Introduction to Big Data Methods

SS 2016

Prof. Dr. Xiaoming Fu

Acknowledgement: Bing Liu (UIC)

Related Concepts

- Machine learning
  - Use *known* properties learned from the *training data* to predict
- Data mining
  - Discover the unknown properties on the data
- Data analytics
  - Apply a mechanical or algorithmic process to derive the insights, e.g. running through several datasets for correction (data fusion: cross different domains)
- Data science
  - A combination of math, statistics, programming, the domain knowledge, data collection & cleaning etc.

What to do with big data?

- Aggregation and Statistics
  - Data warehouse and OLAP
- Indexing, Searching, and Querying
  - Keyword based search
  - Pattern matching (XML/RDF)
- **Knowledge discovery**
  - **Data Preprocessing**
  - **Statistical Modeling**
  - **Data Mining**
    - Unsupervised learning
    - Supervised learning

Statistics basis

Random Sample and Statistics

- *Population: is used to refer to the set or universe of all* entities under study.
- However, looking at the entire population may not be feasible, or may be too expensive.
- Instead, we draw a random sample from the population, and compute appropriate *statistics* from the sample, that give estimates of the corresponding population parameters of interest.

5.9	6.9	6.6	4.6	6.0	4.7	6.5	5.8	6.7	6.7	5.1	5.1	5.7	6.1	4.9
5.0	5.0	5.7	5.0	7.2	5.9	6.5	5.7	5.5	4.9	5.0	5.5	4.6	7.2	6.8
5.4	5.0	5.7	5.8	5.1	5.6	5.8	5.1	6.3	6.3	5.6	6.1	6.8	7.3	5.6
4.8	7.1	5.7	5.3	5.7	5.7	5.6	4.4	6.3	5.4	6.3	6.9	7.7	6.1	5.6
6.1	6.4	5.0	5.1	5.6	5.4	5.8	4.9	4.6	5.2	7.9	7.7	6.1	5.5	4.6
4.7	4.4	6.2	4.8	6.0	6.2	5.0	6.4	6.3	6.7	5.0	5.9	6.7	5.4	6.3
4.8	4.4	6.4	6.2	6.0	7.4	4.9	7.0	5.5	6.3	6.8	6.1	6.5	6.7	6.7
4.8	4.9	6.9	4.5	4.3	5.2	5.0	6.4	5.2	5.8	5.5	7.6	6.3	6.4	6.3
5.8	5.0	6.7	6.0	5.1	4.8	5.7	5.1	6.6	6.4	5.2	6.4	7.7	5.8	4.9
5.4	5.1	6.0	6.5	5.5	7.2	6.9	6.2	6.5	6.0	5.4	5.5	6.7	7.7	5.1

Table 1.2: Iris Dataset: `sepal length`

Statistic

- Let Si denote the random variable corresponding to data point xi , then a *statistic* $\hat{\theta}$ is a function $\hat{\theta} : (S1, S2, \cdots, Sn) \rightarrow R$.

- If we use the value of a statistic to estimate a population parameter, this value is called a *point estimate of the parameter, and the statistic is called as an estimator of the* parameter.

Empirical Cumulative Distribution Function

$$\hat{F}(x) = \frac{\sum_{i=1}^{n} I(S_i \leq x)}{n}$$

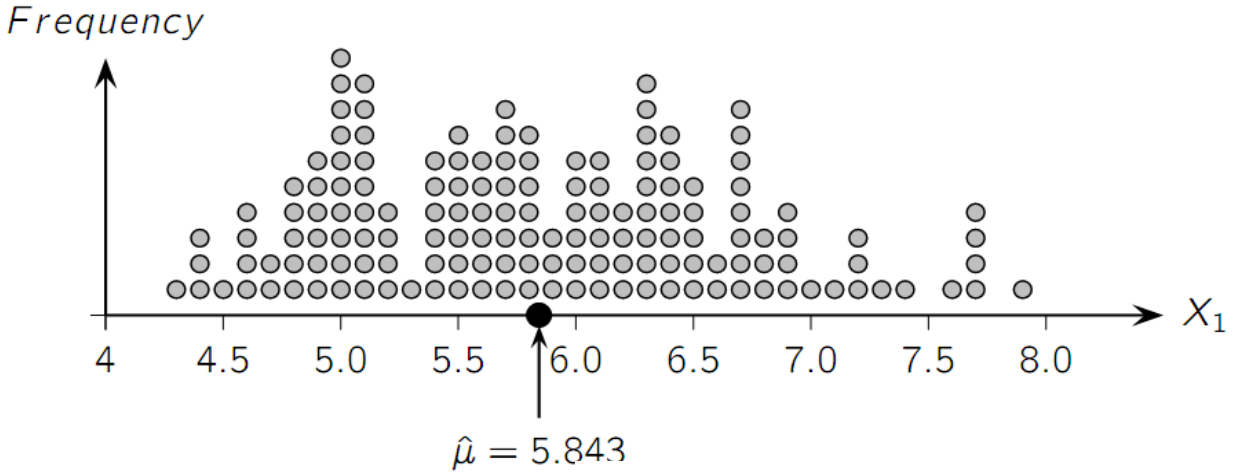
Where

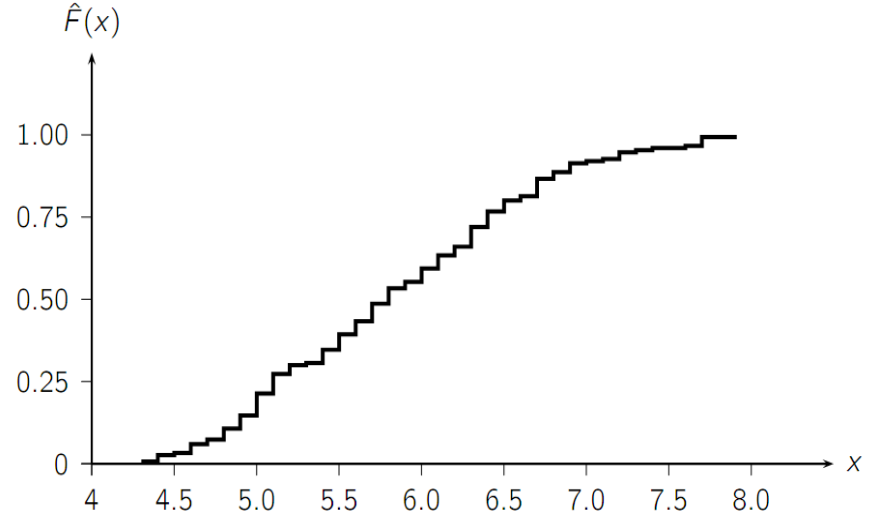
$$I(S_i \leq x) = \begin{cases} 1 & \text{if } S_i \leq x \\ 0 & \text{if } S_i > x \end{cases}$$

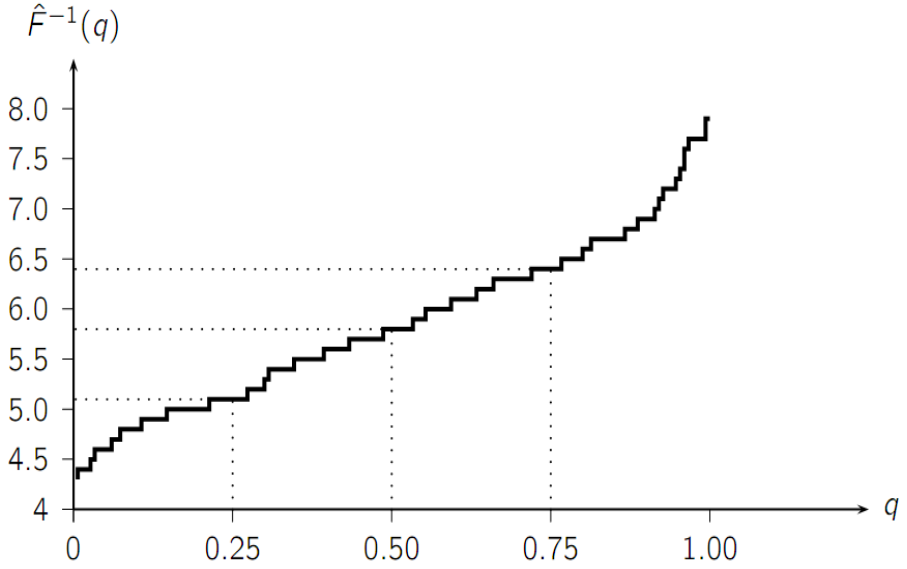
Inverse Cumulative Distribution Function

$$F^{-1}(q) = \min\{x : F(x) > q\} \qquad \text{for } q \in [0, 1]$$

Example


Frequency
4
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
X_1
$\hat{\mu} = 5.843$


$\hat{F}(x)$
1.00
0.75
0.50
0.25
0
4
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
x


$\hat{F}^{-1}(q)$
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4
0
0.25
0.50
0.75
1.00
q

Measures of Central Tendency (Mean)

Population Mean:

$$\mu = E[X] = \sum_x x\, f(x)$$

$$\mu = E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

Sample Mean (Unbiased, not robust):

$$\hat{\mu} = \sum_x x \hat{f}(x) = \sum_x x \left(\frac{\sum_{i=1}^n I(S_i = x)}{n} \right) = \frac{\sum_{i=1}^n S_i}{n}$$

$$E[\hat{\mu}] = E\left[\frac{\sum_{i=1}^n S_i}{n} \right] = \frac{1}{n} \sum_{i=1}^n E[S_i] = \frac{1}{n} \sum_{i=1}^n \mu = \mu$$

Measures of Central Tendency (Median)

Population Median:

$$P(X \leq m) \geq \frac{1}{2} \text{ and } P(X \geq m) \geq \frac{1}{2}$$

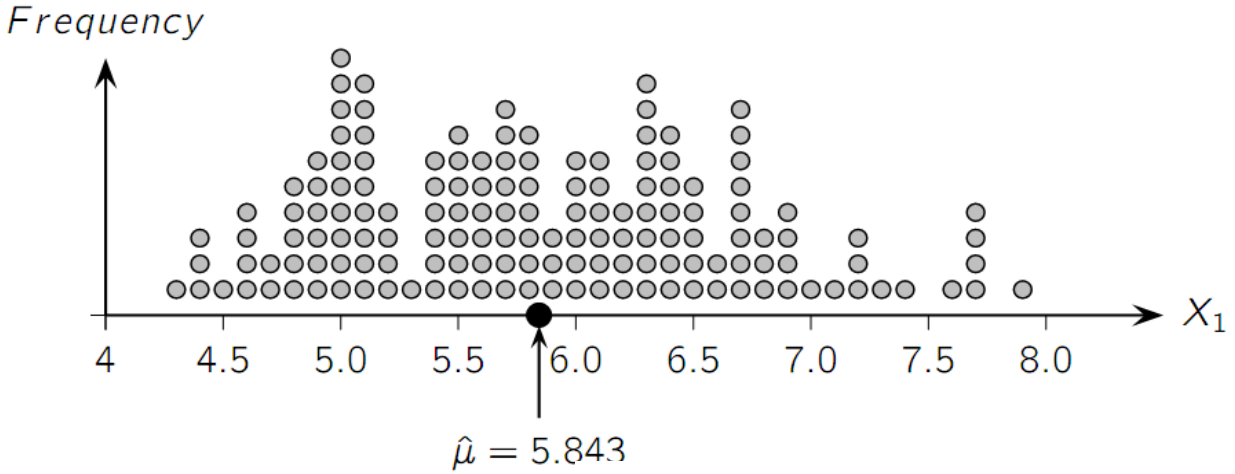
or

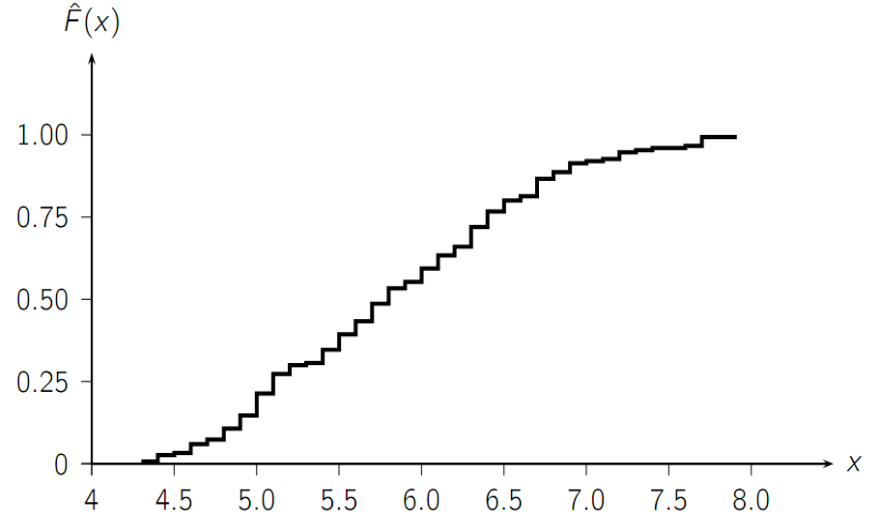
$$F(m) = 0.5 \text{ or } m = F^{-1}(0.5)$$

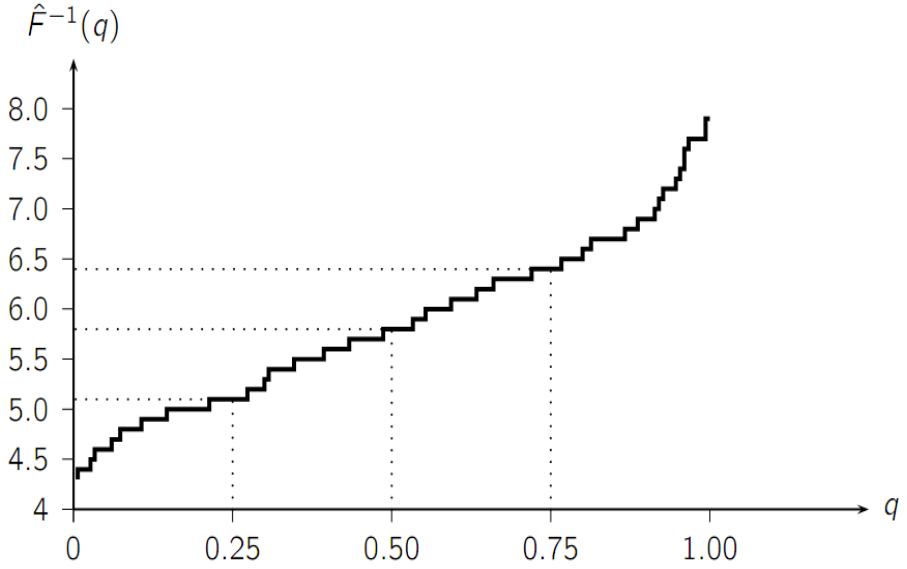
Sample Median:

$$\hat{F}(m) = 0.5 \text{ or } m = \hat{F}^{-1}(0.5)$$

Example


Frequency
4
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
X_1
$\hat{\mu} = 5.843$


