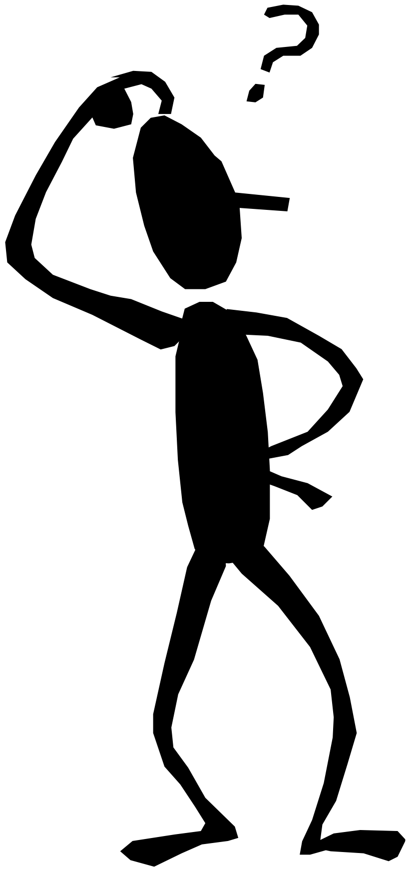
Classification and clustering

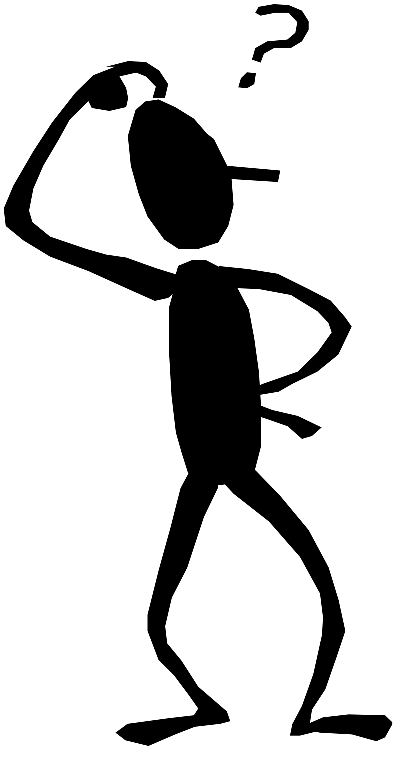


1. **Classification and prediction**
2. Clustering and similarity

Classification and prediction

- **What is classification? What is prediction?**
- **Decision tree induction**
- **Bayesian classification**
- **Other classification methods**
- **Classification accuracy**
- **Summary**

What is classification?

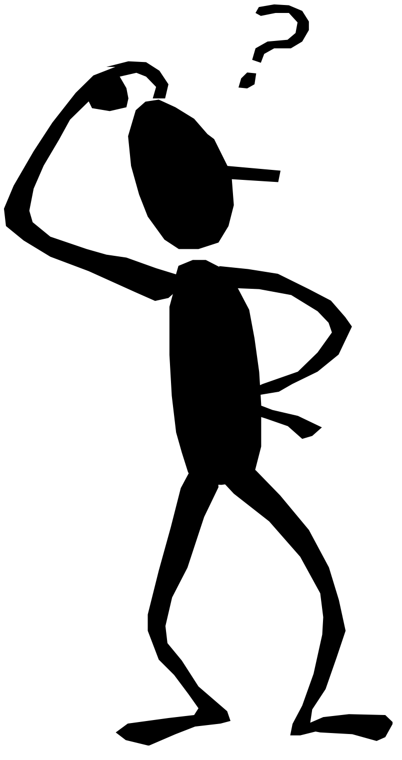


- **Aim:** to predict categorical class labels for new tuples/samples
- **Input:** a training set of tuples/samples, each with a class label
- **Output:** a model (a classifier) based on the training set and the class labels

Typical classification applications

- **Credit approval**
- **Target marketing**
- **Medical diagnosis**
- **Treatment effectiveness analysis**

What is prediction?



