# 

# Index:

[**Index:**](#_6mdawj73agyp) **1**

[**Chapter 1**](#_4maiaajmp6h5) **7**

[What is Data Mining?](#_lbj190zgav8o) 7

[What is NOT Data Mining](#_trgqcdc518kn) 7

[Data Mining Tasks](#_hy3vtshp3x) 7

[Prediction Methods](#_i0fgoblbr3hl) 7

[Description Methods](#_r41m7zyukfw7) 7

[Classification](#_u6qs19kwmzbn) 8

[Definition](#_smsx6wtrw3kq) 8

[Goal of Classification](#_hogm5ocqodud) 8

[Applications of Classification](#_ze16ihjaeuq2) 8

[Clustering](#_7endt9z4y14y) 9

[Definition](#_l1a13trkag44) 9

[Clustering Applications](#_gt04gy5817f) 9

[Association Rule Discovery](#_3xnmochw0as9) 10

[Definition](#_8jv31c891hcn) 10

[Example](#_cx73wk1qmnam) 10

[Application of Association Rule Discovery](#_sy5zz3ehi4pv) 10

[Regression](#_3us7thyhzuyn) 11

[Definition](#_qc1r7dd8tr7w) 11

[Examples](#_kw3jbt59h3h1) 11

[Application of Regression](#_q02tp2eivxcc) 11

[Components of Data Mining Alogirthms](#_ea2eqz4uiwxj) 11

[**Chapter 2a**](#_r0iodw3hnhcv) **12**

[Definitions](#_9wx5gr3nne) 12

[Data](#_8dluc3vmkx3j) 12

[Attributes](#_5xwkj3d45yd3) 12

[Object](#_394s3xcbz3eg) 12

[Types of Attributes](#_jtpiudowwxq5) 12

[Nominal](#_lk9a0h8ueqk6) 12

[Ordinal](#_f2wemfe3n10y) 12

[Interval](#_4o5uh37lj1fh) 12

[Ratio](#_o4tlnb9qs57) 13

[Properties of Attribute Values](#_9pi3htrz47cl) 13

[Discrete and Continuous Attributes](#_rz079untnu5c) 14

[Discrete Attributes](#_i7ek5hw1nfwq) 14

[Continuous Attributes](#_862qj2f2carc) 14

[Data Quality](#_vzvle879ltvn) 14

[Noise](#_tudbcgjdyjo) 14

[Outliers](#_s7l7m850i2in) 14

[Missing Values](#_pdtwoh8qiqu0) 14

[Duplicate data](#_cge2hquu9krw) 14

[Data Preprocessing](#_htxtn7143fae) 15

[Aggregation](#_ihv93mv8xg3u) 15

[Sampling](#_13ld4ccwmlkk) 15

[Types of Sampling](#_61yg348tbuxi) 15

[How to have effective sampling](#_jvtuot7drse) 15

[Dimensionality reduction](#_7lo143ra0jpb) 16

[Curse of Dimensionality](#_vrx5pkq9qyyw) 16

[Why reduce the Dimensions?](#_2cy2jqs4uiih) 16

[Feature subset selection](#_62up06vmeza4) 16

[Feature creation](#_dov3ad5tmyt5) 16

[Attribute Transformation](#_ajbx7zuanaqu) 16

[**Chapter 2b**](#_bev7hw1c2mwp) **17**

[Similarity](#_jr6h8lilv094) 17

[Dissimilarity](#_4gn302cl325t) 17

[Proximity](#_kdf5wh3smskl) 17

[Similarity/Dissimilarity for Simple Attributes](#_h4xvo02o6lqp) 17

[Euclidean Distance](#_muve28e48n5) 18

[Minkowski Distance](#_k595pj4w6wi1) 19

[Common Properties of a distance](#_lzru46ndln3g) 19

[Similarity Between Binary Vectors](#_j542i8omzo99) 19

[Simple Matching Coefficient (SMC):](#_irmdsw2fcqz5) 20

[Jaccard Coefficient](#_y96okruq4yb4) 20

[Cosine Similarity](#_60lz7dcj5bzv) 20

[**Chapter 3**](#_2r8bbkhbslz1) **21**

[What is data exploration?](#_blcb3v1tboug) 22

[**Chapter 8**](#_89y7jnojgae7) **23**

[What is Cluster Analysis](#_rzthd5u5aeah) 23

[What is not Cluster Analysis?](#_379hvl5c90j) 23

[Notion of Clusters can be Ambiguous](#_4azh2korcwqr) 23

[Types of Clusterings](#_c5ccqxfs21bi) 24

[Partitional Clustering](#_r84yxll0o33k) 24

[Hierarchical Clustering](#_wte0oyo5g5py) 24

[Other Distinctions Between Sets of Clusters](#_ugna4636uus) 25

[Types of Clusters](#_e7cdyhrf6uh) 25

[Well-separated clusters](#_hw1vk0yx0pss) 25

[Center-based clusters](#_8l68iw6rrpts) 25

[Contiguous clusters](#_2au3n72mljd0) 25

[Density-based clusters](#_igzfc9scq3vx) 26

[Property or Conceptual Clusters](#_szgwvrenaltu) 26

[Based upon an Objective Function](#_xxvfdv4vk3pl) 27

[Characteristics of the Input Data](#_3ywms7mtdcej) 27

[Clustering Algorithms](#_mixqa0284ill) 27

[K-means Clustering and its variants](#_qz4q62ka70u0) 28

[Convergence and Complexity](#_u8pcwffuz6ie) 28

[Evaluating K-means Clusters](#_sg735lc7eej3) 28

[Solutions to Initial Centroids Problem](#_rj7ma92o0smb) 29

[Handling Empty Clusters](#_oili2ugkklug) 29

[Updating Centers Incrementally](#_2q4rdlunok0o) 29

[Pre-processing](#_p374yxuko2ml) 29

[Post-processing](#_ufe0ll16tn1x) 29

[Bisecting K-means](#_7780yjcmqdf5) 29

[Limitationts of K-means](#_y0chlzeo48p6) 30

[Hierarchical clustering](#_xoyo2iftpykh) 32

[Agglomerative](#_4q4gyjwmc88p) 32

[Time and Space Requirements](#_2jkubnogjesf) 35

[Problems and Limitations](#_slzxa775j958) 35

[Divisive Hierarchical Clustering (MST)](#_u4l4f6oi5ek) 35

[Density-based clustering (DBSCAN)](#_5spn4hkw0mn2) 36

[When DBSCAN Works Well](#_3czr5sojwwg3) 36

[When DBSCAN Does NOT Work Well](#_cgw72ziaq5l6) 37

[Determining EPS and MinPts](#_ba2dggcmn9y8) 37

[Cluster Validity](#_1faotxg7antc) 37

[Different Aspects of Cluster Validation](#_e2yhla17tps6) 38

[Measures of Cluster Validity](#_k2v74ty0s7dh) 38

[Cluster Validity Via Correlation](#_jelu1vtsqcty) 38

[Cluster Validation via Similarity Matrix](#_9q8sdf8wvcyl) 39

[Internal Measures: Sum of Squared Errors](#_nmdliiur2p7) 40

[Framewoork for Cluster Validity](#_84s9wn7o907g) 41

[Statistical Framework for SSE](#_nl90b3hind77) 41

[Statistical Framework for Correlation](#_35niwnaa5pte) 41

[Internal Measures: Cohesion and Separation (1)](#_8qo93jon0cwp) 42

[Internal Measures: Cohesion and Separation (2)](#_g8mp6u28q29) 42

[Internal Measures: Cohesion and Separation (3)](#_vskxmt2p3y0d) 42

[Internal Measures: Silhouette Coefficient](#_qji5mgtaoj01) 43

[External Measures of Cluster Validity](#_pj62x0tlkore) 43

[Purity](#_xv8c0xypdbbg) 43

[Entropy](#_9v4xg0g5ypbw) 43

[**Chapter 8**](#_70vjaxsb47ir) **43**

[Anomaly/Outlier Detection](#_lavy8efe07s) 44

[Importance of Anomaly Detection](#_i1vx0fkszewg) 44

[Anomaly Detection Schemes](#_7heujnz5ub55) 45

[**Chapter 6**](#_ndqc7fxzs464) **46**

[Association Analysis](#_icfo1og3yhgq) 47

[Association Rule Mining](#_1zxrdco0i6al) 47

[Frequent Itemset](#_ltrfk484g0nm) 47

[Association Rule](#_6o44cb861u0) 47

[Mining Association Rules](#_usm6rccrvrky) 48

[Frequent Itemset Generation](#_8x3bkdoqtcv) 48

[Computational Complexity](#_irytgfuuwms) 48

[Frequent Itemset Generation Strategies](#_8bvqf3ey5xue) 49

[Reducing Number of Candidates](#_cfbskp1g0mxq) 49

[Illustrating Apriori Principle](#_1k8kvyt0hx56) 50

[Apriori Algorithm](#_bcjjah5z5ns0) 51

[Reducing Number of Comparisons](#_adrsxecwmt9g) 51

[Generate Hash Tree](#_vdlfkh8z05ti) 51

[Subset Operation Using Hash Tree](#_f9qjgwfs6uac) 52

[Factors Affecting Complexity](#_tw893ee873i) 53

[Factors Affecting Complexity](#_1jop9l8016bi) 53

[Maximal Frequent Itemset](#_qbr4lqttelro) 53

[Maximal vs Closed Itemsets](#_8tg42j9ahycq) 54

[Alternative Methods for Frequent Itemset Generation](#_1krd8ysx2u11) 55

[Rule Generation](#_ttt9uxlg41jg) 55

[Rule Generation for Apriori Algorithm](#_si8d0zh5fqbd) 56

[Pattern Evaluation](#_ssn6su3cj6ak) 56

[Application of Interestingness Measure](#_qycc09g0m35t) 57

[Computing Interestingness Measure](#_bu76efz9062q) 57

[Drawback of Confidence](#_9mogsrrhtecx) 57

[Statistical Independence](#_dgf0ezwwvqww) 57

[Statistical-based Measures](#_uvyt26yqgz5l) 58

[Drawback of Lift and Interest](#_9g773ythlaup) 58

[Properties of A Good Measure](#_u7sjmuwmdjf7) 58

[Property under Variable Permutation](#_e89a8uz0ds46) 59

[Property under Row/Column Scaling](#_bp76u3uc9lwe) 59

[𝝋 -Coefficient](#_ctwsxyd6dcck) 59

[Support-based Pruning](#_jde2tg5h3iwq) 59

[Subjective Interestingness Measure](#_5l2pvxm9i10t) 61

[Interestingness via Unexpectedness](#_xr3uxnyss1yb) 61

[Summar of Association Rule Mining](#_ixjsd1s7tqaq) 61

[**Chapter 5**](#_vzv6kpx80a9) **62**

[Alternative Techniques for Classification](#_28v0n67bx48m) 63

[Rule-based Classifier](#_7q1wwh2n8swz) 63

[Application of a Rule-based Classifier](#_8j7zdfopq5gf) 63

[Rule Coverage and Accuracy](#_f87tkxg4vcu) 63

[Characteristics of Rule-based Classifiers](#_c7f24yxymzc) 63

[From Decision Trees To Rules](#_6fqksyrxo832) 64

[Effect of Rule Simplification](#_ce5uhbuqdi1b) 64

[Ordered Rule Set](#_vdtdmwn0frck) 64

[Rule Ordering Schemes](#_g2akhafo8p82) 65

[Building Classification Rules](#_cbbayni9jn2n) 65

[Direct Method: Sequential Covering](#_obj3bgtuuka0) 65

[Aspects of Sequential Covering](#_6l2xo2dg9isq) 66

[Rule Growing](#_sq9x0o484nkp) 66

[CN2](#_agu1wh27l766) 66

[RIPPER](#_e5dyvyygeozc) 67

[Rule Evaluation Metrics](#_i9yswtreeow0) 67

[Stopping Criterion and Rule Pruning](#_o7mqtcuj39g3) 67

[Summary of Direct Method](#_b8izyh7vinzt) 67

[RIPPER](#_elrjmyslr7a0) 68

[RIPPER: Growing a Rule](#_81z4bnviaspd) 68

[RIPPER: Building a Rule Set](#_1go6ducqkb6r) 68

[RIPPER: Optimize the Rule Set](#_kr4dxl5gtb6b) 68

[Indirect Methods](#_yu2axmuriq41) 69

[C4.5rules](#_ceohmx79lp0d) 69

[C4.5 versus C4.5rules versus RIPPER](#_5lmiinbwt8wh) 69

[Advantages of Rule-Based Classifiers](#_c8kx8c37fqkg) 70

[Instance-Based Classifiers](#_h8u79qjjfs90) 71

[Examples](#_2idtsgmnpzx6) 71

[Nearest Neighbor Classifiers](#_pjf17ix5bvk5) 71

[Definition of K-nearest Neighbour](#_q2suuc9jq8fg) 72

[Nearest Neighbor Classification](#_ublla49tjasj) 72

[Choice of k](#_hhf6p5y0c0us) 72

[Scaling Issues](#_lqbpxe9vgu5q) 72

[Lazy Learning](#_6xdo4oz3z1wr) 73

[Bayes Classifier](#_wuuu8h5h55es) 73

[Example](#_9vner9qzms2h) 73

[Bayesian Classifiers](#_16k7kn8ylr2v) 73

[Naïve Bayes Classifier](#_50gtyrxgp031) 73

[How to Estimate Probabilities from Data: Discrete Data](#_eqtn6gfc94fo) 74

[How to Estimate Probabilities from Data: Continuous Data](#_wz24yjvq0ub) 74

[How to Estimate Probabilities from Data?](#_s5u0imd0i9yc) 74

[Example of Naïve Bayes Classifier](#_uvxyxcjvncj5) 75

[Naïve Bayes Classifier: Handling Zeroes](#_3za52a3kjj15) 76

[Naïve Bayes Summary](#_9jwxaihx74d2) 76

[Artificial Neural Networks (ANN)](#_y2zun45tgbr) 76

[General Structure of ANN](#_4nzvlnlxws2a) 77

[Support Vector Machines](#_cjuu4ntbeuj0) 77

[Support Vector Machines: Constrained Optimization](#_ql6wbien8iwn) 79

[Support Vector Machines: Nonlinearly Separable Case](#_8ytmnjqrm6db) 79

[Nonlinear Support Vector Machines](#_jzklebt1xt8f) 80

[Ensemble Methods](#_26qi2oxd771p) 80

[General Idea](#_5iwv4ohui2f8) 80

[Why does it work?](#_gy2wr1nfr0z3) 80

[Bagging](#_2n4mmi89aeig) 81

[Boosting](#_3xu572beyqkt) 81

[AdaBoost](#_c9zbd8ksc3kq) 81

[Illustrating AdaBoost](#_x05frpodk89d) 82

# 

# Chapter 1

## What is Data Mining?

Data mining has many definitions that we can look at:

* Non-trivial extraction of implicit, previously unknown and potentially useful information from data
* Exploration & analysis, by automatic or semi-automatic means, of large quantities of data in order to discover meaningful patterns
* Data-driven discovery of models and patterns from massive observation data sets

## What is NOT Data Mining

Data mining isn’t a few things:

* Simple querying i.e. Money made this financial year, looking up someone's number in a phone directory or searching for cameras on ebay
* Basically anything that isn’t the process of extracting patterns, correlations, anomalies etc.

