# Classification problem

www.mimuw.edu.pl/~son/datamining/DHCN/DM

Nguyen Hung Son

*This presentation was prepared on the basis of the following public materials:*

1. Jiawei Han and Micheline Kamber, „Data mining, concept and techniques" http://www.cs.sfu.ca
2. Gregory Piatetsky-Shapiro, „kdnuggest", http://www.kdnuggets.com/data_mining_course/

## Lecture plan

- Introduction to classification
- Overview of classification techniques
- K-NN
- Naive Bayes and Bayesian networks
- Classification accuracy

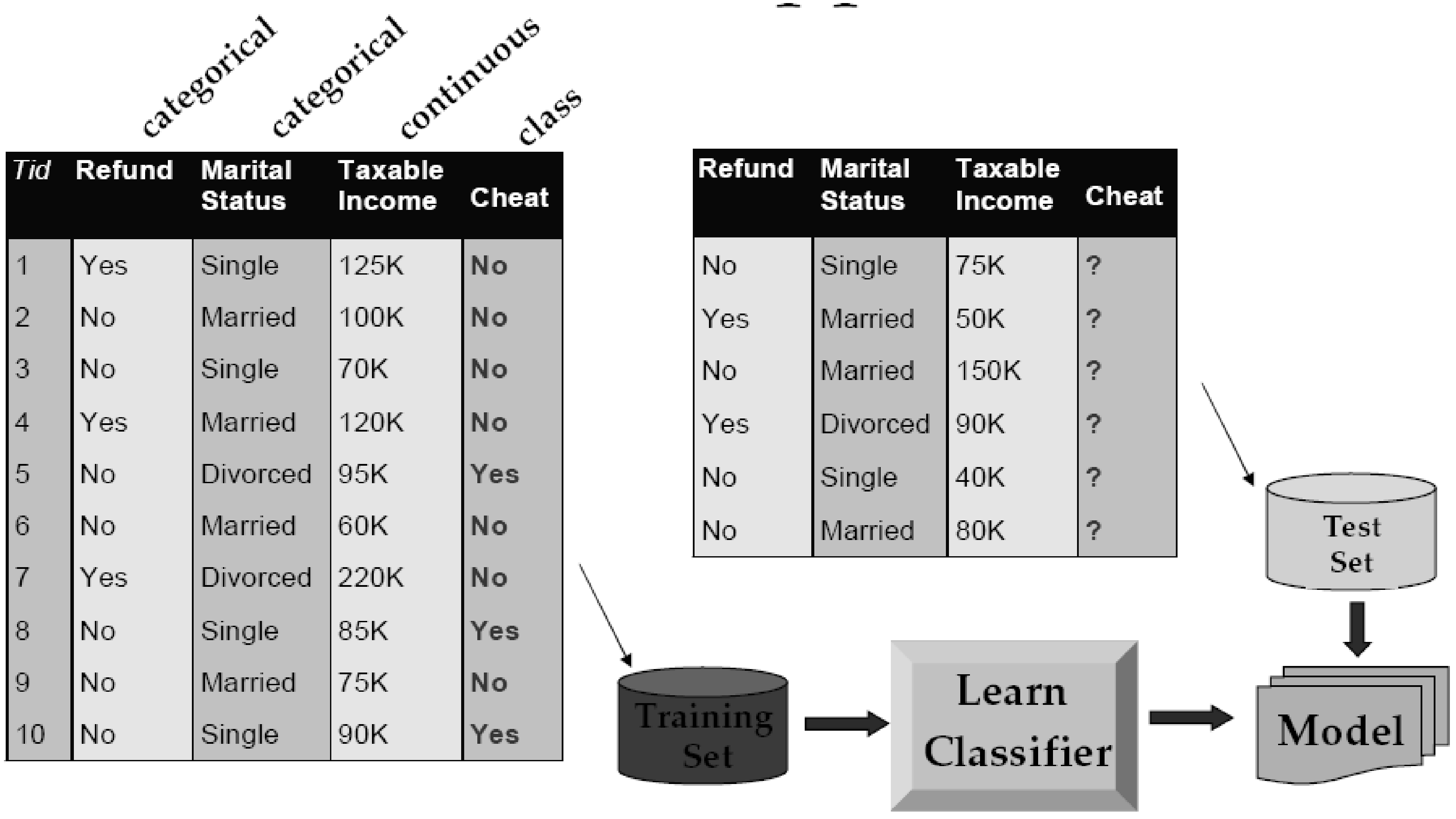
## What is classification

- Given a set of records (called the training set)
  - Each record contains a set of attributes
  - One of the attributes is the class
- Find a model for the class attribute as a function of the values of other attributes
- Goal: Previously unseen records should be assigned to a class as accurately as possible
  - Usually, the given data set is divided into training and test set, with training set used to build the model and test set used to validate it. The accuracy of the model is determined on the test set.

## Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

# General approach



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

| Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|
| No | Single | 75K | ? |
| Yes | Married | 50K | ? |
| No | Married | 150K | ? |
| Yes | Divorced | 90K | ? |
| No | Single | 40K | ? |
| No | Married | 80K | ? |

## Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Memory based reasoning
- Neural Networks
- Genetic Algorithms
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- …

## Classification by Decision Tree Induction

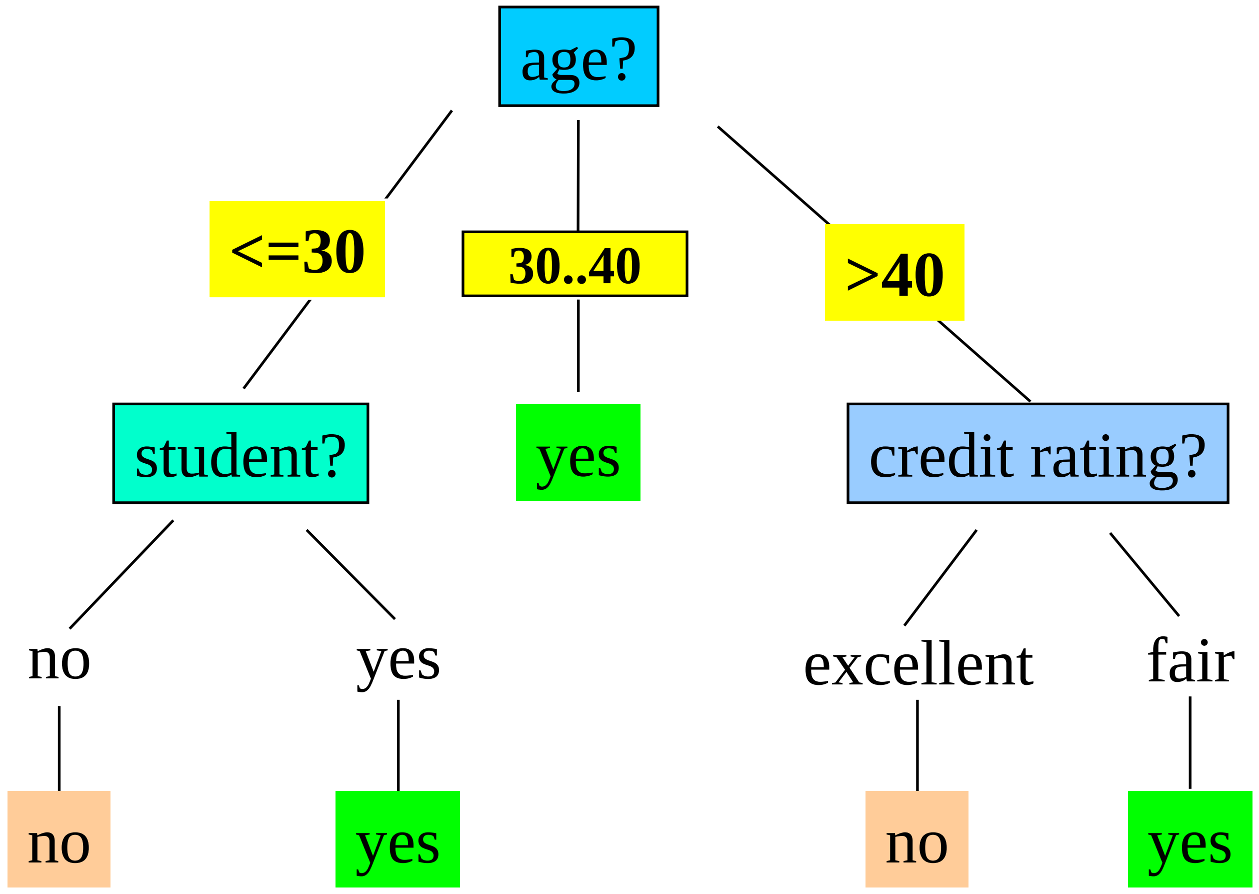
- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases
  - Tree construction
    - At start, all the training examples are at the root
    - Partition examples recursively based on selected attributes
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Classifying an unknown sample
  - Test the attribute values of the sample against the decision tree