$\hat{F}(x)$
1.00
0.75
0.50
0.25
0
4
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
x


$\hat{F}^{-1}(q)$
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4
0
0.25
0.50
0.75
1.00
q

Measures of Dispersion (Range)

Range:

$$r = \max_x\{x\} - \min_x\{x\}$$

Sample Range:

$$\hat{r} = \max_i\{S_i\} - \min_i\{S_i\} = \max_i\{x_i\} - \min_i\{x_i\}$$

- Not robust, sensitive to extreme values

Measures of Dispersion (Inter-Quartile Range)

Inter-Quartile Range (IQR):

$$IQR = F^{-1}(0.75) - F^{-1}(0.25)$$

Sample IQR:

$$\widehat{IQR} = \hat{F}^{-1}(0.75) - \hat{F}^{-1}(0.25)$$

- More robust

Measures of Dispersion (Variance and Standard Deviation)

Variance:

$$var(X) = E[(X-\mu)^2] = \begin{cases} \sum_x (x-\mu)^2 \, f(x) & \text{if } X \text{ is discrete} \\ \int_{-\infty}^{\infty} (x-\mu)^2 \, f(x) \, dx & \text{if } X \text{ is continuous} \end{cases}$$

Standard Deviation:

$$\begin{aligned} \sigma^2 = var(X) &= E[(X-\mu)^2] = E[X^2 - 2\mu X + \mu^2] \\ &= E[X^2] - 2\mu E[X] + \mu^2 = E[X^2] - 2\mu^2 + \mu^2 \\ &= E[X^2] - (E[X])^2 \end{aligned}$$

Measures of Dispersion (Variance and Standard Deviation)

Variance:

$$var(X) = E[(X-\mu)^2] = \begin{cases} \sum_x (x-\mu)^2 f(x) & \text{if } X \text{ is discrete} \\ \int_{-\infty}^{\infty} (x-\mu)^2 f(x)\, dx & \text{if } X \text{ is continuous} \end{cases}$$

Standard Deviation:

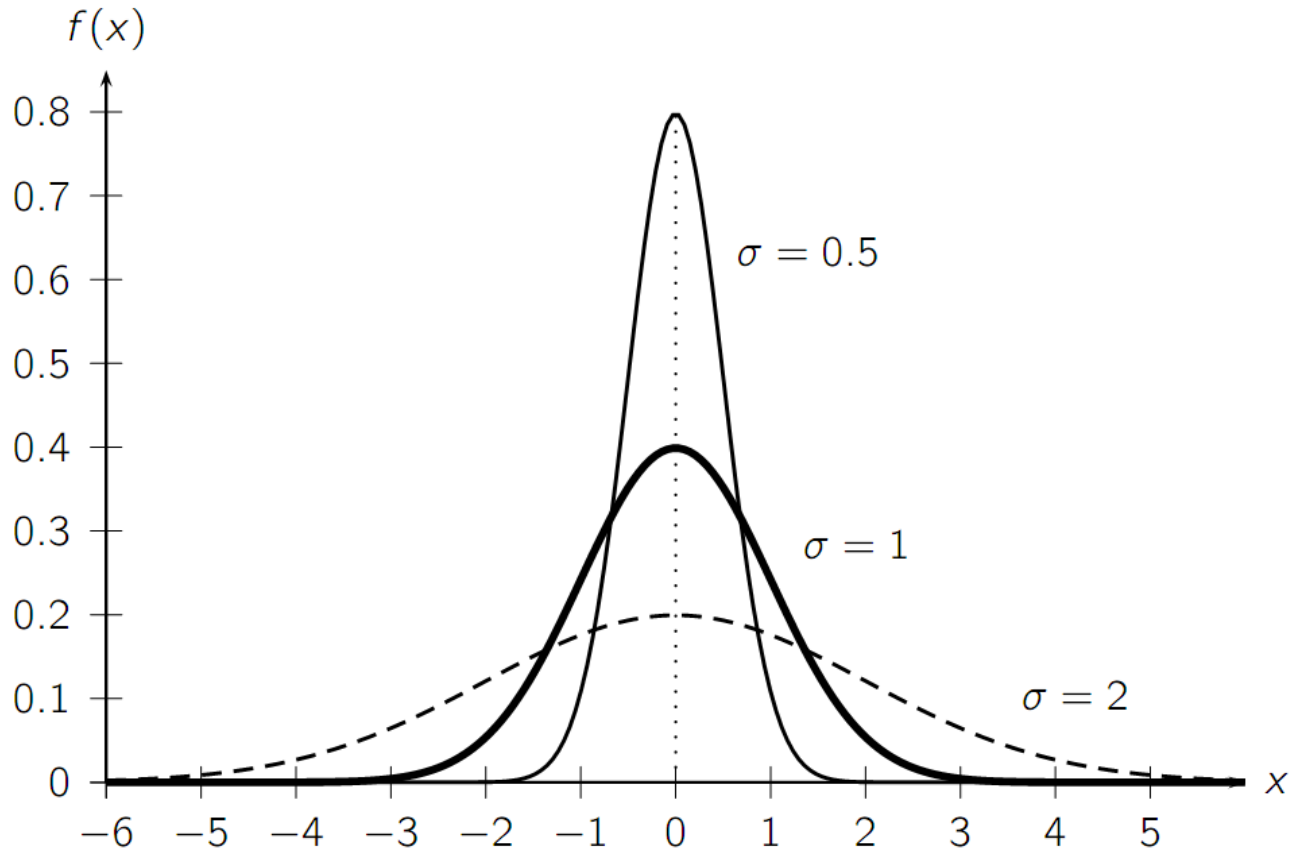
$$\begin{aligned} \sigma^2 = var(X) &= E[(X-\mu)^2] = E[X^2 - 2\mu X + \mu^2] \\ &= E[X^2] - 2\mu E[X] + \mu^2 = E[X^2] - 2\mu^2 + \mu^2 \\ &= E[X^2] - (E[X])^2 \end{aligned}$$

Sample Variance & Standard Deviation:

$$\hat{\sigma}^2 = \sum_x (x-\hat{\mu})^2 \hat{f}(x) = \sum_x (x-\hat{\mu})^2 \left(\frac{\sum_{i=1}^n I(S_i = x)}{n} \right) = \frac{\sum_{i=1}^n (S_i - \hat{\mu})^2}{n}$$

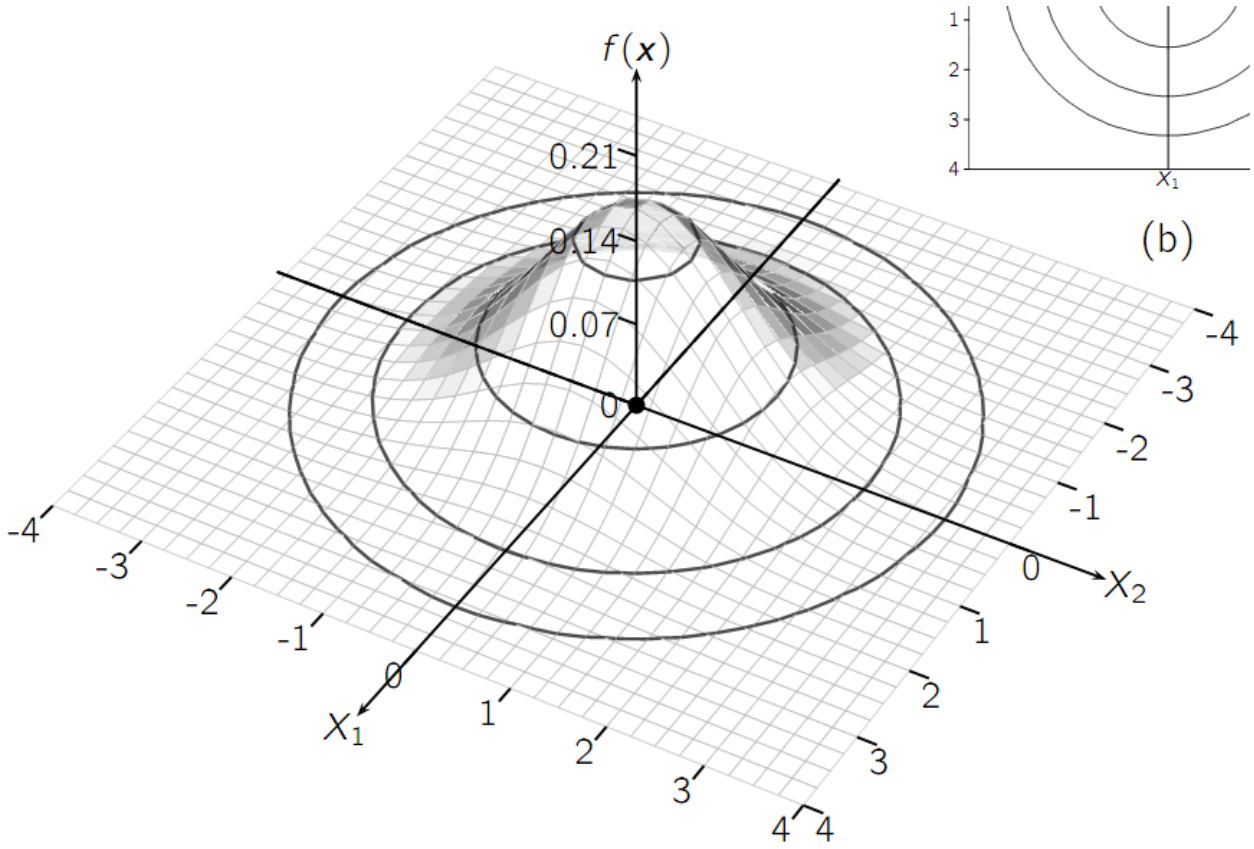
Univariate Normal Distribution

$$f(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$



Multivariate Normal Distribution

$$f(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(\sqrt{2\pi})^d \sqrt{|\boldsymbol{\Sigma}|}} \exp\left\{ -\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}{2} \right\}$$

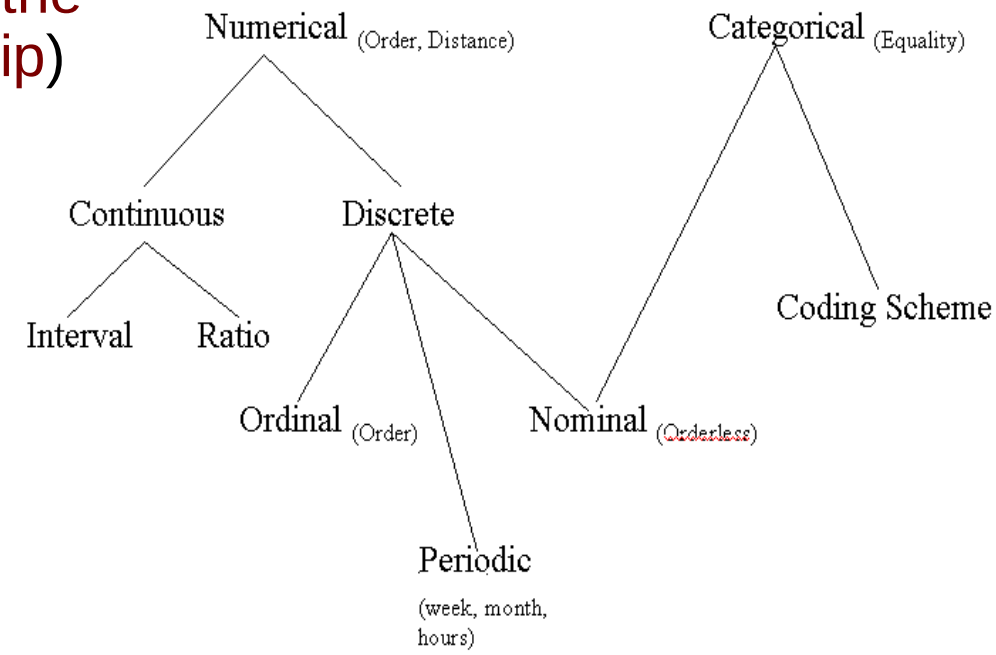


Data Preprocessing

Data Types and Forms

- Attribute-value data:
- Data types
  - numeric, categorical (see the hierarchy for its relationship)
  - static, dynamic (temporal)
- Other kinds of data
  - distributed data
  - text, Web, meta data
  - images, audio/video

A1	A2	…	An	C



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization
- Summary

Why Data Preprocessing?

- Data in the real world is dirty
  - incomplete: missing attribute values, lack of certain attributes of interest, or containing only aggregate data
    - e.g., occupation=“”
  - noisy: containing errors or outliers
    - e.g., Salary=“-10”
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age=“42” Birthday=“03/07/1997”
    - e.g., Was rating “1,2,3”, now rating “A, B, C”
    - e.g., discrepancy between duplicate records

Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
    - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
- Data preparation, cleaning, and transformation comprises the majority of the work in a data mining application (90%).

Multi-Dimensional Measure of Data Quality

- A well-accepted multi-dimensional view:
  - Accuracy
  - Completeness
  - Consistency
  - Timeliness
  - Believability
  - Value added
  - Interpretability
  - Accessibility

Major Tasks in Data Preprocessing

- Data cleaning
  - Fill in missing values, smooth noisy data, identify or remove outliers and noisy data, and resolve inconsistencies
- Data integration
  - Integration of multiple databases, or files
- Data transformation
  - Normalization and aggregation
- Data reduction
  - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization (for numerical data)

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization
- Summary

Data Cleaning

- Importance
  - “Data cleaning is the number one problem in data warehousing”
- Data cleaning tasks
  - Fill in missing values
  - Identify outliers and smooth out noisy data
  - Correct inconsistent data
  - Resolve redundancy caused by data integration

Missing Data

- Data is not always available
  - E.g., many tuples have no recorded values for several attributes, such as customer income in sales data
- Missing data may be due to
  - equipment malfunction
  - inconsistent with other recorded data and thus deleted
  - data not entered due to misunderstanding
  - certain data may not be considered important at the time of entry
  - not register history or changes of the data

How to Handle Missing Data?

- Ignore the tuple
- Fill in missing values manually: tedious + infeasible?
- Fill in it automatically with
  - a global constant : e.g., “unknown”, a new class?!
  - the attribute mean
  - the most probable value: inference-based such as Bayesian formula, decision tree, or EM algorithm

Noisy Data

- Noise: random error or variance in a measured variable.
- Incorrect attribute values may due to
  - faulty data collection instruments
  - data entry problems
  - data transmission problems
  - etc
- Other data problems which requires data cleaning
  - duplicate records, incomplete data, inconsistent data

How to Handle Noisy Data?

