Chapter 26: Data Mining

(Some slides courtesy of
Rich Caruana, Cornell University)

Definition

Data mining is the exploration and analysis of large quantities of data in order to discover valid, novel, potentially useful, and ultimately understandable patterns in data.

Example pattern (Census Bureau Data):
If (relationship = husband), then (gender = male). 99.6%

Definition (Cont.)

Data mining is the exploration and analysis of large quantities of data in order to discover valid, novel, potentially useful, and ultimately understandable patterns in data.

Valid: The patterns hold in general.
Novel: We did not know the pattern beforehand.
Useful: We can devise actions from the patterns.
Understandable: We can interpret and comprehend the patterns.

Why Use Data Mining Today?

Human analysis skills are inadequate:

- Volume and dimensionality of the data
- High data growth rate

Availability of:

- Data
- Storage
- Computational power
- Off-the-shelf software
- Expertise

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

An Abundance of Data

- Supermarket scanners, POS data
- Preferred customer cards
- Credit card transactions
- Direct mail response
- Call center records
- ATM machines
- Demographic data
- Sensor networks
- Cameras
- Web server logs
- Customer web site trails

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Evolution of Database Technology

- 1960s: IMS, network model
- 1970s: The relational data model, first relational DBMS implementations
- 1980s: Maturing RDBMS, application-specific DBMS, (spatial data, scientific data, image data, etc.), OODBMS
- 1990s: Mature, high-performance RDBMS technology, parallel DBMS, terabyte data warehouses, object-relational DBMS, middleware and web technology
- 2000s: High availability, zero-administration, seamless integration into business processes
- 2010: Sensor database systems, databases on embedded systems, P2P database systems, large-scale pub/sub systems, ???

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Computational Power

- Moore's Law:
  In 1965, Intel Corporation cofounder Gordon Moore predicted that the density of transistors in an integrated circuit would double every year. (Later changed to reflect 18 months progress.)

- Experts on ants estimate that there are 10^{16} to 10^{17} ants on earth. In the year 1997, we produced one transistor per ant.

Much Commercial Support

- Many data mining tools
  - http://www.kdnuggets.com/software
- Database systems with data mining support
- Visualization tools
- Data mining process support
- Consultants

Why Use Data Mining Today?

Competitive pressure!
"The secret of success is to know something that nobody else knows."
Aristotle Onassis

- Competition on service, not only on price (Banks, phone companies, hotel chains, rental car companies)
- Personalization, CRM
- The real-time enterprise
- "Systemic listening"
- Security, homeland defense

The Knowledge Discovery Process

Steps:

1. Identify business problem
2. Data mining
3. Action
4. Evaluation and measurement
5. Deployment and integration into businesses processes

Data Mining Step in Detail

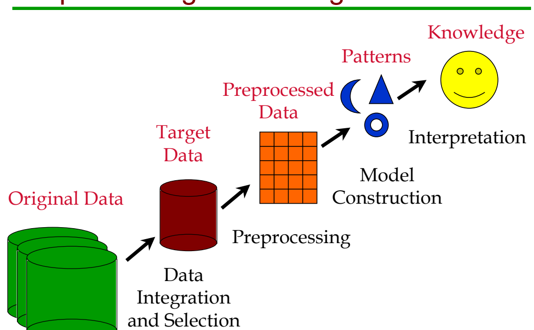
2.1 Data preprocessing

- Data selection: Identify target datasets and relevant fields
- Data cleaning
  - Remove noise and outliers
  - Data transformation
  - Create common units
  - Generate new fields

2.2 Data mining model construction

2.3 Model evaluation

Preprocessing and Mining



Example Application: Sports

IBM Advanced Scout analyzes NBA game statistics
- Shots blocked
- Assists
- Fouls

- Google: "IBM Advanced Scout"

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Advanced Scout

- Example pattern: An analysis of the data from a game played between the New York Knicks and the Charlotte Hornets revealed that "*When Glenn Rice played the shooting guard position, he shot 5/6 (83%) on jump shots.*"
- Pattern is interesting:
  The average shooting percentage for the Charlotte Hornets during that game was 54%.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Example Application: Sky Survey

- Input data: 3 TB of image data with 2 billion sky objects, took more than six years to complete
- Goal: Generate a catalog with all objects and their type
- Method: Use decision trees as data mining model
- Results:
  - 94% accuracy in predicting sky object classes
  - Increased number of faint objects classified by 300%
  - Helped team of astronomers to discover 16 new high red-shift quasars in one order of magnitude less observation time

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Gold Nuggets?

- Investment firm mailing list: Discovered that old people do not respond to IRA mailings
- Bank clustered their customers. One cluster: Older customers, no mortgage, less likely to have a credit card
- "Bank of 1911"
- Customer churn example

What is a Data Mining Model?

A data mining model is a description of a specific aspect of a dataset. It produces output values for an assigned set of input values.

Examples:

- Linear regression model
- Classification model
- Clustering

Data Mining Models (Contd.)

A data mining model can be described at two levels:

- Functional level:
  - Describes model in terms of its intended usage. Examples: Classification, clustering
- Representational level:
  - Specific representation of a model. Example: Log-linear model, classification tree, nearest neighbor method.
- Black-box models versus transparent models

Data Mining: Types of Data

- Relational data and transactional data
- Spatial and temporal data, spatio-temporal observations
- Time-series data
- Text
- Images, video
- Mixtures of data
- Sequence data

- Features from processing other data sources

Types of Variables

- *Numerical*: Domain is ordered and can be represented on the real line (e.g., age, income)
- *Nominal* or *categorical*: Domain is a finite set without any natural ordering (e.g., occupation, marital status, race)
- *Ordinal*: Domain is ordered, but absolute differences between values is unknown (e.g., preference scale, severity of an injury)

Data Mining Techniques

- Supervised learning
  - Classification and regression
- Unsupervised learning
  - Clustering
- Dependency modeling
  - Associations, summarization, causality
- Outlier and deviation detection
- Trend analysis and change detection

Supervised Learning

- F(x): true function (usually not known)
- D: training sample drawn from F(x)

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0	0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0	1
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	0

Supervised Learning

- F(x): true function (usually not known)
- D: training sample (x,F(x))

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1

- G(x): model learned from D

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0	?

- Goal: $E[(F(x)-G(x))^2]$ is small (near zero) for future samples

Supervised Learning

Well-defined goal:

Learn G(x) that is a good approximation to F(x) from training sample D

Well-defined error metrics:

Accuracy, RMSE, ROC, …

Supervised Learning

Training dataset:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	0

Test dataset:

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 ?

Un-Supervised Learning

Training dataset:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	~~0~~
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	~~0~~
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	~~1~~
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	~~1~~
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	~~0~~
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	~~0~~

Test dataset:

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 ?

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Un-Supervised Learning

Training dataset:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	~~0~~
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	~~0~~
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	~~1~~
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	~~1~~
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	~~0~~
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	~~0~~

Test dataset:

~~71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0~~ ?