- **Is similar to classification**
  - o constructs a model
  - o uses the model to predict unknown or missing values
- **Major method: regression**
  - o linear and multiple regression
  - o non-linear regression

Classification vs. prediction

- **Classification:**
  - o predicts categorical class labels
  - o classifies data based on the training set and the values in a classification attribute and uses it in classifying new data
- **Prediction:**
  - o models continuous-valued functions
  - o predicts unknown or missing values

Terminology

- **Classification = supervised learning**
  - o training set of tuples/samples accompanied by class labels
  - o classify new data based on the training set
- **Clustering = unsupervised learning**
  - o class labels of training data are unknown
  - o aim in finding possibly existing classes or clusters in the data

Classification - a two step process

1. step:

Model construction, i.e., build the model from the training set

2. step:

Model usage, i.e., check the accuracy of the model and use it for classifying new data

Model construction

- **Each tuple/sample** is assumed to belong **a prefined class**
- The class of a tuple/sample is determined by the **class label attribute**
- The **training set** of tuples/samples is used for **model construction**
- The model is represented as **classification rules, decision trees** or **mathematical formulae**

Model usage

- **Classify future or unknown objects**
- **Estimate accuracy of the model**
  - o the known class of a test tuple/sample is compared with the result given by the model
  - o accuracy rate = precentage of the tests tuples/samples correctly classified by the model

An example: model construction

Training Data

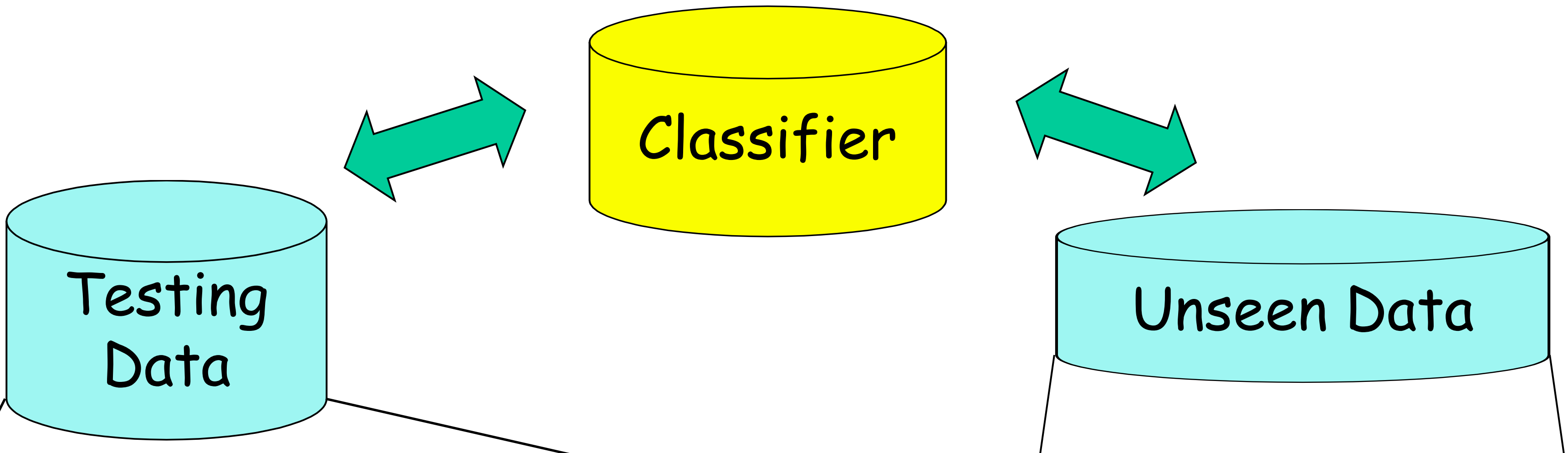
Classification Algorithms

Classifier (Model)