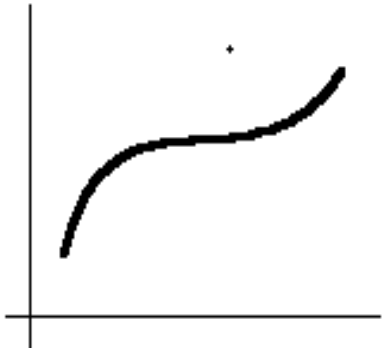
- Binning method:
  - first sort data and partition into (equi-depth) bins
  - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Clustering
  - detect and remove outliers
- Combined computer and human inspection
  - detect suspicious values and check by human (e.g., deal with possible outliers)

Binning Methods for Data Smoothing

- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
- Partition into (equi-depth) bins:
  - Bin 1: 4, 8, 9, 15
  - Bin 2: 21, 21, 24, 25
  - Bin 3: 26, 28, 29, 34
- Smoothing by bin means:
  - Bin 1: 9, 9, 9, 9
  - Bin 2: 23, 23, 23, 23
  - Bin 3: 29, 29, 29, 29
- Smoothing by bin boundaries:
  - Bin 1: 4, 4, 4, 15
  - Bin 2: 21, 21, 25, 25
  - Bin 3: 26, 26, 26, 34

Outlier Removal

- Data points inconsistent with the majority of data
- Different outliers
  - Valid: CEO's salary,
  - Noisy: One's age = 200, widely deviated points
- Removal methods
  - Clustering
  - Curve-fitting
  - Hypothesis-testing with a given model



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization
- Summary

Data Integration

- Data integration:
  - combines data from multiple sources
- Schema integration
  - integrate metadata from different sources
  - Entity identification problem: identify real world entities from multiple data sources, e.g., A.cust-id ≡ B.cust-#
- Detecting and resolving data value conflicts
  - for the same real world entity, attribute values from different sources are different, e.g., different scales, metric vs. British units
- Removing duplicates and redundant data

Data Transformation

- Smoothing: remove noise from data
- Normalization: scaled to fall within a small, specified range
- Attribute/feature construction
  - New attributes constructed from the given ones
- Aggregation: summarization
- Generalization: concept hierarchy climbing

Data Transformation: Normalization

- min-max normalization

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

- z-score normalization

$$v' = \frac{v - mean_A}{stand\_dev_A}$$

- normalization by decimal scaling

$$v' = \frac{v}{10^j}$$ Where j is the smallest integer such that Max($|v'|$)<1

Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization
- Summary

Data Reduction Strategies

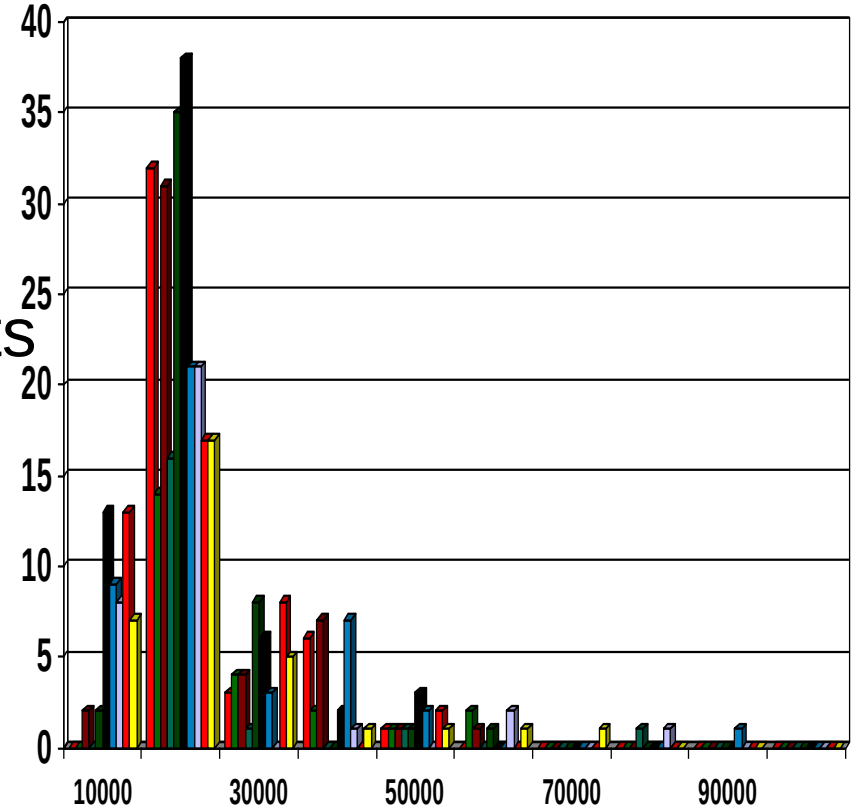
- Data is too big to work with
- Data reduction
  - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- Data reduction strategies
  - Dimensionality reduction — remove unimportant attributes
  - Aggregation and clustering
  - Sampling

Dimensionality Reduction

- Feature selection (i.e., attribute subset selection):
  - Select a minimum set of attributes (features) that is sufficient for the data mining task.
- Heuristic methods (due to exponential # of choices):
  - step-wise forward selection
  - step-wise backward elimination
  - combining forward selection and backward elimination
  - etc

Histograms

- A popular data reduction technique
- Divide data into buckets and store average (sum) for each bucket



Clustering

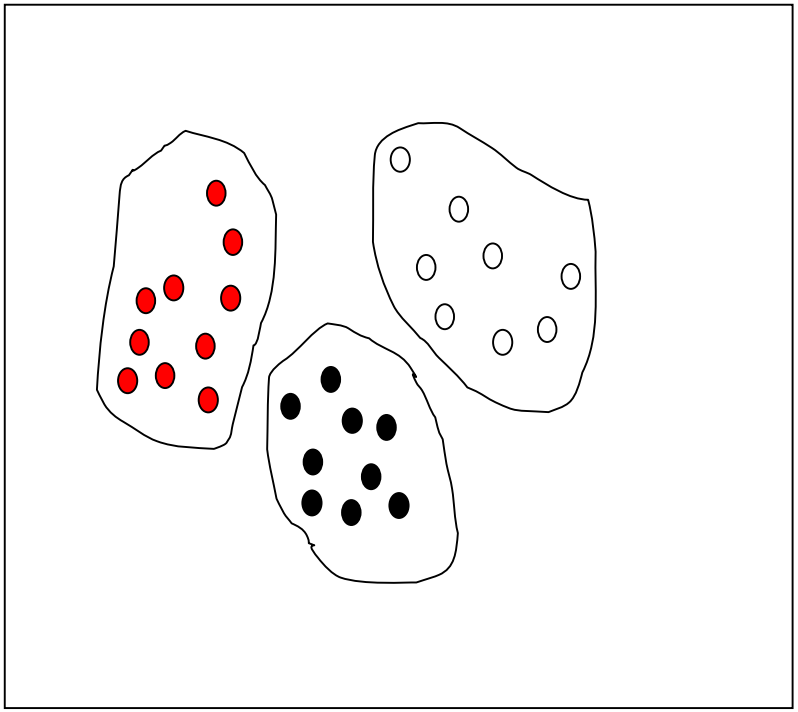
- Partition data set into clusters, and one can store cluster representation only
- Can be very effective if data is clustered but not if data is “smeared”
- There are many choices of clustering definitions and clustering algorithms. We will discuss them later.

Sampling

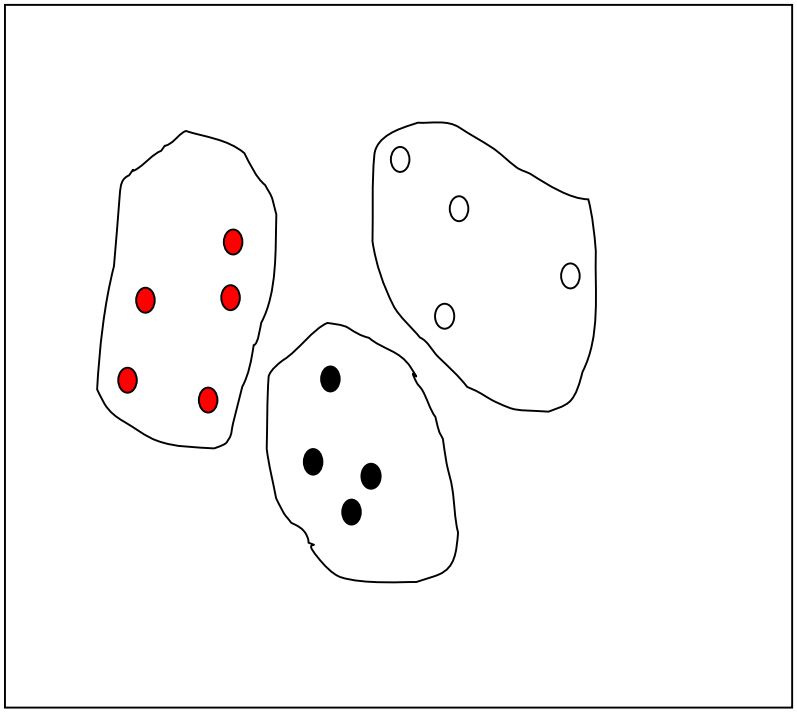
- Choose a representative subset of the data
  - Simple random sampling may have poor performance in the presence of skew.
- Develop adaptive sampling methods
  - Stratified sampling:
    - Approximate the percentage of each class (or subpopulation of interest) in the overall database
    - Used in conjunction with skewed data

Sampling

Raw Data



Cluster/Stratified Sample



Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization
- Summary

Discretization

- Three types of attributes:
  - Nominal — values from an unordered set
  - Ordinal — values from an ordered set
  - Continuous — real numbers
- Discretization:
  - divide the range of a continuous attribute into intervals because some data mining algorithms only accept categorical attributes.
- Some techniques:
  - Binning methods – equal-width, equal-frequency
  - Entropy-based methods

Discretization and Concept Hierarchy

- Discretization
  - reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals. Interval labels can then be used to replace actual data values
- Concept hierarchies
  - reduce the data by collecting and replacing low level concepts (such as numeric values for the attribute age) by higher level concepts (such as young, middle-aged, or senior)

Binning

- Attribute values (for one attribute e.g., age):
  - 0, 4, 12, 16, 16, 18, 24, 26, 28
- Equi-width binning – for bin width of e.g., 10:
  - Bin 1: 0, 4 [-,10) bin
  - Bin 2: 12, 16, 16, 18 [10,20) bin
  - Bin 3: 24, 26, 28 [20,+) bin
  - – denote negative infinity, + positive infinity
- Equi-frequency binning – for bin density of e.g., 3:
  - Bin 1: 0, 4, 12 [-, 14) bin
  - Bin 2: 16, 16, 18 [14, 21) bin
  - Bin 3: 24, 26, 28 [21,+] bin

Entropy-based (1)

- Given attribute-value/class pairs:
  - (0,P), (4,P), (12,P), (16,N), (16,N), (18,P), (24,N), (26,N), (28,N)
- Entropy-based binning via binarization:
  - Intuitively, find best split so that the bins are as pure as possible
  - Formally characterized by maximal information gain.
- Let S denote the above 9 pairs, p=4/9 be fraction of P pairs, and n=5/9 be fraction of N pairs.
- Entropy(S) = - p log p - n log n.
  - Smaller entropy – set is relatively pure; smallest is 0.
  - Large entropy – set is mixed. Largest is 1.

Entropy-based (2)

- Let v be a possible split. Then S is divided into two sets:
  - S1: value <= v and S2: value > v
- Information of the split:
  - I(S1,S2) = (|S1|/|S|) Entropy(S1)+ (|S2|/|S|) Entropy(S2)
- Information gain of the split:
  - Gain(v,S) = Entropy(S) – I(S1,S2)
- **Goal:** split with maximal information gain.
- Possible splits: mid points b/w any two consecutive values.
- For v=14, I(S1,S2) = 0 + 6/9*Entropy(S2) = 6/9 * 0.65 = 0.433
  - Gain(14,S) = Entropy(S) - 0.433
  - maximum *Gain* means minimum *I*.
- The best split is found after examining all possible splits.

Summary

- Data preparation is a big issue for data mining
- Data preparation includes
  - Data cleaning and data integration
  - Data reduction and feature selection
  - Discretization
- Many methods have been proposed but still an active area of research

Data Mining

What is Data Mining?

- Discovery of useful, possibly unexpected, patterns in data
- 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 Mining Process

- Understand the application domain
- Identify data sources and select target data
- Pre-processing: cleaning, attribute selection, etc
- Data mining to extract patterns or models
- Post-processing: identifying interesting or useful patterns/knowledge
- Incorporate patterns/knowledge in real world tasks

Data Mining Tasks

- Classification [Predictive]
- Clustering [Descriptive]
- Association Rule Discovery [Descriptive]
- Sequential Pattern Discovery [Descriptive]
- Regression [Predictive]
- Deviation Detection [Predictive]
- Collaborative Filter [Predictive]

Supervised learning (classification) vs. Unsupervised learning (clustering)

- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
  - Test data are classified into these classes too.
- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

What do we mean by learning?

- Given
  - a data set *D*,
  - a task *T*, and
  - a performance measure *M*,

  a computer system is said to **learn** from *D* to perform the task *T* if after learning the system's performance on *T* improves as measured by *M*.
- In other words, the learned model helps the system to perform *T* better as compared to no learning.

An example

- Data: Loan application data
- Task: Predict whether a loan should be approved or not.
- Performance measure: accuracy.

No learning: classify all future applications (test data) to the majority class (i.e., Yes):

Accuracy = 9/15 = 60%.

- We can do better than 60% with learning.

Fundamental assumption of learning

Assumption: The distribution of training examples is identical to the distribution of test examples (including future unseen examples).

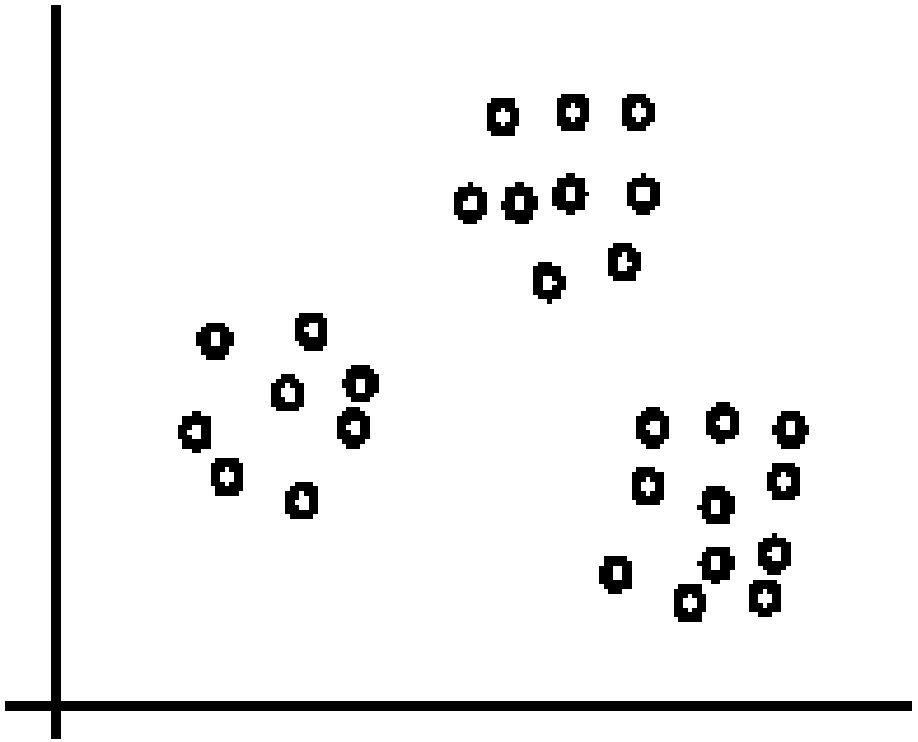
- In practice, this assumption is often violated to certain degree.
- Strong violations will clearly result in poor classification accuracy.
- To achieve good accuracy on the test data, training examples must be sufficiently representative of the test data.