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Un-Supervised Learning

Data Set:

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1
```


Lecture Overview

- Data Mining I: Decision Trees
- Data Mining II: Clustering
- Data Mining III: Association Analysis

Classification Example

- Example training database
  - Two predictor attributes: Age and Car-type (**S**port, **M**inivan and **T**ruck)
  - Age is ordered, Car-type is categorical attribute
  - Class label indicates whether person bought product
  - Dependent attribute is *categorical*

Age	Car	Class
20	M	Yes
30	M	Yes
25	T	No
30	S	Yes
40	S	Yes
20	T	No
30	M	Yes
25	M	Yes
40	M	Yes
20	S	No

Regression Example

- Example training database
  - Two predictor attributes: Age and Car-type (**S**port, **M**inivan and **T**ruck)
  - Spent indicates how much person spent during a recent visit to the web site
  - Dependent attribute is *numerical*

Age	Car	Spent
20	M	$200
30	M	$150
25	T	$300
30	S	$220
40	S	$400
20	T	$80
30	M	$100
25	M	$125
40	M	$500
20	S	$420

Types of Variables (Review)

- *Numerical*: Domain is ordered and can be represented on the real line (e.g., age, income)
- *Nominal* or *categorical*: Domain is a finite set without any natural ordering (e.g., occupation, marital status, race)
- *Ordinal*: Domain is ordered, but absolute differences between values is unknown (e.g., preference scale, severity of an injury)

Definitions

- Random variables $X_1, \ldots, X_k$ (*predictor variables*) and Y (*dependent variable*)
- X_i has domain $dom(X_i)$, Y has domain dom(Y)
- P is a probability distribution on $dom(X_1) \times \ldots \times dom(X_k) \times dom(Y)$
  Training database D is a random sample from P
- A *predictor* d is a function
  $d: dom(X_1) \ldots dom(X_k) \rightarrow dom(Y)$

Classification Problem

- If Y is categorical, the problem is a *classification problem*, and we use C instead of Y. |dom(C)| = J.
- C is called the *class label*, d is called a *classifier.*
- Take r be record randomly drawn from P. Define the *misclassification rate* of d: $RT(d,P) = P(d(r.X_1, \ldots, r.X_k) \neq r.C)$
- Problem definition: Given dataset D that is a random sample from probability distribution P, find classifier d such that RT(d,P) is minimized.

Regression Problem

- If Y is numerical, the problem is a *regression problem.*
- Y is called the dependent variable, d is called a *regression function.*
- Take r be record randomly drawn from P. Define mean squared error rate of d: $RT(d,P) = E(r.Y - d(r.X_1, \ldots, r.X_k))^2$
- Problem definition: Given dataset D that is a random sample from probability distribution P, find regression function d such that RT(d,P) is minimized.

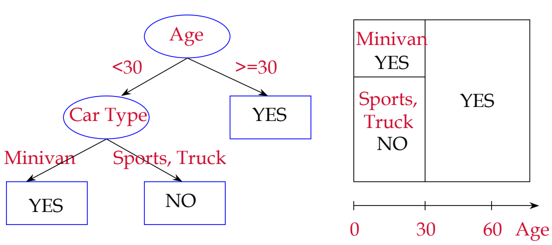
Goals and Requirements

- Goals:
  - To produce an accurate classifier/regression function
  - To understand the structure of the problem
- Requirements on the model:
  - High accuracy
  - Understandable by humans, interpretable
  - Fast construction for very large training databases

Different Types of Classifiers

- Linear discriminant analysis (LDA)
- Quadratic discriminant analysis (QDA)
- Density estimation methods
- Nearest neighbor methods
- Logistic regression
- Neural networks
- Fuzzy set theory
- Decision Trees

What are Decision Trees?



Decision Trees

- A *decision tree* T encodes d (a classifier or regression function) in form of a tree.
- A node t in T without children is called a *leaf node*. Otherwise t is called an *internal node*.

Internal Nodes

- Each internal node has an associated *splitting predicate*. Most common are binary predicates.
  Example predicates:
  - Age <= 20
  - Profession in {student, teacher}
  - 5000*Age + 3*Salary – 10000 > 0

Internal Nodes: Splitting Predicates

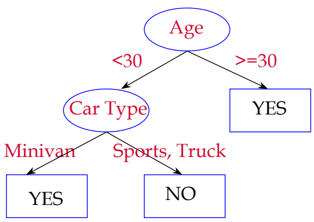
- Binary Univariate splits:
  - Numerical or ordered X: X <= c, c in dom(X)
  - Categorical X: X in A, A subset dom(X)
- Binary Multivariate splits:
  - Linear combination split on numerical variables:
    $\Sigma\, a_i X_i <= c$
- k-ary (k>2) splits analogous

Leaf Nodes

Consider leaf node t

- Classification problem: Node t is labeled with one class label c in dom(C)
- Regression problem: Two choices
  - Piecewise constant model:
    t is labeled with a constant y in dom(Y).
  - Piecewise linear model:
    t is labeled with a linear model
    $Y = y_t + \Sigma\, a_i X_i$

Example



Encoded classifier:

If (age<30 and
carType=Minivan)
Then YES

If (age <30 and
(carType=Sports or
carType=Truck))
Then NO

If (age >= 30)
Then NO

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Evaluation of Misclassification Error

Problem:

- In order to quantify the quality of a classifier d, we need to know its misclassification rate RT(d,P).
- But unless we know P, RT(d,P) is unknown.
- Thus we need to estimate RT(d,P) as good as possible.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Resubstitution Estimate

The *Resubstitution estimate* R(d,D) estimates RT(d,P) of a classifier d using D:

- Let D be the training database with N records.
- R(d,D) = 1/N Σ I(d(r.X) != r.C))
- Intuition: R(d,D) is the proportion of training records that is misclassified by d
- Problem with resubstitution estimate: Overly optimistic; classifiers that overfit the training dataset will have very low resubstitution error.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Test Sample Estimate

- Divide D into D_1 and D_2
- Use D_1 to construct the classifier d
- Then use resubstitution estimate $R(d,D_2)$ to calculate the estimated misclassification error of d
- Unbiased and efficient, but removes D_2 from training dataset D

V-fold Cross Validation

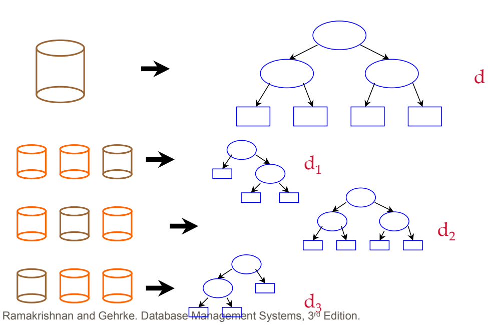
Procedure:

- Construct classifier d from D
- Partition D into V datasets $D_1, \ldots, D_V$
- Construct classifier d_i using $D \setminus D_i$
- Calculate the estimated misclassification error $R(d_i,D_i)$ of d_i using test sample D_i

Final misclassification estimate:

- Weighted combination of individual misclassification errors: $R(d,D) = 1/V\ \Sigma\ R(d_i,D_i)$

Cross-Validation: Example



Cross-Validation

- Misclassification estimate obtained through cross-validation is usually nearly unbiased
- Costly computation (we need to compute d, and $d_1, \ldots, d_V$); computation of d_i is nearly as expensive as computation of d
- Preferred method to estimate quality of learning algorithms in the machine learning literature

Decision Tree Construction

- Top-down tree construction schema:
  - Examine training database and find best splitting predicate for the root node
  - Partition training database
  - Recurse on each child node

Top-Down Tree Construction

BuildTree(Node *t*, Training database *D*,
Split Selection Method ***S***)

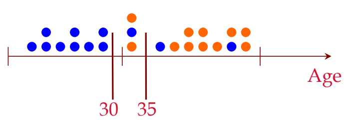
(1) Apply ***S*** to *D* to find splitting criterion
(2) **if** (*t* is not a leaf node)
(3) Create children nodes of *t*
(4) Partition *D* into children partitions
(5) Recurse on each partition
(6) **endif**

Decision Tree Construction

- Three algorithmic components:
  - Split selection (CART, C4.5, QUEST, CHAID, CRUISE, …)
  - Pruning (direct stopping rule, test dataset pruning, cost-complexity pruning, statistical tests, bootstrapping)
  - Data access (CLOUDS, SLIQ, SPRINT, RainForest, BOAT, UnPivot operator)

Split Selection Method

- Numerical or ordered attributes: Find a split point that separates the (two) classes



(Yes: ● No: ●)

Split Selection Method (Contd.)

- Categorical attributes: How to group?

Sport: Truck: Minivan:

(Sport, Truck) -- (Minivan)

(Sport) --- (Truck, Minivan)

(Sport, Minivan) --- (Truck)

Pruning Method

- For a tree T, the misclassification rate R(T,P) and the mean-squared error rate R(T,P) depend on P, but not on D.
- The goal is to do well on records randomly drawn from P, not to do well on the records in D
- If the tree is too large, it overfits D and does not model P. The pruning method selects the tree of the right size.

Data Access Method

- Recent development: Very large training databases, both in-memory and on secondary storage
- Goal: Fast, efficient, and scalable decision tree construction, using the complete training database.

Split Selection Methods

- Multitude of split selection methods in the literature
- In this workshop:
  - CART

Split Selection Methods: CART

- Classification And Regression Trees (Breiman, Friedman, Ohlson, Stone, 1984; considered "the" reference on decision tree construction)
- Commercial version sold by Salford Systems (www.salford-systems.com)
- Many other, slightly modified implementations exist (e.g., IBM Intelligent Miner implements the CART split selection method)

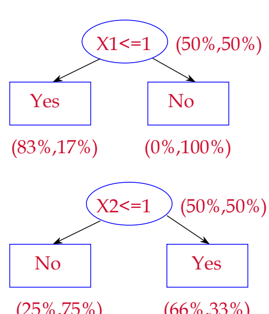
CART Split Selection Method

Motivation: We need a way to choose quantitatively between different splitting predicates

- Idea: Quantify the *impurity* of a node
- Method: Select splitting predicate that generates children nodes with minimum impurity from a space of possible splitting predicates

Intuition: Impurity Function

X1	X2	Class
1	1	Yes
1	2	Yes
1	2	Yes
1	2	Yes
1	2	Yes
1	1	No
2	1	No
2	1	No
2	2	No
2	2	No



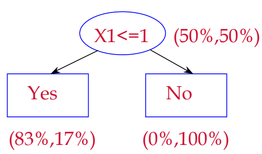
Impurity Function

- Let p(j|t) be the proportion of class j training records at node t
- Node impurity measure at node t:

  i(t) = phi(p(1|t), …, p(J|t))
- phi is symmetric
- Maximum value at arguments (J^{-1}, …, J^{-1}) (maximum impurity)
- phi(1,0,…,0) = … =phi(0,…,0,1) = 0
  (node has records of only one class; "pure" node)

Example

- Root node t:
  p(1|t)=0.5; p(2|t)=0.5
  Left child node t:
  P(1|t)=0.83; p(2|t)=-.17
- Impurity of root node:
  phi(0.5,0.5)
- Impurity of left child node:
  phi(0.83,0.17)
- Impurity of right child node:
  phi(0.0,1.0)



Goodness of a Split

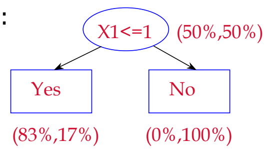
Consider node t with impurity phi(t)

The *reduction in impurity* through splitting predicate s (t splits into children nodes t_L with impurity phi(t_L) and t_R with impurity phi(t_R)) is:

$$\Delta_{phi}(s,t) = \text{phi}(t) - p_L\,\text{phi}(t_L) - p_R\,\text{phi}(t_R)$$

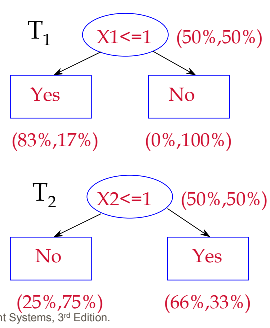
Example (Contd.)

- Impurity of root node:
  phi(0.5,0.5)
- Impurity of whole tree:
  0.6* phi(0.83,0.17)
  + 0.4 * phi(0,1)
- Impurity reduction:
  phi(0.5,0.5)
  - 0.6* phi(0.83,0.17)
  - 0.4 * phi(0,1)



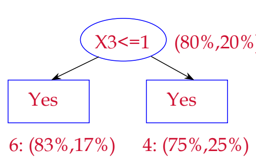
Error Reduction as Impurity Function

- Possible impurity function:
  Resubstitution error R(T,D).
- Example:
  R(no tree, D) = 0.5
  $R(T_1,D) = 0.6*0.17$
  $R(T_2,D) =$
  $0.4*0.25 + 0.6*0.33$



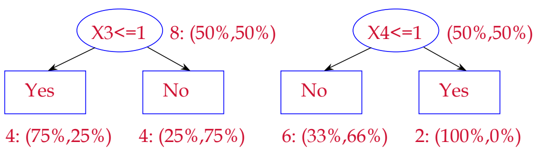
Problems with Resubstitution Error

- Obvious problem:
  There are situations where no split can decrease impurity
- Example:
  R(no tree, D) = 0.2
  $R(T_1,D)$
  =0.6*0.17+0.4*0.25
  =0.2



Problems with Resubstitution Error

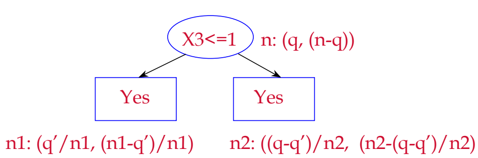
- More subtle problem:



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Problems with Resubstitution Error

Root node: n records, q of class 1
Left child node: n1 records, q' of class 1
Right child node: n2 records, (q-q') of class 1,
n1+n2 = n



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Problems with Resubstitution Error

Tree structure:
- Root node: n records (q/n, (n-q))
- Left child: n1 records (q'/n1, (n1-q')/n1)
- Right child: n2 records ((q-q')/n2, (n2-q')/n2)

Impurity before split:
- Error: q/n

Impurity after split:
- Left child: n1/n * q'/n1 = q'/n
- Right child: n2/n * (q-q')/n2 = (q-q')/n
- Total error: q'/n + (q-q')/n = q/n

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

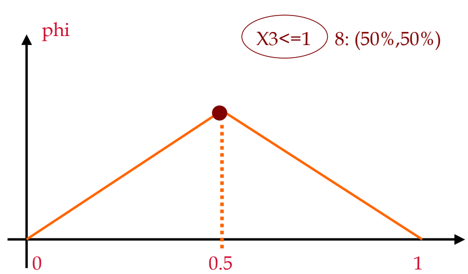
Problems with Resubstitution Error

Heart of the problem:

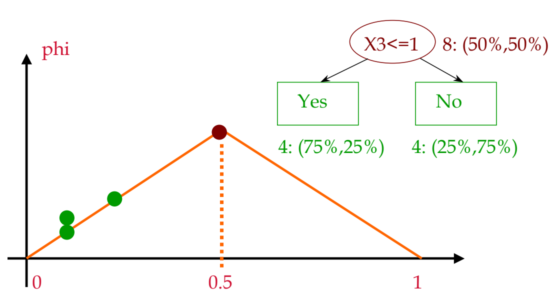
Assume two classes:
phi(p(1|t), p(2|t)) = phi(p(1|t), 1-p(1|t))
= phi (p(1|t))

Resubstitution errror has the following property:
phi(p1 + p2) = phi(p1)+phi(p2)

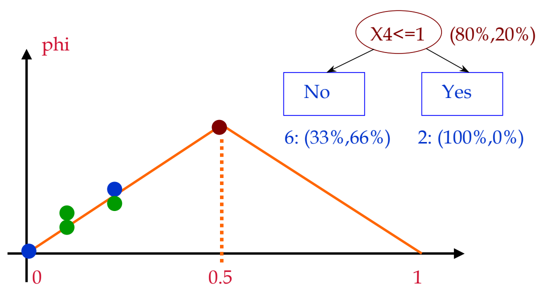
Example: Only Root Node



Example: Split (75,25), (25,75)



Example: Split (33,66), (100,0)



Remedy: Concavity

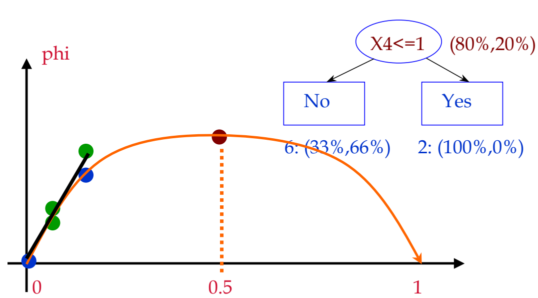
Use impurity functions that are concave:

phi″ < 0

Example impurity functions

- Entropy:
  phi(t) = - Σ p(j|t) log(p(j|t))
- Gini index:
  phi(t) = Σ p(j|t)^2

Example Split With Concave Phi



Nonnegative Decrease in Impurity

Theorem: Let phi(p_1, …, p_J) be a strictly concave function on j=1, …, J, Σ_j p_j = 1.

Then for any split s:

$$\Delta_{phi}(s,t) >= 0$$

With equality if and only if:

$$p(j|t_L) = p(j|t_R) = p(j|t), j = 1, …, J$$

Note: Entropy and gini-index are concave.

CART Univariate Split Selection

- Use gini-index as impurity function
- For each numerical or ordered attribute X, consider all binary splits s of the form
  X <= x
  where x in dom(X)
- For each categorical attribute X, consider all binary splits s of the form
  X in A, where A subset dom(X)
- At a node t, select split s* such that Δ_{phi}(s*,t) is maximal over all s considered

CART: Shortcut for Categorical Splits

Computational shortcut if |Y|=2.

- Theorem: Let X be a categorical attribute with dom(X) = {b_1, …, b_k}, |Y|=2, phi be a concave function, and let
  $p(X=b_1) <= … <= p(X=b_k)$.
  Then the best split is of the form:
  X in {b_1, b_2, …, b_l} for some l < k
- Benefit: We need only to check k-1 subsets of dom(X) instead of $2^{(k-1)}$-1 subsets

CART Multivariate Split Selection

- For numerical predictor variables, examine splitting predicates s of the form:
  $\Sigma_i a_i X_i <= c$
  with the constraint:
  $\Sigma_i a_i^2 = 1$
- Select splitting predicate s* with maximum decrease in impurity.

Problems with CART Split Selection

- Biased towards variables with more splits (M-category variable has 2^{M-1}-1) possible splits, an M-valued ordered variable has (M-1) possible splits
- Computationally expensive for categorical variables with large domains

Pruning Methods

- Test dataset pruning
- Direct stopping rule
- Cost-complexity pruning
- MDL pruning
- Pruning by randomization testing

Top-Down and Bottom-Up Pruning

Two classes of methods:

- Top-down pruning: Stop growth of the tree at the right size. Need a statistic that indicates when to stop growing a subtree.
- Bottom-up pruning: Grow an overly large tree and then chop off subtrees that "overfit" the training data.

Stopping Policies

A stopping policy indicates when further growth of the tree at a node t is counterproductive.

- All records are of the same class
- The attribute values of all records are identical
- All records have missing values
- At most one class has a number of records larger than a user-specified number
- All records go to the same child node if t is split (only possible with some split selection methods)

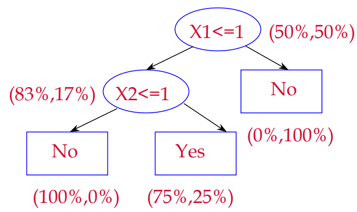
Test Dataset Pruning

- Use an independent test sample D' to estimate the misclassification cost using the resubstitution estimate R(T,D') at each node
- Select the subtree T' of T with the smallest expected cost

Test Dataset Pruning Example

Test set:

X1	X2	Class
1	1	Yes
1	2	Yes
1	2	Yes
1	2	Yes
1	1	Yes
1	2	No
2	1	No
