *y*

- Best Fit
- Find the line that minimize the norm of the *y* errors
- (sum of the squares)

$$\|e\|^2 = \sum_i (y_i - y_i')^2$$

Least Square Approximation

Find a, b to minimize

$$\|e\|^2 = \left\| \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{bmatrix} - \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ \dots & \dots \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 \Rightarrow \begin{cases} a = \dfrac{\sum_{i=1}^{n} (x_i - \mu_x)(y_i - \mu_y)}{\sum_{i=1}^{n} (x_i - \mu_x)^2} \\ b = \mu_y - a\mu_x \end{cases}$$

$$\begin{cases} a = \dfrac{n\sum_{i=1}^{n}(x_i y_i) - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2} \\ b = \dfrac{\sum_{i=1}^{n} y_i - a\sum_{i=1}^{n} x_i}{n} \end{cases}$$

Polynomial Regression

- Minimize the residual between the data points and the curve -- *least-squares regression*

Data	$\{(x_1, y_1), (x_2, y_2), \cdots (x_i, y_i), \cdots (x_n, y_n)\}$
Linear	$y_i = a_0 + a_1 x_i$
Quadratic	$y_i = a_0 + a_1 x_i + a_2 x_i^2$ (Parabola)
Cubic	$y_i = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3$
…	
General	$y_i = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + \cdots + a_m x_i^m$

Find values of the weights *a0* , *a1*, *a2*, … *am*

Polynomial Regression

- Equation

$$y_i = \left(a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + \cdots + a_m x_i^m\right)$$

- Residual

$$e_i = y_i - \left(a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_i^3 + \cdots + a_m x_i^m\right)$$

- Sum of squared residuals

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left[y - \left(a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_m x^m\right)\right]^2$$

Multidimensional Observations

- In general, each observation has several (k) attributes (independent variables) and a numeric class (response).

observations	numeric attributes				numeric class, response
1	a_{11}	a_{11}	…	a_{1k}	b_1
2	a_{21}	a_{21}	…	a_{2k}	b_2
…					…
n	a_{n1}	a_{n1}	…	a_{n1}	b_n

Let's express the response as a linear combination of the attributes:

$$\hat{b}_i = w_0 + w_1 a_{i1} + \ldots + w_k a_{ik} = w_0 + \sum_j w_j a_{ij}$$ (predicted value)

where w_j are the weights of the attributes.

Multidimensional Regression

- Find the weights that minimize the mean square error:

$$\|e\|^2 = \sum_{i=1}^{n} (b_i - \hat{b}_i)^2 = \sum_{i=1}^{n} \left(b_i - \sum_{j=0}^{k} w_j a_{ij} \right)^2$$

observed value — predicted value

where we add an extra attribute a_{i0}, which is always equal to 1.

Least Square Approximation

Find ω_j to minimize

$$\|e\|^2 = \left\| \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} - \begin{bmatrix} 1 & a_{11} & \dots & a_{1k} \\ 1 & a_{21} & & a_{2k} \\ \dots & \dots & \dots & \dots \\ 1 & a_{n1} & \dots & a_{nk} \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \dots \\ \omega_{k+1} \end{bmatrix} \right\|^2$$

Changing notation… **b** - A **x**

b: vector in $\Re^n$ A: matrix in $\Re^n \times \Re^{k+1}$ **x**: vector in $\Re^{k+1}$

In general: if $\mathbf{Ax} = \mathbf{b}$ has no solution, then

Solving $A^TA\mathbf{x} = A^T\mathbf{b}$ produces the least square approximation.

$\mathbf{x} = (A^TA)^{-1} A^T\mathbf{b}$

Classification

Linear Regression for Classification

- Linear regression is used for numeric prediction, i.e. numerical output values.
- **Classification**: how to perform a prediction for nominal attributes, e.g. class labels?

- Multi-response linear regression
  - Perform a regression for each class: output value ranging from 0 to 1
  - Each linear regression approximates a numeric membership function.
  - Given a new instance, compute the membership for each class and select the greatest.

- Problems:
  - The output value may be greater than 1. It cannot be interpreted as a probability function.
  - Errors are not normally distributed with the same standard deviation, as required by the least square approximation.

Pairwise Regression

- Another way of using regression for classification:
  - A regression function for every *pair* of classes, using only instances from these two classes
  - Assign output of +1 to one member of the pair, –1 to the other
- Prediction is done by voting
  - Class that receives most votes is predicted
  - Alternative: "don't know" if there is no agreement
- More likely to be accurate but more expensive

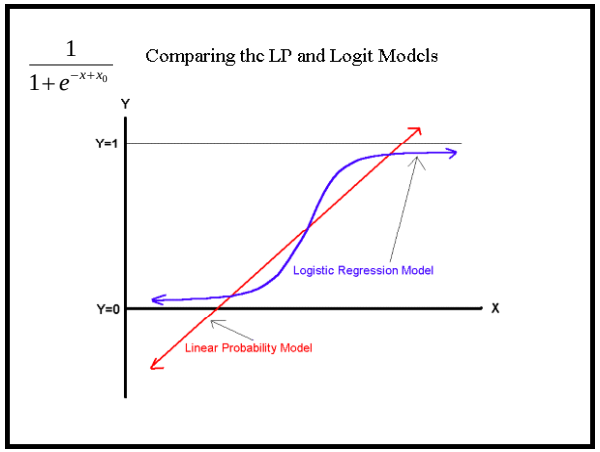
Logistic Regression

- *Logistic* regression: alternative to linear regression
  - Designed for classification problems
  - Let's assume only two classes, e.g. 1 and 0.
    - dichotomous outcome: yes=1, no=0
    - pairwise regression
  - Tries to estimate class probabilities directly
    - by using the *maximum likelihood* method
- Replace the original target variable: $P_1 = P\left[1 \middle| a_1, a_2, ..., a_k\right] = \sum_{i=0}^{k} w_i a_i$
- A ***logistic probability unit*** (**logit**) is computed by taking the natural logarithm of the ratio of the probability of class A over the probability of class B.

$$\log\left(\frac{P_1}{P_0}\right) = \log\left(\frac{P_1}{1-P_1}\right) = w_0 a_0 + w_1 a_1 + \ldots + w_k a_k = \sum_{i=0}^{k} w_i a_i$$

$$\Rightarrow \quad P_1 = \frac{1}{1 + e^{-\sum_{i=0}^{k} w_i a_i}}$$

The Logit Transformation



The Logit Approximation: M*aximum Likelihood* Method

- In the linear regression, find the weights that minimize the mean square error:

$$\|e\|^2 = \sum_{i=1}^{n} \left(b_i - \hat{b_i}\right)^2 = \sum_{i=1}^{n} \left(b_i - \sum_{j=0}^{k} w_j a_{ij}\right)^2$$

- In the logit regression, find the weights that maximize the log-likelihood of the model:

$$\sum_{i=0}^{k} (1-b_i)\log(1-P_1) + b_i \log P_1$$

$$where\, b_i \in \{0,1\}$$

b_i : *observed value*

$\hat{b_i}$: *predicted value*

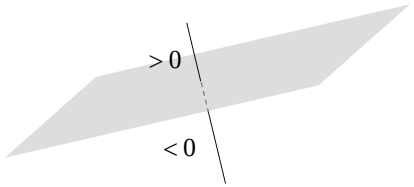
P_1 : *predicted probability for class* 1

Linear Models

Discussion:

- Not appropriate if data exhibits non-linear dependencies
- But: can serve as building blocks for more complex schemes (i.e. model trees)
- Example: multi-response linear regression defines a *hyperplane* for any two given classes:

$$(w_0^{(1)} - w_0^{(2)})a_0 + (w_1^{(1)} - w_1^{(2)})a_1 + (w_2^{(1)} - w_2^{(2)})a_2 + \ldots + (w_k^{(1)} - w_k^{(2)})a_k > 0$$



Summary

- Regression
  - Linear regression
  - Multivariate linear regression (Assignment #2)
  - Polynomial regression
- Classification by regression
  - Multi-response linear regression
  - Pairwise classification
  - Logistic regression

Next:

- Instance-Based Classification
  - Rote Learning
  - Nearest-Neighbors (1-NN, k-NN)
- Model Evaluation

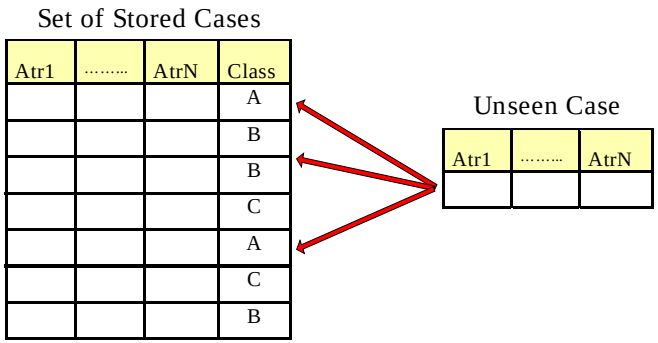
The University of Reading

SE3DM11 - Data Mining

Classification and Model Evaluation

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Instance-Based Classification



- Store the training records
- Use training records to predict the class label of unseen cases

Instance-based Classification

- Simplest form of learning: ***rote learning***
  - Training instances are searched for instance that most closely resembles new instance
  - The instances themselves represent the knowledge
  - Also called *instance-based* learning
  - Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly
- Similarity function (distance) defines what's "learned"
- Instance-based learning is *lazy* learning

- **Nearest Neighbor**:
  - Uses "closest" points (nearest neighbors) for performing classification
    - *nearest-neighbor (1-NN)*
    - *k-nearest-neighbor (k-NN)*

The Distance Function

- Simplest case: one numeric attribute
  - Distance is the difference between the two attribute values involved (or a function thereof)
- Several numeric attributes: normally, Euclidean distance is used and attributes are normalized
- Nominal attributes: distance is set to 1 if values are different, 0 if they are equal.
- Are all attributes equally important?
  - Weighting the attributes might be necessary.

Instance-based Learning

- Distance function defines what's learned
- Most instance-based schemes use *Euclidean distance*:
  $\mathbf{a}^{(1)}$ and $\mathbf{a}^{(2)}$: two instances with *k* attributes

$$d_E = \sqrt{(a_1^{(1)} - a_1^{(2)})^2 + (a_2^{(1)} - a_2^{(2)})^2 + \ldots + (a_k^{(1)} - a_k^{(2)})^2}$$

- Taking the square root is not required when comparing distances
- Other popular metric: *city-block* metric *(*aka *Manhattan* distance*)*
  - Adds differences without squaring them

$$d_M = \sqrt{|a_1^{(1)} - a_1^{(2)}| + |a_2^{(1)} - a_2^{(2)}| + \ldots + |a_k^{(1)} - a_k^{(2)}|}$$

Normalization and Other Issues

- Different attributes are measured on different scales ⇒ need to be *normalized*:

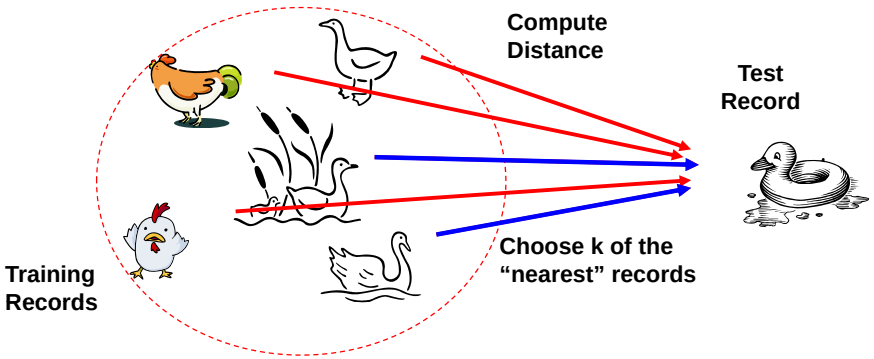
$$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$$

  v_i : the actual value of attribute *i*

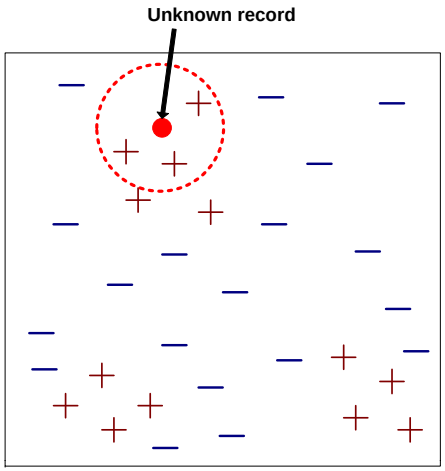
- Common policy for missing values: assumed to be maximally distant (given normalized attributes)

Nearest-Neighbor Classifiers

- **Nearest-Neighbor**:
  - Uses "closest" points (nearest neighbors) for performing classification
- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck

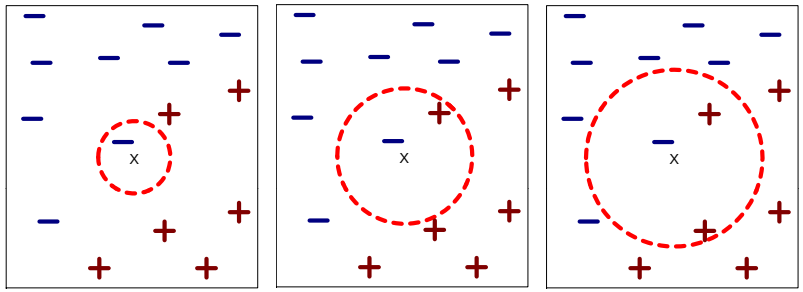


Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve
- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