Clustering

- Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,
  - it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.
- Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.

An illustration

- The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

- Let us see some real-life examples
- Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.
  - Tailor-made for each person: too expensive
  - One-size-fits-all: does not fit all.
- Example 2: In marketing, segment customers according to their similarities
  - To do targeted marketing.

What is clustering for? (cont…)

- Example 3: Given a collection of text documents, we want to organize them according to their content similarities,
  - To produce a topic hierarchy
- In fact, clustering is one of the most utilized data mining techniques.
  - It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.
  - In recent years, due to the rapid increase of online documents, text clustering becomes important.

Aspects of clustering

- A clustering algorithm
  - Partitional clustering, e.g., K-Means
  - Hierarchical clustering
  - …
- A distance (similarity, or dissimilarity) function
- Clustering quality
  - Inter-clusters distance $\Rightarrow$ maximized
  - Intra-clusters distance $\Rightarrow$ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

K-means clustering

- K-means is a partitional clustering algorithm
- Let the set of data points (or instances) D be

  $$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\},$$

  where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.
- The *k*-means algorithm partitions the given data into *k* clusters.
  - Each cluster has a cluster **center**, called **centroid**.
  - *k* is specified by the user

K-means algorithm

- Given *k*, the *k-means* algorithm works as follows:
  1) Randomly choose *k* data points (seeds) to be the initial centroids, cluster centers
  2) Assign each data point to the closest centroid
  3) Re-compute the centroids using the current cluster memberships.
  4) If a convergence criterion is not met, go to 2).

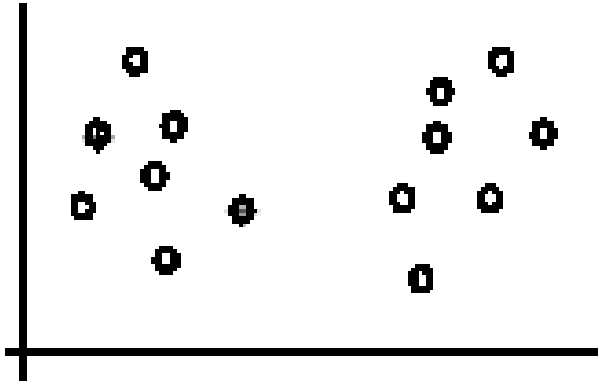
Stopping/convergence criterion

1. no (or minimum) re-assignments of data points to different clusters,
2. no (or minimum) change of centroids, or
3. minimum decrease in the **sum of squared error** (SSE),

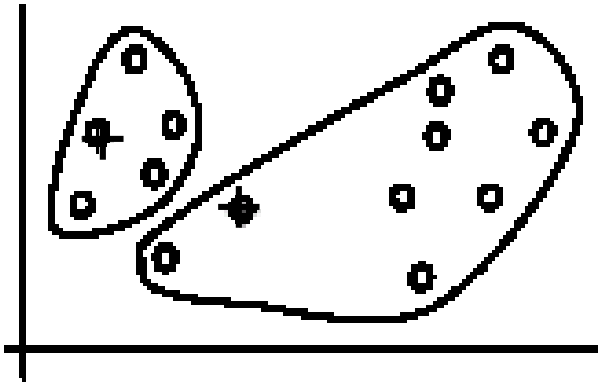
$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2 \quad (1)$$

- C_i is the jth cluster, $\mathbf{m}_j$ is the centroid of cluster C_j (the mean vector of all the data points in C_j), and $dist(\mathbf{x}, \mathbf{m}_j)$ is the distance between data point $\mathbf{x}$ and centroid $\mathbf{m}_j$.

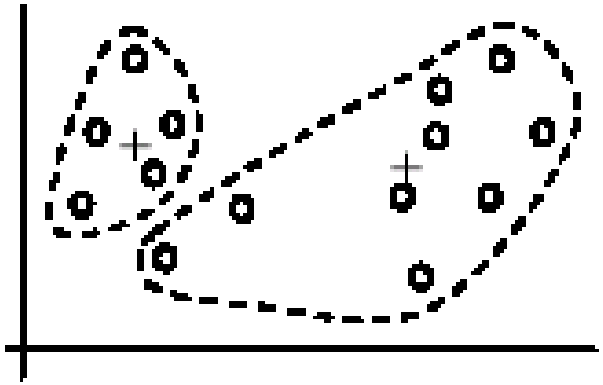
An example



(A). Random selection of k centers

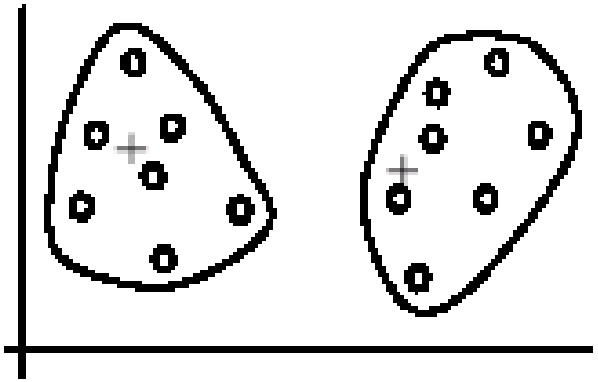


Iteration 1: (B). Cluster assignment

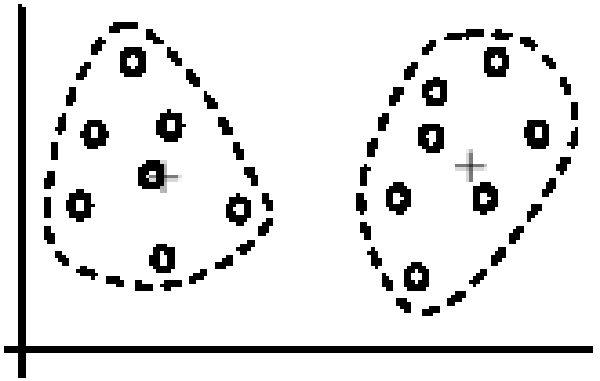


(C). Re-compute centroids

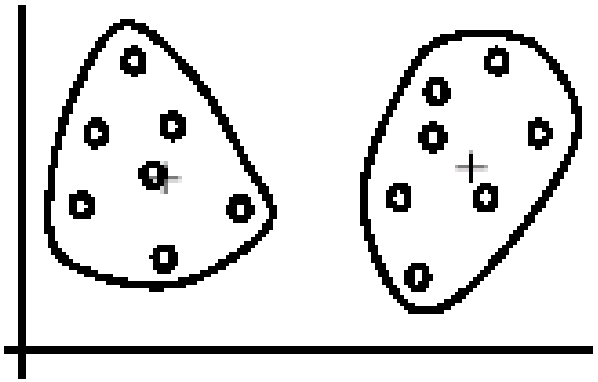
An example (cont …)



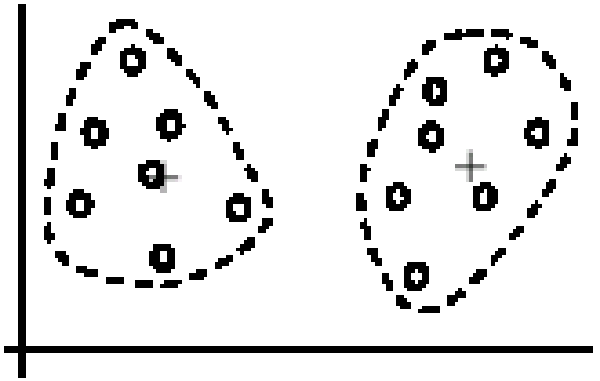
Iteration 2: (D). Cluster assignment



(E). Re-compute centroids



Iteration 3: (F). Cluster assignment



(G). Re-compute centroids

An example distance function

The *k*-means algorithm can be used for any application data set where the **mean** can be defined and computed. In the **Euclidean space**, the mean of a cluster is computed with:

$$\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i \qquad (2)$$

where $|C_j|$ is the number of data points in cluster C_j. The distance from one data point $\mathbf{x}_i$ to a mean (centroid) $\mathbf{m}_j$ is computed with

$$dist(\mathbf{x}_i, \mathbf{m}_j) = \| \mathbf{x}_i - \mathbf{m}_j \| \qquad (3)$$

$$= \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \ldots + (x_{ir} - m_{jr})^2}$$

A disk version of *k*-means

- K-means can be implemented with data on disk
  - In each iteration, it scans the data once.
  - as the centroids can be computed incrementally
- It can be used to cluster large datasets that do not fit in main memory
- We need to control the number of iterations
  - In practice, a limited is set (< 50).
- Not the best method. There are other scale-up algorithms, e.g., BIRCH.

Strengths of k-means

- Strengths:
  - Simple: easy to understand and to implement
  - Efficient: Time complexity: $O(tkn)$,

    where n is the number of data points,

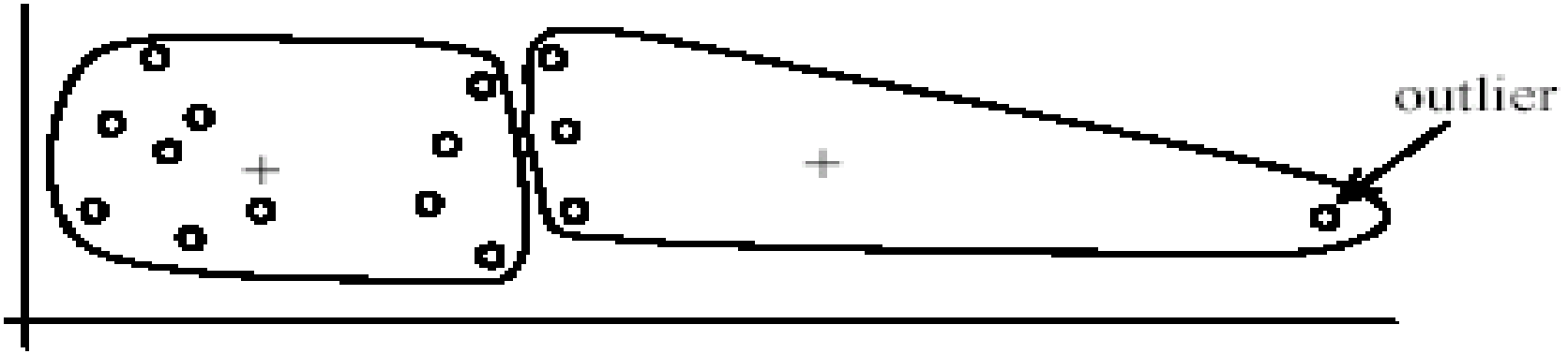
    k is the number of clusters, and

    t is the number of iterations.
  - Since both k and t are small. k-means is considered a linear algorithm.
- K-means is the most popular clustering algorithm.
- Note that: it terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

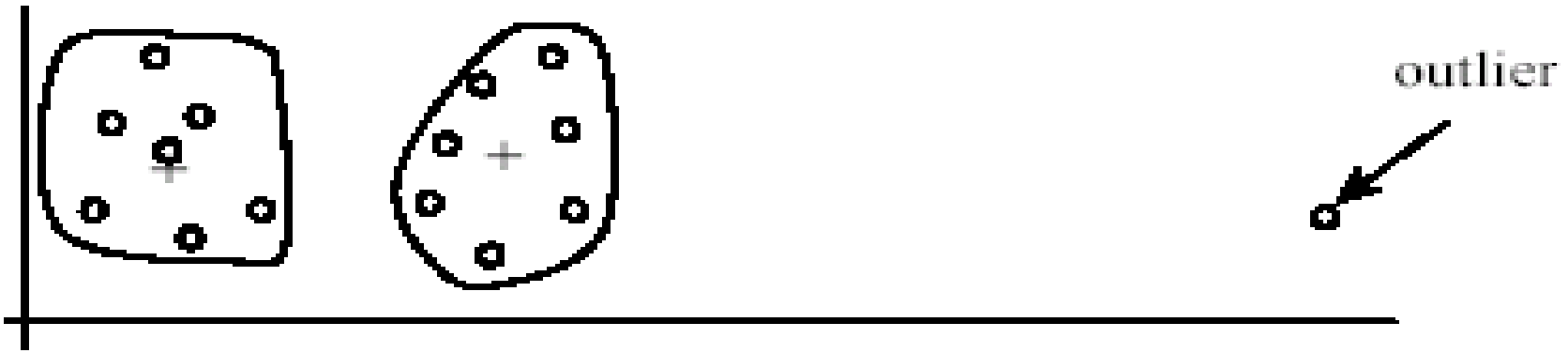
Weaknesses of k-means

- The algorithm is only applicable if the mean is defined.
  - For categorical data, *k*-mode - the centroid is represented by most frequent values.
- The user needs to specify *k*.
- The algorithm is sensitive to **outliers**
  - Outliers are data points that are very far away from other data points.
  - Outliers could be errors in the data recording or some special data points with very different values.

Weaknesses of k-means: Problems with outliers



(A): Undesirable clusters



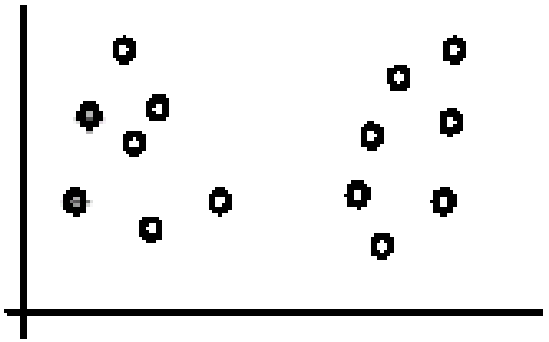
(B): Ideal clusters

Weaknesses of k-means: To deal with outliers

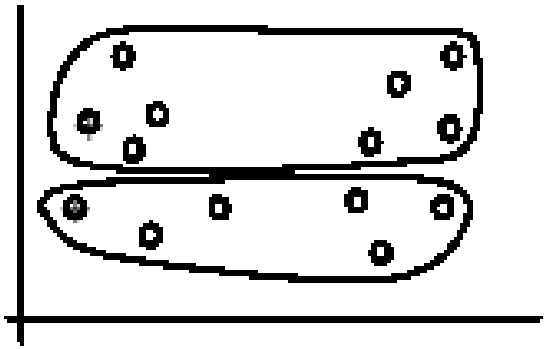
- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

Weaknesses of k-means (cont …)

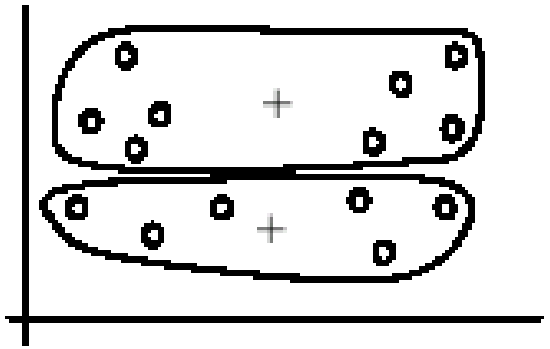
- The algorithm is sensitive to initial seeds.