NAME	RANK	YEARS	TENURED
Mary	Assistant Prof	3	no
James	Assistant Prof	7	yes
Bill	Professor	2	no
John	Associate Prof	7	yes
Mark	Assistant Prof	6	no
Annie	Associate Prof	3	no

IF rank = 'professor'
OR years > 6
THEN tenured = yes

An example: model usage



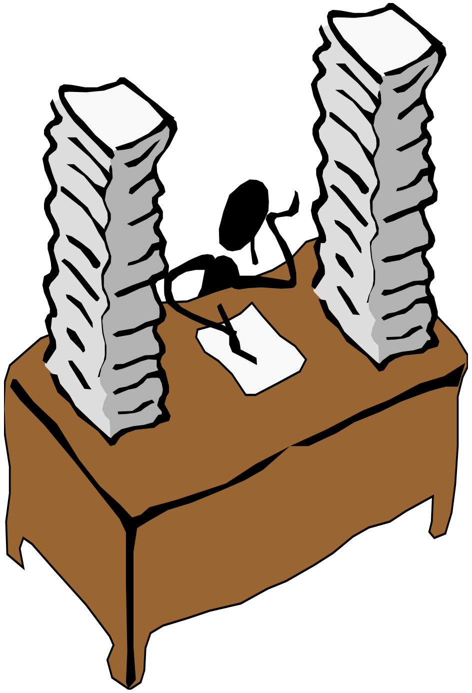
(Jeff, Professor, 4)

NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Lisa	Associate Prof	7	no
Jack	Professor	5	yes
Ann	Assistant Prof	7	yes

Tenured?

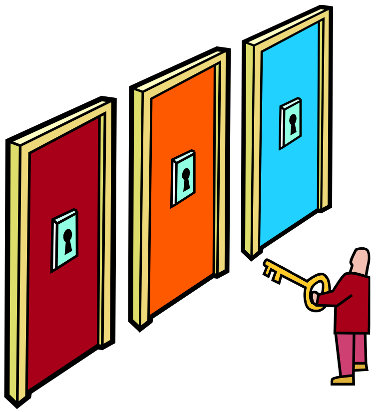
Yes

Data Preparation



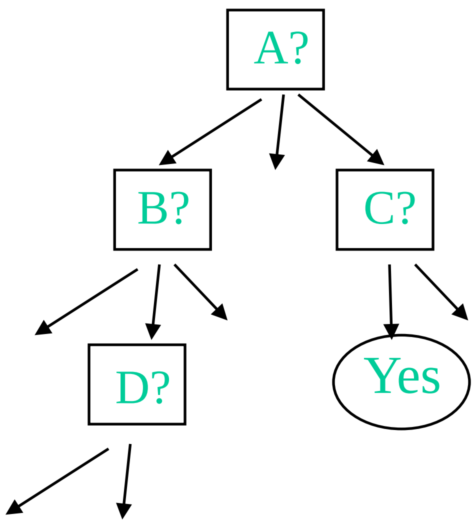
- **Data cleaning**
  - o noise
  - o missing values
- **Relevance analysis** (feature selection)
- **Data transformation**

Evaluation of classification methods



- **Accuracy**
- **Speed**
- **Robustness**
- **Scalability**
- **Interpretability**
- **Simplicity**

Decision tree induction



A decision tree is a tree where

- **internal node** = a test on an attribute
- tree **branch** = an outcome of the test
- **leaf node** = class label or class distribution

Decision tree generation

Two phases of decision tree generation:

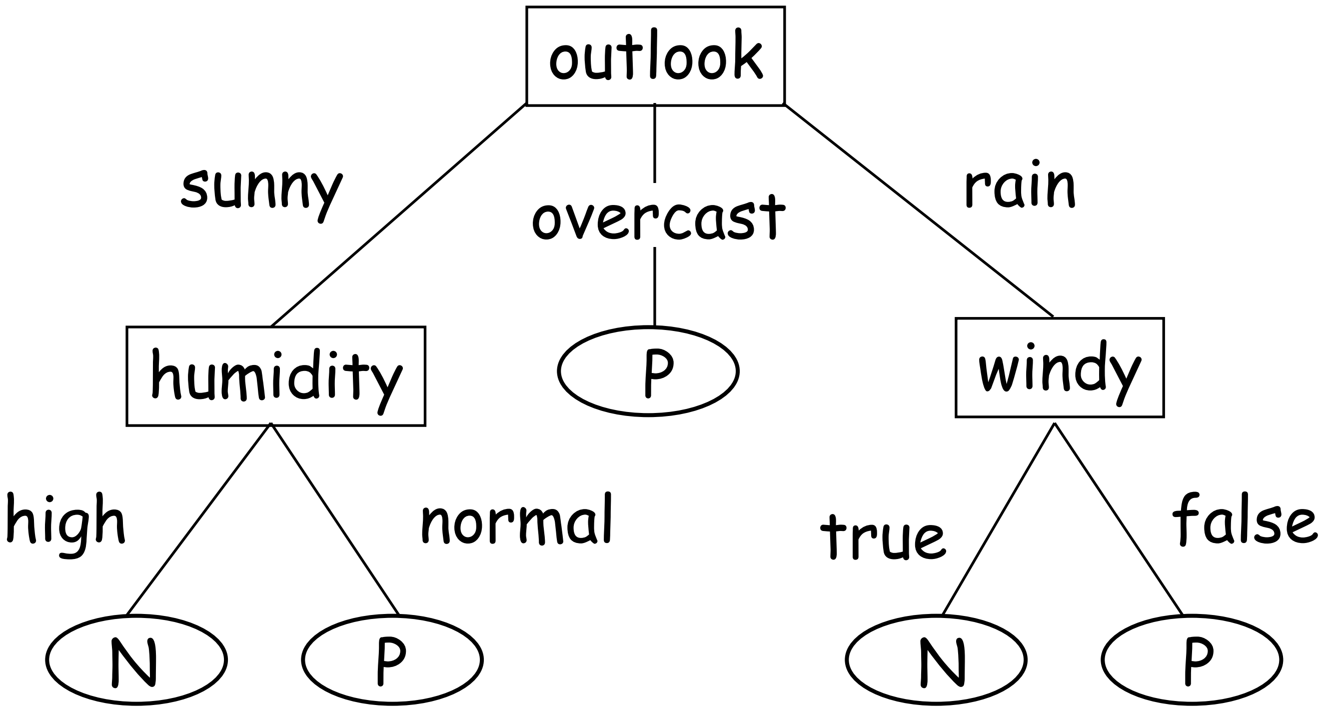
- **tree construction**
  - at start, all the training examples at the root
  - partition examples based on selected attributes
  - test attributes are selected based on a heuristic or a statistical measure
- **tree pruning**
  - identify and remove branches that reflect noise or outliers

Decision tree induction – Classical example: play tennis?

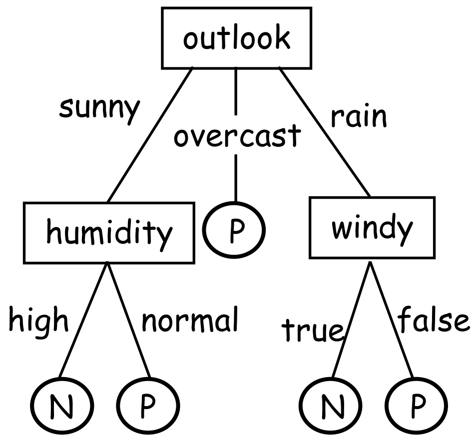
Training set from Quinlan's ID3

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

Decision tree obtained with ID3 (Quinlan 86)



From a decision tree to classification rules



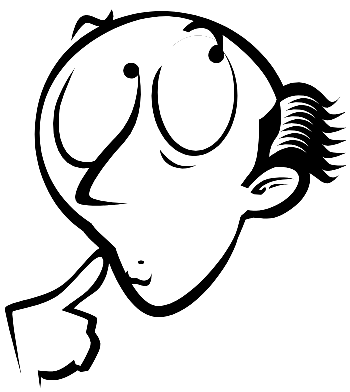
IF outlook=sunny
AND humidity=normal
THEN play tennis