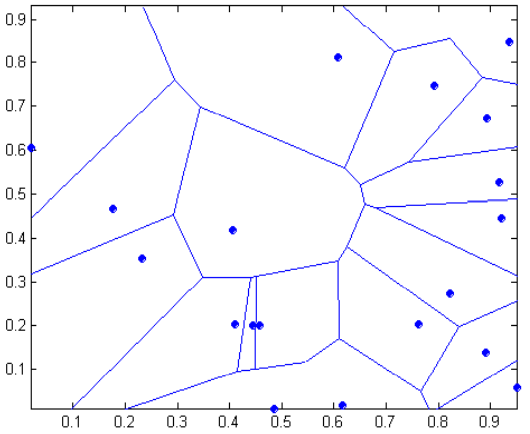
Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record x are data points that have the k smallest distance to x

1-NN: Voronoi Diagram



Nearest-Neighbor Classification

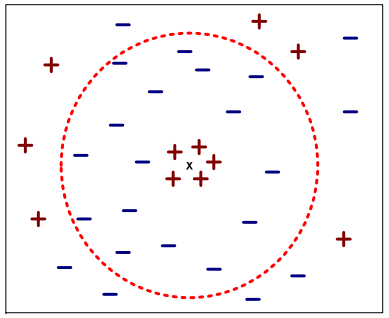
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest-Neighbor Classification

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest-Neighbor Classification

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M

Nearest-Neighbor Classification

- Problem with Euclidean measure:
  - High dimensional data
    - curse of dimensionality
  - Can produce counter-intuitive results

$b_{11}\ b_{10}\ b_9\ b_8\ b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_0$

b_i: 1 or 0 → feature [i] is/is not present

1 1 1 1 1 1 1 1 1 1 1 0		**1 0 0 0 0 0 0 0 0 0 0 0**
0 1 1 1 1 1 1 1 1 1 1 1	vs	**0 0 0 0 0 0 0 0 0 0 0 1**
d_E = 1.4142		**d_E = 1.4142**

Solutions:

- Normalize the vectors to unit length
- Use different metric (e.g. Tanimoto distance)

Nearest-Neighbor Classification

- k-NN classifiers are lazy learners
  - It does not build models explicitly
  - Unlike eager learners such as decision tree induction and rule-based systems
  - Classifying unknown records is relatively expensive

Discussion:

- Often very accurate and slow
  - simple version of 1-NN scans entire training data to derive a prediction
- Assumes all attributes are equally important
  - Remedy: attribute selection or weights
- Possible remedies against noisy instances:
  - Take a majority vote over the *k* nearest neighbors
  - Removing noisy instances from dataset (difficult!)
- Statisticians have used *k*-NN since early 1950s
  - If $n \to \infty$ and $k/n \to 0$, error approaches minimum

When to Consider NN Classifiers

- Instances map to points in R^N
- Less than 20 attributes per instance
- Lots of training data

Advantages:

- Training is very fast
- Learn complex target functions
- Do not loose information

Disadvantages:

- Slow at query time
- Easily fooled by irrelevant attributes

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
  - Rather than how fast it takes to classify or build models, scalability, etc.
- Confusion Matrix:

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)
b: FN (false negative)
c: FP (false positive)
d: TN (true negative)

Accuracy

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{Error rate} = \frac{b+c}{a+b+c+d} = \frac{FN+FP}{TP+TN+FP+FN} = 1 - Accuracy$$

Limitation of Accuracy

- Consider a 2-class problem
  - Number of Class 0 examples = 9990
  - Number of Class 1 examples = 10
- If model predicts everything to be class 0, accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any class 1 example

Cost Matrix

		PREDICTED CLASS	
	C(i|j)	**Class=Yes**	**Class=No**
ACTUAL CLASS	**Class=Yes**	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

C(i|j): Cost of misclassifying class j example as class i

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
	C(i|j)	+	-
ACTUAL CLASS	+	-1	100
	-	1	0

Model M^1	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	150	40
	-	60	250

Accuracy = 80%
Cost = 3910

Model M^2	PREDICTED CLASS		
		+	-
ACTUAL CLASS	+	250	45
	-	5	200

Accuracy = 90%
Cost = 4255

Cost vs Accuracy

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

N = a + b + c + d

Accuracy = (a + d)/N

Cost = p (a + d) + q (b + c)
= p (a + d) + q (N – a – d)
= q N – (q – p)(a + d)
= N [q – (q-p) × Accuracy]

Cost-Sensitive Measures

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

Count	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

- Precision is biased towards C(Yes|Yes) & C(Yes|No)
- Recall is biased towards C(Yes|Yes) & C(No|Yes)
- F-measure is biased towards all except C(No|No)

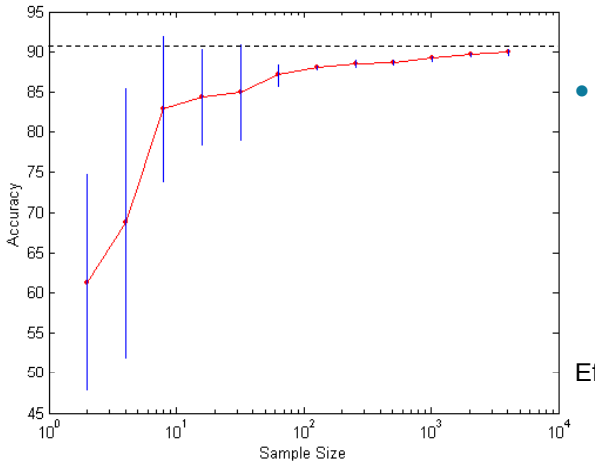
Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

Methods for Performance Evaluation

- How to obtain a reliable estimate of performance?
- Performance of a model may depend on other factors besides the learning algorithm:
  - Class distribution
  - Cost of misclassification
  - Size of training and test sets

Learning Curve



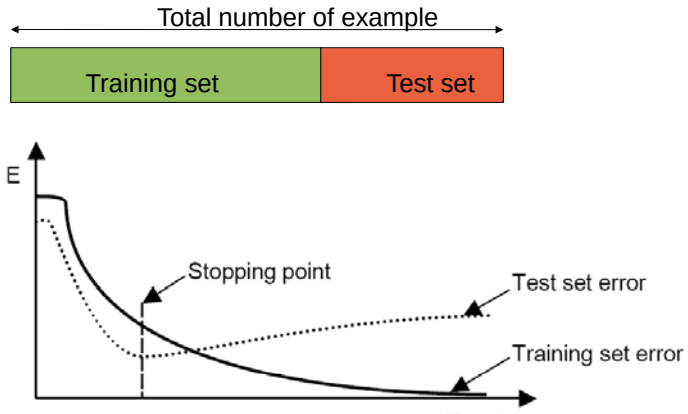
- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve:
  - Arithmetic sampling (Langley, et al)
  - Geometric sampling (Provost et al)

Effect of small sample size:

- Bias in the estimate
- Variance of estimate

The Holdout Method

- Split dataset into two groups
  - Training set: used to train the classifier
  - Test set: used to estimate the error rate of the trained classifier



The Holdout Method

- Holdout method
  - Original data is partitioned into two disjoint sets
  - E.g. 2/3 for training and 1/3 for testing, 50%-50%
  - Several issues
- Issues
  - Less data available for training
  - Bias on the training set
  - Training set and test set are not independent (e.g. there may be unbalanced classes in both)

Methods of Estimation

- Holdout
  - Original data is partitioned into two disjoint sets
- Random subsampling
  - Repeated holdout
  - Still some issues
    - Similar issues to holdout
    - No control over the selected samples (samples may be chosen multiple times or never)
- Cross validation
  - Partition data into k disjoint subsets
  - k-fold: train on k-1 partitions, test on the remaining one
  - Leave-one-out: k=n
- Bootstrap
  - Sampling with replacement
  - .632 bootstrap

Model Evaluation

- Metrics for Performance Evaluation
  - How to evaluate the performance of a model?
- Methods for Performance Evaluation
  - How to obtain reliable estimates?
- Methods for Model Comparison
  - How to compare the relative performance among competing models?

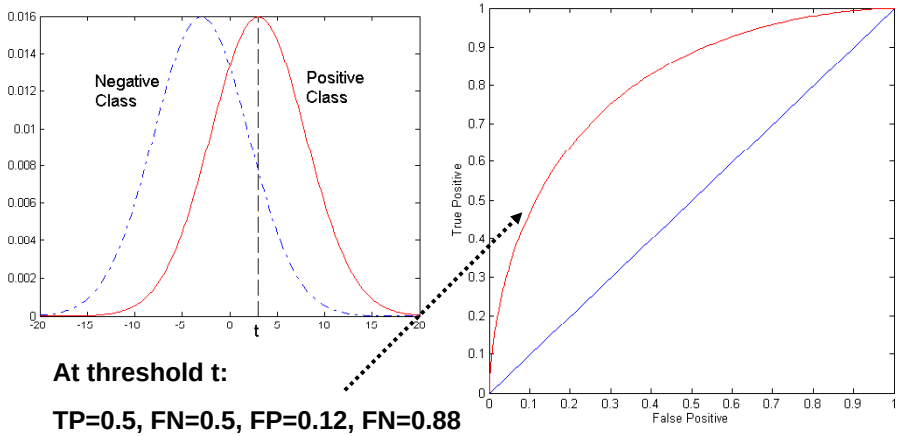
ROC (Receiver Operating Characteristic)

- Developed in 1950s for signal detection theory to analyze noisy signals
  - Characterize the trade-off between positive hits and false alarms
- ROC curve plots TP (on the y-axis) against FP (on the x-axis)
- Performance of each classifier represented as a point on the ROC curve
  - changing the threshold of algorithm, sample distribution or cost matrix changes the location of the point

ROC Curve

- 1-dimensional data set containing 2 classes (positive and negative)

- any points located at x > t is classified as positive



At threshold t:

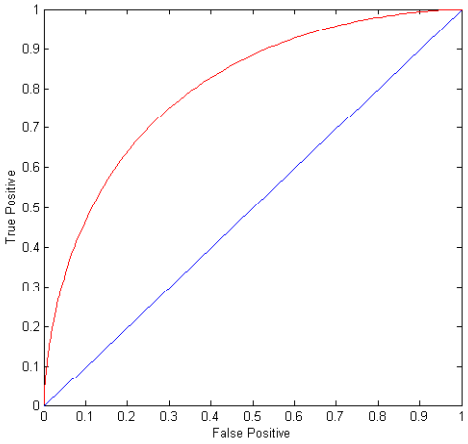
TP=0.5, FN=0.5, FP=0.12, FN=0.88

ROC Curve

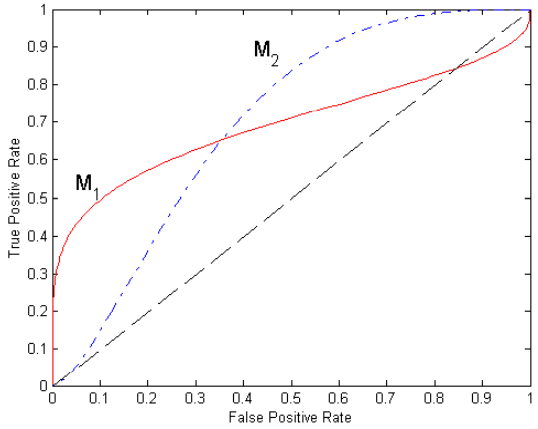
(TP,FP):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (0,1): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class



Using ROC for Model Comparison



- No model consistently outperform the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

Summary

- Regression
  - Linear regression
  - Multivariate linear regression
  - Polynomial regression
- Classification by regression
  - Multi-response linear regression
  - Pairwise classification
  - Logistic regression
- Instance-Based Classification
  - Rote Learning
  - Nearest-Neighbors (1-NN, k-NN)
- Model Evaluation

The University of Reading

SE3DM11 - Data Mining

Decision Trees

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

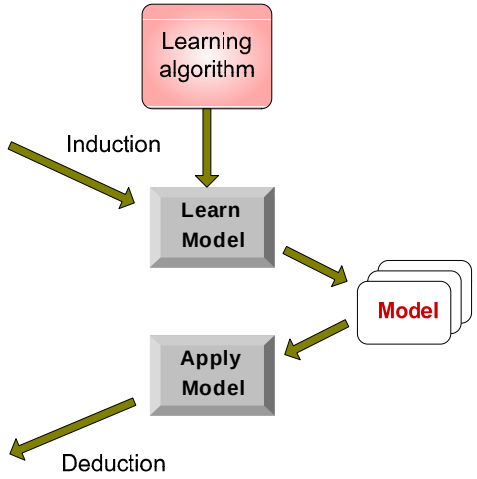
Classification

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



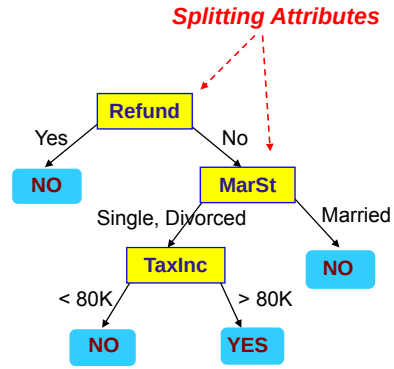
Classification Techniques

- Memory based reasoning
- ➢ **Decision Tree based Methods**
- Rule-based Methods
- Neural Networks
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines

Example of a Decision Tree

	categorical	categorical	continuous	class
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

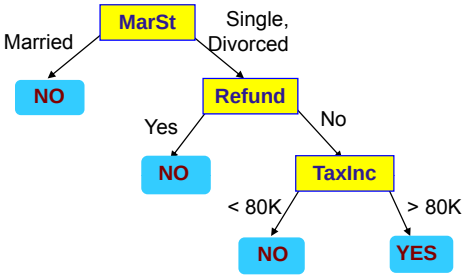
Training Data



Model: Decision Tree

Another Example of Decision Tree

Tid	Refund	Marital Status	Taxable Income	Cheat
	categorical	categorical	continuous	class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

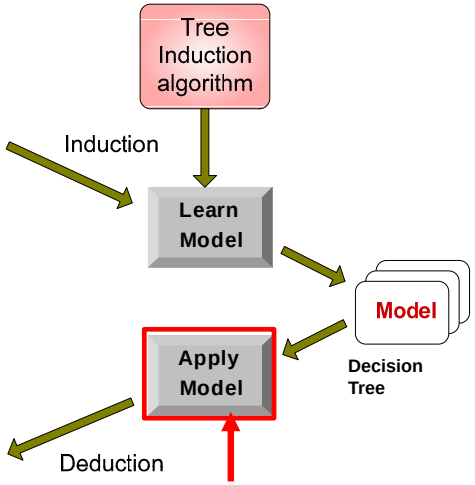
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

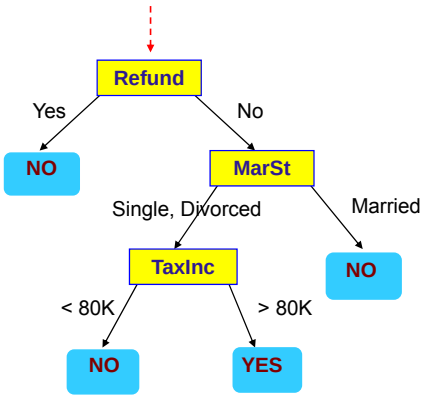
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

Start from the root of tree.



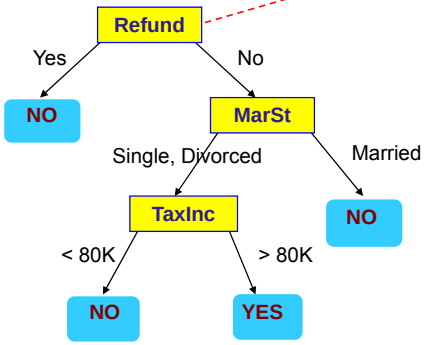
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

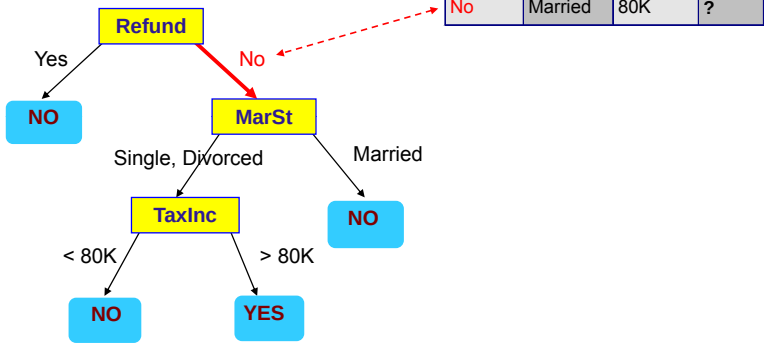
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

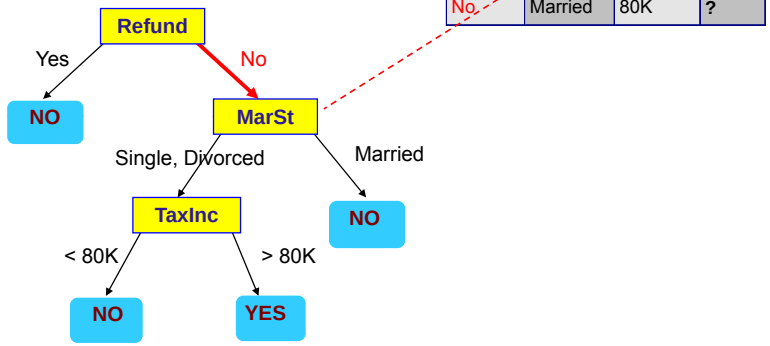
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

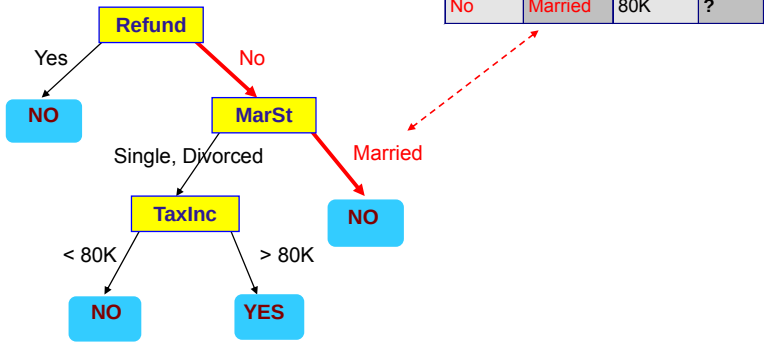
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

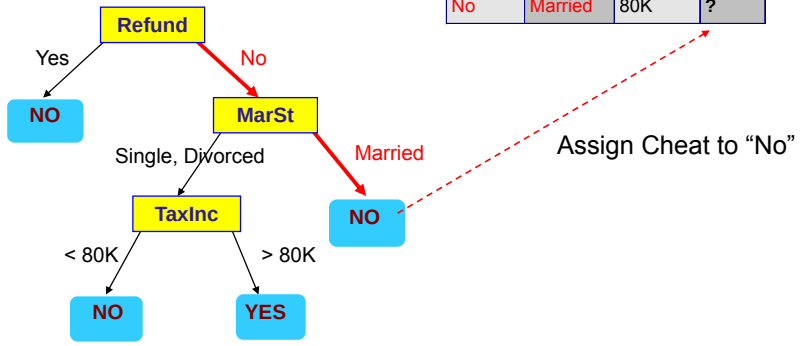
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



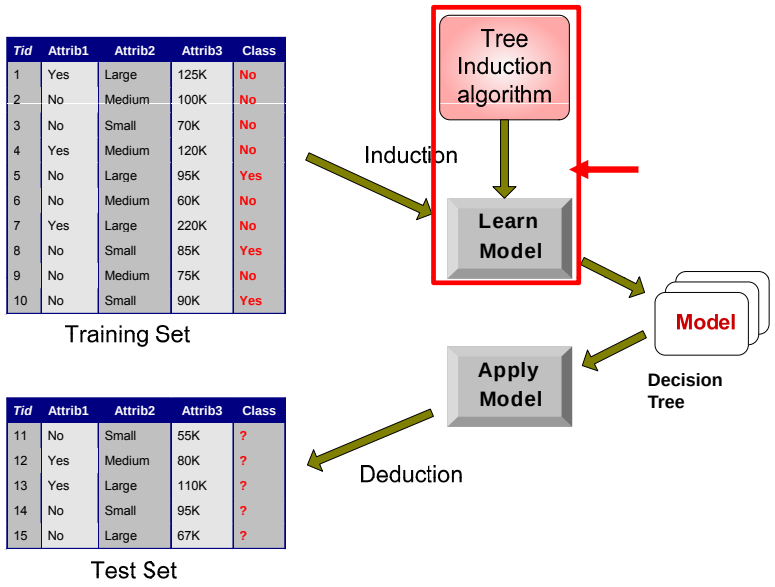
Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Decision Tree Classification Task



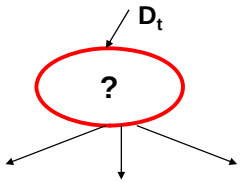
Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5, C5.0
  - SLIQ, SPRINT

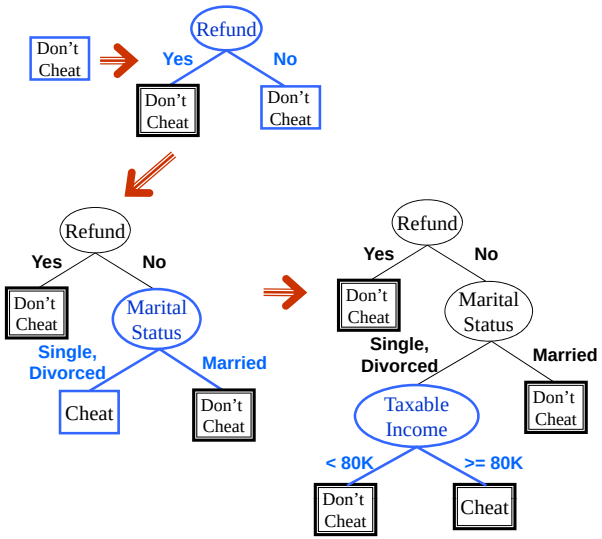
General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t
- General Procedure:
  - If D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t
  - If D_t is an empty set, then t is a leaf node labeled by the default class, y_d
  - If D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Hunt's Algorithm



Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Tree Induction

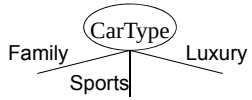
- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to Specify Test Condition?

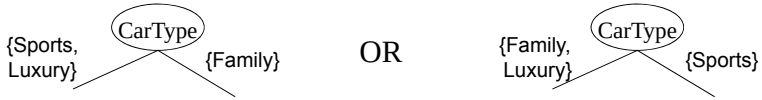
- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

- Multi-way split: Use as many partitions as distinct values.

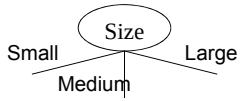


- Binary split: Divides values into two subsets. Need to find optimal partitioning.

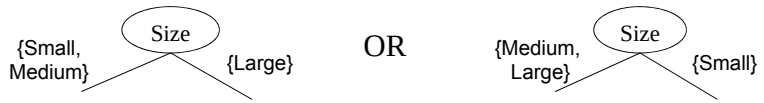


Splitting Based on Ordinal Attributes

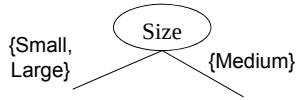
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.



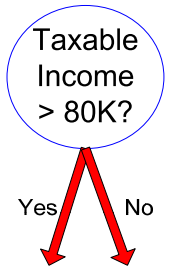
- What about this split?



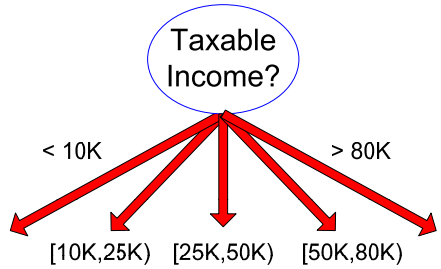
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: (A < v) or (A ≥ v)
    - consider all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes



(i) Binary split



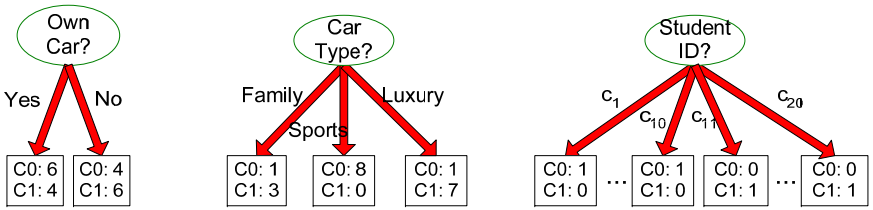
(ii) Multi-way split

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to determine the Best Split

Before Splitting: 10 records of class 0, 10 records of class 1



Which test condition is the best?

How to determine the Best Split

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

Measures of Node Impurity

- Gini Index
- Entropy and Information Gain
- Misclassification error

Measure of Impurity: the GINI Index

- GINI Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

Node t

C1	1
C2	5

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Maximum ($1 - 1/n_c$)
    - when records are equally distributed among all classes,
    - implying least interesting information
  - Minimum (0.0)
    - when all records belong to one class,
    - implying most interesting information

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

Node t

C1	1
C2	5

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – $P(C1)^2$ – $P(C2)^2$ = 1 – 0 – 1 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – $(1/6)^2$ – $(5/6)^2$ = 0.278

C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

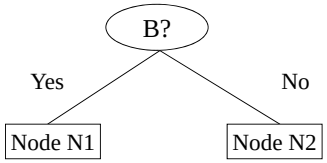
Gini = 1 – $(2/6)^2$ – $(4/6)^2$ = 0.444

Splitting Based on GINI

- When a node p is split into k partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

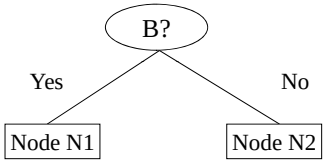
where, n_i = number of records at child i,
n = number of records at node p.