(A). Random selection of seeds (centroids)



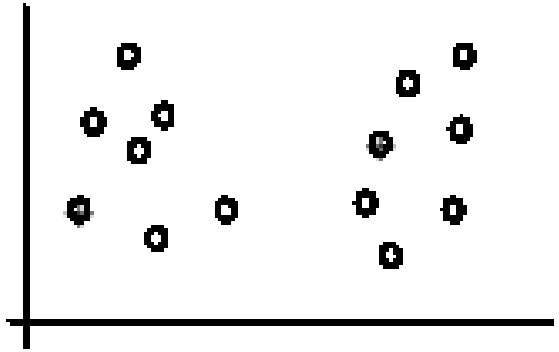
(B). Iteration 1



(C). Iteration 2

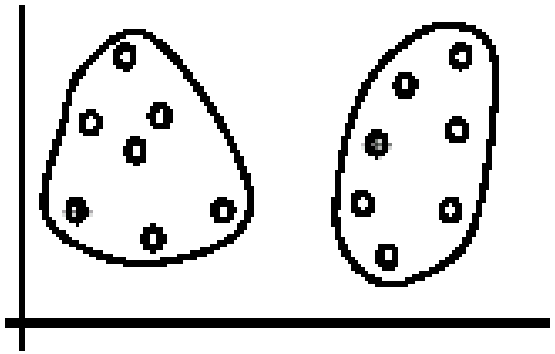
Weaknesses of k-means (cont …)

- If we use different seeds: good results

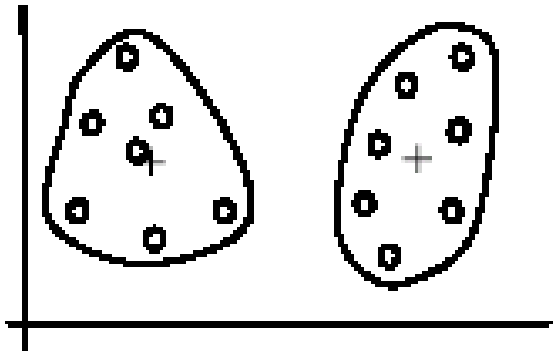


There are some methods to help choose good seeds

(A). Random selection of k seeds (centroids)



(B). Iteration 1



(C). Iteration 2

Distance functions

- Key to clustering. “similarity” and “dissimilarity” can also commonly used terms.
- There are numerous distance functions for
  - Different types of data
    - Numeric data
    - Nominal data
  - Different specific applications

Distance functions for numeric attributes

- Most commonly used functions are
  - Euclidean distance and
  - Manhattan (city block) distance
- We denote distance with: *dist*($\mathbf{x}_i$, $\mathbf{x}_j$), where $\mathbf{x}_i$ and $\mathbf{x}_j$ are data points (vectors)
- They are special cases of Minkowski distance. h is positive integer.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \left((x_{i1} - x_{j1})^h + (x_{i2} - x_{j2})^h + \ldots + (x_{ir} - x_{jr})^h\right)^{\frac{1}{h}}$$

Euclidean distance and Manhattan distance

- If h = 2, it is the Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ir} - x_{jr})^2}$$

- If h = 1, it is the Manhattan distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = | x_{i1} - x_{j1} | + | x_{i2} - x_{j2} | + ... + | x_{ir} - x_{jr} |$$

- Weighted Euclidean distance

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{w_1(x_{i1} - x_{j1})^2 + w_2(x_{i2} - x_{j2})^2 + ... + w_r(x_{ir} - x_{jr})^2}$$

Squared distance and Chebychev distance

- Squared Euclidean distance: to place progressively greater weight on data points that are further apart.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = (x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ir} - x_{jr})^2$$

- Chebychev distance: one wants to define two data points as "different" if they are different on any one of the attributes.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \max(| x_{i1} - x_{j1} |, | x_{i2} - x_{j2} |, ..., | x_{ir} - x_{jr} |)$$

Distance functions for binary and nominal attributes

- Binary attribute: has two values or states but no ordering relationships, e.g.,
  - Gender: male and female.
- We use a confusion matrix to introduce the distance functions/measures.
- Let the *i*th and *j*th data points be $\mathbf{x}_i$ and $\mathbf{x}_j$ (vectors)

Confusion matrix

Data point i \ Data point j	1	0	
1	a	b	$a+b$
0	c	d	$c+d$
	$a+c$	$b+d$	$a+b+c+d$

(10)

a: the number of attributes with the value of 1 for both data points.

b: the number of attributes for which $x_{if} = 1$ and $x_{jf} = 0$, where x_{if} (x_{jf}) is the value of the fth attribute of the data point $\mathbf{x}_i$ ($\mathbf{x}_j$).

c: the number of attributes for which $x_{if} = 0$ and $x_{jf} = 1$.

d: the number of attributes with the value of 0 for both data points.

Symmetric binary attributes

- A binary attribute is **symmetric** if both of its states (0 and 1) have equal importance, and carry the same weights, e.g., male and female of the attribute Gender
- Distance function: Simple Matching Coefficient, proportion of mismatches of their values

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c + d}$$

Symmetric binary attributes: example

x_1	1	1	1	0	1	0	0
x_2	0	1	1	0	0	1	0

$$dist(x_i, x_j) = \frac{2+1}{2+2+1+2} = \frac{3}{7} = 0.429$$

Asymmetric binary attributes

- Asymmetric: if one of the states is more important or more valuable than the other.
  - By convention, state 1 represents the more important state, which is typically the rare or infrequent state.
  - Jaccard coefficient is a popular measure

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{b + c}{a + b + c}$$

  - We can have some variations, adding weights

Nominal attributes

- Nominal attributes: with more than two states or values.
  - the commonly used distance measure is also based on the simple matching method.
  - Given two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, let the number of attributes be r, and the number of values that match in $\mathbf{x}_i$ and $\mathbf{x}_j$ be q.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{r - q}{r}$$

Data standardization

- In the Euclidean space, standardization of attributes is recommended so that all attributes can have equal impact on the computation of distances.
- Consider the following pair of data points
  - $\mathbf{x}_i$: (0.1, 20) and $\mathbf{x}_j$: (0.9, 720).

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{(0.9-0.1)^2 + (720-20)^2} = 700.000457,$$

- The distance is almost completely dominated by (720-20) = 700.
- Standardize attributes: to force the attributes to have a common value range

Interval-scaled attributes

- Their values are real numbers following a linear scale.
  - The difference in Age between 10 and 20 is the same as that between 40 and 50.
  - The key idea is that intervals keep the same importance through out the scale
- Two main approaches to standardize interval scaled attributes, **range** and **z-score**. *f* is an attribute

$$range(x_{if}) = \frac{x_{if} - \min(f)}{\max(f) - \min(f)},$$

Interval-scaled attributes (cont.)

- Z-score: transforms the attribute values so that they have a mean of zero and a **mean absolute deviation** of 1. The mean absolute deviation of attribute *f*, denoted by s_f, is computed as follows

$$s_f = \frac{1}{n}\left(|x_{1f} - m_f| + |x_{2f} - m_f| + \ldots + |x_{nf} - m_f|\right),$$

$$m_f = \frac{1}{n}\left(x_{1f} + x_{2f} + \ldots + x_{nf}\right),$$

Z-score: $$z(x_{if}) = \frac{x_{if} - m_f}{s_f}.$$

Ratio-scaled attributes

- Numeric attributes, but unlike interval-scaled attributes, their scales are exponential,
- For example, the total amount of microorganisms that evolve in a time *t* is approximately given by

  Ae^{Bt},

  - where *A* and *B* are some positive constants.
- Do log transform:

  $$\log(x_{if})$$

  - Then treat it as an interval-scaled attribuete

Nominal attributes

- Sometime, we need to transform nominal attributes to numeric attributes.
- Transform nominal attributes to binary attributes.
  - The number of values of a nominal attribute is *v*.
  - Create *v* binary attributes to represent them.
  - If a data instance for the nominal attribute takes a particular value, the value of its binary attribute is set to 1, otherwise it is set to 0.
- The resulting binary attributes can be used as numeric attributes, with two values, 0 and 1.

Nominal attributes: an example

- Nominal attribute *fruit*: has three values,
  - Apple, Orange, and Pear
- We create three binary attributes called, Apple, Orange, and Pear in the new data.
- If a particular data instance in the original data has Apple as the value for *fruit*,
  - then in the transformed data, we set the value of the attribute Apple to 1, and
  - the values of attributes Orange and Pear to 0

Ordinal attributes

- Ordinal attribute: an ordinal attribute is like a nominal attribute, but its values have a numerical ordering. E.g.,
  - Age attribute with values: Young, MiddleAge and Old. They are ordered.
  - Common approach to standardization: treat is as an interval-scaled attribute.

Mixed attributes

- Our distance functions given are for data with all numeric attributes, or all nominal attributes, etc.
- Practical data has different types:
  - Any subset of the 6 types of attributes,
    - **interval-scaled**,
    - **symmetric binary**,
    - **asymmetric binary**,
    - **ratio-scaled**,
    - **ordinal** and
    - **nominal**

Convert to a single type

- One common way of dealing with mixed attributes is to
  - Decide the dominant attribute type, and
  - Convert the other types to this type.
- E.g, if most attributes in a data set are interval-scaled,
  - we convert ordinal attributes and ratio-scaled attributes to interval-scaled attributes.
  - It is also appropriate to treat symmetric binary attributes as interval-scaled attributes.

Convert to a single type (cont …)

- It does not make much sense to convert a nominal attribute or an asymmetric binary attribute to an interval-scaled attribute,
  - but it is still frequently done in practice by assigning some numbers to them according to some hidden ordering, e.g., prices of the fruits
- Alternatively, a nominal attribute can be converted to a set of (symmetric) binary attributes, which are then treated as numeric attributes.

Combining individual distances

- This approach computes individual attribute distances and then combine them.

$$dist(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{f=1}^{r} \delta_{ij}^{f} d_{ij}^{f}}{\sum_{f=1}^{r} \delta_{ij}^{f}}$$

This distance value is between 0 and 1. r is the number of attributes in the data set. The indicator δ_{ij}^{f} is 1 when both values x_{if} and x_{jf} for attribute f are non-missing, and it is set to 0 otherwise. It is also set to 0 if attribute f is asymmetric and the match is 0-0. Equation (25) cannot be computed if all δ_{ij}^{f} 's are 0. In such a case, some default value may be used or one of the data points is removed.

d_{ij}^{f} is the distance contributed by attribute f, and it is in the 0-1 range.

How to choose a clustering algorithm

- Clustering research has a long history. A vast collection of algorithms are available.
  - We only introduced several main algorithms.
- Choosing the "best" algorithm is a challenge.
  - Every algorithm has limitations and works well with certain data distributions.
  - It is very hard, if not impossible, to know what distribution the application data follow. The data may not fully follow any "ideal" structure or distribution required by the algorithms.
  - One also needs to decide how to standardize the data, to choose a suitable distance function and to select other parameter values.

Choose a clustering algorithm (cont …)

- Due to these complexities, the common practice is to
  - run several algorithms using different distance functions and parameter settings, and
  - then carefully analyze and compare the results.
- The interpretation of the results must be based on insight into the meaning of the original data together with knowledge of the algorithms used.
- Clustering is highly application dependent and to certain extent subjective (personal preferences).

Cluster Evaluation: hard problem

- The quality of a clustering is very hard to evaluate because
  - We do not know the correct clusters
- Some methods are used:
  - User inspection
    - Study centroids, and spreads
    - Rules from a decision tree.
    - For text documents, one can read some documents in clusters.

Cluster evaluation: ground truth

- We use some labeled data (for classification)
- Assumption: Each class is a cluster.
- After clustering, a confusion matrix is constructed. From the matrix, we compute various measurements, entropy, purity, precision, recall and F-score.
  - Let the classes in the data D be $C = (c_1, c_2, \ldots, c_k)$. The clustering method produces k clusters, which divides D into k disjoint subsets, $D_1, D_2, \ldots, D_k$.

Evaluation measures: Entropy

Entropy: For each cluster, we can measure its entropy as follows:

$$entropy(D_i) = -\sum_{j=1}^{k} \Pr_i(c_j) \log_2 \Pr_i(c_j). \quad (29)$$

where $\Pr_i(c_j)$ is the proportion of class c_j data points in cluster i or D_i. The total entropy of the whole clustering (which considers all clusters) is

$$entropy_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times entropy(D_i) \quad (30)$$

Evaluation measures: purity

Purity: This again measures the extent that a cluster contains only one class of data. The purity of each cluster is computed with

$$purity(D_i) = \max_j(\Pr_i(c_j)) \tag{31}$$

The total purity of the whole clustering (considering all clusters) is

$$purity_{total}(D) = \sum_{i=1}^{k} \frac{|D_i|}{|D|} \times purity(D_i) \tag{32}$$

An example

Example 14: Assume we have a text collection D of 900 documents from three topics (or three classes), Science, Sports, and Politics. Each class has 300 documents. Each document in D is labeled with one of the topics (classes). We use this collection to perform clustering to find three clusters. Note that class/topic labels are not used in clustering. After clustering, we want to measure the effectiveness of the clustering algorithm.

Cluster	Science	Sports	Politics		Entropy	Purity
1	250	20	10		0.589	0.893
2	20	180	80		1.198	0.643
3	30	100	210		1.257	0.617
Total	300	300	300		1.031	0.711

A remark about ground truth evaluation

- Commonly used to compare different clustering algorithms.
- A real-life data set for clustering has no class labels.
  - Thus although an algorithm may perform very well on some labeled data sets, no guarantee that it will perform well on the actual application data at hand.
- The fact that it performs well on some label data sets does give us some confidence of the quality of the algorithm.
- This evaluation method is said to be based on **external data** or information.

Evaluation based on internal information

- **Intra-cluster cohesion** (compactness):
  - Cohesion measures how near the data points in a cluster are to the cluster centroid.
  - Sum of squared error (SSE) is a commonly used measure.
- I**nter-cluster separation** (isolation):
  - Separation means that different cluster centroids should be far away from one another.
- In most applications, expert judgments are still the key.

Indirect evaluation

- In some applications, clustering is not the primary task, but used to help perform another task.
- We can use the performance on the primary task to compare clustering methods.
- For instance, in an application, the primary task is to provide recommendations on book purchasing to online shoppers.
  - If we can cluster books according to their features, we might be able to provide better recommendations.
  - We can evaluate different clustering algorithms based on how well they help with the recommendation task.
  - Here, we assume that the recommendation can be reliably evaluated.