# Training Dataset

This follows an example from Quinlan's ID3

| age | income | student | credit_rating |
|---|---|---|---|
| <=30 | high | no | fair |
| <=30 | high | no | excellent |
| 31…40 | high | no | fair |
| >40 | medium | no | fair |
| >40 | low | yes | fair |
| >40 | low | yes | excellent |
| 31…40 | low | yes | excellent |
| <=30 | medium | no | fair |
| <=30 | low | yes | fair |
| >40 | medium | yes | fair |
| <=30 | medium | yes | excellent |
| 31…40 | medium | no | excellent |
| 31…40 | high | yes | fair |
| >40 | medium | no | excellent |

# Output: A Decision Tree for "*buys_computer*"



## Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

## Attribute Selection Measure

- Information gain (ID3/C4.5)
  - All attributes are assumed to be categorical
  - Can be modified for continuous-valued attributes
- Gini index (IBM IntelligentMiner)
  - All attributes are assumed continuous-valued
  - Assume there exist several possible split values for each attribute
  - May need other tools, such as clustering, to get the possible split values
  - Can be modified for categorical attributes

## Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, $P$ and $N$
  - Let the set of examples $S$ contain $p$ elements of class $P$ and $n$ elements of class $N$
  - The amount of information, needed to decide if an arbitrary example in $S$ belongs to $P$ or $N$ is defined as

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

## Information Gain in Decision Tree Induction

- Assume that using attribute A a set $S$ will be partitioned into sets $\{S_1, S_2, \dots, S_v\}$
  - If $S_i$ contains $p_i$ examples of $P$ and $n_i$ examples of $N$, the entropy, or the expected information needed to classify objects in all subtrees $S_i$ is

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

- The encoding information that would be gained by branching on $A$

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

# Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- I(p, n) = I(9, 5) =0.940
- Compute the entropy for *age*:

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

Hence

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

| age | $p_i$ | $n_i$ | $I(p_i, n_i)$ |
|---|---|---|---|
| <=30 | 2 | 3 | 0.971 |
| 30…40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |

Similarly

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

## *Gini* Index (IBM IntelligentMiner)

- If a data set $T$ contains examples from $n$ classes, gini index, $gini(T)$ is defined as

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

where $p_j$ is the relative frequency of class $j$ in $T$.

- If a data set $T$ is split into two subsets $T_1$ and $T_2$ with sizes $N_1$ and $N_2$ respectively, the *gini* index of the split data contains examples from $n$ classes, the *gini* index $gini(T)$ is defined as

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- The attribute provides the smallest $gini_{split}(T)$ is chosen to split the node (*need to enumerate all possible splitting points for each attribute*).

# Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path 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 easier for humans to understand
- Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"
IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
IF *age* = "31…40" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

## Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

## Enhancements to basic decision tree induction

- Allow for continuous-valued attributes
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle missing attribute values
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- Attribute construction
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

# Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why decision tree induction in data mining?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods

# Scalable Decision Tree Induction Methods in Data Mining Studies

- SLIQ (EDBT'96 — Mehta et al.)
  - builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - separates the scalability aspects from the criteria that determine the quality of the tree
  - builds an AVC-list (attribute, value, class label)

## Other Classification Methods

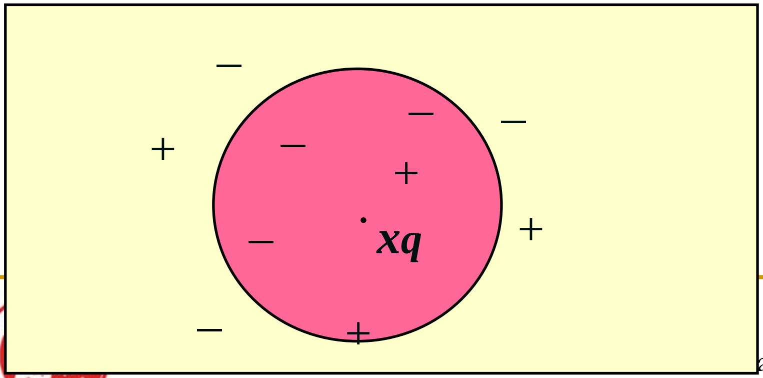
- k-nearest neighbor classifier
- case-based reasoning
- Genetic algorithm
- Rough set approach
- Fuzzy set approaches

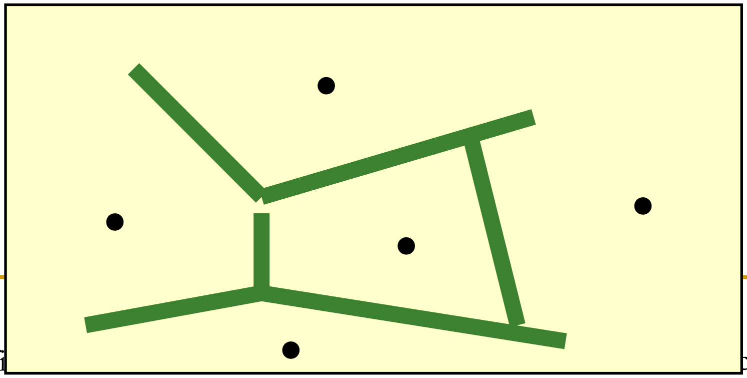
# Instance-Based Methods

- Instance-based learning:
  - Store training examples and delay the processing ("lazy evaluation") until a new instance must be classified
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