2	1	No
2	2	No
2	2	No



Only root: 10% misclassification
Full tree: 30% misclassification

Cost Complexity Pruning

(Breiman, Friedman, Olshen, Stone, 1984)

Some more tree notation

- t: node in tree T
- leaf(T): set of leaf nodes of T
- |leaf(T)|: number of leaf nodes of T
- T_t: subtree of T rooted at t
- {t}: subtree of T_t containing only node t

Notation: Example

leaf(T) = {t1,t2,t3}

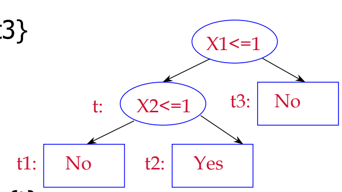
|leaf(T)|=3

Tree rooted at node t: T_t

Tree consisting of only node t: {t}

leaf(T_t)={t1,t2}

leaf({t})={t}



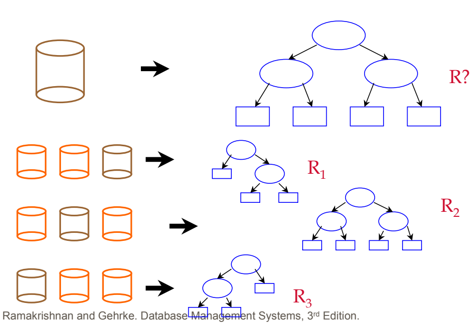
Cost-Complexity Pruning

- Test dataset pruning is the ideal case, if we have a large test dataset. But:
  - We might not have a large test dataset
  - We want to use all available records for tree construction
- If we do not have a test dataset, we do not obtain "honest" classification error estimates
- Remember cross-validation: Re-use training dataset in a clever way to estimate the classification error.

Cost-Complexity Pruning

1. /* cross-validation step */
   Construct tree T using D
2. Partition D into V subsets $D_1, \ldots, D_V$
3. for (i=1; i<=V; i++)
   Construct tree T_i from $(D \setminus D_i)$
   Use D_i to calculate the estimate $R(T_i, D \setminus D_i)$
   endfor
4. /* estimation step */
   Calculate R(T,D) from $R(T_i, D \setminus D_i)$

Cross-Validation Step

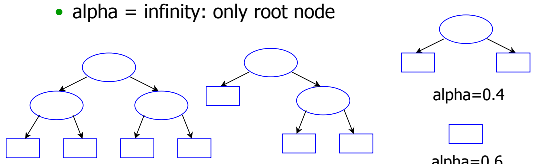


Cost-Complexity Pruning

- Problem: How can we relate the misclassification error of the CV-trees to the misclassification error of the large tree?
- Idea: Use a parameter that has the same meaning over different trees, and relate trees with similar parameter settings.
- Such a parameter is the cost-complexity of the tree.

Cost-Complexity Pruning

- Cost complexity of a tree T:
  $R_{alpha}(T) = R(T) + alpha\ |leaf(T)|$
- For each A, there is a tree that minimizes the cost complexity:
  - alpha = 0: full tree
  - alpha = infinity: only root node



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

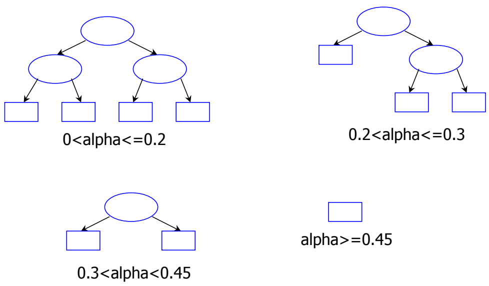
Cost-Complexity Pruning

- When should we prune the subtree rooted at t?
  - $R_{alpha}(\{t\}) = R(t) + alpha$
  - $R_{alpha}(T_t) = R(T_t) + alpha\ |leaf(T_t)|$
  - Define

    $$g(t) = (R(t)-R(T_t)) / (|leaf(T_t)|-1)$$
- Each node has a critical value g(t):
  - Alpha < g(t): leave subtree T_t rooted at t
  - Alpha >= g(t): prune subtree rooted at t to {t}
- For each alpha we obtain a unique minimum cost-complexity tree.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Example Revisited



Cost Complexity Pruning

1. Let $T^1 > T^2 > \ldots > \{t\}$ be the nested cost-complexity sequence of subtrees of T rooted at t.
   Let $alpha_1 < \ldots < alpha_k$ be the sequence of associated critical values of alpha. Define $alpha_{k'}$=squareroot($alpha_k$ * $alpha_{k+1}$)
2. Let T_i be the tree grown from $D \setminus D_i$
3. Let $T^i(alpha_{k'})$ be the minimal cost-complexity tree for $alpha_{k'}$

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Cost Complexity Pruning

4. Let $R'(T_i)(alpha_{k'}))$ be the misclassification cost of $T_i(alpha_{k'})$ based on D_i
5. Define the V-fold cross-validation misclassification estimate as follows: $R^*(T^k) = 1/V\ \Sigma_i\ R'(T_i(alpha_{k'}))$
6. Select the subtree with the smallest estimated CV error

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

k-SE Rule

- Let T* be the subtree of T that minimizes the misclassification error $R(T_k)$ over all k
- But $R(T_k)$ is only an estimate:
  - Estimate the estimated standard error SE(R(T*)) of R(T*)
  - Let T** be the smallest tree such that R(T**) <= R(T*) + k*SE(R(T*)); use T** instead of T*
  - Intuition: A smaller tree is easier to understand.

Cost Complexity Pruning

Advantages:
- No independent test dataset necessary
- Gives estimate of misclassification error, and chooses tree that minimizes this error

Disadvantages:
- Originally devised for small datasets; is it still necessary for large datasets?
- Computationally very expensive for large datasets (need to grow V trees from nearly all the data)

Missing Values

- What is the problem?
  - During computation of the splitting predicate, we can selectively ignore records with missing values (note that this has some problems)
  - But if a record r misses the value of the variable in the splitting attribute, r can not participate further in tree construction

Algorithms for missing values address this problem.

Mean and Mode Imputation

Assume record r has missing value r.X, and splitting variable is X.

- Simplest algorithm:
  - If X is numerical (categorical), impute the overall mean (mode)
- Improved algorithm:
  - If X is numerical (categorical), impute the mean(X|t.C) (the mode(X|t.C))

Decision Trees: Summary

- Many application of decision trees
- There are many algorithms available for:
  - Split selection
  - Pruning
  - Handling Missing Values
  - Data Access
- Decision tree construction still active research area (after 20+ years!)
- Challenges: Performance, scalability, evolving datasets, new applications

Lecture Overview

- Data Mining I: Decision Trees
- Data Mining II: Clustering
- Data Mining III: Association Analysis

Supervised Learning

- F(x): true function (usually not known)
- D: training sample drawn from F(x)

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	0

Supervised Learning

- F(x): true function (usually not known)
- D: training sample (x,F(x))

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1

- G(x): model learned from D

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	?

- Goal: $E[(F(x)-G(x))^2]$ is small (near zero) for future samples

Supervised Learning

Well-defined goal:

Learn G(x) that is a good approximation to F(x) from training sample D

Well-defined error metrics:

Accuracy, RMSE, ROC, …

Supervised Learning

Training dataset:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	1
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	1
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	1
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	1
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	0

Test dataset:

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 ?

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Un-Supervised Learning

Training dataset:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	~~0~~
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	~~0~~
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	~~1~~
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	~~1~~
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	~~0~~
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	~~0~~

Test dataset:

71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 ?

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Un-Supervised Learning

Training dataset:

57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0	~~0~~
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0	~~0~~
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0	~~1~~
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0	~~1~~
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0	~~0~~
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~0~~
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0	~~1~~
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1	~~0~~

Test dataset:

~~71,M,160,1,130,105,38,20,1,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0~~ ?

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Un-Supervised Learning

Data Set:

```
57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0
78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0
69,F,180,0,115,85,40,22,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0
18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
54,F,135,0,115,95,39,35,1,1,0,0,0,1,0,0,0,1,0,0,0,0,1,0,0,0,1,0,0,0,0
84,F,210,1,135,105,39,24,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0
89,F,135,0,120,95,36,28,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0,0,1,0,0
49,M,195,0,115,85,39,32,0,0,0,1,1,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0,0
40,M,205,0,115,90,37,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
74,M,250,1,130,100,38,26,1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0
77,F,140,0,125,100,40,30,1,1,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,1,1
```

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Supervised vs. Unsupervised Learning

Supervised	Unsupervised
• $y=F(x)$: true function • D: labeled training set • D: $\{x_i,F(x_i)\}$ • Learn: G(x): model trained to predict labels D • Goal: $E[(F(x)-G(x))^2] \approx 0$ • Well defined criteria: Accuracy, RMSE, ...	• Generator: true model • D: unlabeled data sample • D: $\{x_i\}$ • Learn ?????????? • Goal: ?????????? • Well defined criteria: ??????????

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

What to Learn/Discover?

- Statistical Summaries
- Generators
- Density Estimation
- Patterns/Rules
- Associations (see previous segment)
- Clusters/Groups (this segment)
- Exceptions/Outliers
- Changes in Patterns Over Time or Location

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Clustering: Unsupervised Learning

- Given:
  - Data Set D (training set)
  - Similarity/distance metric/information
- Find:
  - Partitioning of data
  - Groups of similar/close items

Similarity?

- Groups of similar customers
  - Similar demographics
  - Similar buying behavior
  - Similar health
- Similar products
  - Similar cost
  - Similar function
  - Similar store
  - …
- Similarity usually is domain/problem specific

Distance Between Records

- *d*-dim vector space representation and distance metric

r_1: 57,M,195,0,125,95,39,25,0,1,0,0,0,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0

r_2: 78,M,160,1,130,100,37,40,1,0,0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0

…

r_N: 18,M,165,0,110,80,41,30,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

Distance (r_1,r_2) = ???

- Pairwise distances between points (no *d*-dim space)
  - Similarity/dissimilarity matrix (upper or lower diagonal)
  - Distance: 0 = near, ∞ = far
  - Similarity: 0 = far, ∞ = near

--	1	2	3	4	5	6	7	8	9	10
1	-	d	d	d	d	d	d	d	d	d
2		-	d	d	d	d	d	d	d	d
3			-	d	d	d	d	d	d	d
4				-	d	d	d	d	d	d
5					-	d	d	d	d	d
6						-	d	d	d	d
7							-	d	d	d
8								-	d	d
9									-	d

Properties of Distances: Metric Spaces

- A metric space is a set *S* with a global distance function *d*. For every two points *x, y* in *S*, the distance *d(x,y)* is a nonnegative real number.
- A metric space must also satisfy
  - d(x,y) = 0 iff *x = y*
  - d(x,y) = d(y,x) (symmetry)
  - d(x,y) + d(y,z) >= d(x,z) (triangle inequality)

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Minkowski Distance (L_p Norm)

- Consider two records $x=(x_1,\ldots,x_d)$, $y=(y_1,\ldots,y_d)$:

$$d(x,y)=\sqrt[p]{|x_1-y_1|^p+|x_2-y_2|^p+\ldots+|x_d-y_d|^p}$$

Special cases:

- p=1: Manhattan distance

$$d(x,y)=|x_1-y_1|+|x_2-y_2|+\ldots+|x_p-y_p|$$

- p=2: Euclidean distance

$$d(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2+\ldots+(x_d-y_d)^2}$$

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Only Binary Variables

2x2 Table:

	0	1	Sum
0	a	b	a+b
1	c	d	c+d
Sum	a+c	b+d	a+b+c+d

- Simple matching coefficient: (symmetric) $$d(x,y)=\frac{b+c}{a+b+c+d}$$
- Jaccard coefficient: (asymmetric) $$d(x,y)=\frac{b+c}{b+c+d}$$

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Nominal and Ordinal Variables

- Nominal: Count number of matching variables
  - m: # of matches, d: total # of variables

$$d(x,y) = \frac{d-m}{d}$$

- Ordinal: Bucketize and transform to numerical:
  - Consider record x with value x_i for i^{th} attribute of record x; new value x_i':

$$x_i' = \frac{x_i - 1}{dom(X_i) - 1}$$

Mixtures of Variables

- Weigh each variable differently
- Can take "importance" of variable into account (although usually hard to quantify in practice)

Clustering: Informal Problem Definition

Input:

- A data set of *N* records each given as a *d*-dimensional data feature vector.

Output:

- Determine a natural, useful "partitioning" of the data set into a number of (k) clusters and noise such that we have:
  - High similarity of records within each cluster (intra-cluster similarity)
  - Low similarity of records between clusters (inter-cluster similarity)

Types of Clustering

- Hard Clustering:
  - Each object is in one and only one cluster
- Soft Clustering:
  - Each object has a probability of being in each cluster

Clustering Algorithms

- Partitioning-based clustering
  - K-means clustering
  - K-medoids clustering
  - EM (expectation maximization) clustering