Classification / Supervised Learning

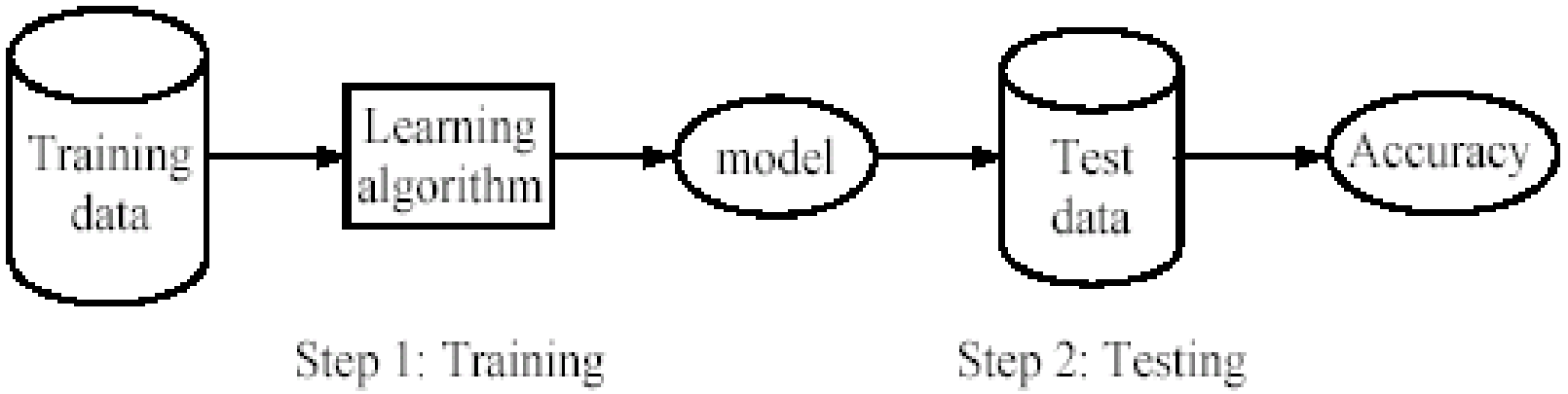
Classification: Definition

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

Supervised learning process: two steps

- Learning (training): Learn a model using the training data
- Testing: Test the model using unseen test data to assess the model accuracy

$$Accuracy = \frac{\text{Number of correct classifications}}{\text{Total number of test cases}},$$



An example application

- An emergency room in a hospital measures 17 variables (e.g., blood pressure, age, etc) of newly admitted patients.
- A decision is needed: whether to put a new patient in an intensive-care unit.
- Due to the high cost of ICU (Intensive Care Unit), those patients who may survive less than a month are given higher priority.
- Problem: to predict high-risk patients and discriminate them from low-risk patients.

Another application

- A credit card company receives thousands of applications for new cards. Each application contains information about an applicant,
  - age
  - Marital status
  - annual salary
  - outstanding debts
  - credit rating
  - etc.
- Problem: to decide whether an application should approved, or to classify applications into two categories, approved and not approved.

Machine learning and our focus

- Like human learning from past experiences.
- A computer does not have “experiences”.
- A computer system learns from data, which represent some “past experiences” of an application domain.
- Our focus: learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved, and high-risk or low risk.
- The task is commonly called: Supervised learning, classification, or inductive learning.

The data and the goal

- Data: A set of data records (also called examples, instances or cases) described by
  - *k* attributes: A_1, A_2, … A_k.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.

An example: data (loan application)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

An example: the learning task

- Learn a classification model from the data
- Use the model to classify future loan applications into
  - Yes (approved) and
  - No (not approved)
- What is the class for following case/instance?

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

Decision Tree Learning

- One of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - it is very efficient.
- The classification model is a tree, called decision tree.
- C4.5 by Ross Quinlan is perhaps the best known system. It can be downloaded from the Web.

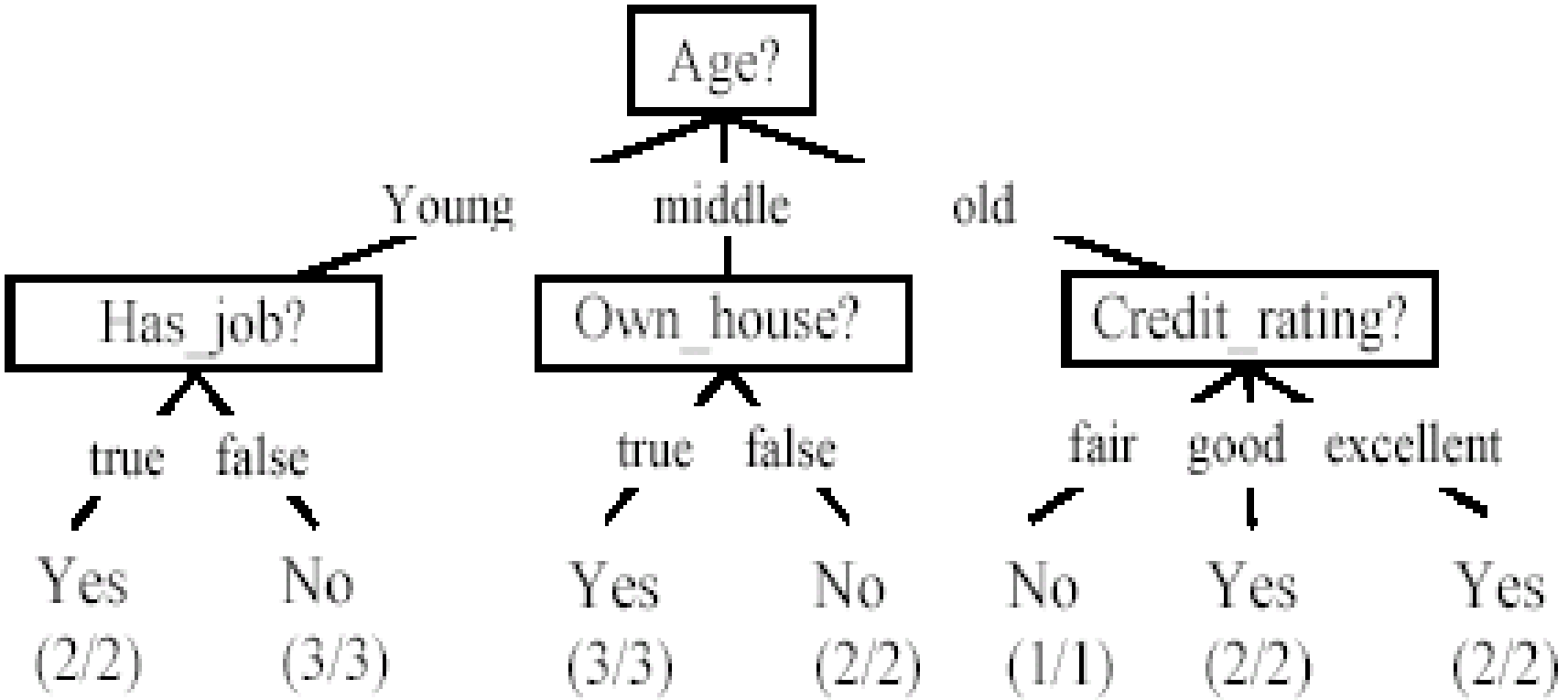
The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

A decision tree from the loan data

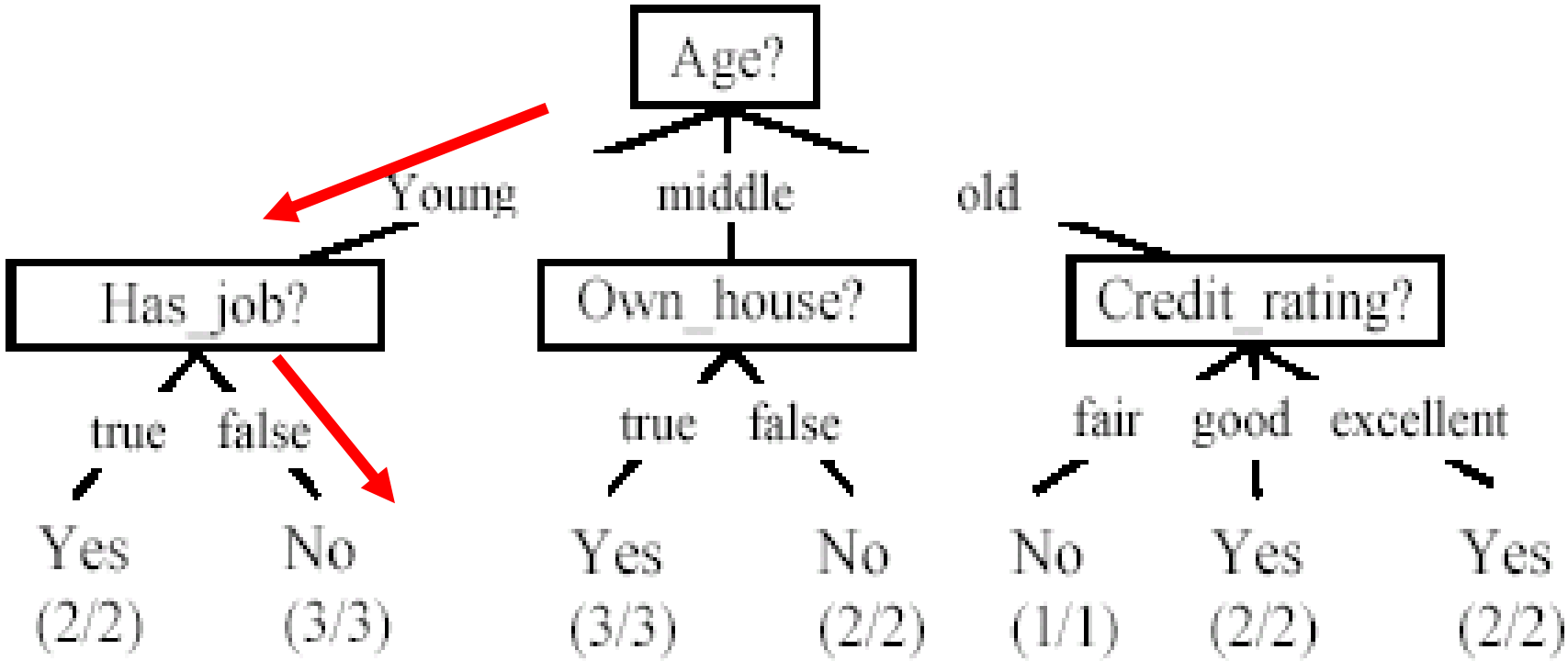
- Decision nodes and leaf nodes (classes)



Use the decision tree

Age	Has_Job	Own_house	Credit-Rating	Class
young	false	false	good	?

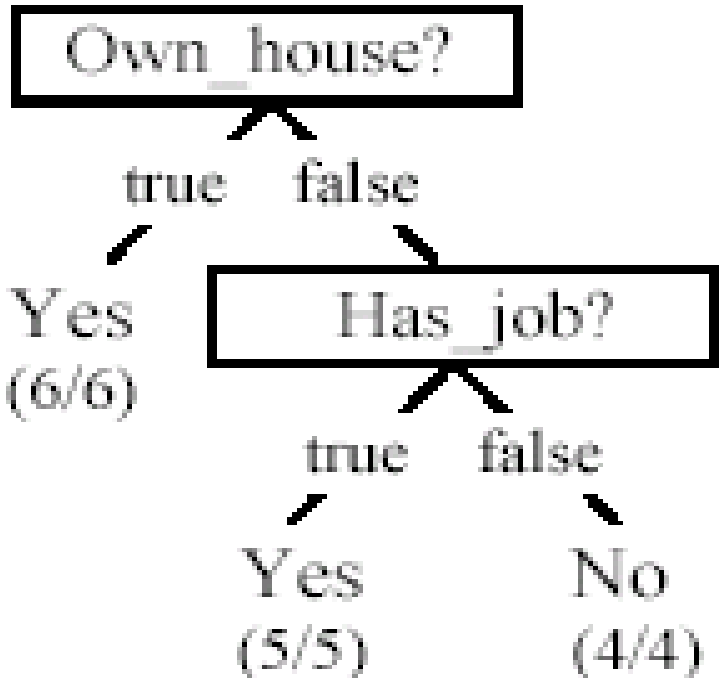
No



Is the decision tree unique?

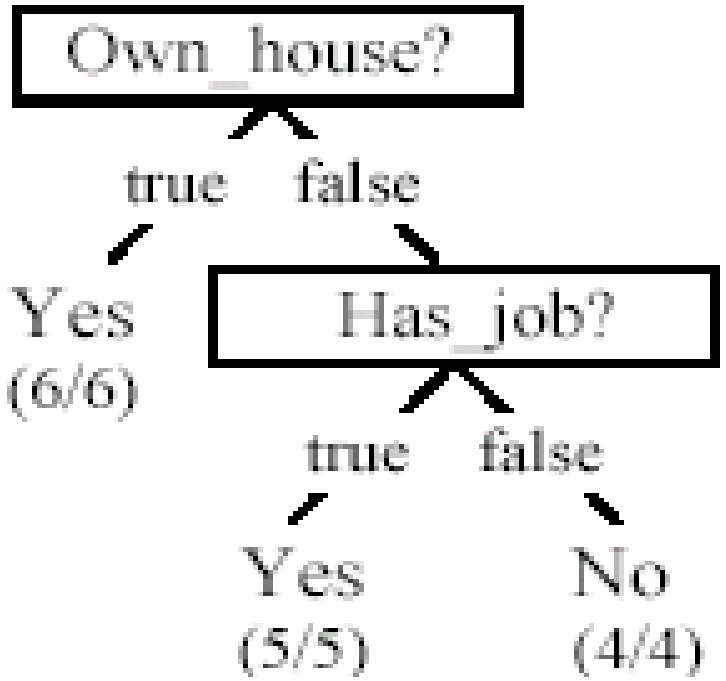
- No. Here is a simpler tree.
- We want smaller tree and accurate tree.
  - Easy to understand and perform better.

- Finding the best tree is NP-hard.
- All current tree building algorithms are heuristic algorithms



From a decision tree to a set of rules

- A decision tree can be converted to a set of rules
- Each path from the root to a leaf is a rule.