## Data Mining Tasks

### Prediction Methods

Use previously obtained data to predict unkown or future values of other variables i.e.

* Classification
* Regression
* Time Series Analysis
* Predicition
* Deviation detection

### Description Methods

Find human-interpretable patterns (since humans can’t find patterns in objects with 8 dimensions as cool as that would be) that describe data i.e

* Clustering
* Summarization
* Association Rules / Association rule discovery
* Sequence Discovery

## Classification

### Definition

Classfication is a function that assigns (usually never seen before) items/objects to a target category or class. An example would be identifying loan applicants as either a low, medium or high credit risk.

### Goal of Classification

The goal of classification is so we can predict a new item (or items) class based on a Test set of data. A test set being a set of data used to train a function/algorithm usually split into two groups one to train/build the alrogithm and the other to test its accuracy.

### Applications of Classification

Examples of how classification is used:

**Example 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.

**Example 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.

## 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:

* Euchlidean distance if attributes are continuous
* OTHER ONES

### Clustering Applications

Examples on how clustering is applicable.

**Example 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 the same cluster vs. those from different clusters.

**Example 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.

### Example

Set of Records when shopping

| 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}

### Application of Association Rule Discovery

**Application 1**

Marketing and sales promotion

* Suppose the discovered rule is

{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!

**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 (on Thursday)

## Regression

### Definition

Simply put regression is when we predict a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency.

### Examples

* Predicting sales amount of a 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.

### Application of Regression

**Example 1: Predicting newspaper sales**

Goal: Optimise single-copy sales of De Telegraaf

Approach:

* Learn from past sales
* Let outlets learn from one another

Example 2: Deviation / Anomaly Detection in Paper Mills

Goal: Alert operators when the paper mill starts behaving “weirdly”

Approach:

* Visualize the dynamics by cleverly projecting the measurements of hundreds of sensors

## Components of Data Mining Alogirthms

Representation:

* Determining the nature and structure of the representation to be used

Score function:

* Quantifying and comparing how well different representations fit the data

Search/Optimization method:

* Choosing an algorithmic process to optimize the score function

Data Management:

* Deciding what principles of data management are required to implement the algorithms efficiently

# Chapter 2a

## Definitions

### Data

A collection of data objects and their attributes.

### Attributes

A property or characteristic of an object i.e. eye colour, temperature, age.

### Object

A collection of attributes to describe something i.e. Height, Eye Colour, Age, Gender could describe a person object.

## Types of Attributes

There are 4 different types of attributes that an attribute could lie under

### Nominal

Nominal data is data that can be labelled or classified into mutually exclusive categories. There is no meaningful order that can be applied to this data.

**Examples:**

* ID Numbers
* Eye Colour
* Zip Codes
* Hair colour
* Name of a person
* Nationalities

### Ordinal

Oridinal data is Categorical data that has some ranking/order to it. Distance between these rankings does not exist (A scale from 1 - 10 can only be a full integer for example)

**Examples:**

* Education Level → (High School Diploma, Bachelors, Masters, PhD)
* Satisfaction rating → (Extremely Dislike, Dislike, Neutral, Like, Extremely Like)
* Categories of Height → (Short, Medium, Tall)

### Interval

Interval data is like an integer. The data can be measured along some scale in which each point is equally spread apart. A ruler is a great example we go up by 1 milimeter everytime.

**Examples:**

* A wooden ruler
* Calendar dates
* Temperature in Celsius

### Ratio

Ratio data is quantitative numeric data on a continuous scale that can have a true zero. Ration Data can be categorized, ranked, evenly spaced and has a natural zero.

Examples:

* Temperature in Kelvin
* Length
* Time
* Crime Rate
* Unemployment Rate
* Weight

## Properties of Attribute Values

An easier representation of these properties can be seen with mathematical properties/operations we use in Computer Science:

* Distinctness: =, !=
* Order: <, >
* Addition: +, -
* Multiplication: \*, /

The type of data depends on which of the above rules apply:

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

## Discrete and Continuous Attributes

### Discrete Attributes

* 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 Attributes

* 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.

## Data Quality

Data is of high quality if the data if fit for our intended use and correctly represents the phenomena they correspond to. Problems in data quality can come from:

* Noise
* Outliers
* Missing values
* Duplicate data

### Noise

Noise refers to the modification of our original data values. For example a distortion of a person's voice when talking with a poor quality microphone vs a high quality microphone.

### Outliers

Outliers are data objects with characteristics that are considerably different from most of the other data objects in a given set.

### Missing Values

We can have missing values for a number of reasons:

* Information is not collected e.g. People decline to give their age or weight
* Attributes may not be applicable like annual income for children

How do we handle missing values?

* Eliminate these data objects
* Estimate missing values
* Ignore the missing values during analysis if they aren’t used
* Replace with all possible values (weighted by their probailities)

### Duplicate data

Data sets may include duplicate data objects or almost duplicates like a person making multiple accounts with the same email address. This is a major issue when merging data from multiple sources.

## Data Preprocessing

### Aggregation

Combining two or more attributes (or objects) into a single attribute (or object). The purpose of this is:

* Data reduction: Reduce the number of attributes or objects
* Change of scale: Cities aggregate into regions, states, countries etc.
* More “stable” data: Aggregate data tends to have less variablity

### Sampling

Sampling is the main technique employed for data selection (For both preliminary investigation of data and the final analysis). We use sampling since using an entire set of data is too expensive and time consuming.

#### 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

Sampling with replacement

* Objects are not removed from the population as they are selected for the sample: the same object can be picked up more than once

Stratified sampling

* Split the data into several partitions; then draw random samples from each partition

#### How to have effective sampling

* 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

### Dimensionality reduction

#### Curse of Dimensionality

Explanation 1:

When dimensionality increases, data becomes increasingly sparse in the space that it occupies. Definitions of density and distance between points, which is critical for clustering and outlier detection, become less meaningful.

Explanation 2:

Its the phenomena of data being explosive in nature when increasing data dimensions and its resulting exponential increase in computational efforts for processing and/or analysis.

Explanation 3:

When analysing high-dimensional data we run into problems and/or unusual activity that doesn’t occur in low-dimensional spaces.

#### Why reduce the Dimensions?

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:

* Principal Component Analysis (PCA)
* Singular Value Decomposition (SVD)
* Others: supervised and non-linear techniques

### Feature subset selection

There are different techniques when trying to ‘create’ a feature subset selection

* 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

The process in which to create new attributes that still capture important information in the data set but in a more efficient way. Combining features is a good way like using BMI instead of length and weight separately.

### Attribute Transformation

A function that maps the entire set of values of a given attribute to a new set of values in a way each old value can still be identified with the new values (Transforming data from one format to another). A more abstract examplewould be turning a Microsoft Word Document into a PDF.

# Chapter 2b

## Similarity

A numerical measure of how alike two data objects are (The higher similarity the more alike the objects are). Often fall in a range of [0, 1]

## Dissimilarity

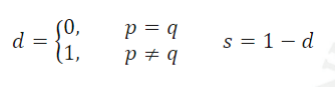
A numerical measure of how different two data objects are (The higher the value is the less alike the objects are). Lowest dissimilarity is often 0 (Representing they are absolutely nothing alike). The upper limit can vary depending on the data set.

### Proximity

Refers to a similarity or dissimilarity of data objects.

## Similarity/Dissimilarity for Simple Attributes

**Nominal attributes** *p* and *q*:



**Ordinal attributes**: Map n distinct values to integers from 0 to n - 1

**Interval or ratio attributes**:

## Euclidean Distance

The Euchlidean distance is the distance between two objects (p and q) with *n* attribites.

This can also be viewed as the distance on a ruler. Standardization is necessary, if scales differ.

**Example:**

**Object Data**

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

Using the formula:

Eventually you get this kinda matrix

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

## Minkowski Distance

The minkowski is a more generalised form of the Euclidean distance:

*r* can have a few different values here each corresponding to a different distance scheme

Manhattan Distance / City Block (r = 1)

The Manhattan Distance is the distance between two points is the sum of the absolute differences of their Cartesian coordinates (You can only move along one axis at a time). Reduces to hamming distance, which just counts the number of differences in case of binary variables.

Euchlidean distance (r = 2)

Just the normal Euchlidean distance nothing fancy

Chebyshev distance ( r = ∞)

The Chebyshev Distance measures distance between two points as the maximum difference over any of their axis values. In a 2D grid, for instance, if we have two points (x1, y1), and (x2, y2), the Chebyshev distance between is max(y2 - y1, x2 - x1).

## Common Properties of a distance

Distances, such as the Euclidean distance, have some well-known properties

1. Positive definiteness d(p, q) ≥ 0 for all p and q and d(p, q) = 0 iff p = q
2. Symmetry: d(p, q) = d(q, p) for all p and q
3. Triangle inequality d(p, r) ≤ d(p, q) + d(q, r) for all p, q and r

A distance that satisfies these properties is called a metric

## Similarity Between Binary Vectors

### Simple Matching Coefficient (SMC):

**Example**

p = [1 0 0 0 0 0 0 0 0 0]

q = [0 0 0 0 0 0 1 0 0 1]

|  | 0 | 1 |
| --- | --- | --- |
| 0 | 7 | 1 |
| 1 | 2 | 0 |

### Jaccard Coefficient

**Example**

p = [1 0 0 0 0 0 0 0 0 0]

q = [0 0 0 0 0 0 1 0 0 1]

|  | 0 | 1 |
| --- | --- | --- |
| 0 | 7 | 1 |
| 1 | 2 | 0 |

### Cosine Similarity

Specifically for document vectors

Example:

p = [3 2 0 5 0 0 0 2 0 0]

q = [1 0 0 0 0 0 0 1 0 2]

# 

# 

# 

# 

# 

# 

# 

# 

# 

# Chapter 3

## What is data exploration?

A preliminary exploration of the data to better understand its characteristics. Key motivations of data exploration include:

* Helping to select the right tool for preprocessing or analysis
* Making use of humans’ abilities to recognize patterns

Related to the area of Exploratory Data Analysis (EDA)

* Created by statistician John Tukey
* Seminal book is Exploratory Data Analysis by Tukey
* A nice online introduction can be found in Chapter 1 of the NIST Engineering Statistics Handbook http://www.itl.nist.gov/div898/handbook/index.htm

# 

# Chapter 8

## What is Cluster Analysis

Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups. This means the Intra-cluster distances are as small as we can get them and the Inter-cluster distances are as big as we can get them (The clusters are preferably grouped together and the clusters are far apart).

Cluster Analysis helps us group objects/data together in a way that is useful for us to identify patterns we may not be able to see (easier to visualise). It also helps reduce the size of large data sets especially when we visually represent the data

### What is not Cluster Analysis?

*Supervised classification*

* Have class label information

*Simple segmentation*

* Dividing students into different registration groups alphabetically, by last name

*Results of a query*

* Groupings are a result of an external specification

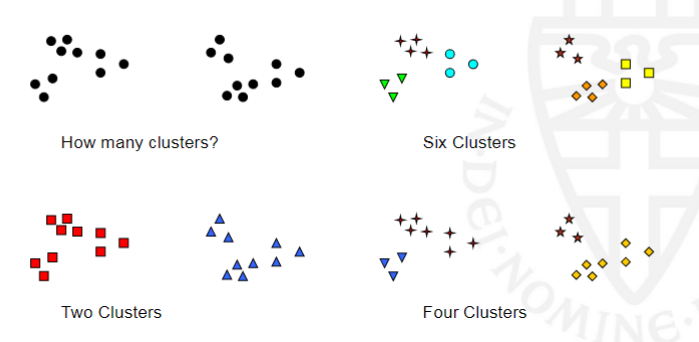
*Graph partitioning*

* Some mutual relevance and synergy, but areas are not identical

Basically anything that's not identifying clusters/patterns in a given data set that's not yet classed (Not results of a query).

## Notion of Clusters can be Ambiguous

Depending on how you view/see given data depends on how you may interpret a cluster. Below is a good example of this. There currently isn’t a way we know that will help us pick the correct amount of clusters.

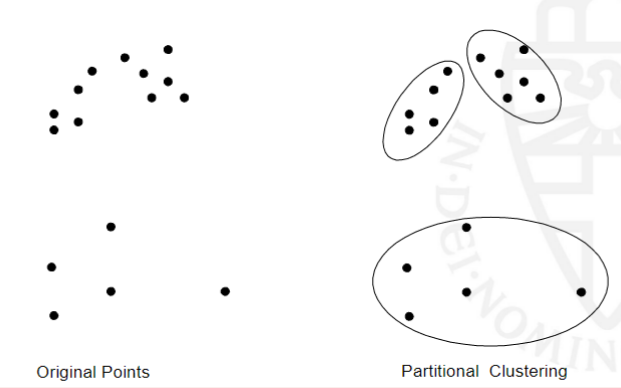


## Types of Clusterings

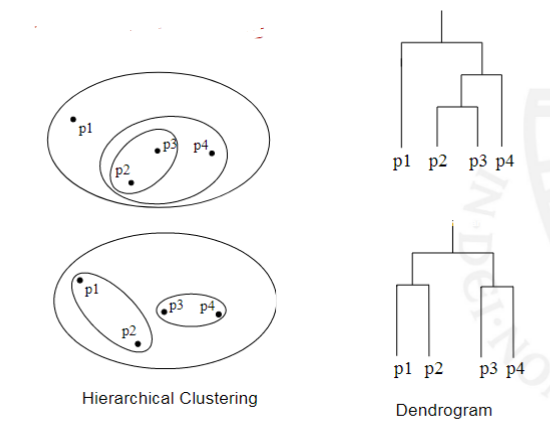
A **clustering** is a set of clusters and not just one particular cluster. There is an important distinction between hierarchical and partitional sets of clusters:

### Partitional Clustering

Can be described as a division of the objects into non-overlapping subsets where one data object can only lie in one cluster (One point lies in one cluster).