- Hierarchical clustering
  - Divisive clustering (top down)
  - Agglomerative clustering (bottom up)
- Density-Based Methods
  - Regions of dense points separated by sparser regions of relatively low density

K-Means Clustering Algorithm

Initialize k cluster centers
Do
  Assignment step: Assign each data point to its closest cluster center
  Re-estimation step: Re-compute cluster centers
While (there are still changes in the cluster centers)

Visualization at:

- http://www.delft-cluster.nl/textminer/theory/kmeans/kmeans.html

Issues

Why is K-Means working:

- How does it find the cluster centers?
- Does it find an optimal clustering
- What are good starting points for the algorithm?
- What is the right number of cluster centers?
- How do we know it will terminate?

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

K-Means: Distortion

- Communication between sender and receiver
- Sender encodes dataset: $x_i \rightarrow \{1,\ldots,k\}$
- Receiver decodes dataset: $j \rightarrow center_j$
- Distortion: $D = \sum_{1}^{N} \left(x_i - center_{encode(x_i)}\right)^2$
- A good clustering has minimal distortion.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Properties of the Minimal Distortion

- Recall: Distortion $D = \sum_{1}^{N} \left(x_i - center_{encode(x_i)}\right)^2$
- Property 1: Each data point x_i is encoded by its nearest cluster center $center_j$. (Why?)
- Property 2: When the algorithm stops, the partial derivative of the Distortion with respect to each center attribute is zero.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Property 2 Followed Through

- Calculating the partial derivative:

$$D=\sum_{1}^{N}\left(x_i - center_{encode(x_i)}\right)^2=\sum_{j=1}^{k}\sum_{i\in Cluster(center_j)}\left(x_i - center_j\right)^2$$

$$\frac{\partial D}{\partial center_j}=\frac{\partial}{\partial center_j}\sum_{i\in Cluster(c_j)}\left(x_i - center_j\right)^2=-2\sum_{i\in Cluster(c_j)}\left(x_i - center_j\right)\overset{!}{=}0$$

- Thus at the minimum:

$$center_j=\frac{1}{|\{i\in Cluster(center_j)\}|}\sum_{i\in Cluster(center_j)}x_i$$

K-Means Minimal Distortion Property

- Property 1: Each data point x_i is encoded by its nearest cluster center $center_j$
- Property 2: Each center is the centroid of its cluster.
- How do we improve a configuration:
  - Change encoding (encode a point by its nearest cluster center)
  - Change the cluster center (make each center the centroid of its cluster)

K-Means Minimal Distortion Property (Contd.)

- Termination? Count the number of distinct configurations …
- Optimality? We might get stuck in a local optimum.
  - Try different starting configurations.
  - Choose the starting centers smart.
- Choosing the number of centers?
  - Hard problem. Usually choose number of clusters that minimizes some criterion.

K-Means: Summary

- Advantages:
  - Good for exploratory data analysis
  - Works well for low-dimensional data
  - Reasonably scalable
- Disadvantages
  - Hard to choose k
  - Often clusters are non-spherical

K-Medoids

- Similar to K-Means, but for categorical data or data in a non-vector space.
- Since we cannot compute the cluster center (think text data), we take the "most representative" data point in the cluster.
- This data point is called the medoid (the object that "lies in the center").

Agglomerative Clustering

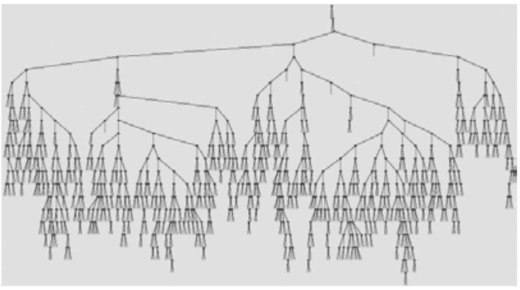
Algorithm:

- Put each item in its own cluster (all singletons)
- Find all pairwise distances between clusters
- Merge the two *closest* clusters
- Repeat until everything is in one cluster

Observations:

- Results in a hierarchical clustering
- Yields a clustering for each possible number of clusters
- Greedy clustering: Result is not "optimal" for any cluster size

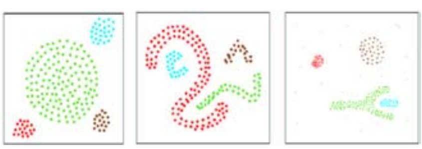
Agglomerative Clustering Example



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Density-Based Clustering

- A cluster is defined as a connected dense component.
- Density is defined in terms of number of neighbors of a point.
- We can find clusters of arbitrary shape



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

DBSCAN

E-neighborhood of a point

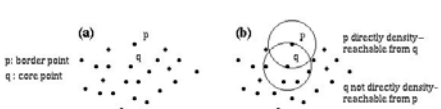
- NE(p) = {q ∈D | dist(p,q) ≤ E}

Core point

- |NE(q)| ≥ MinPts

Directly density-reachable

- A point p is *directly* density-reachable from a point q wrt. E, MinPts if
  1) p ∈ NE(q) and
  2) |NE(q)| ≥ MinPts (core point condition).



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

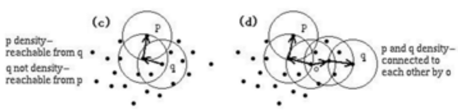
DBSCAN

<u>Density-reachable</u>

- A point p is density-reachable from a point q wrt. E and MinPts if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

<u>Density-connected</u>

- A point p is density-connected to a point q wrt. E and MinPts if there is a point o such that both, p and q are density-reachable from o wrt. E and MinPts.



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

DBSCAN

<u>Cluster</u>

- A cluster C satisfies:
  1) $\forall$ p, q: if p $\in$ C and q is density-reachable from p wrt. E and MinPts, then q $\in$ C. (Maximality)