Own_house = true → Class =Yes [sup=6/15, conf=6/6]
Own_house = false, Has_job = true → Class = Yes [sup=5/15, conf=5/5]
Own_house = false, Has_job = false → Class = No [sup=4/15, conf=4/4]

Algorithm for decision tree learning

- Basic algorithm (a greedy **divide-and-conquer** algorithm)
  - Assume attributes are categorical now (continuous attributes can be handled too)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

Decision tree learning algorithm

. **Algorithm** decisionTree(D, A, T)

```
1    if D contains only training examples of the same class c_j ∈ C then
2        make T a leaf node labeled with class c_j;
3    elseif A = ∅ then
4        make T a leaf node labeled with c_j, which is the most frequent class in D
5    else  // D contains examples belonging to a mixture of classes. We select a single
6          // attribute to partition D into subsets so that each subset is purer
7        p_0 = impurityEval-1(D);
8        for each attribute A_i ∈ {A_1, A_2, ..., A_k} do
9            p_i = impurityEval-2(A_i, D)
10       end
11       Select A_g ∈ {A_1, A_2, ..., A_k} that gives the biggest impurity reduction,
             computed using p_0 − p_i;
12       if p_0 − p_g < threshold then    // A_g does not significantly reduce impurity p_0
13         make T a leaf node labeled with c_j, the most frequent class in D.
14       else                             // A_g is able to reduce impurity p_0
15           Make T a decision node on A_g;
16           Let the possible values of A_g be v_1, v_2, ..., v_m. Partition D into m
                 disjoint subsets D_1, D_2, ..., D_m based on the m values of A_g.
17           for each D_j in {D_1, D_2, ..., D_m} do
18               if D_j ≠ ∅ then
19                   create a branch (edge) node T_j for v_j as a child node of T;
20                   decisionTree(D_j, A-{A_g}, T_j)// A_g is removed
21               end
22           end
23       end
24   end
```

Choose an attribute to partition data

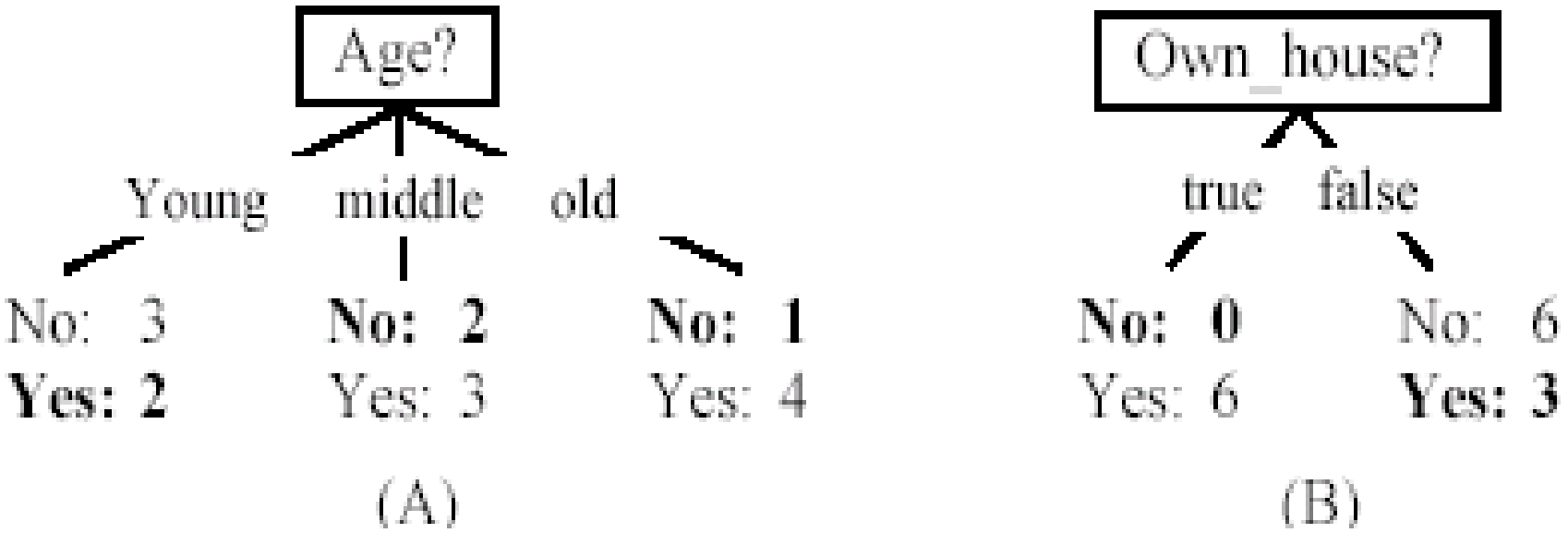
- The *key* to building a decision tree - which attribute to choose in order to branch.
- The objective is to reduce impurity or uncertainty in data as much as possible.
  - A subset of data is pure if all instances belong to the same class.
- The *heuristic* in C4.5 is to choose the attribute with the maximum Information Gain or Gain Ratio based on information theory.

The loan data (reproduced)

Approved or not

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	**No**
2	young	false	false	good	**No**
3	young	true	false	good	**Yes**
4	young	true	true	fair	**Yes**
5	young	false	false	fair	**No**
6	middle	false	false	fair	**No**
7	middle	false	false	good	**No**
8	middle	true	true	good	**Yes**
9	middle	false	true	excellent	**Yes**
10	middle	false	true	excellent	**Yes**
11	old	false	true	excellent	**Yes**
12	old	false	true	good	**Yes**
13	old	true	false	good	**Yes**
14	old	true	false	excellent	**Yes**
15	old	false	false	fair	**No**

Two possible roots, which is better?



- Fig. (B) seems to be better.

Information theory: Entropy measure

- The entropy formula,

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

$$\sum_{j=1}^{|C|} \Pr(c_j) = 1,$$

- $\Pr(c_j)$ is the probability of class c_j in data set D
- We use entropy as a measure of impurity or disorder of data set D. (Or, a measure of information in a tree)

Entropy measure: let us get a feeling

1. The data set D has 50% positive examples ($\Pr(positive) = 0.5$) and 50% negative examples ($\Pr(negative) = 0.5$).

$$entropy(D) = -0.5 \times \log_2 0.5 - 0.5 \times \log_2 0.5 = 1$$

2. The data set D has 20% positive examples ($\Pr(positive) = 0.2$) and 80% negative examples ($\Pr(negative) = 0.8$).

$$entropy(D) = -0.2 \times \log_2 0.2 - 0.8 \times \log_2 0.8 = 0.722$$

3. The data set D has 100% positive examples ($\Pr(positive) = 1$) and no negative examples, ($\Pr(negative) = 0$).

$$entropy(D) = -1 \times \log_2 1 - 0 \times \log_2 0 = 0$$

- As the data become purer and purer, the entropy value becomes smaller and smaller. This is useful to us!

Information gain

- Given a set of examples *D*, we first compute its entropy:

$$entropy(D) = -\sum_{j=1}^{|C|} \Pr(c_j) \log_2 \Pr(c_j)$$

- If we make attribute A_i, with v values, the root of the current tree, this will partition *D* into v subsets D_1, D_2 …, D_v . The expected entropy if A_i is used as the current root:

$$entropy_{A_i}(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times entropy(D_j)$$

Information gain (cont …)

- Information gained by selecting attribute A_i to branch or to partition the data is

$$gain(D, A_i) = entropy(D) - entropy_{A_i}(D)$$

- We choose the attribute with the highest gain to branch/split the current tree.

An example

$$entropy(D) = \frac{6}{15} \times \log_2 \frac{6}{15} + \frac{9}{15} \times \log_2 \frac{9}{15} = 0.971$$

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	excellent	No
3	young	true	false	good	Yes
4	young	true	true	good	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

$$entropy_{Own\_house}(D) = \frac{6}{15} \times entropy(D_1) + \frac{9}{15} \times entropy(D_2)$$
$$= \frac{6}{15} \times 0 + \frac{9}{15} \times 0.918$$
$$= 0.551$$

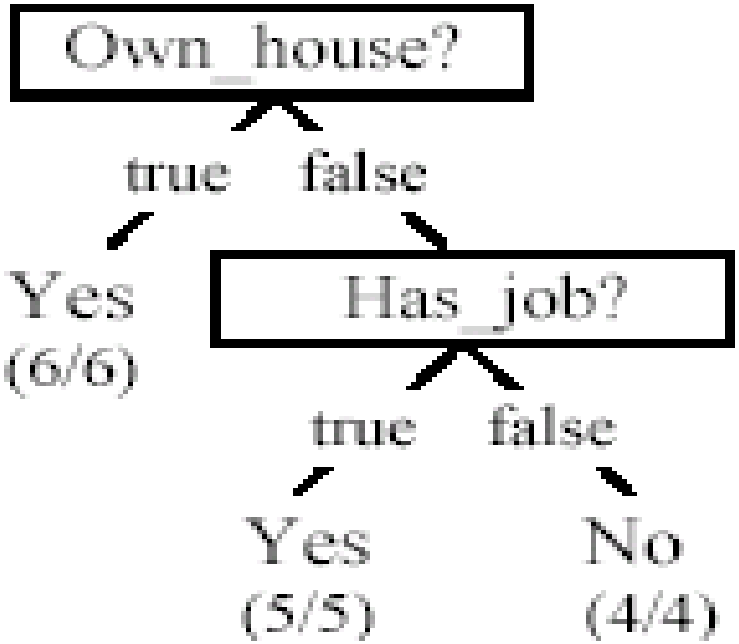
$$entropy_{Age}(D) = \frac{5}{15} \times entropy(D_1) + \frac{5}{15} \times entropy(D_2) + \frac{5}{15} \times entropy(D_3)$$
$$= \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.971 + \frac{5}{15} \times 0.722$$
$$= 0.888$$

Age	Yes	No	entropy(Di)
young	2	3	0.971
middle	3	2	0.971
old	4	1	0.722

- Own_house is the best choice for the root.

$$gain(D, \text{Age}) = 0.971 - 0.888 = 0.083$$
$$gain(D, \text{Own\_house}) = 0.971 - 0.551 = 0.420$$
$$gain(D, \text{Has\_Job}) = 0.971 - 0.647 = 0.324$$
$$gain(D, \text{Credit\_Rating}) = 0.971 - 0.608 = 0.363$$

We build the final tree



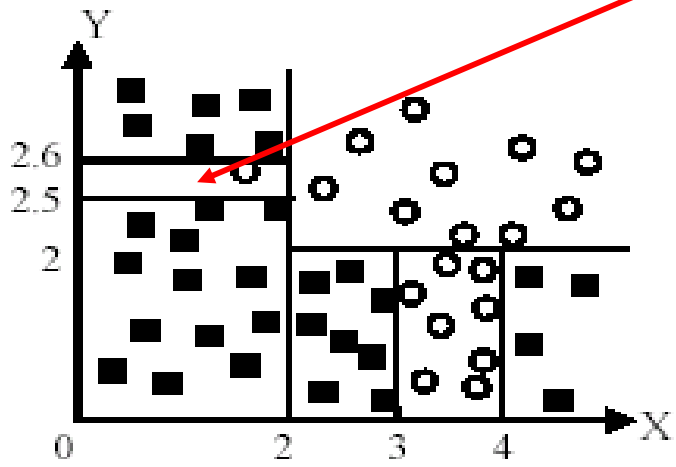
- We can use information gain ratio to evaluate the impurity as well

Avoid overfitting in classification

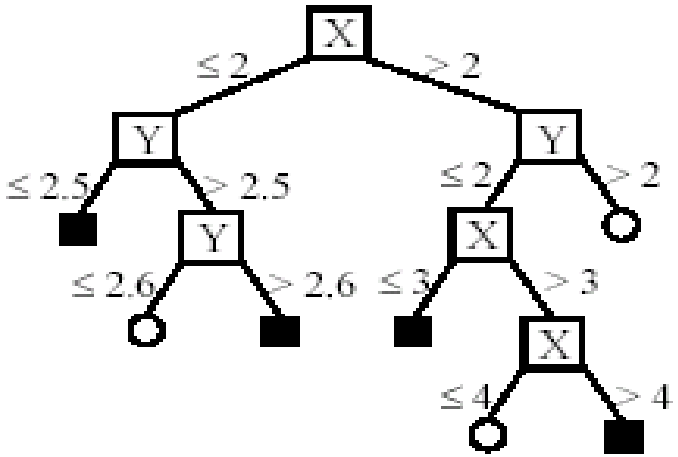
- Overfitting: A tree may overfit the training data
  - Good accuracy on training data but poor on test data
  - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Two approaches to avoid overfitting
  - Pre-pruning: Halt tree construction early
    - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
  - Post-pruning: Remove branches or sub-trees from a "fully grown" tree.
    - This method is commonly used. C4.5 uses a statistical method to estimates the errors at each node for pruning.
    - A validation set may be used for pruning as well.

An example

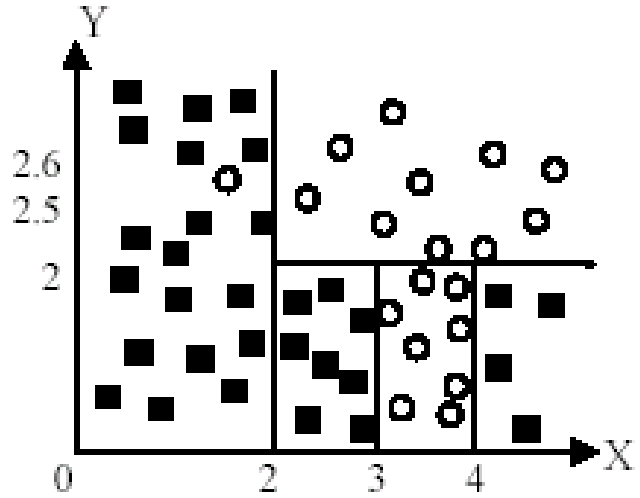
Likely to overfit the data

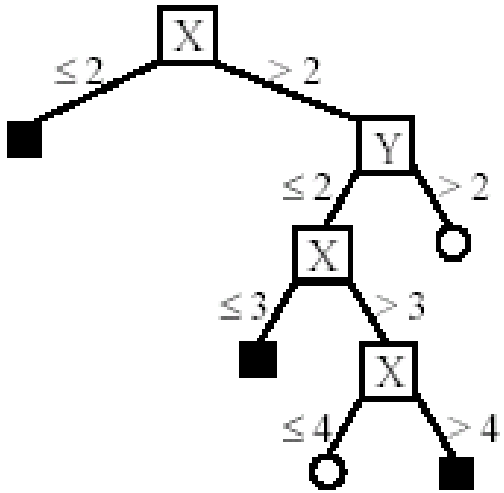


(A) A partition of the data space



(B). The decision tree





Naïve Bayesian classification

- Probabilistic view: Supervised learning can naturally be studied from a probabilistic point of view.
- Let A_1 through A_k be attributes with discrete values. The class is C.
- Given a test example *d* with observed attribute values a_1 through a_k.
- Classification is basically to compute the following posteriori probability. The prediction is the class c_j such that

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

is maximal

Apply Bayes' Rule

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|})}$$

$$= \frac{\Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_j)\Pr(C = c_j)}{\sum_{r=1}^{|C|} \Pr(A_1 = a_1, \ldots, A_{|A|} = a_{|A|} \mid C = c_r)\Pr(C = c_r)}$$

- $\Pr(C=c_j)$ is the class *prior* probability: easy to estimate from the training data.

Computing probabilities

- The denominator $P(A_1=a_1,...,A_k=a_k)$ is irrelevant for decision making since it is the same for every class.
- We only need $P(A_1=a_1,...,A_k=a_k \mid C=c_i)$, which can be written as

  $Pr(A_1=a_1|A_2=a_2,...,A_k=a_k, C=c_j)* Pr(A_2=a_2,...,A_k=a_k \mid C=c_j)$
- Recursively, the second factor above can be written in the same way, and so on.
- Now an assumption is needed.