### Hierarchical Clustering

Like a russian nesting doll where each cluster is inside a one big cluster (root cluster). A set of nested clusters organised into a hierarchical tree. 

### Other Distinctions Between Sets of Clusters

***Exclusive versus non-exclusive***

* In non-exclusive clusters, points may belong to multiple clusters.
* A data object can represent multiple classes or ‘border’ points in the clustering

***Fuzzy versus non-fuzzy***

* In fuzzy clustering, a point belongs to every cluster with some weight between 0 and 1 the weights must sum to 1 in the whole data set
* Probabilistic clustering has similar characteristics

***Partial versus complete***

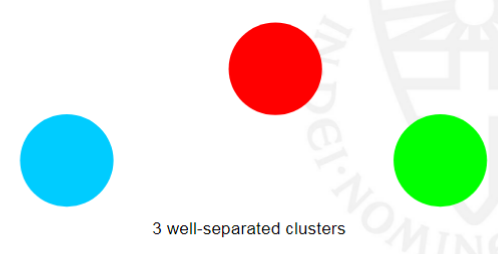
* In some cases, we only want to cluster some of the data

***Heterogeneous versus homogeneous***

* Cluster of widely different sizes, shapes, and densities

## Types of Clusters

### Well-separated clusters

A well-separated 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.

### Center-based clusters

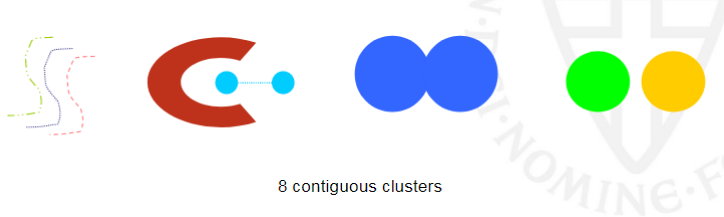
Each object in a centre-based cluster is closer to the centre of the cluster than to the centres of any other clusters.

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.



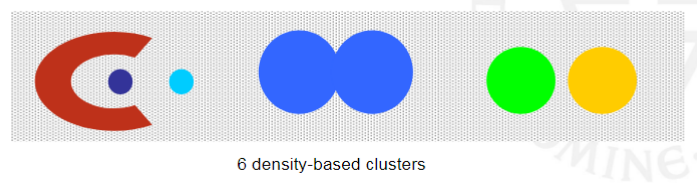
### Contiguous clusters

Each object in a contiguity-based cluster is closer to some other object in the cluster than to any point in a different cluster (A object is in the cluster that its nearest neighbour is part of or the most similar to).



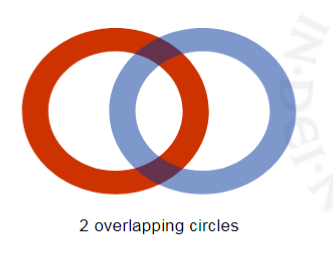
### Density-based clusters

* A density-based 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



### Property or Conceptual Clusters

A conceptual (shared-property) cluster contains points that jointly share some common property or represent a particular concept



### Based upon an Objective Function

Many objective functions define the “goodness” of a clustering. A rating of how ‘good’ of a cluster it is. There are specific algorithms for optimizing such objective functions. Finding the optimal solution often requires enumerating all possibleways of dividing the points into clusters (NP Hard). Can have global or local objectives:

* Hierarchical clustering algorithms typically have local objectives
* Partitional algorithms typically have global objective

A variation of the global objective function approach is to fit the data to a parameterized statistical model:

* Parameters for the model are determined from the data.
* Mixture models assume that the data is a ‘mixture' of a number of statistical distributions.

Alternative approach is to map the clustering problem to a different domain and solve a related problem in that domain

* E.g., proximity matrix defines a weighted graph, where the nodes are the points being clustered, and the weighted edges represent the proximities between points.
* Clustering is equivalent to breaking the graph into connected components, one for each cluster: minimum spanning tree algorithm

## Characteristics of the Input Data

When looking at input data we need to look at the characteristics of the data since this will play a role in how we cluster / identidy clusters (and how well we cluster the data). These characteristics include:

*Type of proximity or density measure*

* This is a derived measure from the data, but central to clustering.

*Sparseness*

* Dictates the type of similarity we will use between data points
* Adds to the efficiency of the cluster

*Attribute type*

* Dictates type of similarity we will use

*Type of Distribution*

* Dictates type of similarity we will use
* Other characteristics, e.g., autocorrelation

## Clustering Algorithms

### K-means Clustering and its variants

* K-means is a Partitional clustering approach
* Each cluster is associated with a centroid (The center point of a cluster or the most representative point of the cluster)
* Each point is assigned to the cluster with the closest centroid
* The number of clusters, K, must be specified in the execution
* The basic algorithm is very simple. Below is a basic implementation of K-means clustering

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 centroids don’t change
5. **Until** The centroids don’t change

* Initial centroids are often chosen randomly (since when we keep calling the algorithm we should eventually get the correct centroids)
* 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.

#### Convergence and Complexity

* K-means will converge for common similarity measures (In reference two the similarity of two data points)
* Most of the convergence happens in the first few iterations (The general shape of the cluster is formed)
* Often the stopping condition is changed to ‘Until relatively few points change clusters’
* Complexity is O(n\*K\*I\*d)

n = number of points, K = number of clusters,

I = number of iterations, d = number of attributes

### Evaluating K-means Clusters

The most common measure is Sum of Squared Error (SSE)

* For each point, the error is the distance to the nearest cluster
* To get SSE, we square these errors and sum them:
* X is a data point in cluster and is the representative point for cluster

Given two clusterings, the one with the smallest error is the “better” one. One easy way to reduce SSE is to increase K (the number of clusters), so it typically only makes sense to compare SSE’s for clusterings with the same K.

* K-means has problems when the data contains outliers

#### Solutions to Initial Centroids Problem

* Have multiple runs
  + Helps, but probability is not on your side since we have so many data points selecting them at random isn’t an efficient option
* Sample points 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 points
* Post Processing
* Bisecting K-means
  + Split up larger clusters into smaller subsets
  + Not as susceptible to initialization issues

#### Handling Empty Clusters

Basic K-means algorithms can yield empty clusters. Idea: set the cluster center of an empty cluster equal to one of the data points and continue updating

Several strategies

* Choose the data point that contributes most to SSE
* Choose a data point from the cluster with the highest SSE
* If there are several empty clusters, the above can be repeated several times

#### Updating Centers Incrementally

* In the basic K-means algorithm, centroids are updated after all points are assigned to a centroid
* An alternative is to update the centroids after each assignment (incremental approach)
  + Each assignment updates zero or two centroids
  + More expensive
  + Introduces an order dependency
  + Never get an empty cluster

##### Pre-processing

* Normalize the data
* Eliminate outliers

##### Post-processing

* Eliminate small clusters that may represent outliers
* Split ‘loose’ clusters, i.e., clusters with relatively high SSE
* Merge clusters that are ‘close’ and that have relatively low SSE
* Can use these steps during the clustering process (done in an algorithm called ISODATA)

#### 

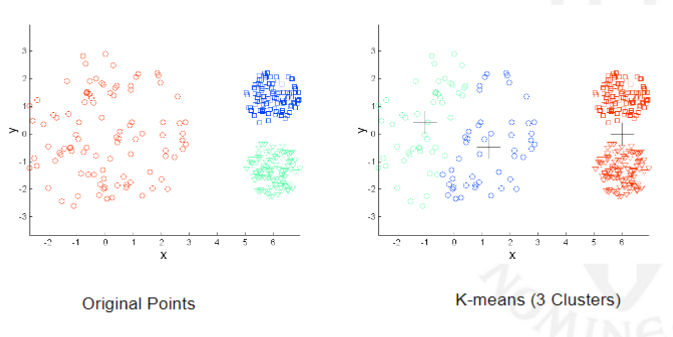
#### Bisecting K-means

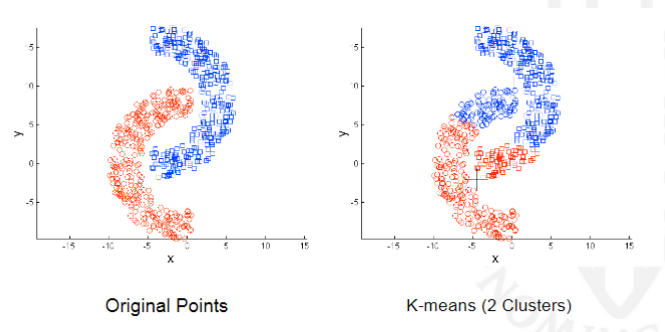
Variant of K-means that can produce a partitional or a hierarchical clustering. Can be viewed as splitting a cluster into multiple clusters and finding the lowest SSE combination to get the ‘best’ cluster. A written example of how it works

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

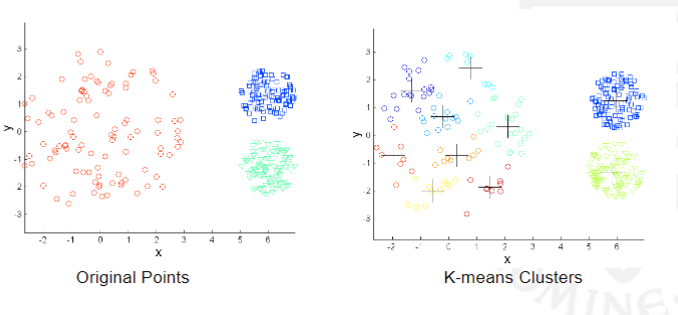
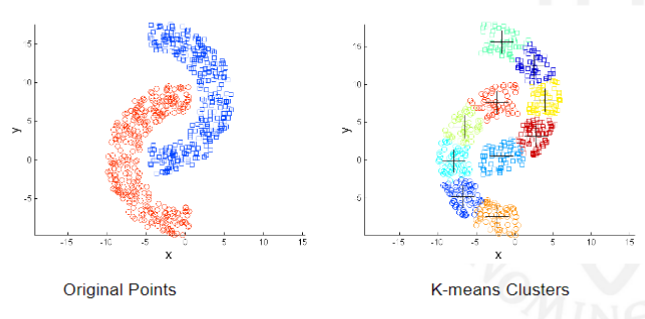
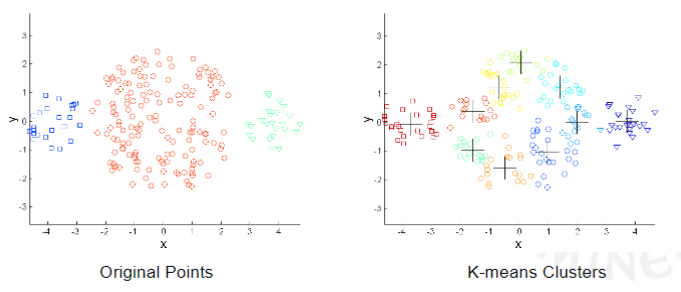
#### Limitationts of K-means

K-means has problems when clusters that are of differing

* Sizes: When each cluster differs in sizes its easy for the centroid to not be where you would normally expect. Think about it like a shadow of Mickey Mouse's head that all the clusters aren’t the same sizes so the centroid would not be ideal.
* Densities:
* Non-globular shapes

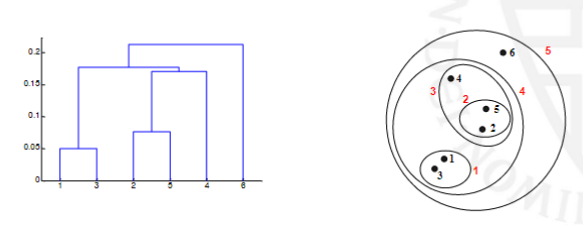


One solution is to use many clusters and simply combine the ‘best’ pairs together.



## Hierarchical clustering

Produces a set of nested clusters organized as a hierarchical tree. It can be visualized as a dendrogram (A tree like diagram that records the sequences of merges or splits)

**Strengths of Hierarchical Clustering**

* You 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, ...)