- Greedy optimisation approach:
  - choose the split which minimises $GINI_{index}$
- Used in CART, SLIQ, SPRINT

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



	Parent
C1	6
C2	6
Gini = 0.500	

GINI(N1)
= 1 – $(5/7)^2$ – $(2/7)^2$
= 0.408

GINI(N2)
= 1 – $(1/5)^2$ – $(4/5)^2$
= 0.320

	N1	N2
C1	5	1
C2	2	4
Gini=0.371		

$GINI_{split}$
= 7/12 * 0.408 +
5/12 * 0.320
= 0.371

Categorical Attributes: Computing Gini Index

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

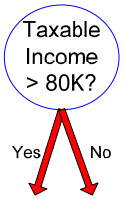
	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values = Number of distinct values +1
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A<v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



Continuous Attributes: Computing Gini Index...

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing Gini index
  - Choose the split position that has the least Gini index

10 records: YES: 3, NO: 7

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	0.300	0.343	0.375	0.400	0.420

Alternative Splitting Criteria based on Information Theory

- Entropy at a given node t:

Node t

C1	1
C2	5

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

- It measures the homogeneity (impurity) of a node
- Maximum ($\log_2 n_c$)
  - when records are equally distributed among all classes,
  - implying least information
- Minimum (0.0)
  - when all records belong to one class,
  - implying most information
- Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

C1	0
C2	6

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	1
C2	5

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

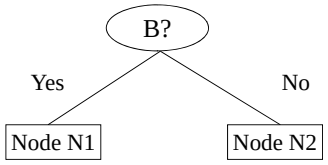
C1	2
C2	4

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on Information Theory

- Information Gain:



$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

Parent Node, p is split into k partitions;
n_i is number of records in partition i

- Measures Reduction in Entropy achieved because of the split.
  - ➢ Choose the split that achieves most reduction (maximizes GAIN)
- Used in ID3
- Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on Information Theory

- Gain Ratio:

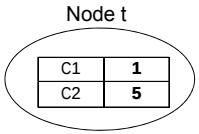
$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions
n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain (ID3)

Splitting Criterion based on Classification Error

- Classification Error at a node t :



$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures misclassification error made by a node.
  - Maximum (1 - 1/n_c)
    - when records are equally distributed among all classes,
    - implying least interesting information
  - Minimum (0.0)
    - when all records belong to one class,
    - implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1
Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

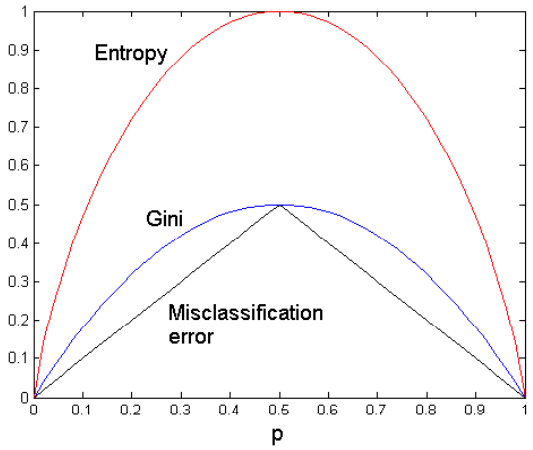
P(C1) = 1/6 P(C2) = 5/6
Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes a certain criterion.
- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Stopping Criteria for Tree Induction

Stop expanding a node when

- Case 0: node is empty
- Case1: all the records have similar attribute values
- Case 2: all the records belong to the same class
- Case 3: early termination (to be discussed later)

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

Example: C4.5

- Simple depth-first construction.
- Evolution of ID3.
- Uses Information Gain, Gain Ratio.
- Sorts continuous attributes at each node.
- Needs entire data to fit in memory.
- Unsuitable for large datasets.
  - Needs out-of-core sorting.

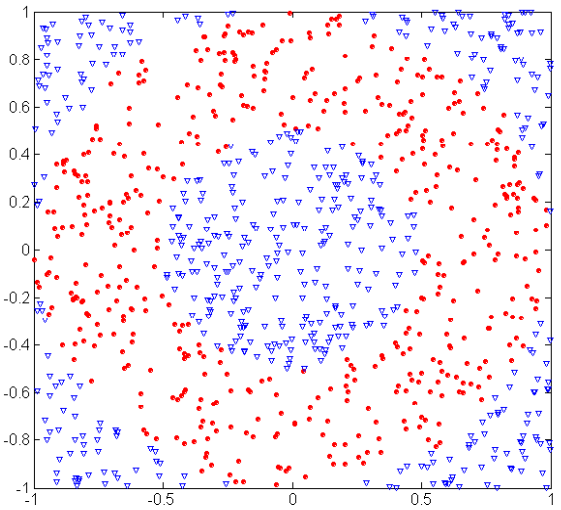
- You can download the source code from: http://www.rulequest.com/Personal/c4.5r8.tar.gz

Ross Quinlan

Practical Issues of Classification

- Underfitting and Overfitting
- Missing Values
- Costs of Classification

Underfitting and Overfitting (Example)



500 circular and 500 triangular data points.

Circular points:

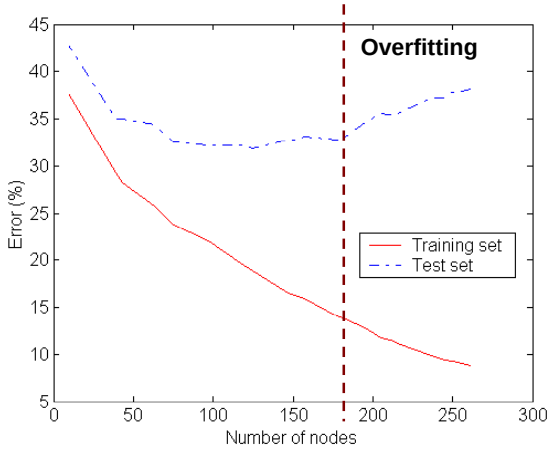
$0.5 \leq \text{sqrt}(x_1^2+x_2^2) \leq 1$

Triangular points:

$\text{sqrt}(x_1^2+x_2^2) > 0.5$ or

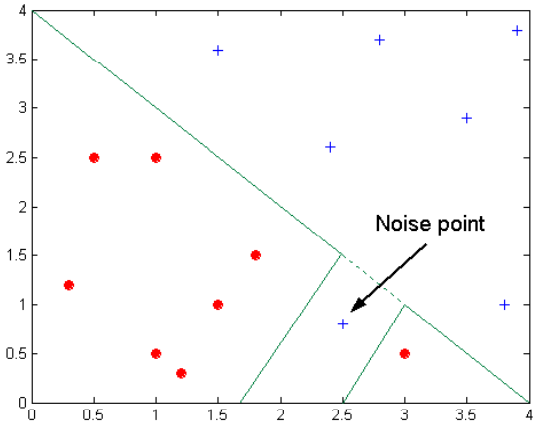
$\text{sqrt}(x_1^2+x_2^2) < 1$

Underfitting and Overfitting



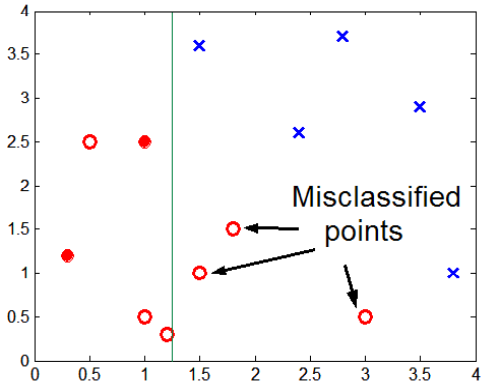
Underfitting: when model is too simple, both training and test errors are large

Overfitting due to Noise



Decision boundary is distorted by noise point

Overfitting due to Insufficient Examples



Lack of data points in the lower half of the diagram makes it difficult to predict correctly the class labels of that region

- Insufficient number of training records in the region causes the decision tree to predict the test examples using other training records that are irrelevant to the classification task

Notes on Overfitting

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records
- Need new ways for estimating errors

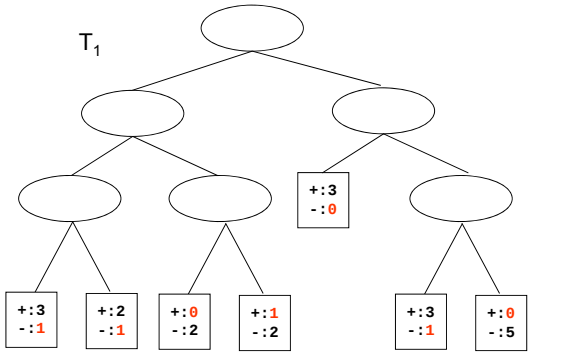
Estimating Generalization Errors

- Re-substitution error: error on training data $e(T) = \Sigma\, e(t)$
- Generalization error: error on testing data $e'(T) = \Sigma\, e'(t)$
  - T is a decision tree, t is a leaf node, sum over all leaf nodes
- Methods for estimating generalization errors:
  1. Use validation (test) data to compute $e'(T)$ directly

  If validation data are not available, use training data:
  2. Optimistic approach: $e'(t) = e(t)$
  3. Pessimistic approach:
     - For each leaf node: $e'(t) = e(t) + \alpha$ (typically $\alpha \geq 0.5$)
     - For the entire tree: $e'(T) = e(T) + N \times \alpha$ (num. of leaf nodes: N)
     - Example:
       - For a tree with 30 leaf nodes and 10 errors on training (out of 1000 instances):
         - » Re-substitution error = 10/1000 = 1%
         - » Generalization error = (10 + 30×0.5)/1000 = 2.5%
- Reduced Error Pruning (REP):
  - Post-processing: use validation data to estimate generalization error, and to prune the tree to optimize $e'(T)$

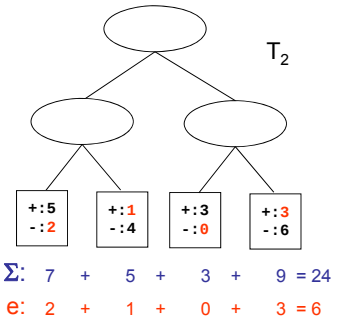
Exercise: Estimating Generalization Error (optimistic approach)

- Let's compare the two decision trees T_1 and T_2 using the re-substitution error



- training records: 24
- misclassifications on training records: 4

$$e'(T_1) = e(T_1) = \frac{4}{24} = 0.167$$

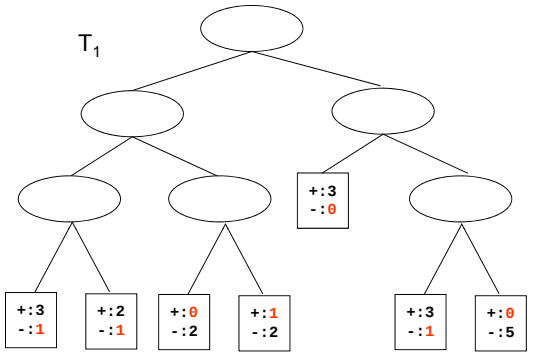


- training records: 24
- misclassifications on training records: 6

$$e'(T_2) = e(T_2) = \frac{6}{24} = 0.25$$

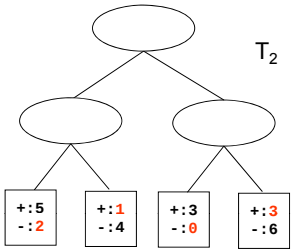
Exercise: Estimating Generalization Error (pessimistic approach)

- Let's compare the two decision trees T_1 and T_2 using a penalty term ($\alpha = 0.9$)



- training records: 24
- misclassifications on training records: 4
- number of leaf nodes: 7

$$e'(T_1) = e(T_1) + \alpha \cdot N = \frac{4 + 0.9 \cdot 7}{24} = 0.429$$



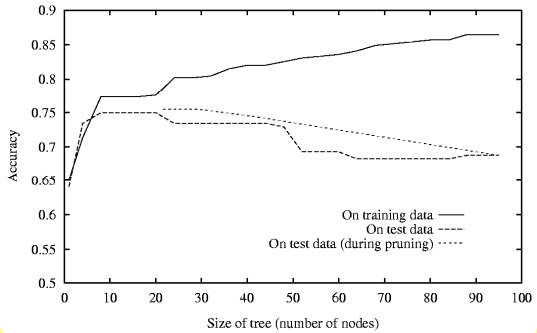
- training records: 24
- misclassifications on training records: 6
- number of leaf nodes: 4

$$e'(T_1) = e(T_1) + \alpha \cdot N = \frac{6 + 0.9 \cdot 4}{24} = 0.4$$

Reduced Error Pruning

- Split data into training and validation sets.
- Build tree on training set
- Post-processing (pruning): produces smallest version of most accurate subtree over validation set:
  - Repeat pruning step until further pruning is 'harmful':
    - evaluate impact on validation set of pruning each possible node (plus those below it)
    - greedily remove the one that most improves validation set accuracy

It requires that a lot of data be available.