Conditional independence assumption

- All attributes are conditionally independent given the class $C = c_j$.
- Formally, we assume,

  $\Pr(A_1=a_1 \mid A_2=a_2, ..., A_{|A|}=a_{|A|}, C=c_j) = \Pr(A_1=a_1 \mid C=c_j)$

  and so on for A_2 through $A_{|A|}$. I.e.,

$$\Pr(A_1 = a_1,..., A_{|A|} = a_{|A|} \mid C = c_i) = \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

Final naïve Bayesian classifier

$$\Pr(C = c_j \mid A_1 = a_1, \ldots, A_{|A|} = a_{|A|})$$

$$= \frac{\Pr(C = c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)}{\sum_{r=1}^{|C|} \Pr(C = c_r) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_r)}$$

- We are done!
- How do we estimate $P(A_i = a_i | C=c_j)$? Easy!.

Classify a test instance

- If we only need a decision on the most probable class for the test instance, we only need the numerator as its denominator is the same for every class.
- Thus, given a test example, we compute the following to decide the most probable class for the test instance

$$c = \arg\max_{c_j} \Pr(c_j) \prod_{i=1}^{|A|} \Pr(A_i = a_i \mid C = c_j)$$

An example

- Compute all probabilities required for classification

A	B	C
m	b	t
m	s	t
g	q	t
h	s	t
g	q	t
g	q	f
g	s	f
h	b	f
h	q	f
m	b	f

$Pr(C = t) = 1/2$, $Pr(C = f) = 1/2$

$Pr(A=m \mid C=t) = 2/5$
$Pr(A=m \mid C=f) = 1/5$
$Pr(B=b \mid C=t) = 1/5$
$Pr(B=b \mid C=f) = 2/5$

$Pr(A=g \mid C=t) = 2/5$
$Pr(A=g \mid C=f) = 2/5$
$Pr(B=s \mid C=t) = 2/5$
$Pr(B=s \mid C=f) = 1/5$

$Pr(A=h \mid C=t) = 1/5$
$Pr(A=h \mid C=n) = 2/5$
$Pr(B=q \mid C=t) = 2/5$
$Pr(B=q \mid C=f) = 2/5$

Now we have a test example:

$A = m \quad B = q \quad C = ?$

An Example (cont …)

- For C = t, we have

$$\Pr(C=t)\prod_{j=1}^{2}\Pr(A_j=a_j\,|\,C=t)=\frac{1}{2}\times\frac{2}{5}\times\frac{2}{5}=\frac{2}{25}$$

- For class C = f, we have

$$\Pr(C=f)\prod_{j=1}^{2}\Pr(A_j=a_j\,|\,C=f)=\frac{1}{2}\times\frac{1}{5}\times\frac{2}{5}=\frac{1}{25}$$

- C = t is more probable. t is the final class.

Additional issues

- **Numeric attributes**: Naïve Bayesian learning assumes that all attributes are categorical. Numeric attributes need to be discretized.
- **Zero counts**: An particular attribute value never occurs together with a class in the training set. We need smoothing.

$$\Pr(A_i = a_i \mid C = c_j) = \frac{n_{ij} + \lambda}{n_j + \lambda n_i}$$

- **Missing values**: Ignored

On naïve Bayesian classifier

- Advantages:
  - Easy to implement
  - Very efficient
  - Good results obtained in many applications
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy when the assumption is seriously violated (those highly correlated data sets)

SVM

- Support vector machines were invented by V. Vapnik and his co-workers in 1970s in Russia and became known to the West in 1992.
- SVMs are linear classifiers that find a hyperplane to separate two class of data, positive and negative.
- Kernel functions are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.
- It is perhaps the best classifier for text classification.

Basic concepts

- Let the set of training examples D be

  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_r, y_r)\}$,

  where $\mathbf{x}_i = (x_1, x_2, \ldots, x_n)$ is an **input vector** in a real-valued space $X \subseteq R^n$ and y_i is its **class label** (output value), $y_i \in \{1, -1\}$.

  1: positive class and -1: negative class.

- SVM finds a linear function of the form (**w**: weight vector)

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$$

$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

k-Nearest Neighbor Classification (kNN)

- Unlike all the previous learning methods, kNN does not build model from the training data.
- To classify a test instance *d*, define *k*-neighborhood *P* as *k* nearest neighbors of *d*
- Count number *n* of training instances in *P* that belong to class c_j
- Estimate Pr(c_j|*d*) as *n*/*k*
- No training is needed. Classification time is linear in training set size for each test case.

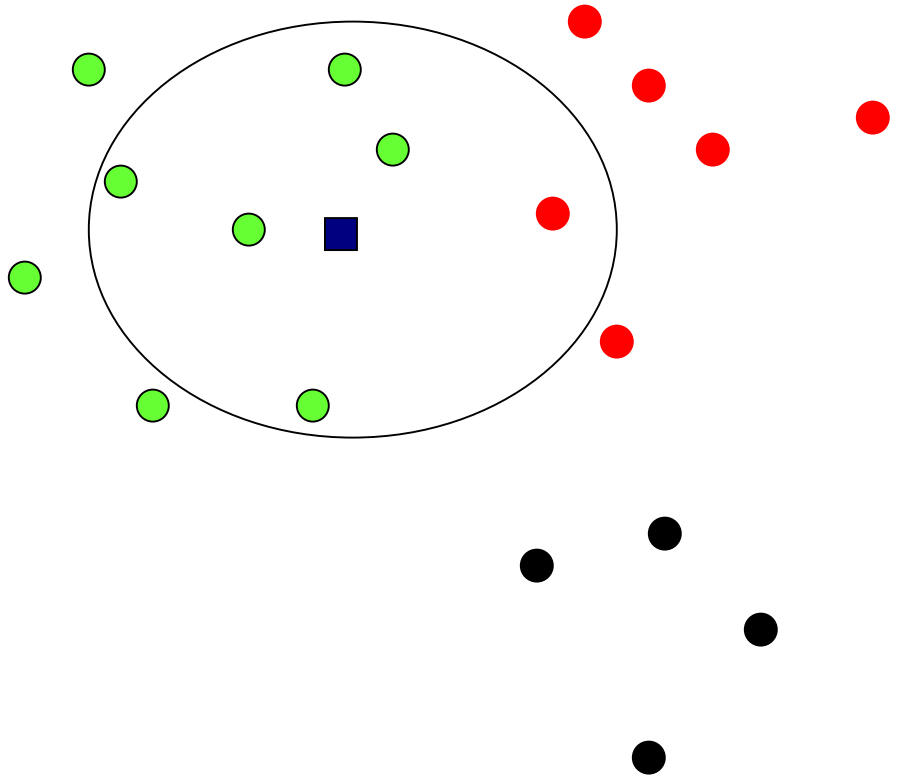
kNNAlgorithm

Algorithm kNN(D, d, k)

1 Compute the distance between d and every example in D;
2 Choose the k examples in D that are nearest to d, denote the set by P ($\subseteq D$);
3 Assign d the class that is the most frequent class in P (or the majority class);

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- Distance function is crucial, but depends on applications.

Example: k=6 (6NN)



- Government
- Science
- Arts

A new point ■
Pr(science|■)?

Discussions

- kNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of kNN can be quite strong and in many cases as accurate as those elaborated methods.
- kNN is slow at the classification time
- kNN does not produce an understandable model

Road Map

- Basic concepts
- Decision tree induction
- Evaluation of classifiers
- Rule induction
- Classification using association rules
- Naïve Bayesian classification
- Naïve Bayes for text classification
- Support vector machines
- K-nearest neighbor
- **Ensemble methods: Bagging and Boosting**
- Summary

Ensemble methods: Bagging and Boosting

- So far, we have only discussed individual classifiers, i.e., how to build them and use them.
- Can we combine multiple classifiers to produce a better classifier?
- Yes, sometimes
- We discuss two main algorithms:
  - Bagging
  - Boosting

Bagging

- Breiman, 1996
- Bootstrap Aggregating = Bagging
  - Application of bootstrap sampling
    - Given: set *D* containing *m* training examples
    - Create a sample *S*[*i*] of *D* by drawing *m* examples at random *with replacement* from *D*
    - *S*[*i*] of size *m*: expected to leave out 0.37 of examples from *D*

Bagging (cont…)

- Training
  - Create *k* bootstrap samples *S*[1], *S*[2], …, *S*[*k*]
  - Build a distinct classifier on each *S*[*i*] to produce *k* classifiers, using the same learning algorithm.
- Testing
  - Classify each new instance by voting of the *k* classifiers (equal weights)

Bagging Example

Original	1	2	3	4	5	6	7	8
Training set 1	2	7	8	3	7	6	3	1
Training set 2	7	8	5	6	4	2	7	1
Training set 3	3	6	2	7	5	6	2	2
Training set 4	4	5	1	4	6	4	3	8

Bagging (cont …)

- When does it help?
  - When learner is unstable
    - Small change to training set causes large change in the output classifier
    - True for decision trees, neural networks; not true for *k*-nearest neighbor, naïve Bayesian, class association rules
  - Experimentally, bagging can help substantially for unstable learners, may somewhat degrade results for stable learners

Boosting

- A family of methods:
  - We only study **AdaBoost** (Freund & Schapire, 1996)
- Training
  - Produce a sequence of classifiers (the same base learner)
  - Each classifier is dependent on the previous one, and focuses on the previous one's errors
  - Examples that are incorrectly predicted in previous classifiers are given higher weights
- Testing
  - For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

AdaBoost

Weighted training set

(x_1, y_1, w_1)
(x_2, y_2, w_2)
...
(x_n, y_n, w_n)

called a weaker classifier

- Build a classifier h_t whose accuracy on training set > ½ (better than random)

Non-negative weights sum to 1

Change weights

AdaBoost algorithm

Algorithm AdaBoost.M1

Input: sequence of m examples $\langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$
with labels $y_i \in Y = \{1, \ldots, k\}$
weak learning algorithm **WeakLearn**
integer T specifying number of iterations

Initialize $D_1(i) = 1/m$ for all i.

Do for $t = 1, 2, \ldots, T$:

1. Call **WeakLearn**, providing it with the distribution D_t.
2. Get back a hypothesis $h_t : X \rightarrow Y$.
3. Calculate the error of h_t: $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$.

   If $\epsilon_t > 1/2$, then set $T = t - 1$ and abort loop.
4. Set $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
5. Update distribution D_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \beta_t & \text{if } h_t(x_i) = y_i \\ 1 & \text{otherwise} \end{cases}$$

   where Z_t is a normalization constant (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$h_{fin}(x) = \arg\max_{y \in Y} \sum_{t:h_t(x)=y} \log \frac{1}{\beta_t}.$$

Bagging, Boosting and C4.5

C4.5's mean error rate over the 10 cross-validation.

Bagged C4.5 vs. C4.5.

Boosted C4.5 vs. C4.5.

Boosting vs. Bagging

	C4.5	Bagged C4.5 vs C4.5			Boosted C4.5 vs C4.5			Boosting vs Bagging	
	err (%)	err (%)	w-l	ratio	err (%)	w-l	ratio	w-l	ratio
anneal	7.67	6.25	10-0	.814	4.73	10-0	.617	10-0	.758
audiology	22.12	19.29	9-0	.872	15.71	10-0	.710	10-0	.814
auto	17.66	19.66	2-8	1.113	15.22	9-1	.862	9-1	.774
breast-w	5.28	4.23	9-0	.802	4.09	9-0	.775	7-2	.966
chess	8.55	8.33	6-2	.975	4.59	10-0	.537	10-0	.551
colic	14.92	15.19	0-6	1.018	18.83	0-10	1.262	0-10	1.240
credit-a	14.70	14.13	8-2	.962	15.64	1-9	1.064	0-10	1.107
credit-g	28.44	25.81	10-0	.908	29.14	2-8	1.025	0-10	1.129
diabetes	25.39	23.63	9-1	.931	28.18	0-10	1.110	0-10	1.192
glass	32.48	27.01	10-0	.832	23.55	10-0	.725	9-1	.872
heart-c	22.94	21.52	7-2	.938	21.39	8-0	.932	5-4	.994
heart-h	21.53	20.31	8-1	.943	21.05	5-4	.978	3-6	1.037
hepatitis	20.39	18.52	9-0	.908	17.68	10-0	.867	6-1	.955
hypo	.48	.45	7-2	.928	.36	9-1	.746	9-1	.804
iris	4.80	5.13	2-6	1.069	6.53	0-10	1.361	0-8	1.273
labor	19.12	14.39	10-0	.752	13.86	9-1	.725	5-3	.963
letter	11.99	7.51	10-0	.626	4.66	10-0	.389	10-0	.621
lymphography	21.69	20.41	8-2	.941	17.43	10-0	.804	10-0	.854
phoneme	19.44	18.73	10-0	.964	16.36	10-0	.842	10-0	.873
segment	3.21	2.74	9-1	.853	1.87	10-0	.583	10-0	.684
sick	1.34	1.22	7-1	.907	1.05	10-0	.781	9-1	.861
sonar	25.62	23.80	7-1	.929	19.62	10-0	.766	10-0	.824
soybean	7.73	7.58	6-3	.981	7.16	8-2	.926	8-1	.944
splice	5.91	5.58	9-1	.943	5.43	9-0	.919	6-4	.974
vehicle	27.09	25.54	10-0	.943	22.72	10-0	.839	10-0	.889
vote	5.06	4.37	9-0	.864	5.29	3-6	1.046	1-9	1.211
waveform	27.33	19.77	10-0	.723	18.53	10-0	.678	8-2	.938
average	*15.66*	*14.11*		*.905*	*13.36*		*.847*		*.930*

Does AdaBoost always work?

- The actual performance of boosting depends on the data and the base learner.
  - It requires the base learner to be unstable as bagging.
- Boosting seems to be susceptible to noise.
  - When the number of outliners is very large, the emphasis placed on the hard examples can hurt the performance.

Other learning method: Association Rule Discovery

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

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

Collaborative Filtering

- Goal: predict what movies/books/… a person may be interested in, on the basis of
  - Past preferences of the person
  - Other people with similar past preferences
  - The preferences of such people for a new movie/book/…
- One approach based on repeated clustering
  - Cluster people on the basis of preferences for movies
  - Then cluster movies on the basis of being liked by the same clusters of people
  - Again cluster people based on their preferences for (the newly created clusters of) movies
  - Repeat above till equilibrium
- Above problem is an instance of **collaborative filtering**, where users collaborate in the task of filtering information to find information of interest

Other Types of Mining

- **Text mining**: application of data mining to textual documents
- **Graph Mining**:
  - Deal with graph data

Text Mining

- Data mining on text
  - Due to online texts on the Web and other sources
  - Text contains a huge amount of information of almost any imaginable type!
  - A major direction and tremendous opportunity!
- Main topics
  - Text classification and clustering
    - cluster Web pages to find related pages
    - cluster pages a user has visited to organize their visit history
    - classify Web pages automatically into a Web directory
  - Information retrieval
  - Information extraction
  - Opinion mining