There are different kinds of Hierarchical Clustering that one can use (besides the traditional algorithm for merging or splitting a cluster at a time). There is:

### Agglomerative

You start with all the points as individual clusters and then combine the closest pair clusters until only one cluster is left (or until the desired amount of clusters is achieved). The basic algorithm is simple:

1. Compute the proximity matrix

2. Let each data point be a cluster

3. **Repeat**

4. Merge the two closest clusters

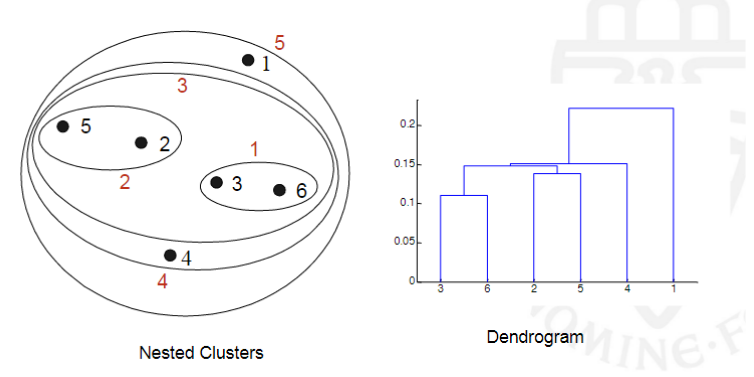
5. Update the proximity matrix

6. **Until** only a single cluster remains

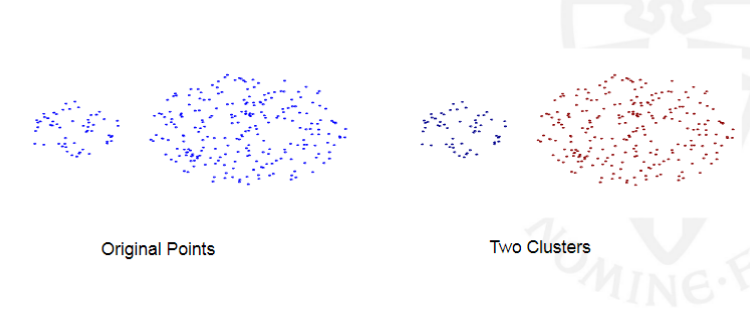
*(Different approaches to defining the distance between two clusters in step 5 distinguish the different algorithms)*

We can define cluster similarity in different ways depending on our data and its characteristics. We can have:

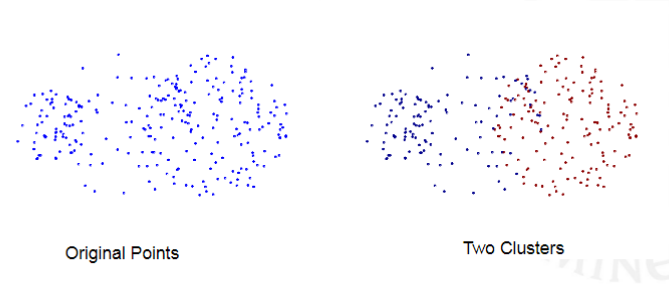
* *Min*: The closest data point between the two clusters

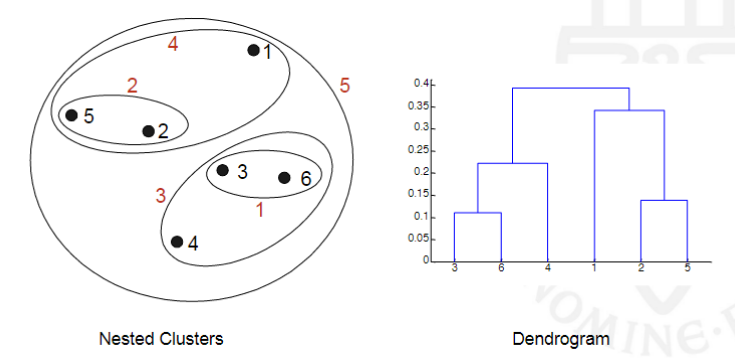
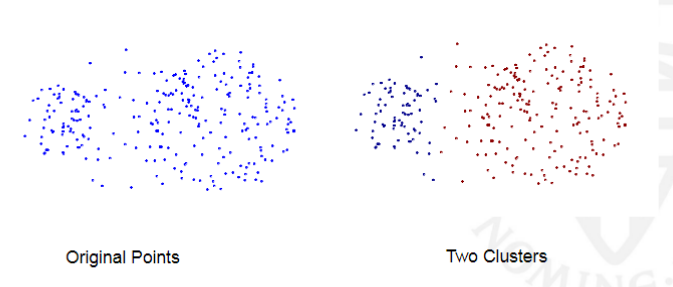


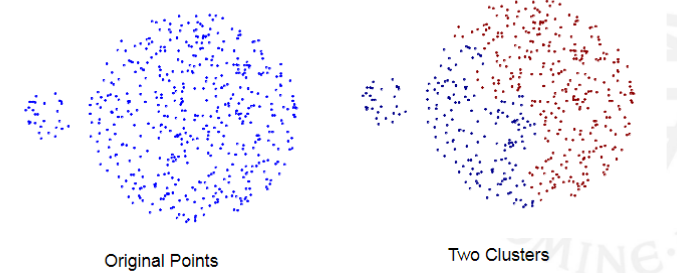
* + Strengths: Can handle non-spherical shapes

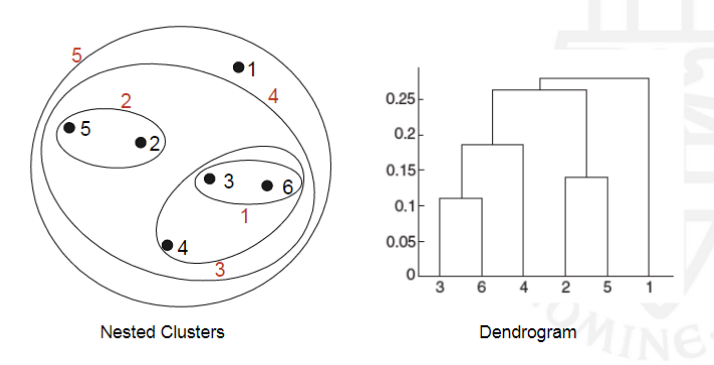


* + Weakness: Sensitive to noise



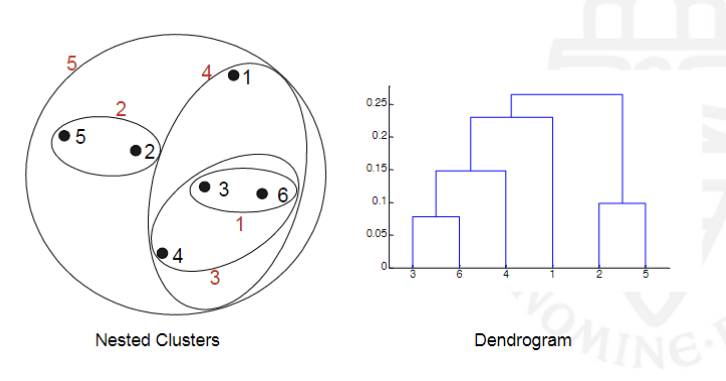
* *Max*: The furthest data point between the two clusters
  + Strengths: Less susceptible to noise and outliers
  + Weakness: Tends to break large clusters and is biased towards global clusters



* *Group average*: Average the pairwise proximity of all points in the two clusters. Its a compromise between min and max methods.
  + Strengths: Less susceptible to noise and outliers
  + Weaknesses: Biased towards global clusters
* *Distance between centroids*: The distance between the centroids of the clusters

Other methods driven by an objective function includes:

* *Ward’s method*: The similarity of two clusters is based on the increase in square error when two clusters are merged (Similar to group average if you were to square the distance between points).



* + Strength: Less susceptible to noise and outliers
  + Weakness: Biased towards globular clusters.

Comparisons:

##### Time and Space Requirements

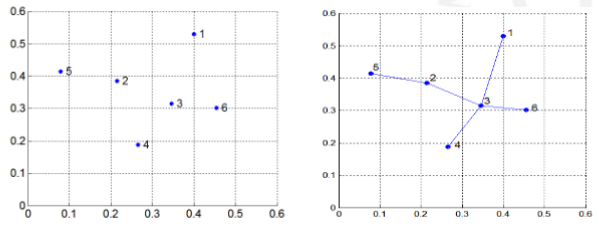
* space since it uses the proximity matrix
  + N is the number of points
* time in many cases
  + There are N steps and at each step the size, N 2 , proximity matrix must be updated and searched
  + Complexity can be reduced to O(N 2 log(N) ) time for some approaches

##### 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

### Divisive Hierarchical Clustering (MST)

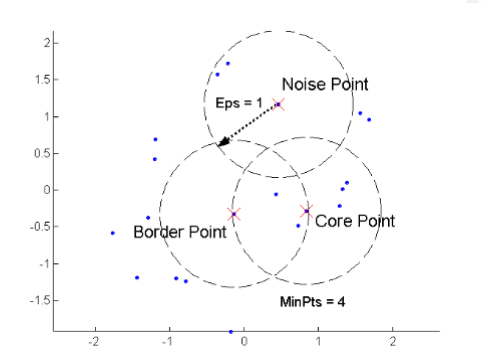
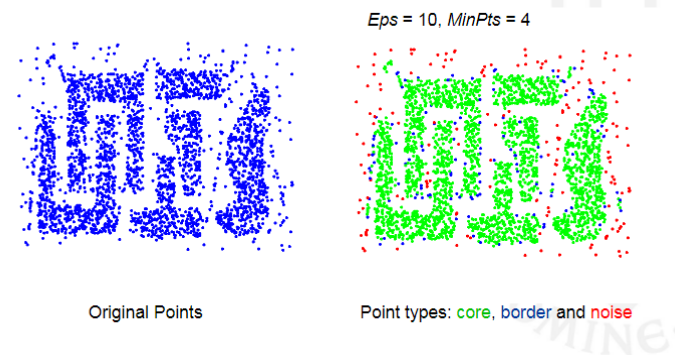
Simple terms we: Start with one all inclusive cluster and each step you split a cluster until you end up with each data point being its own cluster (or until the desired amount of clusters is achieved). How we actually make a MST is we:

* 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
* Use the MST for constructing hierarchy of clusters

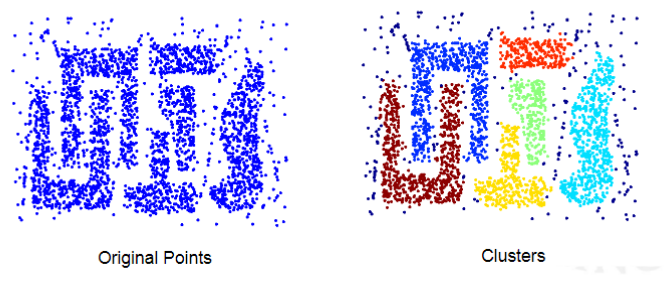
1. Compute a minimum spanning tree for proximity graph.
2. repeat
   1. Create a new cluster by breaking the link corresponding to the largest distance
3. Unil only singleton clusters remain

### Density-based clustering (DBSCAN)

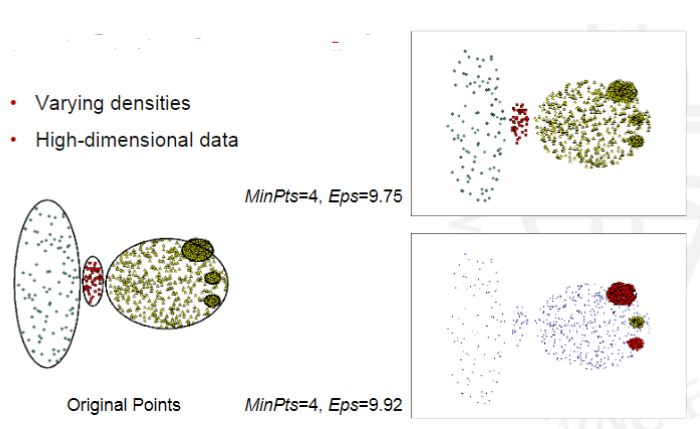
* DBSCAN is a density-based algorithm where the Density = number of points within a specified radius (Eps).
* A point is a core point if it has more than a specified number of points (MinPts) within Eps (These are points that are at the interior of a cluster).
* A border point has fewer than MinPts within 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 on a graph

#### When DBSCAN Works Well

DBSCAN works really well when we have a lot of noise and when we have clusters of different shapes and sizes. E.g.

#### When DBSCAN Does NOT Work Well



#### Determining EPS and MinPts

* Idea is that for points in a cluster, their kth nearest neighbors, with k equal to MinPts, are at roughly the same distance
* Noise points have the kth nearest neighbor at farther distance
* So, plot sorted distance of every point to its kth nearest neighbor and look for sharp increase
* In this example to the right: choose Eps ≈10

## Cluster Validity

* For supervised classification we have a variety of measures to evaluate how good our model is: Accuracy, precision, recall, area under the curve, etc.
* For cluster analysis, the analogous question is how to evaluate the “goodness” of the resulting clusters since “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 sets of clusters
  + To compare two clusters

### Different Aspects of Cluster Validation

1. Determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels
3. Evaluating how well the results of a cluster analysis fit the data without reference to external information
4. Comparing the results of two different sets of cluster analyses to determine which is better
5. Determining the ‘correct’ number of clusters

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

### Measures of Cluster Validity

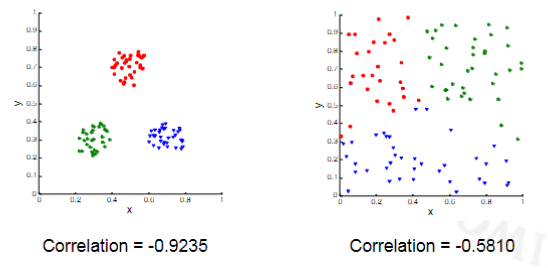
Numerical measures that are applied to judge various aspects of cluster validity. They are classified into the following three types:

* *External Index*: Used to measure the extent to which cluster labels match externally supplied class labels; example: entropy
* *Internal Index*: Used to measure the goodness of a clustering structure without respect to external information; example: sum of squared errors (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

#### Cluster Validity Via Correlation

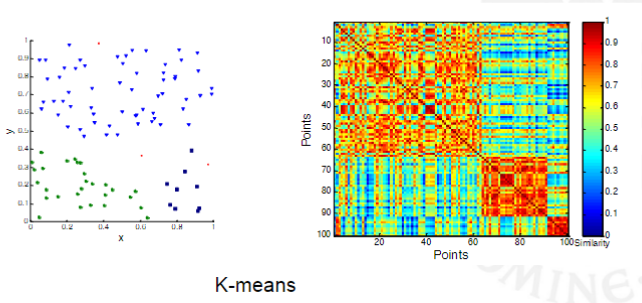
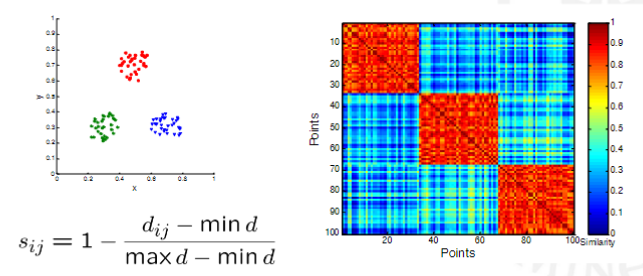
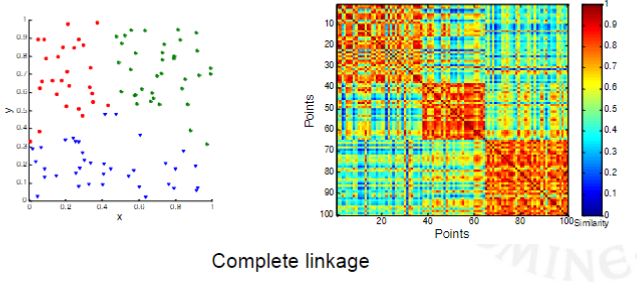
* Two matrices
  + Proximity Matrix (e.g., used as input to the clustering algorithm)
  + “Incidence” Matrix (computed from the clustering result)
* Incidence matrix:
  + One row and one column for each data point
  + An entry is 1 if the associated pair of points belong to the same cluster
  + An entry is 0 if the associated pair of points belongs to different clusters
* Compute the correlation between the two matrices
* Since the matrices are symmetric, only the correlation between n (n-1)/2 entries needs to be calculated
* High correlation (in absolute sense) indicates that points that belong to the same cluster are close to each other
* Not a good measure for some density or contiguity based clusters

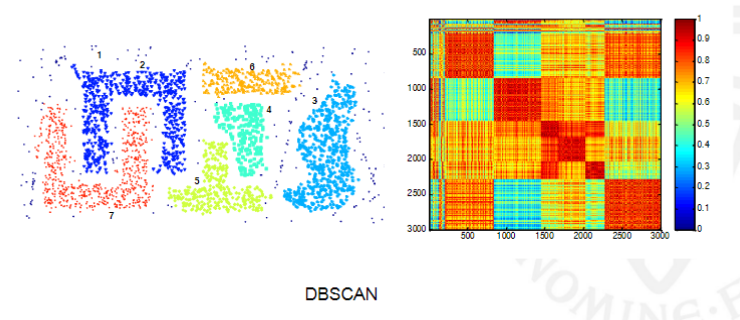
*Example*: Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets



#### Cluster Validation via Similarity Matrix

Order the similarity matrix with respect to cluster labels and inspect it visually. Below we have an example with visually apparent clusters:

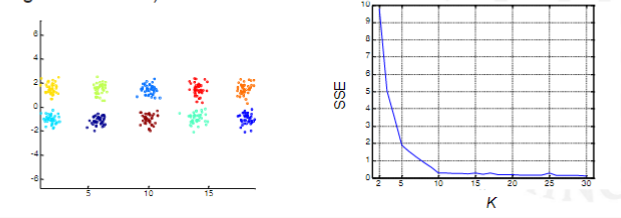
Here we have an example of a similarity matrix with random data we can see its just not as crisp.