Occam's Razor

"Pluralitas non est ponenda sine necessitate"


William of Ockham
(1285-1349)

- Given two models of similar generalization errors, one should prefer the simpler model over the more complex model.
  - For complex models, there is a greater chance that it was fitted accidentally by errors in data.
  - Therefore, one should include model complexity when evaluating a model.

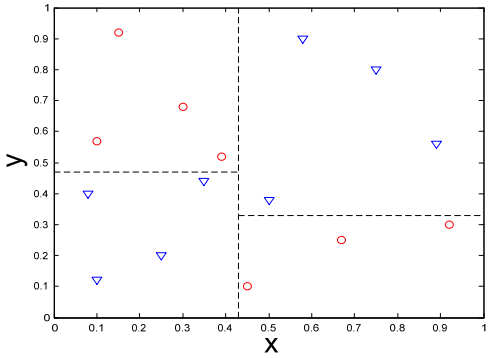
How to Address Overfitting

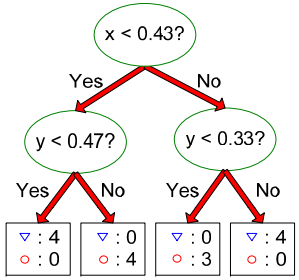
- Case 3: Pre-Pruning (Early Stopping Rule)
  - Stop the algorithm before it becomes a fully-grown tree
  - Typical stopping conditions for a node (cases 0, 1, 2):
    - Stop if all instances belong to the same class
    - Stop if all the attribute values are the same
  - More restrictive conditions:
    - Stop if number of instances is less than some user-specified threshold
    - Stop if class distribution of instances are independent of the available features (e.g., using χ^2 test)
    - Stop if expanding the current node does not improve impurity measures (e.g., Gini or Information Gain).

How to Address Overfitting…

- Post-pruning
  - Grow decision tree to its entirety
  - Trim the nodes of the decision tree in a bottom-up fashion
  - If generalization error improves after trimming, replace sub-tree by a leaf node.
  - Class label of leaf node is determined from majority class of instances in the sub-tree
  - Can use MDL for post-pruning

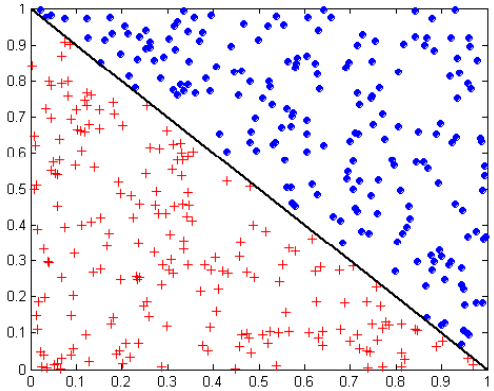
Decision Boundary

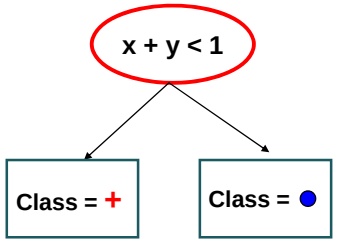




- **Border line between two neighboring regions of different classes is known as decision boundary**
- **Decision boundary is parallel to axes because test condition involves a single attribute at-a-time**

Oblique Decision Trees





- **Test condition may involve multiple attributes**
- **More expressive representation**
- **Finding optimal test condition is computationally expensive**

Summary

- Top-down induction of Decision Trees:
  - ID3, algorithm developed by Ross Quinlan
  - Gain ratio just one modification of this basic algorithm
  - C4.5: deals with numeric attributes, missing values, noisy data
- There are many other attribute selection criteria! (But little difference in accuracy of result)

Additional material:

- ID3
  - Induction: ID3 generates the model, i.e. the decision tree
    - Nominal attributes only, multi-way split
    - Information Gain / Gain Ratio
    - Opt.: early stopping (pre-pruning)
  - Deduction: apply the model to classify a set of unlabeled data
- Visualization of decision trees
  - Visualize the model produced by the tree induction (model)
  - Visualize the results produced by the deduction (classification)

ID3 Algorithm

```
ID3(in T : table; C : classification attribute) return node /* root node of decision tree */
{
    if (T is empty)
        then return(null);                              /* case 0: no more records */
    N := a new node;
    if (there are no predictive attributes in T)        /* case1: no more attributes */
        then label N with most common value of C in T
    elseif (all instances in T have the same value V of C)   /* case 2: homogeneous set of records */
        then label N, "X.C=V"
    else
    {
        for each attribute A in T
            compute AVGENTROPY(A,C,T);
        AS := the attribute for which AVGENTROPY(A,C,T) is minimal;
        if (AVGENTROPY(AS,C,T) is not substantially smaller than ENTROPY(C,T))   /* case 3: (opt) early stop */
            then label N with most common value of C in T
        else
        {
            label N with AS;
            for each value V of AS do
            {
                N1 := ID3(SUBTABLE(T,AS,V),C)          /* Recursive call */
                if (N1 !=null)
                    then make an arc from N to N1 labeled V;
            }
        }
    }
    return N;
}
```


ID3 Algorithm

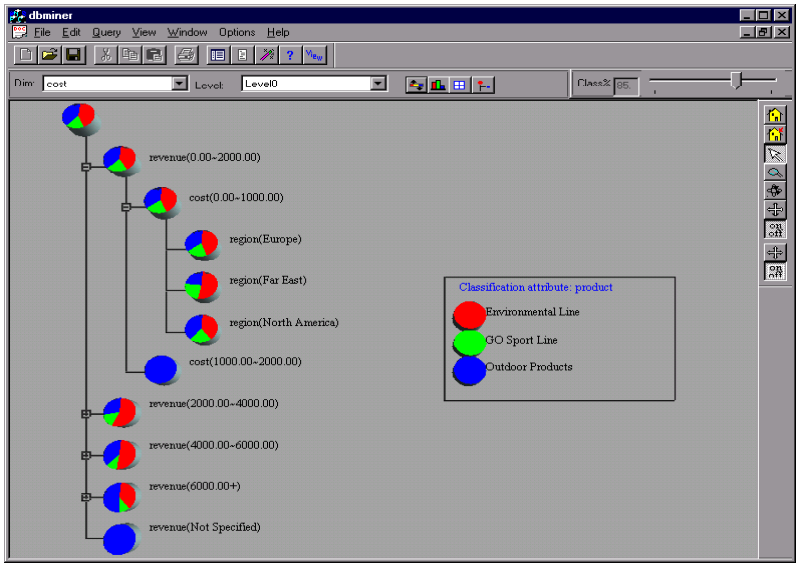
```
SUBTABLE(in T : table; A : predictive attribute; V : value) return table;
{
    T1 := the set of instance X in T such that X.A = V;
    T1 := delete column A from T1;
    return T1
}

ENTROPY(in C : classification attribute; T : table) return real number;
{
    for each value V of C, let p(V) := FREQUENCY(C,V,T);
    return -Σ_V p(V)log_2(p(V))          /* By convention, we consider (0 · log_2(0)) to be 0. */
}

AVGENTROPY(in A: predictive attribute; C : classification attribute; T : table)
    return real number;
{
    return Σ_V FREQUENCY(A,V,T) · ENTROPY(C,SUBTABLE(T,A,V))
}

FREQUENCY(in B : attribute; V : value; T : table) return real number;
{
    return #{X in T |X.B=V } / size(T);
}
```


Presentation of Classification Trees



The University of Reading

SE3DM11 - Data Mining

Association Rule Mining

Dr. Giuseppe Di Fatta
G.DiFatta@reading.ac.uk

Overview

- Association Rule Mining (ARM)
- Apriori algorithm
- Database layout
- Relevant subsets of frequent itemsets
  - Maximal and Closed Frequent Itemsets
- Taxonomy of ARM algorithms
- Extended Association Rules
- Other ARM algorithms
- References

What is Association Rule Mining (ARM)?

- ARM was introduced in 1993:
  R.Agrawal, T. Imielinski, A Swami, "Mining Association Rules between Sets of Items in Large Databases", Proc. of ACM-SIGMOD93 Conference.
- Given a set of attributes (*items*) I, and a set T of objects (*transactions*) containing a subset of the items,

List of Items
I = {Bread, Milk, Diaper, Beer, Eggs, Coke}

|I| = 6
|T| = 5

TID	*Transactions*
1	**Bread, Milk**
2	**Bread, Diaper, Beer, Eggs**
3	**Milk, Diaper, Beer, Coke**
4	**Bread, Milk, Diaper, Beer**
5	**Bread, Milk, Diaper, Coke**

- find rules that express the occurrence of items based on the occurrences of other items in the transactions.

Examples:

{Diaper} ⇒ {Beer}

{Milk, Bread} ⇒ {Eggs, Coke}

{Beer, Bread} ⇒ {Milk}

NB: Implication "⇒" means co-occurrence, not causality!

Diapers and Beer

Gösta Grahne and Jianfei Zhu's paper "Efficiently Using Prefix-trees in Mining Frequent Itemsets" received the best implementation award at FIMI03, the Workshop on Frequent Itemset Mining Implementations 2003.

Applications

- ✓ Marketing and Sales Promotion
  - What other products should the store stocks up?
  - Shelf management
  - Customer behavior analysis
- ✓ Prediction of the failure of telecommunication networks
  - What set of events occur prior to failure of a switching node?
- ✓ Medical applications
  - In Singapore, about 10 percent of the population is diabetic. In 1992 a regular screening program for the diabetic patients was introduced. Patient information, clinical symptoms, eye-disease diagnosis and treatments, etc., were captured in a database.
  - As a result of data mining, the doctors gained a much better understanding of how diabetes progresses over time and how different treatments affect its progress.
- ✓ Health Insurance Commission (HIC)
  - Associations on episode database for pathology services
    - 6.8 million records for 120 attributes (3.5GB)
    - 15 months preprocessing then 2 weeks data mining
  - Goal: find associations between tests
    - minConf = 50% and minSupp = 1%, 0.5%, 0.25% (1% of 6.8 million = 68,000)
    - Unexpected/unnecessary combination of services
    - Refusing cover would save $550,000 per year
- ✓ Tools for the drug discovery process

Definition of ARM

Association Rule Mining (ARM)

- Let I be a set of items, I = {A,B,C,…} with |I| = d. A set X = $\{i_1,\ldots,i_k\} \subseteq$ I is called an itemset.
- A transaction over I is a pair t = <tid, S>, where tid is the transaction identifier and S is an itemset. T is the set of transactions (|T| = n).
- A transaction t = <tid, S> is said to support an itemset X ⊆ I, if X ⊆ S.
  - Ex.: the transaction t=<09CF44, {bier, diapers, milk}> does support the itemset X={bier, diapers} and does not support the itemset Y={bier, coke}.
- X and Y are proper sets of items, X ⊂ I and Y ⊂ I, and X ∩ Y = ∅
- ARM problem: find all frequent and strong rules:

X ⇒ Y, "if X then Y" (body ⇒ head) (antecedent ⇒ consequent)

with sufficient support and confidence

Support of the rule = Prob(X ∪ Y) → joint probability

Confidence of the rule = Prob(Y | X) → conditional probability

Definition of Frequent Itemset

- Itemset
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- Support of an itemset
  - The support of an itemset c, c ⊆ I, is the frequency (either absolute or relative value) of its occurrence in the transactions, i.e. the fraction of transactions that contain the itemset.

$$s(c) = Support(c) = \frac{\left|\{t \mid t \in T, c \subseteq itemset(t)\}\right|}{|T|} = \frac{\sigma(c)}{|T|}$$

(σ(c): support count)

- Frequent Itemset (FI)
  - An itemset whose support is greater than or equal to a ***minSup*** threshold.

s(c) ≥ minSup

Interesting Association Rules

- ARM problem: find all frequent and strong rules:

  X ⇒ Y, "if X then Y"

  with sufficient support and confidence
- Given an association rule X ⇒ Y, where X, Y ⊆ I,
  - Def.: **support of a rule** is given by the fraction of the transactions that contain both X and Y.

    Support(X ⇒ Y) = Support(X ∪ Y)
  - Def.: **confidence of a rule** is a measures how often items in Y appear in transactions that contain X

    Confidence(X ⇒ Y) = Support(X ∪ Y) / Support(X)

  Given two thresholds minSup and minConf, interesting rules are those such that:
  - Support(X ⇒ Y) ≥ ***minSup***
  - Confidence(X ⇒ Y) ≥ ***minConf***

Example

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Association Rule Mining Task

- Given a set of transactions T, the goal of association rule mining is to find ALL rules having
  - support ≥ *minSup* threshold
  - confidence ≥ *minConf* threshold

- **Brute-force approach**:
  1. List all possible association rules
  2. Compute the support and confidence for each rule
  3. Prune rules that fail the *minSup* and *minConf* thresholds

⇒ Computationally prohibitive!