- One **rule** is generated for each **path** in the tree from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are generally simpler to understand than trees

Decision tree algorithms

- **Basic algorithm**
  - o constructs a tree in a **top-down** recursive **divide-and-conquer** manner
  - o attributes are assumed to be categorical
  - o greedy (may get trapped in local maxima)
- **Many variants**: ID3, C4.5, CART, CHAID
  - o main difference: divide (split) criterion / attribute selection measure

Attribute selection measures



- **Information gain**
- **Gini index**
- **χ^2 contingency table statistic**
- **G-statistic**

Information gain (1)

- Select the attribute with the **highest information gain**
- Let ***P*** and ***N*** be two classes and **S** a dataset with *p* ***P***-elements and *n* ***N***-elements
- The amount of information needed to decide if an arbitrary example belongs to ***P*** or ***N*** is

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information gain (2)

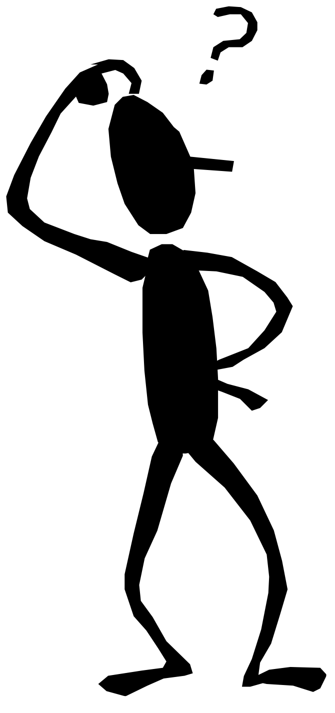
- Let sets {$\mathbf{S_1}$, $\mathbf{S_2}$, …, $\mathbf{S_v}$} form a partition of the set **S,** when using the attribute *A*
- Let each $\mathbf{S}_i$ contain p_i examples of ***P*** and n_i examples of ***N***
- The **entropy**, or the expected information needed to classify objects in all the subtrees $\mathbf{S}_i$ is

$$E(A) = \sum_{i=1}^{\nu} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The information that would be gained by branching on *A* is

$$Gain(A) = I(p,n) - E(A)$$

Information gain – Example (1)



Assumptions:

- Class ***P***: plays_tennis = “yes”
- Class ***N***: plays_tennis = “no”
- Information needed to classify a given sample:

$$I(p,n) = I(9,5) = 0.940$$

Information gain – Example (2)

Compute the entropy for the attribute *outlook*:

outlook	p_i	n_i	$I(p_i, n_i)$
sunny	2	3	0,971
overcast	4	0	0
rain	3	2	0,971

Now

$$E(outlook) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

Hence

$$Gain(outlook) = I(9,5) - E(outlook) = 0.246$$

Similarly

$$Gain(temperature) = 0.029$$

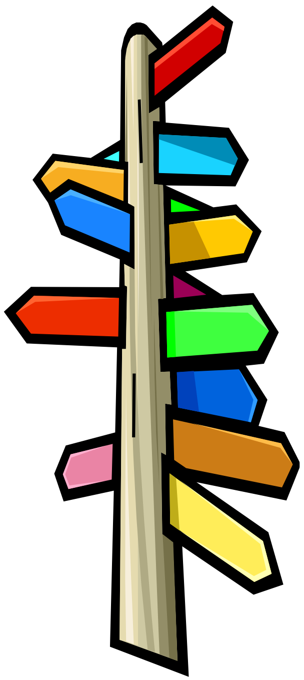
$$Gain(humidity) = 0.151$$

$$Gain(windy) = 0.048$$

Other criteria used in decision tree construction

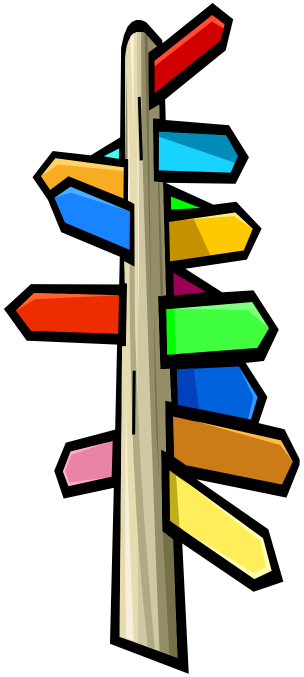
- **Conditions for stopping partitioning**
  - o all samples belong to the same class
  - o no attributes left for further partitioning => majority voting for classifying the leaf
  - o no samples left for classifying
- **Branching scheme**
  - o binary vs. *k*-ary splits
  - o categorical vs. continuous attributes
- **Labeling rule**: a leaf node is labeled with the class to which most samples at the node belong

Overfitting in decision tree classification



- **The generated tree may overfit the training data**
  - o too many branches
  - o poor accuracy for unseen samples
- **Reasons for overfitting**
  - o noise and outliers
  - o too little training data
  - o local maxima in the greedy search

How to avoid overfitting?



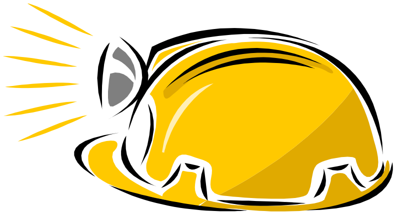
Two approaches:

- **prepruning:** Halt tree construction early
- **postpruning:** Remove branches from a "fully grown" tree

Classification in Large Databases

- **Scalability**: classifying data sets with millions of samples and hundreds of attributes with reasonable speed
- **Why decision tree induction in data mining?**
  - relatively faster learning speed than other methods
  - convertible to simple and understandable classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy

Scalable decision tree induction methods in data mining studies



- **SLIQ** (EDBT'96 — Mehta et al.)
- **SPRINT** (VLDB'96 — J. Shafer et al.)
- **PUBLIC** (VLDB'98 — Rastogi & Shim)
- **RainForest** (VLDB'98 — Gehrke, Ramakrishnan & Ganti)

Bayesian Classification: Why? (1)

- **Probabilistic learning**:
  - calculate explicit probabilities for hypothesis
  - among the most practical approaches to certain types of learning problems
- **Incremental**:
  - each training example can incrementally increase/decrease the probability that a hypothesis is correct
  - prior knowledge can be combined with observed data

Bayesian Classification: Why? (2)

- **Probabilistic prediction**:
  - o predict multiple hypotheses, weighted by their probabilities
- **Standard**:
  - o even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayesian classification

- The classification problem may be formalized using **a-posteriori probabilities:**

  P($C|X$**)** **=** probability that the sample tuple $X = \langle x_1, \ldots, x_k \rangle$ is of the class C

- For example

  P(class**=*N* |** outlook=sunny,windy=true,…**)**

- **Idea:** assign to sample X the class label C such that **P(**$C|X$**)** is maximal

Estimating a-posteriori probabilities

- **Bayes theorem**:

$$\mathbf{P}(C|X) = \mathbf{P}(X|C)\cdot\mathbf{P}(C) \,/\, \mathbf{P}(X)$$

- $\mathbf{P}(X)$ is constant for all classes
- $\mathbf{P}(C)$ = relative freq of class **C** samples
- C such that $\mathbf{P}(C|X)$ is maximum =
  C such that $\mathbf{P}(X|C)\cdot\mathbf{P}(C)$ is maximum
- **Problem**: computing $\mathbf{P}(X|C)$ is unfeasible!