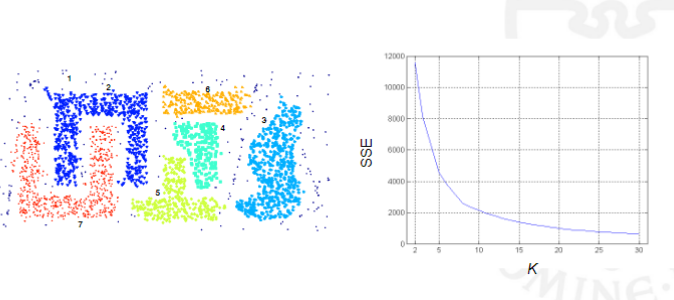
#### Internal Measures: Sum of Squared Errors

*Internal index*: used to measure the goodness of a clustering structure without respect to external information

SSE can be used for comparing two clusterings (e.g., based on different numbers of clusters) or two clusters (small SSE: tight; high SSE: loose)



SSE curve for a more complicated data set and K-means



### Framewoork for Cluster Validity

Need a framework to interpret any measure.

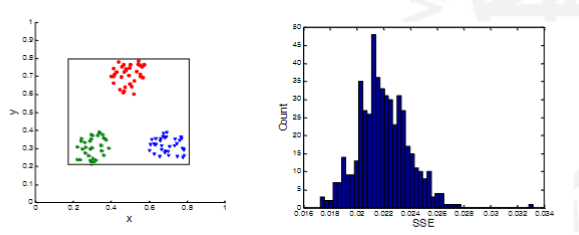
* For example, if our measure of evaluation has the value 10, is that good, fair, or poor?

Statistics, in particular Monte Carlo sampling, provides a framework for cluster validity

* The more “atypical” a clustering result is, the more likely it represents valid structure in the data
* Can compare the values of an index obtained by clustering actual data with those obtained by clustering random data in the same range
* If the value of the index is unlikely, then the cluster results are valid

### Statistical Framework for SSE

* Compare SSE of 0.005 against three clusters in random data
* Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range [0.2,0.8] for x and y values.
* Kinda like comparing what we think is a cluster to random data to make sure we have a lesser SSE than random data



### Statistical Framework for Correlation

Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.

### 

### Internal Measures: Cohesion and Separation (1)

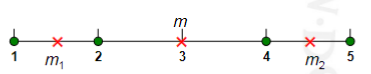
Cluster Cohesion measures how closely related are objects in a cluster, e.g., through the within cluster sum of squared errors:

Cluster Separation measure how distinct or well-separated a cluster is from other clusters, e.g., through the between cluster sum of squares:

with is the size of (i.e., number of data points belonging to) cluster i and m the overall mean of the data.

### Internal Measures: Cohesion and Separation (2)

It can also be shown that

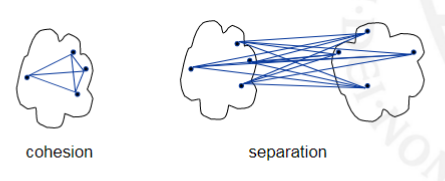
I.e., independent of the clustering

### Internal Measures: Cohesion and Separation (3)

A proximity graph based approach can also be used for cohesion and separation

**-** Cluster cohesion is the sum of the weight of all links within a cluster

**-** Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster



### Internal Measures: Silhouette Coefficient

* The silhouette coefficient combines ideas of both cohesion and separation
* Silhouette for data point i: si = (b i – a i ) / max(a i ,b i ), where:

= average distance of i to the points in the same cluster

= min (average distance of i to points in another cluster), where the minimum is taken over all other clusters

* Typically b i ≥ a i and then 0 ≤ si ≤ 1; the higher the better
* Often also averaged over clusters and clusterings

### External Measures of Cluster Validity

* When given labels (e.g., afterwards), there are better/easier ways to evaluate clusterings using so-called external measures
* In the above clustering, which of the clusters performs best?

Cluster 3 appears to be the best just by looking at the table. It has the highest number of any article/document type being sport at 671 (out of a total 738 amongst all the clusters). It doesn’t do well in all categories but our definition of “performing the best” is very up to interpretation

### Purity

“Probability” that a member of cluster j belongs to class i:

with the number of data points belonging to cluster j and class i

Purity of a cluster:

Simply put: The number of correctly matched class and cluster labels divided by the number of total data points

### Entropy

Entropy is the purity of a cluster (or sub split in terms of decision trees). The lower entropy (The less impurities in a subset/cluster) the better a given cluster or set is.

Entropy of a cluster:

Total entropy/purity of a clustering:

# Chapter 8

## Anomaly/Outlier Detection

**What are anomalies/outliers?**

* The set of data points that are considerably different from the remainder of the data

**Variants of Anomaly/Outlier Detection Problems**

* Given a database **D**, find all the data points with anomaly scores greater than some threshold **t**
* Given a database **D**, find all the data points having the top-n largest anomaly scores f(**x**)
* Given a database **D**, containing mostly normal (but unlabeled) data points, and a test point **x**, compute the anomaly score of **x** with respect to **D**

**Applications:**

* Credit card fraud detection, telecommunication fraud detection, network intrusion detection, fault detection

**Challenges:**

* How many outliers are there in the data?
* Method is unsupervised, so validation can be quite challenging (just like for clustering)
* Finding needle in a haystack

**Working assumption:**

* There are considerably more “normal” observations than “abnormal” observations (outliers/anomalies) in the data

### Importance of Anomaly Detection

* In 1985 three researchers (Farman, Gardinar, and Shanklin) were puzzled by data gathered by the British Antarctic Survey showing that ozone levels for Antarctica had dropped 10% below normal levels
* Why did the Nimbus 7 satellite, which had instruments aboard for recording ozone levels, not record similarly low ozone concentrations?
* The ozone concentrations recorded by the satellite were so low they were being treated as outliers by a computer program and discarded!

### Anomaly Detection Schemes

**General Steps**

* Build a profile of the “normal” behavior, e.g., patterns or summary statistics for the overall population
* Use the “normal” profile to detect anomalies: Anomalies are observations whose characteristics differ significantly from the normal profile

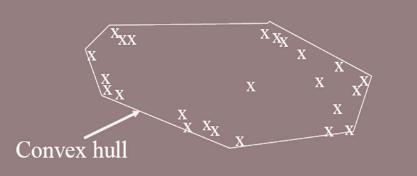
**Types of anomaly detection schemes**

*Graphical-based:* Box Plots (1-D), Scatter Plot (2-D) or Spin Plot (3-D).

However, they are time consuming and very subjective.

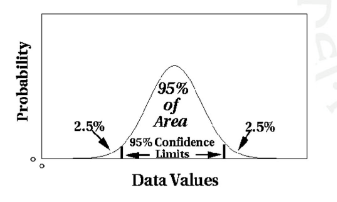
*Convex Hull Method*

* Extreme points are assumed to be outliers
* Use convex hull method to detect extreme values



*Statistical-based:*

* Assume a parametric model describing the distribution of the data (e.g., normal distribution)
* Apply a statistical test that depends on
  + Data distribution
  + Parameter of distribution (e.g., mean, variance)
  + Number of expected outliers (confidence limit)



* Limitations of Statistical Appraches:
  + Most of the tests are for a single attribute
  + In many cases, data distribution may not be known
  + For high dimensional data, it may be difficult to estimate the true distribution

*Grubb’s Test*

* Detect outliers in univariate data (Data with only 1 variable)
* Assume data comes from normal distribution
  + Detects one outlier at a time, remove the outlier, and repeat
  + : There is no outlier in data
  + : There is at least one outlier
* Grubbs’ test statistic:
* Reject if:

*Distance-based:*

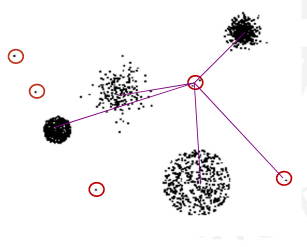
* Data is represented as a vector of features
* Three major approaches
  + Nearest-neighbor based
  + Density based
  + Clustering based

*Nearest-Neighbor Based Approach*

* Compute the distance between every pair of data points
* There are various ways to define outliers:
  + Data points for which there are fewer than **p** neighboring points within a distance **D**
  + The top **n** data points whose distance to the **k**th nearest neighbor is greatest
  + The top **n** data points whose average distance to the **k** nearest neighbors is greatest

*Density-Based: Local Outlier Factor*

* For each point, compute the density of its local neighborhood
* Compute local outlier factor (LOF) of a sample p as the average of the ratios of the density of sample p and the density of its nearest neighbors
* Outliers are points with largest LOF value



Clustering-Based

* Cluster the data into groups of different density
* Choose points in small cluster as candidate outliers
* Compute the distance between candidate points and non-candidate clusters: if these candidate points are far from all other non-candidate points, they are outliers

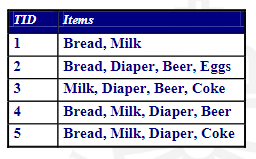
# Chapter 6

## Association Analysis

### Association Rule Mining

Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

### Frequent Itemset

*Itemset*

* A collection of one or more items
* Example: {Milk, Bread, Diaper}
* K-itemset
  + An itemset that contains k items

*Support count ()*

* Frequency of occurrence of an itemset
* E.g. ({Milk, Bread, Diaper}) = 2

*Support*

* Fraction of transactions that contain an itemset (How often the set appears in the transactions).
* E.g. s({Milk, Bread, Diaper}) = ⅖

*Frequent itemset*

* An itemset whose support is greater than or equal to a minsup threshold

### Association Rule

*Association Rule*

* An implication expression of the form:

X → Y, where X and Y are itemsets

* Example: {Milk, Diaper} → {Beer}

*Rule evaluation metrics:*

* Support (s): Fraction of transactions that contain both X and Y
  + Example:
* Confidence (c): Measures how often items in Y appear in transactions that contain X
  + Example:

**Association Rule Mining Task**

Given a set of transactions T, the goal of association rule mining is to find all rules having

* Support (s) ≥ **minsup** threshold (**Minsup:** a user-specified variable that stands for the minimum support threshold for itemsets.)
* Confidence (c) ≥ **minconf** threshold (**Minconf:** a user-specified variable that stands for the minimum confidence threshold for rules.)

Brute-force approach:

* List all possible association rules
* Compute the support and confidence for each rule
* Prune rules that fail the minsup and minconf thresholds

⇒ Computationally prohibitive!

*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.

### Mining Association Rules

Two-step approach:

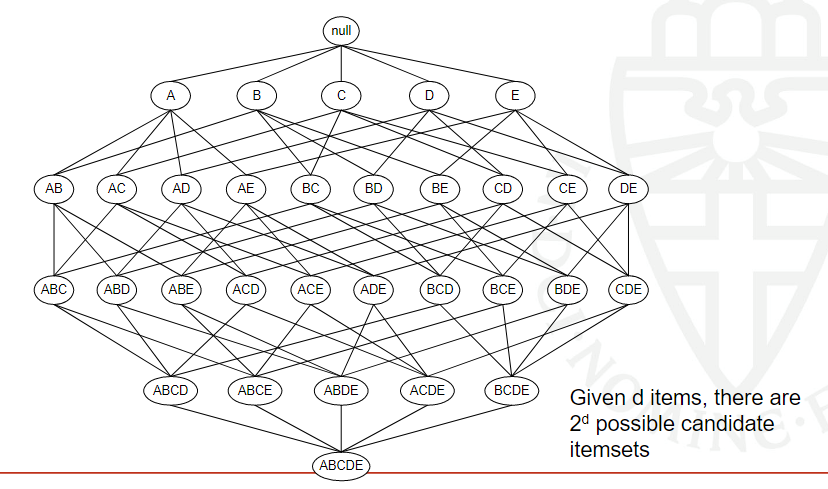
1. Frequent itemset generation

* + Generate all itemsets whose support minsup

2. Rule generation

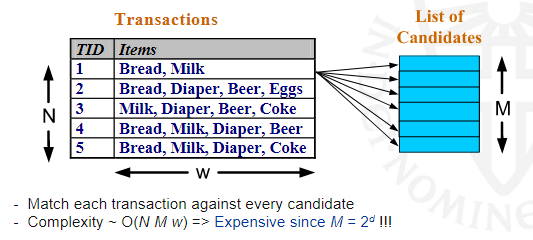
* + Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset
* Frequent itemset generation is still computationally expensive

#### Frequent Itemset Generation



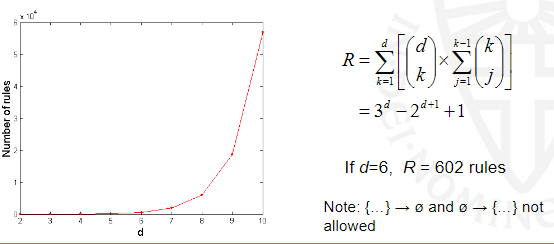
*Brute-force approach:*

* Each itemset in the lattice is a candidate frequent itemset
* Count the support of each candidate by scanning the database