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$$

Proof derived from the binomial theorem:

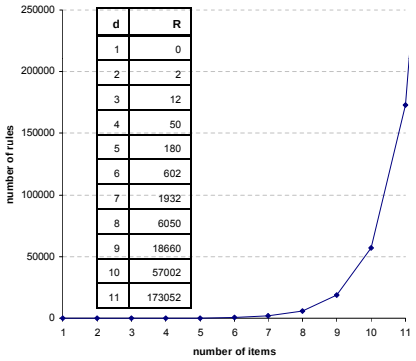
$$\sum_{k=0}^{N}\binom{N}{k} \times r^k = (1+r)^N$$

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$$

d	R
1	0
2	2
3	12
4	50
5	180
6	602
7	1932
8	6050
9	18660
10	57002
11	173052



Example for I={A,B,C}, d=3:

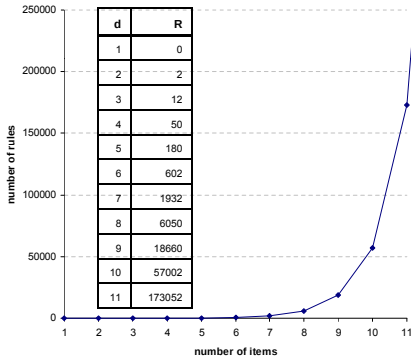
1. A→B	7. C→A
2. A→C	8. C→B
3. A→BC	9. C→AB
4. B→A	10. AB→C
5. B→C	11. AC→B
6. B→AC	12. BC→A

Total Number of Rules

- Given a set I with d unique items, |I|=d:
  - the total number of itemsets is the cardinality of the power set, |Power(I)| = 2^d
  - the total number R of possible association rules is:

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right] = 3^d - 2^{d+1} + 1$$

d	R
1	0
2	2
3	12
4	50
5	180
6	602
7	1932
8	6050
9	18660
10	57002
11	173052



Example for I={A,B,C}, d=3:

1. A→B	7. A→BC
2. A→C	8. B→AC
3. B→A	9. C→AB
4. B→C	10. AB→C
5. C→A	11. AC→B
6. C→B	12. BC→A

Support and Confidence

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} ⇒ {Beer} (s=0.4, c=0.67)
{Milk,Beer} ⇒ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} ⇒ {Milk} (s=0.4, c=0.67)
{Beer} ⇒ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} ⇒ {Milk,Beer} (s=0.4, c=0.5)
{Milk} ⇒ {Diaper,Beer} (s=0.4, c=0.5)

- All the above rules are binary partitions of the same itemset: {Milk, Diaper, Beer}
- Rules originating from the same itemset have **identical support** but can have **different confidence**.
- Thus, we may **decouple the support and confidence** requirements

Two Phases Process

Association Rule Mining (ARM)

Discover all association rules <X ⇒ Y>
with a minSupp and a minConf (user parameters)

Two phases:

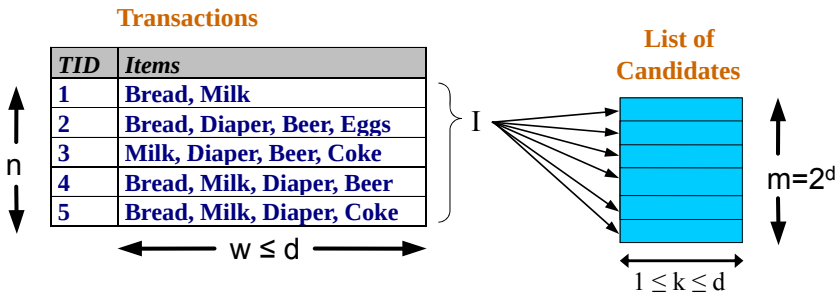
1. find all Frequent Itemsets (FI) ← Most of the complexity
   - all possible itemsets with a minimum support
     FI = {c, c ⊆ I| s(c) ≥ minSupp}
2. build strong association rules
   - Use the frequent itemsets FI to build rules with a minimum confidence:
     Rules = {X⇒Y | X,Y ⊂ I, X∩Y=∅, X∪Y∈FI, c(X⇒Y)≥minConf}

Frequent Itemsets Generation

- In order to find all frequent itemsets we have to match each transaction (|T| = n) against every candidate (m=|Power(I)|=2^d, where |I|=d): $O(n \cdot 2^d)$.

  Def.: a **candidate itemset** is an itemset that has been generated and is potentially frequent. We still need to count its support.

- Each match requires a subset test, at most (w·k) or max(w,k) comparisons: $O(d)$.



Complexity ~ O(n · d · 2^d) => Still exponential!

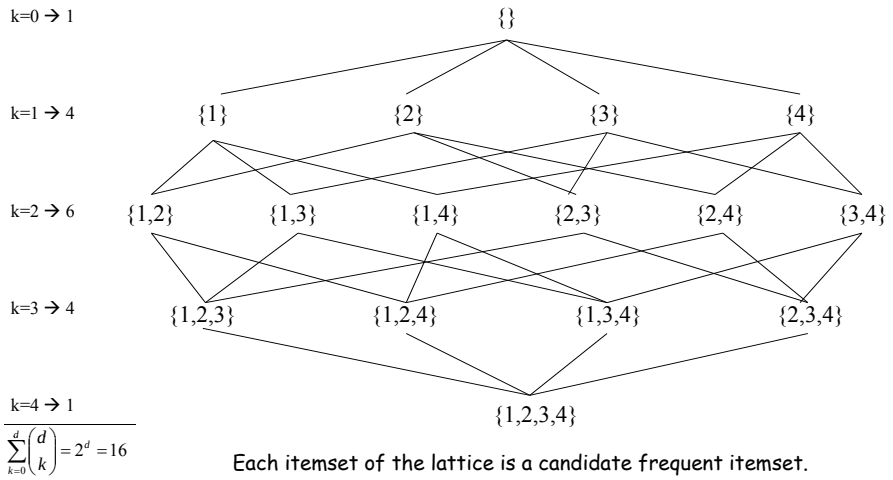
Frequent Itemsets

How can we efficiently generate FI?

- Reducing the **number of candidates** (m)
  - Complete search: m=2^d
  - Use pruning techniques to reduce m
- Reducing the **number of transactions** (n)
  - Reduce size of T as the size of itemsets increases
  - Use a subsample of the transactions
- Reducing the **number of comparisons** (w·k)
  - Use efficient data structures to store the candidates or the transactions
  - No need to match every candidate against every transaction

Lattice of Subsets of I={1,2,3,4}

$$\binom{d}{k} = \frac{d!}{k!(d-k)!}$$



$$\sum_{k=0}^{d}\binom{d}{k} = 2^d = 16$$

Each itemset of the lattice is a candidate frequent itemset.

…reducing the number of candidates

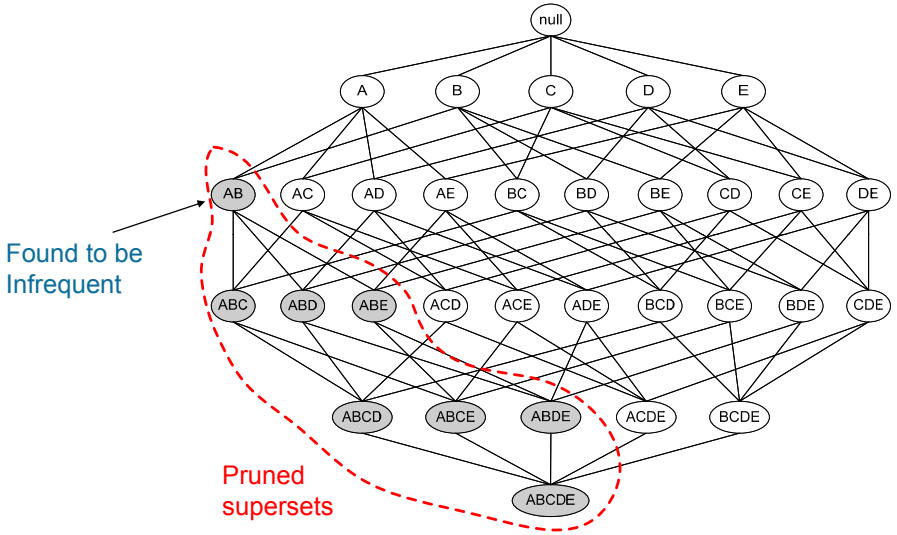
Apriori algorithm was introduced in 1994:

"Fast algorithms for mining association rules", R. Agrawal and R. Srikant, in VLDB'94, the Twentieth Very Large Data Base Conference, 1994.

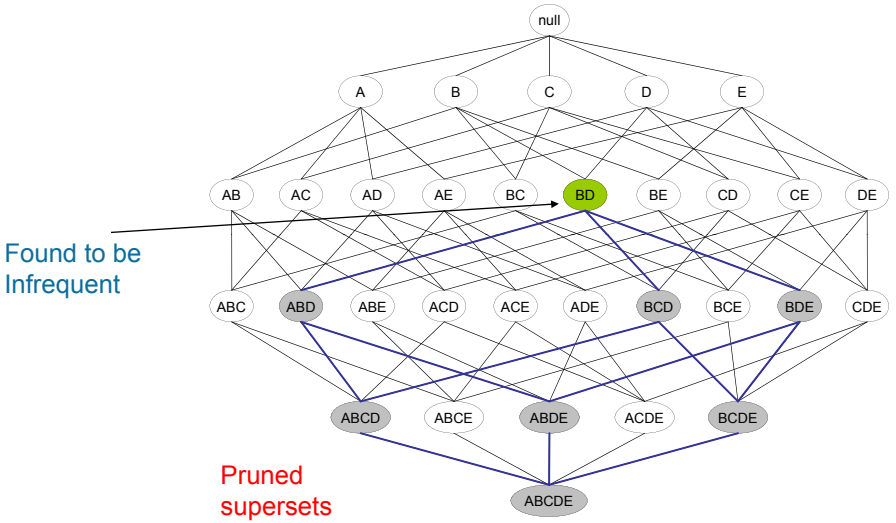
- Apriori principle:
  if an itemset is frequent, then all of its subsets must also be frequent
  - Apriori principle holds due to the following property of the support measure:
  - The support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of the support.

$$\forall X, Y : (X \subseteq Y) \rightarrow s(X) \geq s(Y)$$

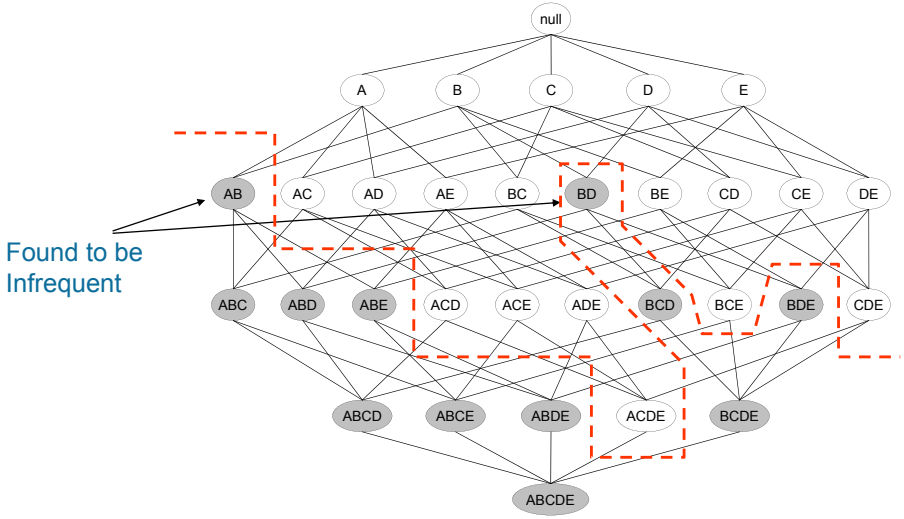
Anti-monotone Property



Anti-monotone Property

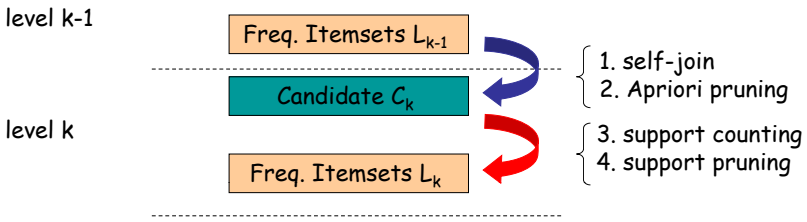


Border of Frequent Itemsets



Apriori Algorithm

- Bread First Search (BFS) of all possible candidates:
  1. At level k, generate all candidate k-itemsets C_k (of length k), using frequent itemsets L_{k-1} generated at the previous level: self-join.
  2. Prune the search tree using the anti-monotone property of the support:
     $x2 \supseteq x1 \rightarrow support(x2) \leq support(x1)$
  3. Count the support of the candidate C_k: one DB scan.
  4. Prune infrequent candidates and generate L_k.
- Efficient computation of the support
  - Candidate itemsets are stored in a hash-tree



Apriori Algorithm

1. L_1 = {frequent 1-itemsets};
2. for (k=2; $L_{k-1} \neq \emptyset$; k++) do begin
3. C_k = ***apriori-gen***(L_{k-1}); //New candidates
4. for all trans t∈T do begin
5. C_t = ***subset***(C_k,t); //Candidates contained in t
6. for all candidates c ∈C_t do
7. c.count++;
8. end
9. L_k = {c ∈C_k | c.count ≥ minSup}
10. end
11. return FI = $\cup_k L_k$

Apriori-gen Method

Let's assume that the items in L_{k-1} are listed in an order.