Naïve Bayesian classification

- Naïve assumption: **attribute independence**

$$\mathbf{P}(x_1,\ldots,x_k|C) = \mathbf{P}(x_1|C)\cdot\ldots\cdot\mathbf{P}(x_k|C)$$

- If i-th attribute is **categorical**:
  $\mathbf{P}(x_i|C)$ is estimated as the relative frequency of samples having value x_i as i-th attribute in the class C
- If i-th attribute is **continuous**:
  $\mathbf{P}(x_i|C)$ is estimated thru a Gaussian density function
- Computationally easy in both cases

Naïve Bayesian classification – Example (1)

- Estimating $\mathbf{P}(x_i|C)$

P(p) = 9/14
P(n) = 5/14

Outlook	
P(sunny | p) = 2/9	P(sunny | n) = 3/5
P(overcast | p) = 4/9	P(overcast | n) = 0
P(rain | p) = 3/9	P(rain | n) = 2/5
Temperature	
P(hot | p) = 2/9	P(hot | n) = 2/5
P(mild | p) = 4/9	P(mild | n) = 2/5
P(cool | p) = 3/9	P(cool | n) = 1/5

Humidity	
P(high | p) = 3/9	P(high | n) = 4/5
P(normal | p) = 6/9	P(normal | n) = 1/5
Windy	
P(true | p) = 3/9	P(true | n) = 3/5
P(false | p) = 6/9	P(false | n) = 2/5

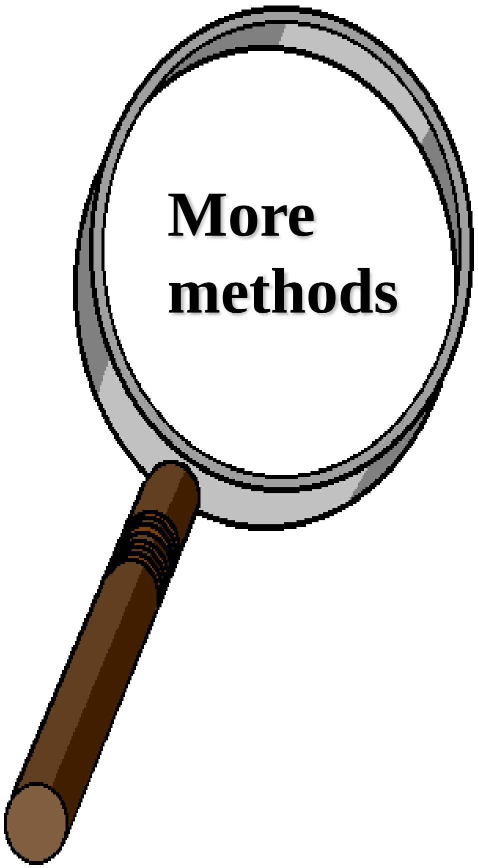
Naïve Bayesian classification – Example (2)

- Classifying *X:*
  - an unseen sample X = <*rain*, *hot*, *high*, *false*>
  - $\mathbf{P(}X|p\mathbf{)\cdot P(}p\mathbf{)} =$ $\mathbf{P(}rain|p\mathbf{)\cdot P(}hot|p\mathbf{)\cdot P(}high|p\mathbf{)\cdot P(}false|p\mathbf{)\cdot P(}p\mathbf{)} =$ $3/9\cdot 2/9\cdot 3/9\cdot 6/9\cdot 9/14 = 0.010582$
  - $\mathbf{P(}X|n\mathbf{)\cdot P(}n\mathbf{)} =$ $\mathbf{P(}rain|n\mathbf{)\cdot P(}hot|n\mathbf{)\cdot P(}high|n\mathbf{)\cdot P(}false|n\mathbf{)\cdot P(}n\mathbf{)} =$ $2/5\cdot 2/5\cdot 4/5\cdot 2/5\cdot 5/14 = \mathbf{0.018286}$
  - Sample *X* is classified in class *n* (don't play)

Naïve Bayesian classification – the independence hypothesis

- … makes computation possible
- … yields optimal classifiers when satisfied
- … but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
  - **Bayesian networks**, that combine Bayesian reasoning with causal relationships between attributes
  - **Decision trees**, that reason on one attribute at the time, considering most important attributes first