  2) $\forall$ p, q $\in$ C: p is density-connected to q wrt. E and MinPts. (Connectivity)

<u>Noise</u>

Those points not belonging to any cluster

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

DBSCAN

Can show

(1) Every density-reachable set is a cluster:
The set
O = {o | o is density-reachable from p wrt. Eps and MinPts}
is a cluster wrt. Eps and MinPts.

(2) Every cluster is a density-reachable set:
Let C be a cluster wrt. Eps and MinPts and let p be any point in C with $|N_{Eps}(p)| \geq$ MinPts. Then C equals to the set
O = {o | o is density-reachable from p wrt. Eps and MinPts}.

This motivates the following algorithm:

- For each point, DBSCAN determines the Eps-environment and checks whether it contains more than MinPts data points
- If so, it labels it with a cluster number
- If a neighbor *q* of a point *p* has already a cluster number, associate this number with *p*

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

DBSCAN

Arbitrary shape clusters found by DBSCAN

DBSCAN: Summary

- Advantages:
  - Finds clusters of arbitrary shapes
- Disadvantages:
  - Targets low dimensional spatial data
  - Hard to visualize for >2-dimensional data
  - Needs clever index to be scalable
  - How do we set the magic parameters?

Lecture Overview

- Data Mining I: Decision Trees
- Data Mining II: Clustering
- Data Mining III: Association Analysis

Market Basket Analysis

- Consider shopping cart filled with several items
- Market basket analysis tries to answer the following questions:
  - Who makes purchases?
  - What do customers buy together?
  - In what order do customers purchase items?

Market Basket Analysis

Given:

- A database of customer transactions
- Each transaction is a set of items
- Example: Transaction with TID 111 contains items {Pen, Ink, Milk, Juice}

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Market Basket Analysis (Contd.)

- Coocurrences
  - 80% of all customers purchase items X, Y and Z together.
- Association rules
  - 60% of all customers who purchase X and Y also buy Z.
- Sequential patterns
  - 60% of customers who first buy X also purchase Y within three weeks.

Confidence and Support

We prune the set of all possible association rules using two interestingness measures:

- Confidence of a rule:
  - X → Y has confidence c if P(Y|X) = c
- Support of a rule:
  - X → Y has support s if P(XY) = s

We can also define

- Support of an itemset (a coocurrence) XY:
  - XY has support s if P(XY) = s

Example

Examples:

- {Pen} => {Milk}
  Support: 75%
  Confidence: 75%
- {Ink} => {Pen}
  Support: 100%
  Confidence: 100%

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Example

- Find all itemsets with support >= 75%?

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Example

- Can you find all association rules with support >= 50%?

TID	CID	Date	Item	Qty
111	201	5/1/99	Pen	2
111	201	5/1/99	Ink	1
111	201	5/1/99	Milk	3
111	201	5/1/99	Juice	6
112	105	6/3/99	Pen	1
112	105	6/3/99	Ink	1
112	105	6/3/99	Milk	1
113	106	6/5/99	Pen	1
113	106	6/5/99	Milk	1
114	201	7/1/99	Pen	2
114	201	7/1/99	Ink	2
114	201	7/1/99	Juice	4

Market Basket Analysis: Applications

- Sample Applications
  - Direct marketing
  - Fraud detection for medical insurance
  - Floor/shelf planning
  - Web site layout
  - Cross-selling

Applications of Frequent Itemsets

- Market Basket Analysis
- Association Rules
- Classification (especially: text, rare classes)
- Seeds for construction of Bayesian Networks
- Web log analysis
- Collaborative filtering

Association Rule Algorithms

- More abstract problem redux
- Breadth-first search
- Depth-first search

Problem Redux

Abstract:
- A set of items {1,2,…,k}
- A dabase of transactions (itemsets) D={T1, T2, …, Tn}, Tj subset {1,2,…,k}

GOAL:
Find all itemsets that appear in at least x transactions

("appear in" == "are subsets of")
I subset T: T supports I

For an itemset I, the number of transactions it appears in is called the support of I.
x is called the minimum support.

Concrete:
- I = {milk, bread, cheese, …}
- D = { {milk,bread,cheese}, {bread,cheese,juice}, …}

GOAL:
Find all itemsets that appear in at least 1000 transactions

{milk,bread,cheese} supports {milk,bread}

Problem Redux (Contd.)

Definitions:
- An itemset is frequent if it is a subset of at least x transactions. (FI.)
- An itemset is maximally frequent if it is frequent and it does not have a frequent superset. (MFI.)

GOAL: Given x, find all frequent (maximally frequent) itemsets (to be stored in the FI (MFI)).

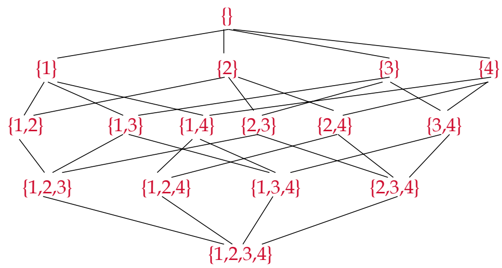
Obvious relationship:
MFI subset FI

Example:
D={ {1,2,3}, {1,2,3}, {1,2,3}, {1,2,4} }
Minimum support x = 3

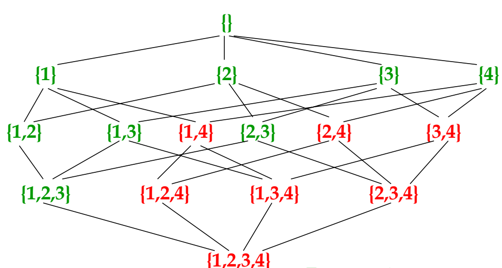
{1,2} is frequent
{1,2,3} is maximal frequent
Support({1,2}) = 4

All maximal frequent itemsets:
{1,2,3}

The Itemset Lattice



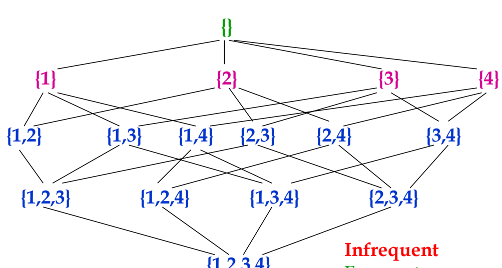
Frequent Itemsets



Frequent itemsets

Infrequent itemsets

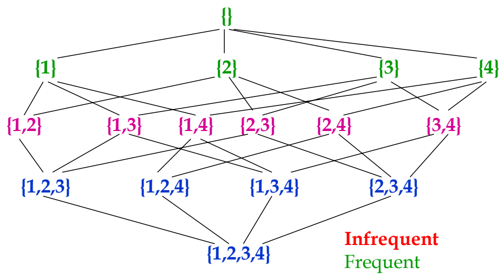
Breath First Search: 1-Itemsets



Infrequent
Frequent
Currently examined
Don't know

The Apriori Principle:
I infrequent ➔ (I union {x}) infrequent

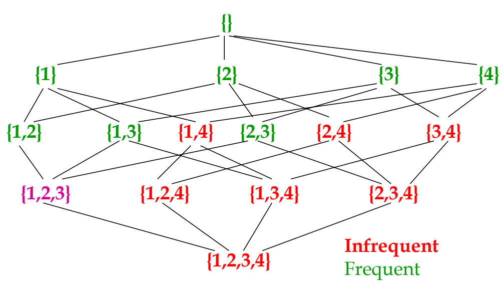
Breath First Search: 2-Itemsets



Infrequent
Frequent
Currently examined
Don't know

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Breath First Search: 3-Itemsets



Infrequent
Frequent
Currently examined
Don't know

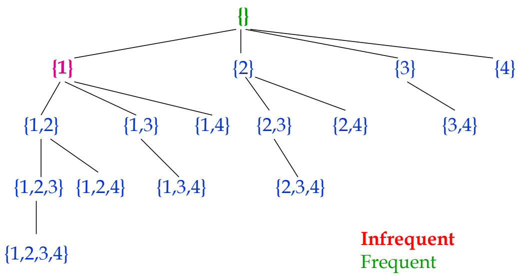
Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Breadth First Search: Remarks

- We prune infrequent itemsets and avoid to count them
- To find an itemset with k items, we need to count all 2^k subsets

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

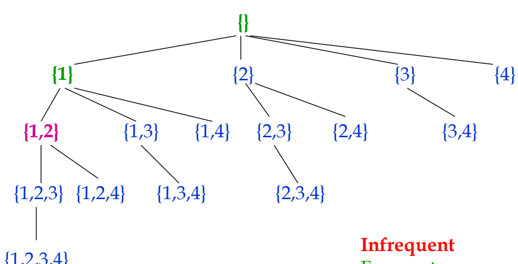
Depth First Search (1)



Infrequent
Frequent
Currently examined
Don't know

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

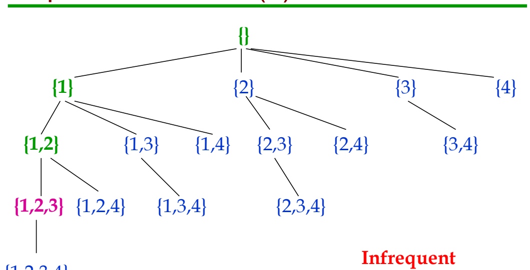
Depth First Search (2)



Infrequent
Frequent
Currently examined
Don't know

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

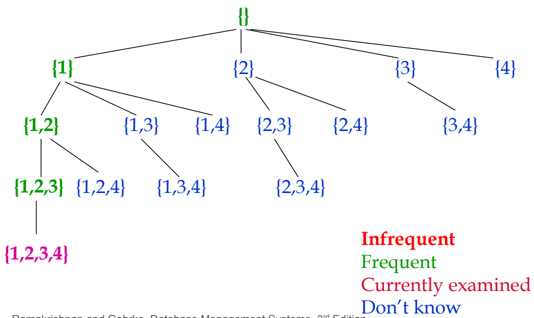
Depth First Search (3)



Infrequent
Frequent
Currently examined
Don't know

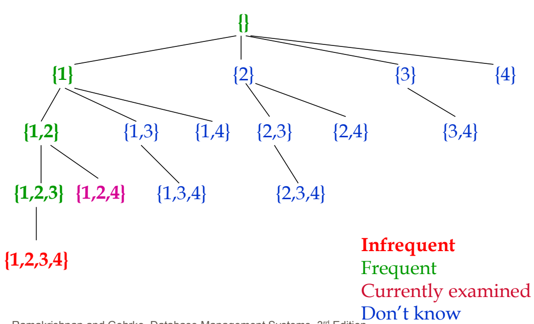
Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Depth First Search (4)



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Depth First Search (5)



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Depth First Search: Remarks

- We prune frequent itemsets and avoid counting them (works only for maximal frequent itemsets)
- To find an itemset with k items, we need to count k prefixes

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

BFS Versus DFS

Breadth First Search
- Prunes infrequent itemsets
- Uses anti-monotonicity: Every superset of an infrequent itemset is infrequent

Depth First Search
- Prunes frequent itemsets
- Uses monotonicity: Every subset of a frequent itemset is frequent

Extensions

- Imposing constraints
  - Only find rules involving the dairy department
  - Only find rules involving expensive products
  - Only find "expensive" rules
  - Only find rules with "whiskey" on the right hand side
  - Only find rules with "milk" on the left hand side
  - Hierarchies on the items
  - Calendars (every Sunday, every 1st of the month)

Itemset Constraints

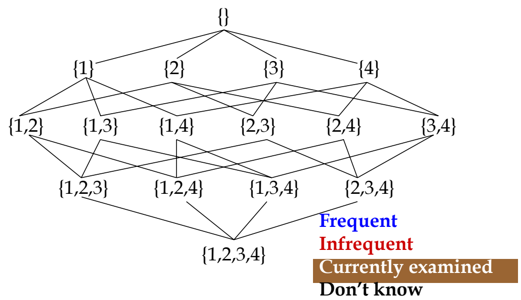
Definition:
- *A constraint* is an arbitrary property of itemsets.

Examples:
- The itemset has support greater than 1000.
- No element of the itemset costs more than $40.
- The items in the set average more than $20.

Goal:
- Find all itemsets satisfying a given constraint **P**.

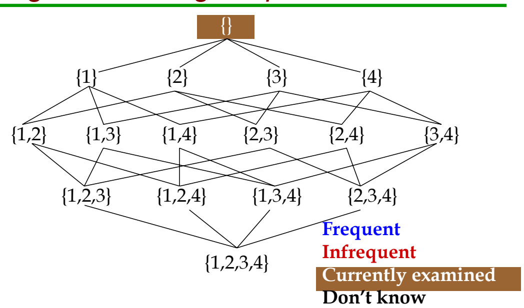
"Solution":
- If **P** is a support constraint, use the *Apriori* Algorithm.

Negative Pruning in *Apriori*



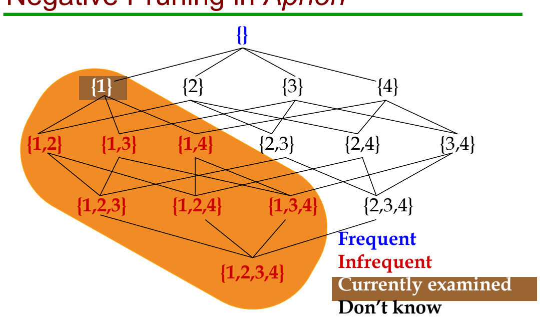
Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Negative Pruning in *Apriori*



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Negative Pruning in *Apriori*

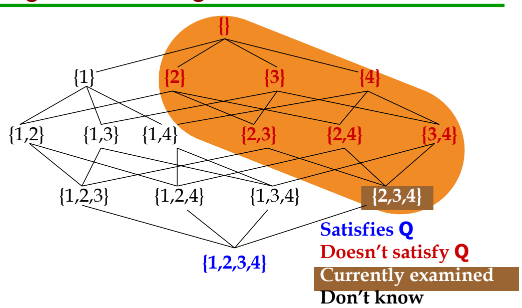


Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Two Trivial Observations

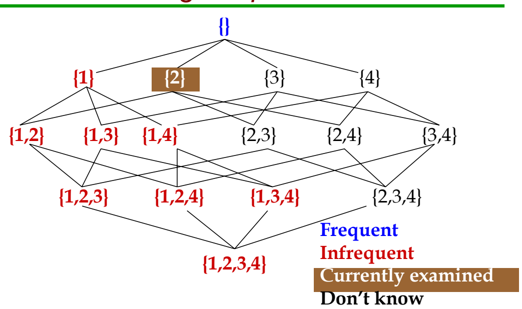
- *Apriori* can be applied to any constraint **P** that is antimonotone.
  - Start from the empty set.
  - Prune supersets of sets that do not satisfy **P**.
- Itemset lattice is a boolean algebra, so *Apriori* also applies to a monotone **Q**.
  - Start from set of all items instead of empty set.
  - Prune subsets of sets that do not satisfy **Q**.

Negative Pruning a Monotone **Q**



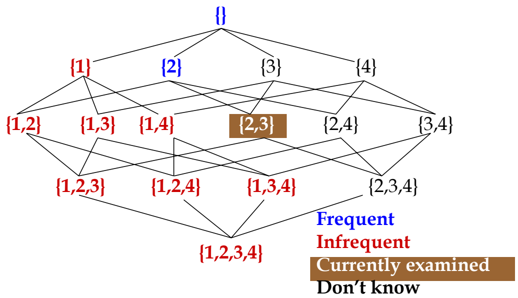
Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Positive Pruning in *Apriori*



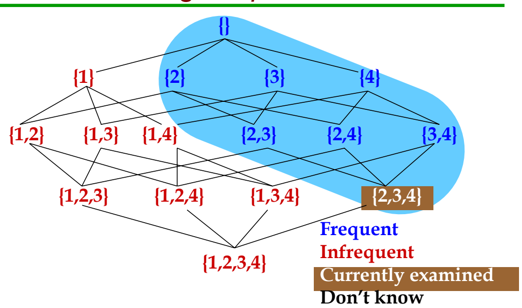
Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Positive Pruning in *Apriori*



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Positive Pruning in *Apriori*



Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

Classifying Constraints

Antimonotone:
- **support(I) > 1000**
- **max(I) < 100**

Monotone:
- **sum(I) > 3**
- **min(I) < 40**

Neither:
- **average(I) > 50**
- **variance(I) < 2**
- **3 < sum(I) < 50**

These are the constraints we really want.

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.

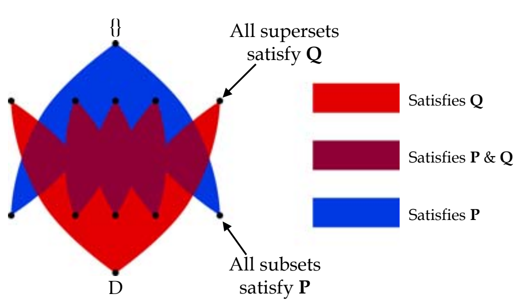
The Problem Redux

Current Techniques:
- Approximate the difficult constraints.
- Monotone approximations are common.

New Goal:
- Given constraints **P** and **Q**, with **P** antimonotone (support) and **Q** monotone (statistical constraint).
- Find all itemsets that satisfy both **P** and **Q**.

Recent solutions:
- Newer algorithms can handle both **P** and **Q**

Conceptual Illustration of Problem



Applications

- Spatial association rules
- Web mining
- Market basket analysis
- User/customer profiling

Extensions: Sequential Patterns

Customer ID (CID)	Transaction ID (TID)	Itemset
1	1	$\{a, b, d\}$
1	3	$\{c, d\}$
1	6	$\{b, c, d\}$
2	2	$\{b\}$
2	4	$\{a, b, c\}$
3	5	$\{a, b\}$
3	7	$\{b, c, d\}$

Customer ID (CID)	Sequence
1	$\langle\{a, b, d\}, \{c, d\}, \{b, c, d\}\rangle$
2	$\langle\{b\}, \{a, b, c\}\rangle$
3	$\langle\{a, b\}, \{b, c, d\}\rangle$

Ramakrishnan and Gehrke. Database Management Systems, 3rd Edition.