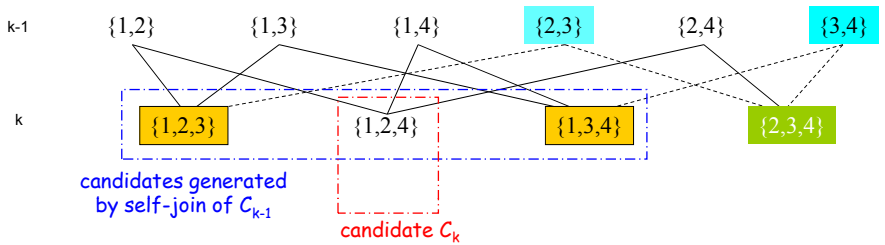
C_k = apriori-gen(L_{k-1}):
For all itemsets X,Y in L_{k-1} , X[i]=Y[i] for $1 \le i \le k-2$, and X[k-1] < Y[k-1]
I = X U {Y[k-1]}
for each J⊂I, |J|=k-1
if J∈L_{k-1} then C_k = C_k U I

Candidate Generation: Example

Generation step:
1. {1,2} U {1,3} = {1,2,3}
2. {1,2} U {1,4} = {1,2,4}
3. {1,3} U {1,4} = {1,3,4}
✓ {2,3} is not frequent → skipped
✓ {2,4} has no frequent partner
✓ {3,4} is not frequent → skipped

Pruning step:
1. {1,2,3} pruned because its subset {2,3} is infrequent
2. {1,3,4} pruned because its subset {3,4} is infrequent

not frequent
pruned
not generated



…reducing the number of comparisons

- **Apriori candidate counting**:
  - We need to scan the database of transactions to determine the support of each candidate k-itemset
  - To reduce the number of comparisons, store the candidates in a hash structure (x buckets)
    - Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets

Transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

n

Hash Structure

candidate of level k

$\le \binom{d}{k}$

x

x buckets to store the candidate of level k

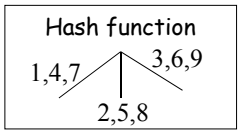
Hash Tree

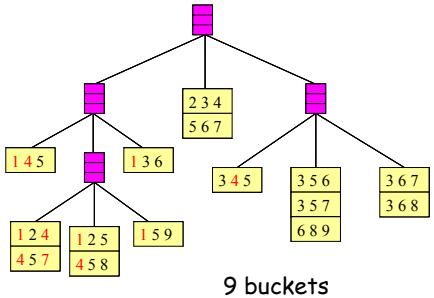
Suppose you have 15 candidate itemsets of length 3:

{1 4 5}, {1 2 4}, {4 5 7}, {1 2 5}, {4 5 8},
{1 5 9}, {1 3 6}, {2 3 4}, {5 6 7}, {3 4 5},
{3 5 6}, {3 5 7}, {6 8 9}, {3 6 7}, {3 6 8}

We need:

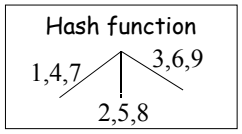
- Hash function
- Max leaf size: max number of itemsets stored in a leaf node
  (if number of candidate itemsets exceeds max leaf size, split the node)

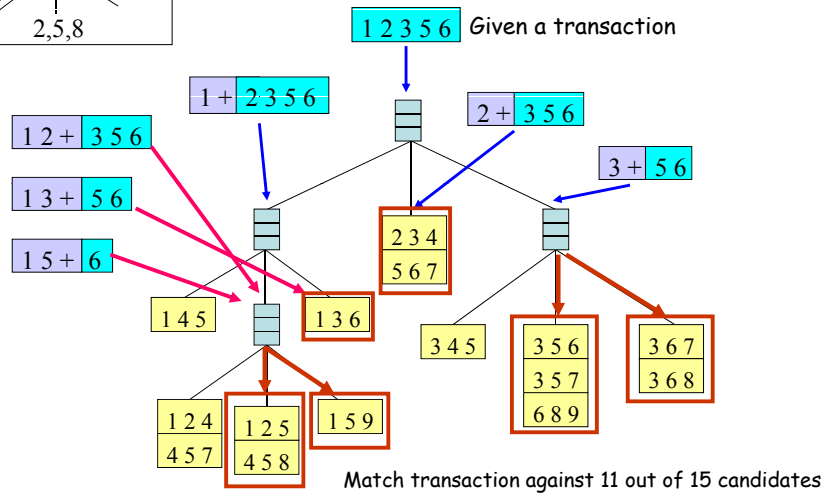




9 buckets

Support Counting





Match transaction against 11 out of 15 candidates

Database Layout

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

- **Horizontal layout**:
  TransactionID – list of items (Transactional)

TID	Bread	Beer	Coke	Diapers	Eggs	Milk
1	1	0	0	0	0	1
2	1	1	0	1	1	0
3	0	1	1	1	0	1
4	1	1	0	1	0	1
5	1	0	1	1	0	1

- **Vertical layout**:
  Item – List of transactions (TID-list)

TID	Bread	Beer	Coke	Diapers	Eggs	Milk
1	1	0	0	0	0	1
2	1	1	0	1	1	0
3	0	1	1	1	0	1
4	1	1	0	1	0	1
5	1	0	1	1	0	1

TID	1	2	3	4	5
Bread	1	1	0	1	1
Beer	0	1	1	1	0
Coke	0	0	1	0	1
Diapers	0	1	1	1	1
Eggs	0	1	0	0	0
Milk	1	0	1	1	1

- In the **vertical layout** all data for a particular item is available in one record.
  - to count the itemset {beer, diaper} **intersect** TID-list of the item "beer" with TID-list of item "diaper".

Database Projection

Given a transaction database and an itemset X

- Database ***selection*** operation:
  select only those records that contain X.
- Databases ***projection*** operation:
  delete the items of X from the selected records.

TID	Transactions
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

{beer}-projection

TID	Transactions
2	Bread, Diaper, Eggs
3	Milk, Diaper, Coke
4	Bread, Milk, Diaper

Compact Representations

Some itemsets are **redundant** because they have identical support as their supersets.

A method of reducing the large amount of frequent itemsets is to use so-called **compact or condensed representations**, such as

- **Maximal Frequent Itemset** (MFI) and
- **Closed Frequent Itemset** (CFI).

MFI are frequent itemsets for which none of their supersets is also frequent. Clearly, each frequent itemset is a subset of a maximal frequent itemset.

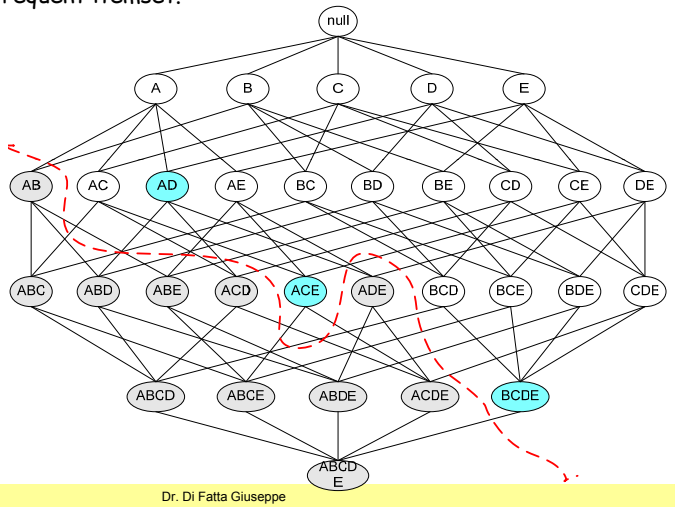
CFI are itemsets that completely characterize their associated set of transactions. That is, a frequent itemset is closed if it contains all items that occur in all transaction in which it is bought, i.e. it is the intersection of its supporting transactions.

The underlying idea of both concepts is that the set of all maximal/closed frequent itemsets represents all frequent itemsets but is far smaller.

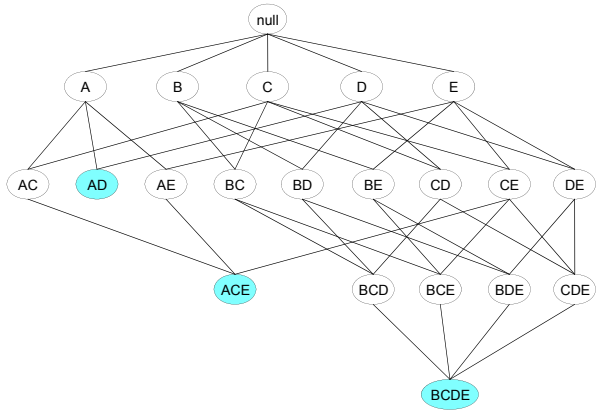
Maximal Frequent Itemsets

***Definition 1**: Maximal Frequent Itemset (MFI)*

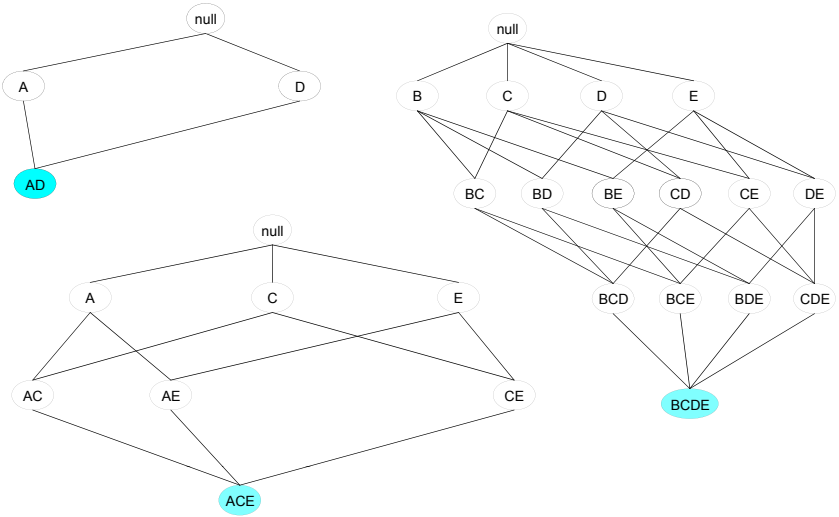
A frequent itemset is maximal if it is not a proper subset of any other frequent itemset.



Maximal Frequent Itemsets



Sublattice induced by MFI



Closed Frequent Itemsets

Problem with maximal frequent itemsets:
support of their subsets is not known and additional DB scans are needed.

***Definition 2**: Closed Frequent Itemset (CFI)*
A closed frequent itemset is a frequent itemset whose support is higher than the supports of all its proper supersets.
(All maximal frequent itemsets are closed.)

Example: minSup=2

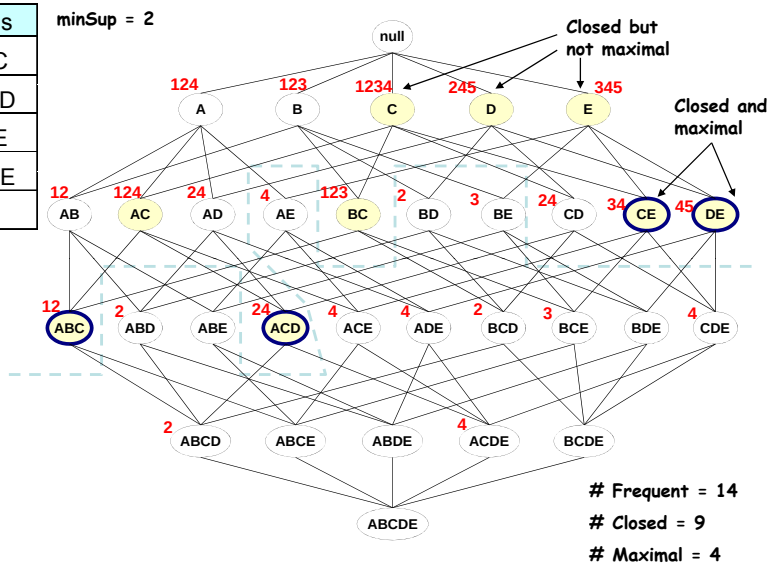
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	3
{A,B}	4
{A,C}	2
{A,D}	2
{B,C}	3
{B,D}	3
{C,D}	2

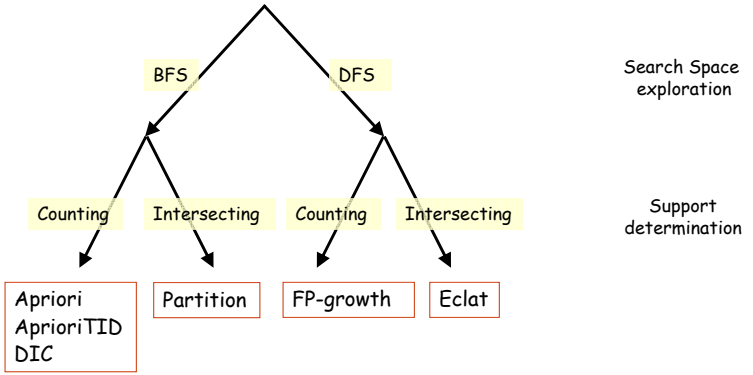
Itemset	Support
{A,B,C}	2
{A,B,D}	2
{A,C,D}	1
{B,C,D}	2
{A,B,C,D}	1

MFI vs CFI

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

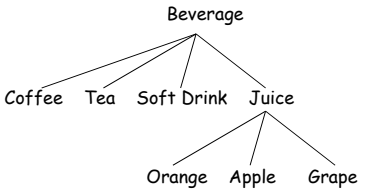


Taxonomy of ARM Approaches



Extended Association Rules

- Generalized Association Rules
  - hierarchical taxonomy (concept hierarchy)
- Quantitative Association Rules
  - categorical and quantitative data
- Interval Data Association Rules
  - E.g., partition the age into 5-year-increment ranges
- Maximal Association Rules
- Sequential Association Rules
  - temporal data
  - E.g., first buy a PC, then CDROMs, and, finally, a digital camera



Other ARM algorithms

- AprioriTID and AprioriHybrid
- PARTITION
- DIC
- ECLAT
- Prefix-Tree based
  - FP-Growth
  - FP-Tree

AprioriTID and AprioriHybrid

The database is not directly used for the counting phase.