Other classification methods (not covered)



- **Neural networks**
- **k-nearest neighbor classifier**
- **Case-based reasoning**
- **Genetic algorithm**
- **Rough set approach**
- **Fuzzy set approaches**

Classification accuracy

Estimating error rates:

- **Partition**: training-and-testing (large data sets)
  - o use two independent data sets, e.g., training set (2/3), test set(1/3)
- **Cross-validation** (moderate data sets)
  - o divide the data set into *k* subsamples
  - o use *k-1* subsamples as training data and one sub-sample as test data --- *k*-fold cross-validation
- **Bootstrapping**: leave-one-out (small data sets)

Summary (1)

- **Classification is an extensively studied problem**
- **Classification is probably one of the most widely used data mining techniques with a lot of extensions**

Summary (2)

- **Scalability is still an important issue for database applications**
- **Research directions: classification of non-relational data, e.g., text, spatial and multimedia**

Course on Data Mining

Thanks to
Jiawei Han from Simon Fraser University
for his slides which greatly helped
in preparing this lecture!

Also thanks to
Fosca Giannotti and Dino Pedreschi from Pisa
for their slides of classification.

References - classification

- C. Apte and S. Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13, 1997.
- F. Bonchi, F. Giannotti, G. Mainetto, D. Pedreschi. Using Data Mining Techniques in Fiscal Fraud Detection. In Proc. DaWak'99, First Int. Conf. on Data Warehousing and Knowledge Discovery, Sept. 1999.
- F. Bonchi , F. Giannotti, G. Mainetto, D. Pedreschi. A Classification-based Methodology for Planning Audit Strategies in Fraud Detection. In Proc. KDD-99, ACM-SIGKDD Int. Conf. on Knowledge Discovery & Data Mining, Aug. 1999.
- J. Catlett. *Megainduction: machine learning on very large databases*. PhD Thesis, Univ. Sydney, 1991.
- P. K. Chan and S. J. Stolfo. Metalearning for multistrategy and parallel learning. In Proc. 2nd Int. Conf. on Information and Knowledge Management, p. 314-323, 1993.
- J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufman, 1993.
- J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81-106, 1986.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. In Proc. KDD'95, August 1995.

References - classification

- J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest: A framework for fast decision tree construction of large datasets. In Proc. 1998 Int. Conf. Very Large Data Bases, pages 416-427, New York, NY, August 1998.
- B. Liu, W. Hsu and Y. Ma. Integrating classification and association rule mining. In Proc. KDD'98, New York, 1998.
- J. Magidson. The CHAID approach to segmentation modeling: Chi-squared automatic interaction detection. In R. P. Bagozzi, editor, *Advanced Methods of Marketing Research*, pages 118-159. Blackwell Business, Cambridge Massechusetts, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. In Proc. 1996 Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, March 1996.
- S. K. Murthy, Automatic Construction of Decision Trees from Data: A Multi-Diciplinary Survey. *Data Mining and Knowledge Discovery* 2(4): 345-389, 1998
- J. R. Quinlan. Bagging, boosting, and C4.5. In Proc. 13th Natl. Conf. on Artificial Intelligence (AAAI'96), 725-730, Portland, OR, Aug. 1996.
- R. Rastogi and K. Shim. Public: A decision tree classifer that integrates building and pruning. In Proc. 1998 Int. Conf. Very Large Data Bases, 404-415, New York, NY, August 1998.

References - classification

- J. Shafer, R. Agrawal, and M. Mehta. SPRINT : A scalable parallel classifier for data mining. In Proc. 1996 Int. Conf. Very Large Data Bases, 544-555, Bombay, India, Sept. 1996.
- S. M. Weiss and C. A. Kulikowski. *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Morgan Kaufman, 1991.
- D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning internal representation by error propagation. In D. E. Rumelhart and J. L. McClelland (eds.) *Parallel Distributed Processing*. The MIT Press, 1986