#### Computational Complexity

* Given d unique items:
  + Total number of itemsets
  + Total number of possible association rules:



#### Frequent Itemset Generation Strategies

*Reduce the number of candidates (M)*

* Complete search:
* Use pruning techniques to reduce M

*Reduce the number of transactions (N)*

* Reduce size of N as the size of itemset increases
* Used by Direct Hash & Pruning and vertical-based mining algorithms

*Reduce the number of comparisons (N M)*

* Use efficient data structures to store the candidates or transactions
* No need to match every candidate against every transaction

#### Reducing Number of Candidates

* *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:
* Support (s) of an itemset never exceeds the support (s) of its subsets
* This is known as the monotone property of support

##### Illustrating Apriori Principle

****

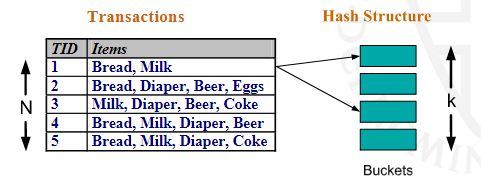
****

### Apriori Algorithm

* Let k = 1
* Generate frequent itemsets of length 1
* Repeat until no new frequent itemsets are identified
  + Generate length (k+1) candidate itemsets from length k frequent itemsets
  + Prune candidate itemsets containing subsets of length k that are infrequent
  + Count the support of each candidate by scanning the DB
  + Eliminate candidates that are infrequent, leaving only those that are frequent

#### Reducing Number of Comparisons

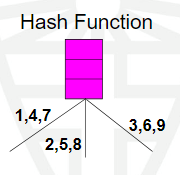
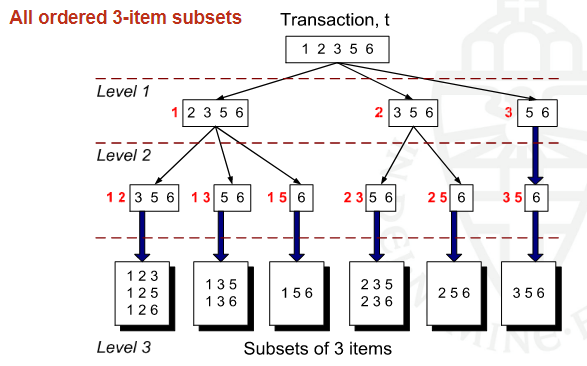
*Candidate counting:*

* Scan the database of transactions to determine the support of each candidate itemset
* To reduce the number of comparisons, store the candidates in a hash structure
* Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets
* Example: candidate itemsets with 3 items out of 9 products:

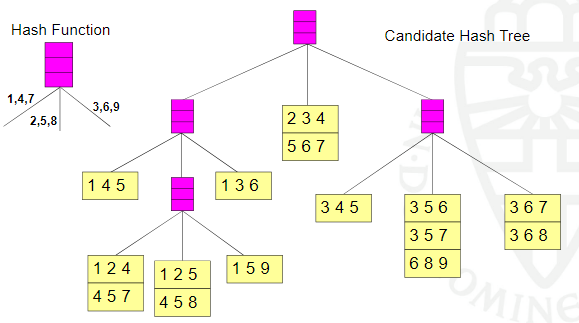
{1,2,4}, {1,2,5}, {1,3,6}, {1,4,5}, {1,5,9},

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

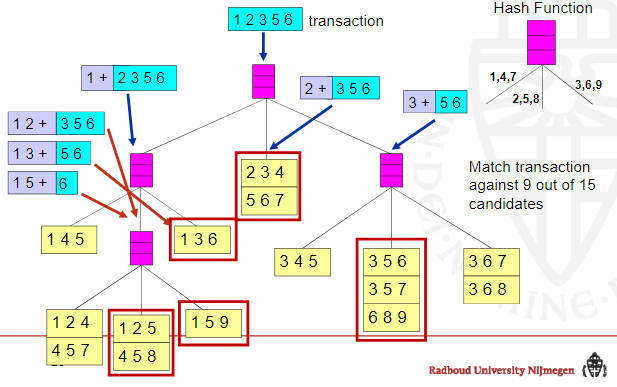
{4,5,7}, {4,5,8}, {5,6,7}, {6,8,9}

* New transaction t: {1,2,3,5,6}
* Plan:
  + Consider all ordered 3-item subsets in the transaction
  + Compare them against all candidate itemset
  + If there’s a match, the support of the corresponding candidate itemset gets +1
  + Do this for all transactions
* Additional trick: store the candidate itemsets in a (hash) structure

#### Generate Hash Tree



#### Subset Operation Using Hash Tree



#### Factors Affecting Complexity

Choice of minimum support threshold (**minsup**)

* Lowering support threshold results in more frequent itemsets
* This may increase number of candidates and max length of frequent itemsets

Dimensionality (number of items) of the data set

* More space is needed to store support count of each item
* If number of frequent items also increases, both computation and I/O costs may also increase

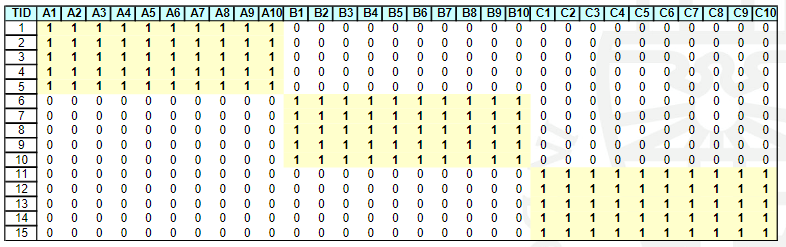
Size of database

* Since Apriori makes multiple passes, run time of algorithm may increase with number of transactions

Average transaction width

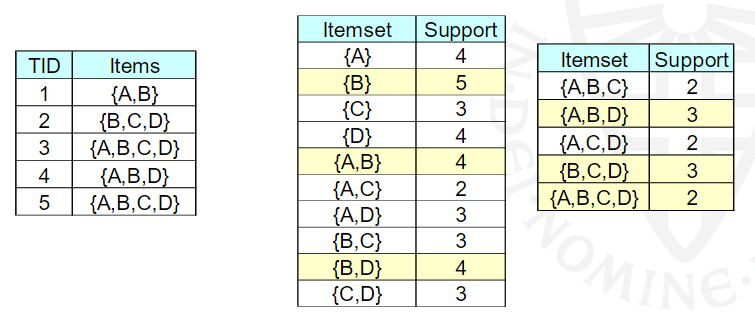
* Transaction width increases with denser data sets
* This may increase max length of frequent itemsets and traversals of hash tree (number of subsets in a transaction increases with its width)

**Compact Representation of Frequent Itemsets**

* Some itemsets are redundant because they have identical support as their supersets
* Number of frequent itemsets =
* Need a compact representation

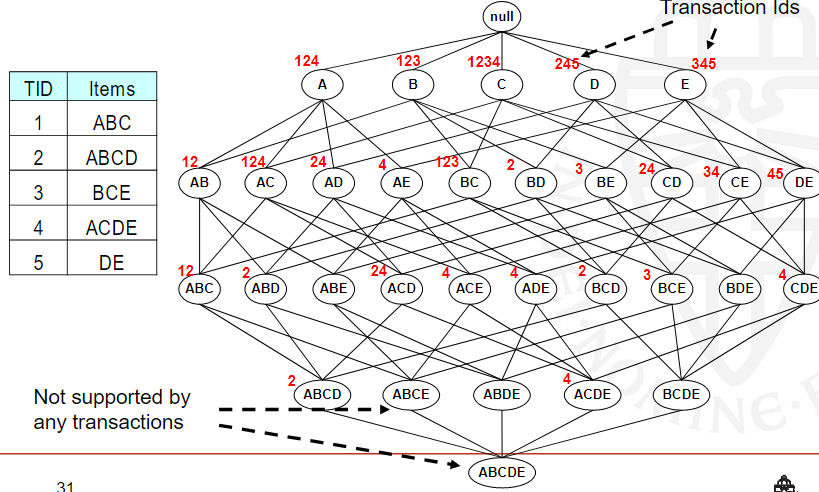
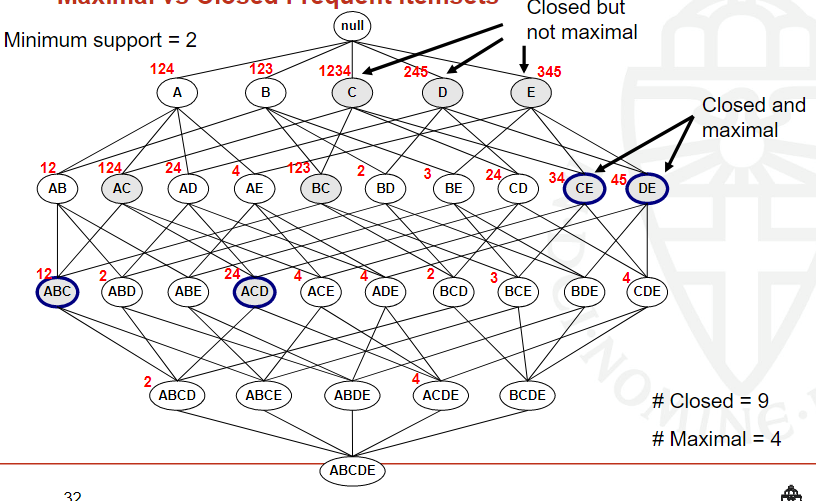
#### Factors Affecting Complexity

* An itemset is **closed** if none of its immediate supersets has the same support as the itemset i.e. The subset can’t provide the exact same answers (which is what would happen if the support of the superset is the same).
* Compact representation of itemsets without loss of support info



### Maximal Frequent Itemset

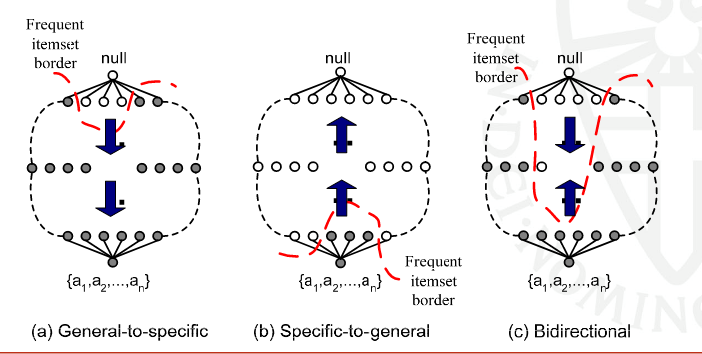
### Maximal vs Closed Itemsets



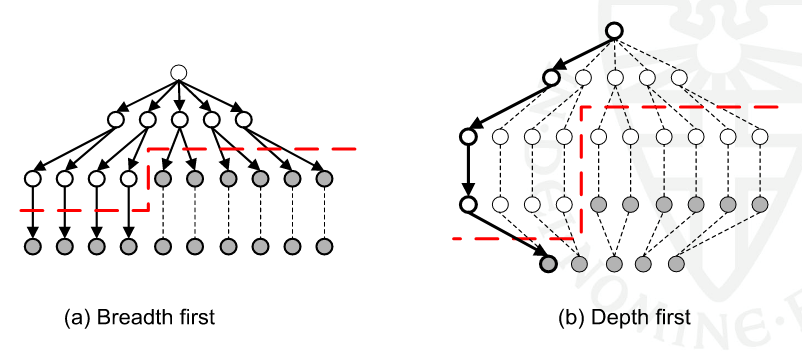


#### Alternative Methods for Frequent Itemset Generation

General-to-specific (as Apriori) vs Specific-to-general



Breadth-first vs depth-first



### Rule Generation

* Given a frequent itemset L, find all non-empty subsets f L such that f → L – f satisfies the minimum confidence requirement
* If {A,B,C,D} is a frequent itemset, candidate rules:

ABC → D, ABD → C, ACD → B, BCD → A,

A → BCD, B → ACD, C → ABD, D → ABC,

AB → CD, AC → BD, AD → BC, BC → AD,

BD → AC, CD → AB

* If |L| = k, then there are candidate association rules (ignoring L → ø and L → ø)

**How to efficiently generate rules from frequent itemsets?**

* In general, confidence does not have a monotone property:

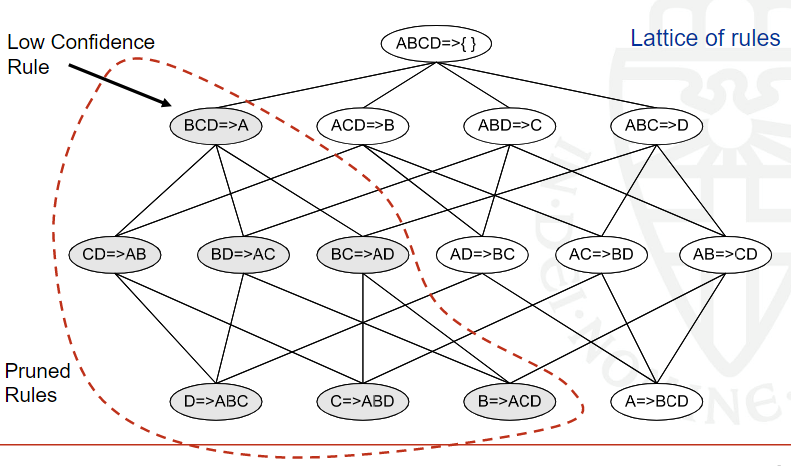
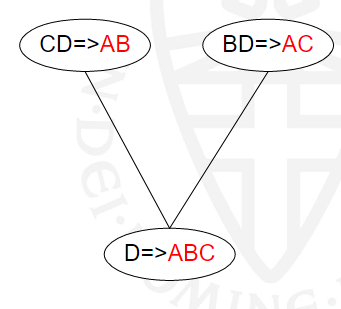
c(ABC → D) can be larger or smaller than c(AB → D)

* But confidence of rules generated from the same itemset does have a monotone property
* For example, L = {A,B,C,D}:

c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)

* Confidence is a decreasing function of the number of items on the RHS of the rule

#### Rule Generation for Apriori Algorithm



* Candidate rule is generated by merging two rules that share the same prefix in the rule consequent
* join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC
* Prune rule D=>ABC if its subset AD=>BC does not have high confidence

### Pattern Evaluation

* Association rule algorithms tend to produce too many rules
  + many of them are uninteresting or redundant
  + redundant if {A,B,C} → {D} and {A,B} → {D} have the same support and confidence
* Interestingness measures can be used to prune/rank the derived patterns
* In the original formulation of association rules, support and confidence are the only measures used

#### Application of Interestingness Measure

#### 

#### Computing Interestingness Measure

Given a rule X Y, information needed to compute rule interestingness can be obtained from a contingency table.