Let be C_k the set of candidate itemsets.

At each iteration k:

- generate the set $\overline{C}_k = \{ \langle tid, \{c \in C_k \mid c \subseteq itemset(tid) \}\rangle \}$
  (replacing every transaction in the database by the set of candidates that occur in that transaction)

If a transaction does not contain any candidate k-itemset, it will not have an entry in $\overline{C}_k$. Thus, the number of transactions will decrease for larger k.

But for small k it may not fit into main memory: in a transaction there may be more candidates than items.

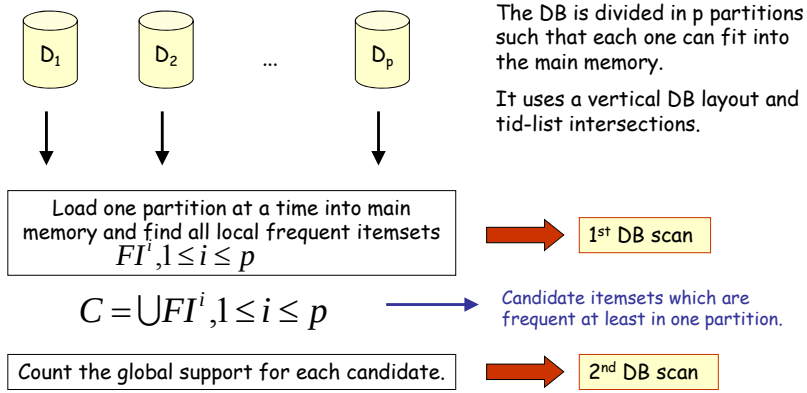
For small k AprioriTID can be slower than Apriori.

For large k it will be more efficient.

The size of the set $\overline{C}_k$ is proportional to the sum of the support of all candidates in C_k.

AprioriHybrid adopts a heuristic to switch between Apriori and AprioriTID.

PARTITION



The DB is divided in p partitions such that each one can fit into the main memory.

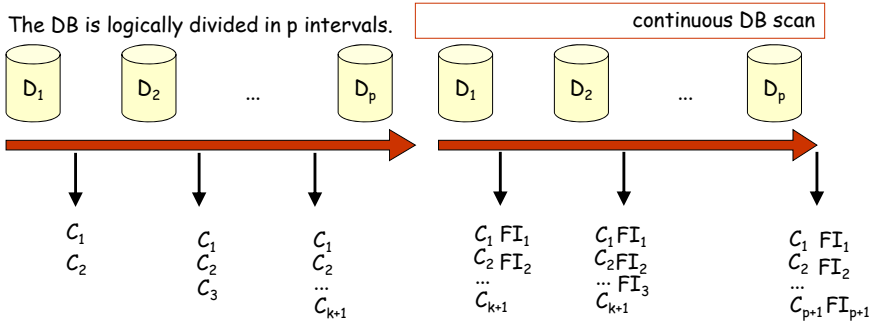
It uses a vertical DB layout and tid-list intersections.

For homogeneous data distribution the algorithm performs well.

However, for skewed data distributions many false candidates will be generated in the first scan and counted in the second scan.

DIC

Dynamic Itemset Counting



At the end of each interval k generate and count candidate upto(k+1)-itemsets.

If interval size is small enough, the algorithm complete in **two** scans.

Similarly to PARTITION, for homogeneous data distribution the algorithm performs well.

ECLAT

- First DFS approach (1997, M. J. Zaki et al.)
- Vertical database layout (tid-list "fast intersection")
- The Apriori principle cannot be fully exploited
  - only the join step of Apriori; all itemsets necessary for the pruning step are not available.
- Each step it generates the frequent k-itemsets with the same (k-1)-prefix.
- 3 traversal approaches: top-down, bottom-up and hybrid
- vs. Apriori: more efficient support counting
- vs. Partition: the number of tid-list in main memory is much less
- MaxEclat generates only maximal frequent itemsets

Prefix-Tree (Trie)

Definition: Trie is a tree for storing strings in which there is one node for every common prefix. The strings are stored in extra leaf nodes.

Note: The name comes from re**TRIE**val and is pronounced "tree".

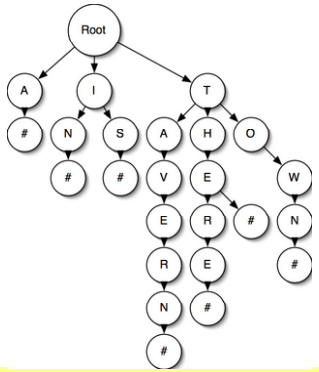
Trie in ARM

The idea is that all itemsets sharing a common *prefix* (stem) hang off a common node.
When the strings are words over {a..z}, a node has at most 27 children - one for each letter plus a terminator.
With itemsets a node has, in general, less potential children: it depends on the level and on the item of the last branch.

A trie can be adopted to store candidates, to count the support and to generates new candidates.
To process a transaction we can follow the same procedure as in the Apriori hash-tree. The difference is the number of branches from each node.

- Hash-tree: # of branches are defined by the hash function. It is not a prefix-tree.
- Trie: each different item generates a different branch. It is a prefix-tree.

Example:
{a, in, is, tavern, the, there, town}

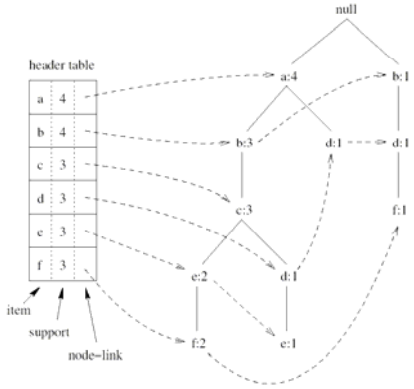


FP-Growth

FP-Growth, Frequent Pattern Growth

- (2000, J. Han, J. Pei and Y. Yin)
- DFS approach
- Combination of horizontal and vertical database layout:
  - trie data structure called FP-tree (compressed DB representation)
  - header table

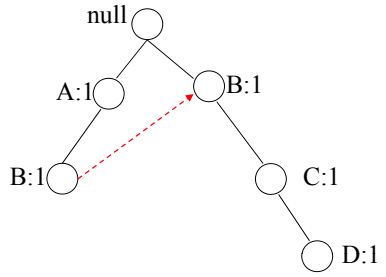
TID	Itemset
1	{a,b,c,d,e,f}
2	{a,b,c,d,e}
3	{a,d}
4	{b,d,f}
5	{a,b,c,d,e,f}



FP-Tree

- First, items are sorted by their frequency in decreasing order
- Then, the FP-Tree is built by scanning the DB transactions
- Finally, recursive divide-and-conquer approach in frequency-increasing order.

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{A}
11	{B,C,E}



FP-Tree

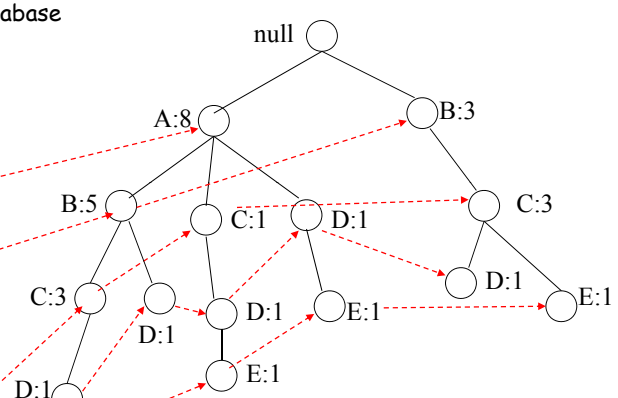
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{A}
11	{B,C,E}

Transaction Database

minSup=2

Header table

Item	s	Pointer
A	8	
B	8	
C	7	
D	5	
E	3	

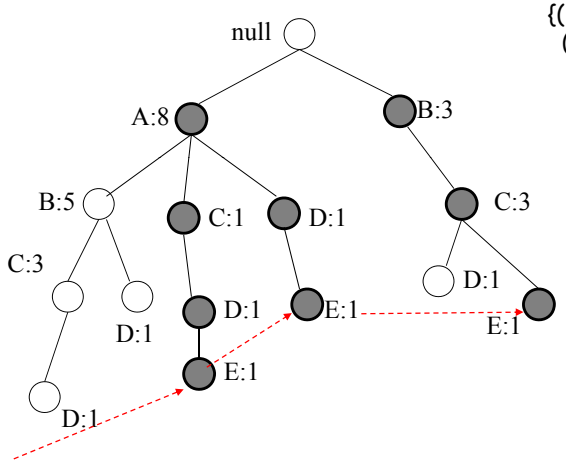


Pointers are used to assist frequent itemset generation

FP-Growth

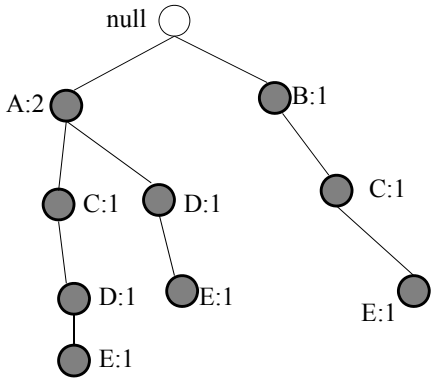
Build conditional pattern base for E:
prefix paths for E are

{(A,C,D),
(A,D),
(B,C)}



FP-Growth

Conditional tree for E:

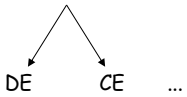


Conditional Pattern base for E:
P_E = {(A:1,C:1,D:1,E:1),
(A:1,D:1,E:1),
(B:1,C:1,E:1)}

Count for E is $|P_E| = 3$

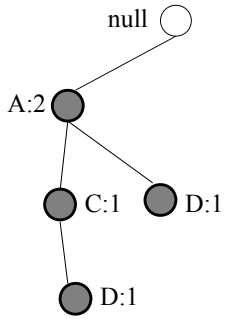
{E} is a frequent itemset

Recursively apply FP-growth on P_E



FP-Growth

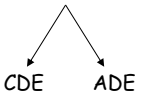
Conditional tree for D within conditional tree for E:



Conditional pattern base for D within conditional base for E:
P_{DE} = {(A:1,C:1,D:1),
(A:1,D:1)}

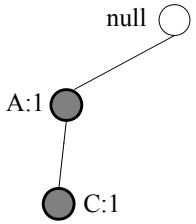
Count for D is 2: {D,E} is frequent itemset

Recursively apply FP-growth on P_{DE}



FP-Growth

Conditional tree for C within D within E:



Conditional pattern base for C within D within E:

P_{CDE} = {(A:1,C:1)}

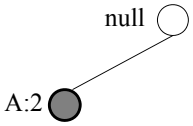
Count for C is 1: {C,D,E} is NOT frequent itemset

---backtrack---

Next step: ADE

FP-Growth

Conditional tree for A within D within E:



Count for A is 2:

P_{ADE} = {(A:1,D:1,E:1), (A:1,C:1,D:1,E:1)}

{A,D,E} is frequent itemset

---backtrack---

Next step: CE

Construct conditional tree C within conditional tree E

Continue until exploring conditional tree for A (which has only node A)

ARM References

- R. Agrawal, T. Imielinski, and A. Swami, Mining Association Rules Between Sets of Items in Large Database. Proc. ACM-SIGMOD International Conference, May 1993.
- R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules. Proc International Conference on Very Large Databases, Santiago, Chile, September 1994.
- R. Srikant and R. Agrawal, Mining Quantitative Association Rules in Large Relational Tables, SIGMOD, June 1996.
- Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In Proceedings of ACM SIGMOD'00,pages 1–12, May 2000.
- M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In D. Heckerman, H. Mannila, and D. Pregibon, editors, Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, pages 283–286. AAAI Press, 1997.
- A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules in large databases. In Dayal et al. [17], pages 432–444.
- H. Toivonen. Sampling large databases for association rules. In T.M. Vijayaraman, A.P. Buchmann, C. Mohan, and N.L. Sarda, editors, Proceedings 22nd International Conference on Very Large Data Bases, pages 134–145. Morgan Kaufmann, 1996.
- S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data, volume 26(2) of SIGMOD Record, pages 255–264. ACM Press, 1997.
- Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh, "Algorithms for Association Rule Mining – A General Survey and Comparison", ACM SIGKDD, July 2000.