#### Drawback of Confidence

* Association rule: Tea → Coffee
* Confidence = P(Coffee|Tea) = 0.75
* But P(Coffee) = 0.9
* Although confidence is high, rule is uninteresting
* P(Coffee|) = 0.9375

### Statistical Independence

Population of 1000 students

* 600 students know how to swim (S)
* 700 students know how to bike (B)
* 420 students know how to swim and bike (S,B)
* P(S⋀B) = 420/1000 = 0.42
* P(S) ✕ P(B) = 0.6 0.7 = 0.42
* P(S⋀B) = P(S) ✕ P(B) => Statistical independence
* P(S⋀B) > P(S) ✕ P(B) => Positively correlated
* P(S⋀B) < P(S) ✕ P(B) => Negatively correlated

### Statistical-based Measures

Measures that take into account statistical dependence

Lift = also called Interest

PS = P(X, Y) - P(X)P(Y)

φ - coefficient =



***Example:***

* Association rule: Tea → Coffee
* Confidence = P(Coffee|Tea) = 0.75
* But P(Coffee) = 0.9
* Lift = 0.75/0.9= 0.8333 (< 1, therefore is negatively associated)

#### Drawback of Lift and Interest

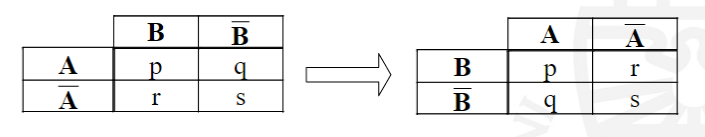
Not invariant under inversion operation (0 →1 and 1→0)

### Properties of A Good Measure

Piatetsky-Shapiro: 3 properties a good measure M must satisfy:

* M(A,B) = 0 if A and B are statistically independent
* M(A,B) increase monotonically with P(A,B) when P(A) and P(B) remain unchanged
* M(A,B) decreases monotonically with P(A) [or P(B)] when P(A,B) and P(B) [or P(A)] remain unchanged

#### Property under Variable Permutation



* Does M(A,B) = M(B,A)?
* Symmetric measures:

support, lift, collective strength, cosine, Jaccard, etc

* Asymmetric measures:

confidence, conviction, Laplace, J-measure, etc

#### Property under Row/Column Scaling

#### 𝝋 -Coefficient

* 𝝋 - coefficient is analogous to correlation coefficient for continuous variables
* Invariant under inversion operation
* Two binary variables are considered positively associated if most of the data falls along the diagonal cells.
* In contrast, two binary variables are considered negatively associated if most of the data falls off the diagonal.

### Support-based Pruning

* Definition: Support-based pruning is trimming a search space based on any support measure.
* Most of the association rule mining algorithms use support measure to prune rules and itemsets

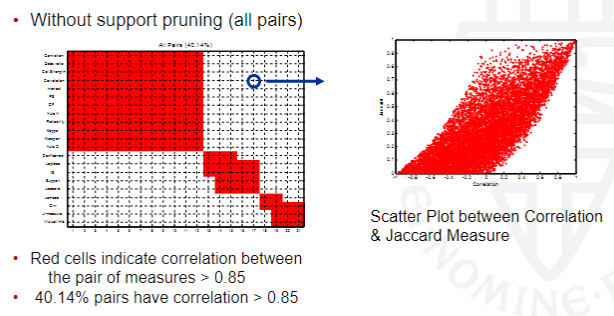
*How are we going to see this?* Study the effect of support pruning on correlation of itemsets

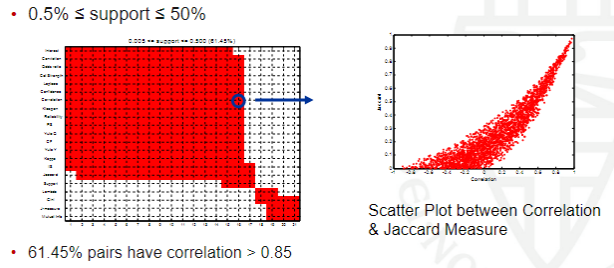
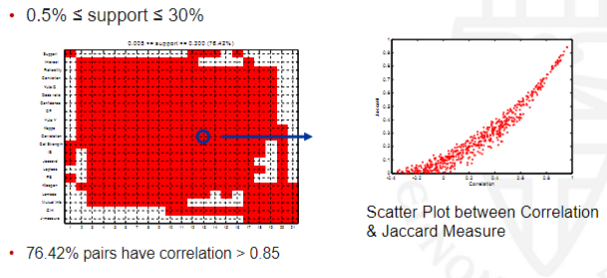
* Generate 10000 random contingency tables
* Compute support and pairwise correlation for each table
* Apply support-based pruning and examine the tables that are removed



How does support-based pruning affect other measures?

Similar to before we will:

* Generate 10000 contingency tables
* Rank each table according to the different measures
* Compute the pair-wise correlation between the measures



### Subjective Interestingness Measure

*Objective measure:*

* Rank patterns based on statistics computed from data
* e.g., 21 measures of association (support, confidence, Laplace, Gini, mutual information, Jaccard, etc).

*Subjective measure* (Silberschatz & Tuzhilin):

* Rank patterns according to user’s interpretation
* A pattern is subjectively interesting if it contradicts the expectation of a user
* A pattern is subjectively interesting if it is actionable

#### Interestingness via Unexpectedness

* Interestingness measures are used to select and rank patterns according to the interest it may have on a potential user (Better measures alsoinclude time and space costs for the mining process)
* Unexpectedness is a measure of a hidden pattern we did not see during the selection of a pattern (I think I can't really find a lot about this).

### Summar of Association Rule Mining

* Find potentially interesting association rules
* Main technical challenge is the computational complexity
* Various tricks to make this work in practice:
  + First search for frequent itemsets, only then for interesting rules
  + Monotonicity properties of support and confidence
  + Clever ways to compare transactions against candidate itemsets
* Many different measures of “interestingness”, typically highly correlated in practice
* Real challenge is to find associations that are surprising and actionable

# Chapter 5

## Alternative Techniques for Classification

### Rule-based Classifier

* Classify records by using a collection of “if...then...” rules
* Rule: (Condition) → y

Where:

* + Condition is a conjunctions of attributes
  + y is the class label
  + LHS: rule antecedent or condition
  + RHS: rule consequent
* Examples of classification rules:
* (Blood Type=Warm) ⴷ (Lay Eggs=Yes) → Birds
* (Taxable Income < 50K) ⴷ (Refund=Yes) → Evade=No

### Application of a Rule-based Classifier

* A rule r covers an instance x if the attributes of the instance satisfy the condition of the rule

R1: (Give Birth = no) ⴷ (Can Fly = yes) → Birds

R2: (Give Birth = no) ⴷ (Live in Water = yes) → Fishes

R3: (Give Birth = yes) ⴷ (Blood Type = warm) → Mammals

R4: (Give Birth = no) ⴷ (Can Fly = no) → Reptiles

R5: (Live in Water = sometimes) → Amphibians

### 

### Rule Coverage and Accuracy

*Coverage of a rule:*

Fraction (relative to the total number) of records that satisfy the antecedent of a rule

*Accuracy of a rule:*

Fraction (relative to those satisfying the antecedent) of records that satisfy both the antecedent and consequent of a rule

### Characteristics of Rule-based Classifiers

*Mutually exclusive rules*

* Classifier contains mutually exclusive rules if the rules are independent of each other
* Every record is covered by at most one rule

*Exhaustive rules*

* Classifier has exhaustive coverage if it accounts for every possible combination of attribute values
* Each record is covered by at least one rule

### From Decision Trees To Rules

### Effect of Rule Simplification

Rules are no longer mutually exclusive

* A record may trigger more than one rule
* Solution?
  + Ordered rule set or
  + Unordered rule set – use voting schemes

Rules are no longer exhaustive

* A record may not trigger any rules
* Solution? Use a default class

### Ordered Rule Set

Rules are rank ordered according to their priority

* An ordered rule set is known as a decision list

When a test record is presented to the classifier

* It is assigned to the class label of the highest ranked rule it has triggered
* If none of the rules fired, it is assigned to the default class



### Rule Ordering Schemes

*Rule-based ordering:* Individual rules are ranked based on their quality

*Class-based ordering:* Rules that belong to the same class appear together

### Building Classification Rules

*Direct method:*

* Extract rules directly from data
* e.g.: RIPPER, CN2, Holte’s 1R

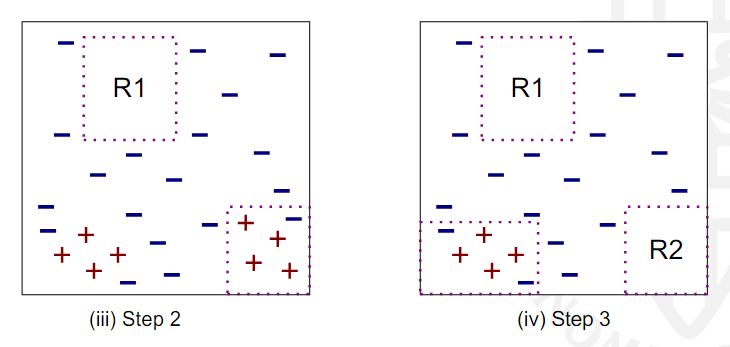
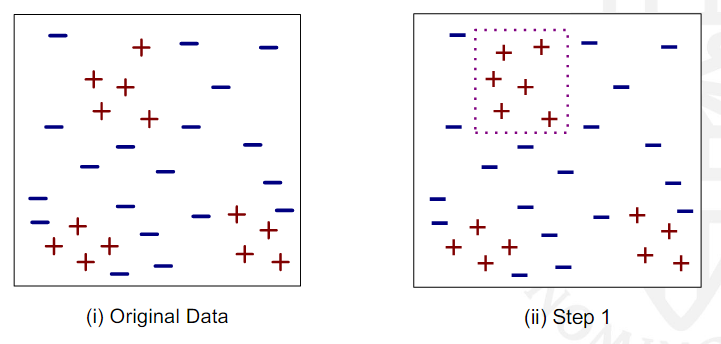
*Indirect method:*

* Extract rules from other classification models (e.g. decision trees, neural networks,).
* e.g: C4.5rules

### Direct Method: Sequential Covering

1. Start from an empty rule
2. Grow a rule using the Learn-One-Rule function
3. Remove training records covered by the rule
4. Repeat Step (2) and (3) until stopping criterion is met

Example:

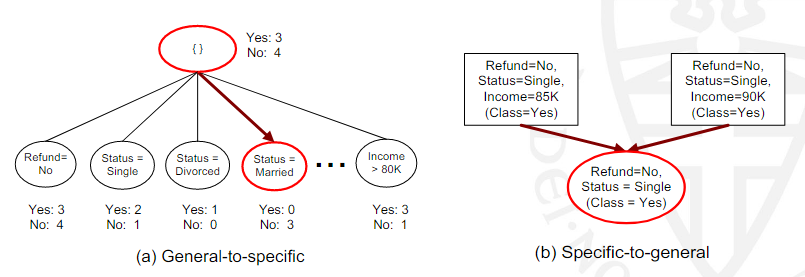


#### Aspects of Sequential Covering

* Rule Growing
* Instance Elimination
* Rule Evaluation
* Stopping Criterion
* Rule Pruning

### Rule Growing

Two common strategies



#### CN2

* Start from an empty conjunct: {}
* Greedily add conjuncts that minimize some entropy measure: {A}, {A,B}, ...
* Determine the rule consequent by taking majority class of instances covered by the rule

#### RIPPER

* Start from an empty rule: {} => class
* Add conjuncts that maximizes FOIL’s information gain measure:
  + R0: {A} => class (initial rule)
  + R1: {A Λ B} => class (rule after adding conjunct: more specific)
  + Gain (R0, R1) = t [ log (p1/(p1+n1)) – log (p0/(p0 + n0)) ]
  + Where

t: number of positive instances covered by both R0 and R1

p0: number of positive instances covered by R0

n0: number of negative instances covered by R0

p1: number of positive instances covered by R1

n1: number of negative instances covered by R1

(note: p1 ≤ p0 and n1 ≤ n0)

### Rule Evaluation Metrics

* Accuracy =
* Laplace =
* M-estimate =

### Stopping Criterion and Rule Pruning

*Stopping criterion*

* Compute the gain
* If gain is not significant, discard the new rule

*Rule Pruning*

* Similar to post-pruning of decision trees
* Reduced Error Pruning:
  + Remove one of the conjuncts in the rule
  + Compare error rate on validation set before and after pruning
  + If error improves, prune the conjunct

### Summary of Direct Method

* Grow a single rule
* Prune the rule (if necessary)
* Add rule to Current Rule Set
* Remove Instances from rule
* Repeat

#### RIPPER

* For 2-class problem, choose one of the classes as positive class, and the other as negative class
  + Learn rules for positive class
  + Negative class will be default class
* For multi-class problem
  + Order the classes according to increasing class prevalence (fraction of instances that belong to a particular class)
  + Learn the rule set for smallest class first, treat the rest as negative class
  + Repeat with next smallest class as positive class

#### RIPPER: Growing a Rule

* Start from empty rule
* Add conjuncts as long as they improve FOIL’s information gain
* Stop when rule no longer covers negative examples
* Prune the rule immediately using incremental reduced error pruning
* Measure for pruning: v = (p-n)/(p+n)
  + p: number of positive examples covered by the rule in the validation set
  + n: number of negative examples covered by the rule in the validation set
* Pruning method: delete conjunct if without it v is higher on a validation set

#### RIPPER: Building a Rule Set

* Use sequential covering algorithm
  + Finds the best rule that covers the current set of positive examples
  + Eliminate both positive and negative examples covered by the rule
* Each time a rule is added to the rule set, compute the new description length
  + stop adding new rules when the new description length is d bits (default: 64) longer than the smallest description length obtained so far

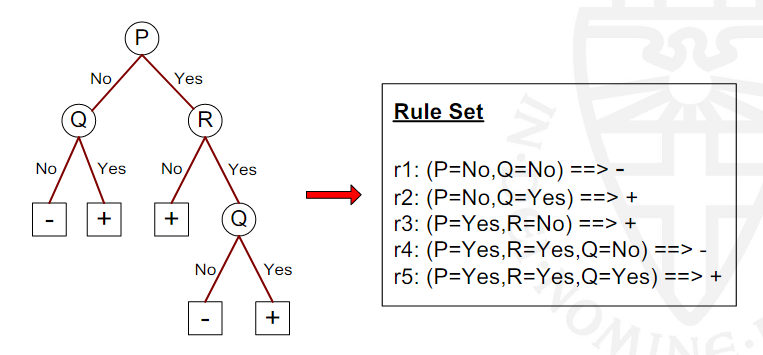
#### RIPPER: Optimize the Rule Set

For each rule r in the rule set R

* Consider 2 alternative rules:
  + Replacement rule (r\*): grow new rule from scratch
  + Revised rule (r’): add conjuncts to extend the rule r
* Compare the rule set for r against the rule set for r\* and r’
* Choose rule set that minimizes the description length

Repeat rule generation and rule optimization for the remaining positive examples

### Indirect Methods



#### C4.5rules

* Extract rules from an unpruned decision tree
* For each rule, r: A → y,
  + consider an alternative rule r’: A’ → y where A’ is obtained by removing one of the conjuncts in A
  + Compare the pessimistic error rate for r against all r’-s
  + Prune if one of the r’-s has lower pessimistic error rate
  + Repeat until we can no longer improve generalization error
* Use class-based ordering: order subsets of rules that correspond to the same class

#### C4.5 versus C4.5rules versus RIPPER

#### Advantages of Rule-Based Classifiers

* As highly expressive as decision trees
* Easy to interpret
* Easy to generate
* Can classify new instances rapidly
* Performance comparable to decision trees

### Instance-Based Classifiers

Instance-based learning is a family of learning algorithms that, instead of performing explicit generalization, compare new problem instances with instances seen in training, which have been stored in memory.

#### Examples

*Rote-learner*

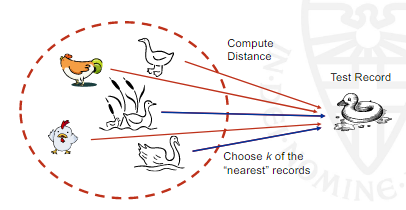
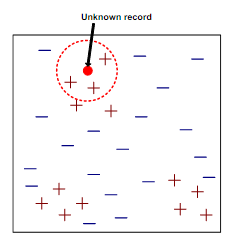
* Memorizes entire training data and performs classification only if attributes of record match one of the training examples exactly

*Nearest neighbor*

* Uses k “closest” points (nearest neighbors) for performing classification

#### Nearest Neighbor Classifiers

If it walks like a duck, quacks like a duck, then it’s probably a duck



For Nearest-Nieghbour classifiers we have some things we require:

1. The set of stored records (Lazy-learning algorithm)

2. A distance metric to compute distance between records

3. 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)

##### Definition of K-nearest Neighbour

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

#### Nearest Neighbor Classification

* Compute distance between two points, e.g., using Euclidean distance:
* 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, e.g., with weight factor,

##### Choice of k

* If k is too small, sensitive to noise points
* If k is too large, neighborhood may include points from other classes

##### 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
* Can also work with other distance measures

##### Lazy Learning

*Definition*: Lazy Learning is simply an algorithm that generalizes the data AFTER a query is made. We store the data points and then use it when a query is made.

* “Lazy learners” do not build models explicitly
* To be contrasted with “eager learners” such as decision tree induction, rule-based systems, neural networks, ...
* Classifying unknown records is relatively expensive

### Bayes Classifier

A probabilistic framework for solving classification problems (Minimizes the probability of misclassification).

* Conditional probability:
* Bayes Theorem:

#### Example

Given:

* A doctor knows that meningitis causes stiff neck 50% of the time
* People without meningitis have a stiff neck with probability 1/20
* Prior probability of any patient having meningitis is 1/50,000

If a patient has a stiff neck, what’s the probability he/she has meningitis?

#### Bayesian Classifiers

* Consider each attribute and class label as random variables
* Given a record with attributes (, ,...,)
  + Goal is to predict class C
  + Specifically, we want to find the value of C that maximizes P(C|, ,...,)
* Can we estimate P(C|, ,...,) directly from data?

*Approach*:

* Compute the posterior probability P(C|, ,...,) for all values of C using Bayes theorem
* Choose the value of C that maximizes P(C|, ,...,)
* Equivalent to choosing value of C that maximizes P(C|, ,...,)P(C)
* How to estimate P(C|, ,...,)?

#### Naïve Bayes Classifier

Assume independence among attributes Ai when class is given:

* Can estimate for all and C from training data.
* New point is classified to C=j if is maximal.
* This handles both continuous and discrete data and is highly scalable.

### How to Estimate Probabilities from Data: Discrete Data

* Class:
* e.g., P(No) = 7/10,

P(Yes) = 3/10

* For discrete attributes:

* where is the number of instances having attribute and belonging to class k
* Examples:

P(Status = Married | No) = 4/7

P(Homeowner = Yes | Yes) = 0

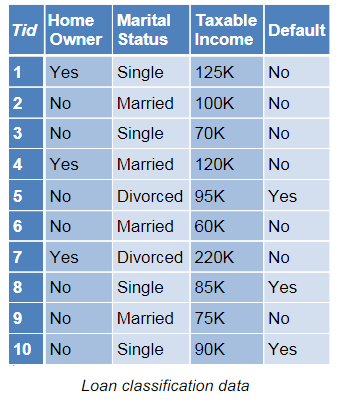
### How to Estimate Probabilities from Data: Continuous Data

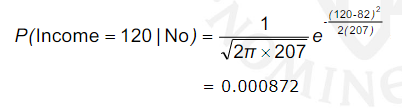
* Discretize the range into bins (Trying to view the continuous data as discrete data almost)
  + One ordinal attribute per bin
  + May violate the independence assumption...
* Two-way split: (A < v) or (A > v)
  + Choose one of the two splits as new attribute
* Probability density estimation:
  + Assume attribute follows, for example: a normal distribution
  + Use data to estimate parameters of distribution (e.g., mean and standard deviation)
  + Once probability distribution is known, use it to estimate the conditional probability P( |C)

### How to Estimate Probabilities from Data?

Normal distribution:

One for each pair.

* For (Income, Home Owner = No):
  + If Home Owner = No:
    - Sample mean = 82
    - Sample variance = 207





#### Example of Naïve Bayes Classifier

Given a Test Record: X = (HomeOwner = No, Married, Income = 120K)

* P(X | Class=No) = P(Owner=No | Class=No)

⨉ P(Married | Class=No)

⨉ P(Income=120K | Class=No)

= 4/7 ́ 4/7 ́ 0.0072 = 0.0024

* P(X|Class=Yes) = P(Owner=No | Class=Yes)

⨉ P(Married | Class=Yes)

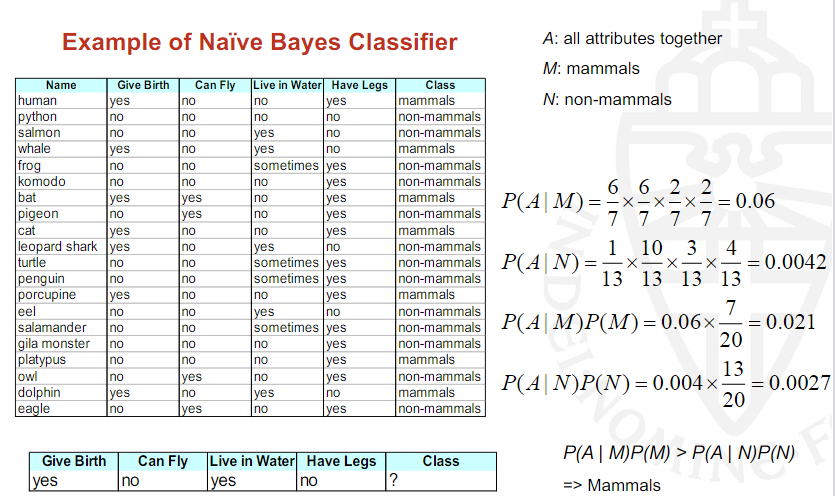
⨉ P(Income=120K | Class=Yes)

= 1 ⨉ 0 ⨉ 1.2e-9 = 0

Since P(X | No)P(No) > P(X | Yes)P(Yes)

Therefore P(No | X) > P(Yes | X)

=> Class = No



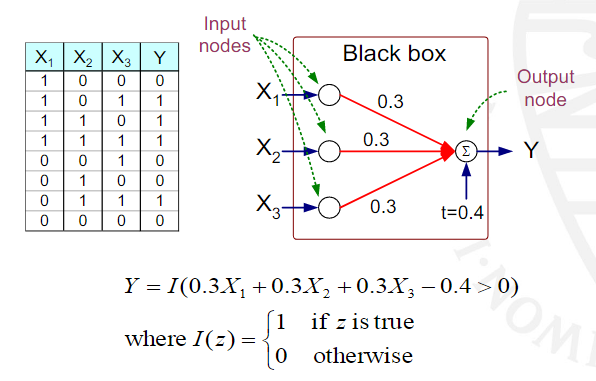
### Naïve Bayes Classifier: Handling Zeroes

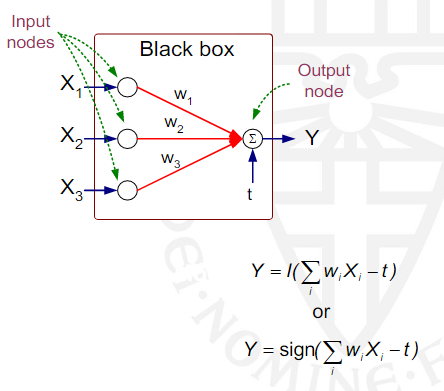
* If one of the conditional probabilities is zero, then the entire expression becomes zero...
* Probability estimation:

### Naïve Bayes Summary

* Robust to isolated noise points.
* Handle missing values by ignoring the instance during probability estimate calculations.
* Robust to irrelevant attributes.
* Conditional independence assumption may not hold for some attributes
  + Use other techniques such as Bayesian networks to model the probability distribution of the attributes given the class

## Artificial Neural Networks (ANN)





* Model is an assembly of inter-connected nodes and weighted links
* Output node sums up each of its input value according to weights of its links
* Compare output node against some threshold t

### General Structure of ANN

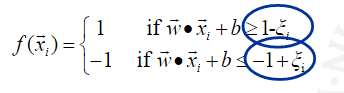
* Initialize the weights (, , ..., )
* Adjust the weights in such a way that the output of ANN is consistent with the class labels of the training examples
  + Objective function:
  + Find the weights that minimize the above objective function e.g., using the so-called backpropagation algorithm
* Revival of “deep learning”: clever ways to initialize the weights in combination with lots of data to learn huge neural networks

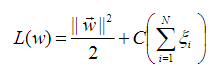
### Support Vector Machines

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

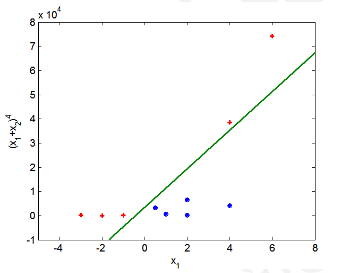
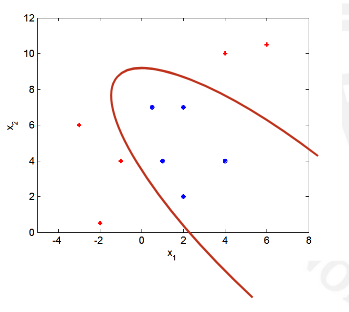
### Support Vector Machines: Constrained Optimization

### Support Vector Machines: Nonlinearly Separable Case

* Introduce “slack” variables ξ to allow for errors
* Constraints now become
* Still want to minimize the norm, but also the amount of errors, so now:



## Nonlinear Support Vector Machines

* Transform data into higher dimensional space and apply a linear SVM in this space
* The so-called “kernel trick” makes the transformation implicit instead of explicit
* Transformation back to the original space gives a nonlinear boundary

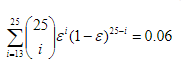
### Ensemble Methods

* Construct a set of classifiers from the training data
* Predict class label of previously unseen records by aggregating predictions made by multiple classifiers
* Examples: bagging and boosting

#### General Idea

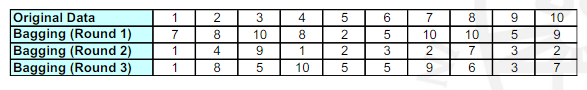
#### Why does it work?

* Suppose there are 25 base classifiers
  + Each classifier has an error rate, ε = 0.35
  + Assume classifiers are independent
  + Probabiliy that ensembles classifier makes a wrong prediection:



* Catch: even classifiers trained on different parts of the data may be clearly dependent
* Basic argument especially applies for unstable classifiers such as decision trees and neural networks

### Bagging

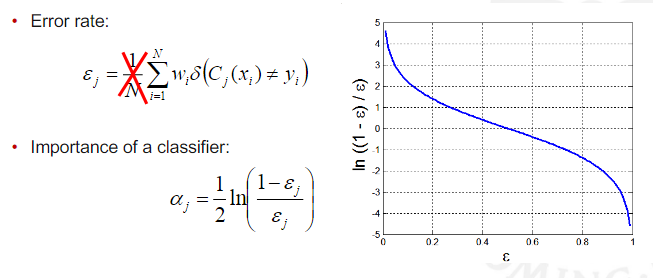
* Sampling with replacement
* Build classifier on each bootstrap sample
* Each same has probability for large N of being selected at least once

### Boosting

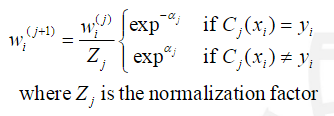
* An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
* Initially, all N records are assigned equal weights
* Unlike bagging, weights may change at the end of boosting round
  + Records that are wrongly classified will have their weights increased
  + Records that are classified correctly will have their weights decreased

### AdaBoost

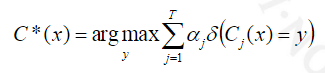
* Base classifiers:



* Weight update:



* Fallback: if any intermediate rounds produce error rate higher than 50%, the weights are reverted back 1/N and the resampling procedure is repeated.
* Classification:



#### Illustrating AdaBoost