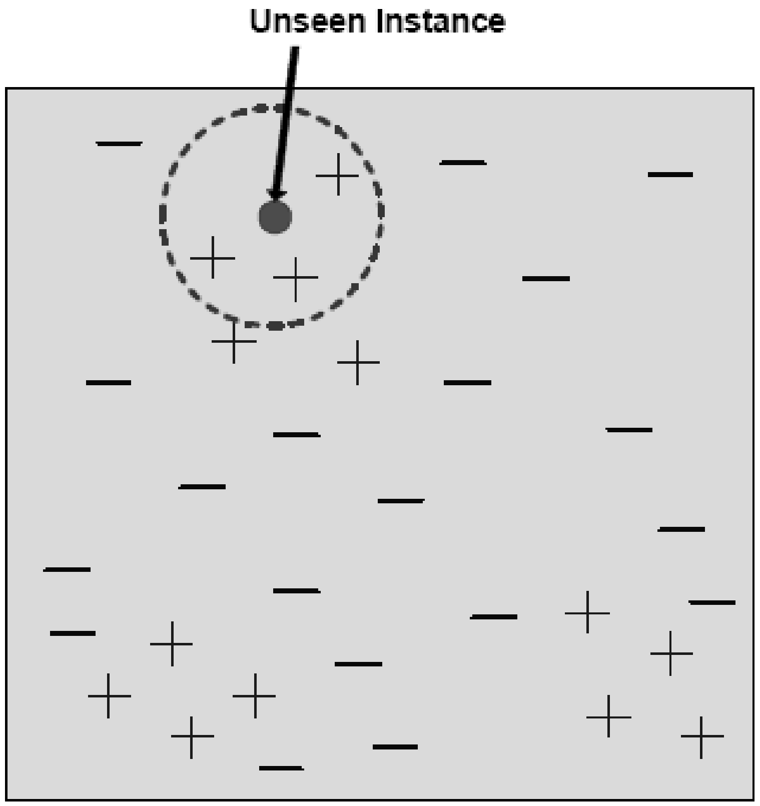
# The *k*-Nearest Neighbor Algorithm

- All instances correspond to points in the n-D space.
- The nearest neighbor are defined in terms of Euclidean distance.
- The target function could be discrete- or real- valued.
- For discrete-valued, the *k*-NN returns the most common value among the k training examples nearest to $x_q$.
- Vonoroi diagram: the decision surface induced by 1-NN for a typical set of training examples.





# Nearest neighbours classifiers



- For data set with continuous attributes
- Requires
  - Set of sorted instances
  - Distance metric to compute distance between instances
  - Value of *k*, the number of nearest neighbors to retrieve
- For classification
  - Retrieve the *k* nearest neighbors
  - Use class labels of the nearest neighbors to determine the class label of the unseen instance (e.g. by taking majority vote)

# Nearest neighbours classifiers

- If $k = 1$, we can illustrate the decision boundary of each class by using a *Voronoi diagram*
- Distance metric is required to compute the distance between two instances
- A nearest neighbor classifier represents each instance as a data point embedded in a $d$ dimensional space, where $d$ is the number of continuous attributes
- Euclidean Distance

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

- Weighted Distance
  - Weight factor, $w = 1/d^2$
  - Weight the vote according to the distance
- Other popular metric: *city-block (Manhattan) metric*

## Other issues

- Nearest neighbor classifiers are lazy learners
  - It does not build models explicitly
  - Classifying instances is expensive
- Scaling of attributes
  - Person:
    - (Height in meters, Weight in pounds, Class)
    - Height may vary from 1.5 m to 1.85 m
    - Weight may vary from 45 kg to 120 kg
  - Distance measure could be dominated by difference in weights

## Nearsest neighbours classifiers

- If *k* is too small, classifier is sensitive to noise points
- If *k* is too large –
  - Computationally intensive
  - Neighborhood may include points from other classes
- Problems with Euclidean distance
  - High dimensional data
    - Curse of dimensionality
  - Can produce counter intuitive results (e.g. text document classification)
    - Solution: Normalization

## Discussion on the *k*-NN Algorithm

- The k-NN algorithm for continuous-valued target functions
  - Calculate the mean values of the *k* nearest neighbors
- Distance-weighted nearest neighbor algorithm
  - Weight the contribution of each of the k neighbors according to their distance to the query point $x_q$
    - giving greater weight to closer neighbors
  - Similarly, for real-valued target functions

$$w \equiv \frac{1}{d(x_q, x_i)^2}$$

- Robust to noisy data by averaging k-nearest neighbors
- Curse of dimensionality: distance between neighbors could be dominated by irrelevant attributes.
  - To overcome it, axes stretch or elimination of the least relevant attributes.

## Case-Based Reasoning

- Also uses: lazy evaluation + analyze similar instances
- Difference: Instances are not “points in a Euclidean space”
- Example: Water faucet problem in CADET (Sycara et al’92)
- Methodology
  - Instances represented by rich symbolic descriptions (e.g., function graphs)
  - Multiple retrieved cases may be combined
  - Tight coupling between case retrieval, knowledge-based reasoning, and problem solving
- Research issues
  - Indexing based on syntactic similarity measure,  and when failure, backtracking, and adapting to additional cases

# Eager vs. Lazy learning

***eager learners***

- intend to learn the model as soon as possible, once the training data is available:
  - An *inductive step* for constructing classification models from data
  - A *deductive step* for applying the derived model to previously unseen instances
- For decision tree induction and rule based classifiers, the models are constructed immediately after the training set is provided

***lazy learners***

- An opposite strategy would be to delay the process of generalizing the training data until it is needed to classify the unseen instances
- An example of lazy learner is the *Rote Classifier*, which memorizes the entire training data and perform classification only if the attributes of a test instance matches one of the training instances exactly

## Remarks on Lazy vs. Eager Learning

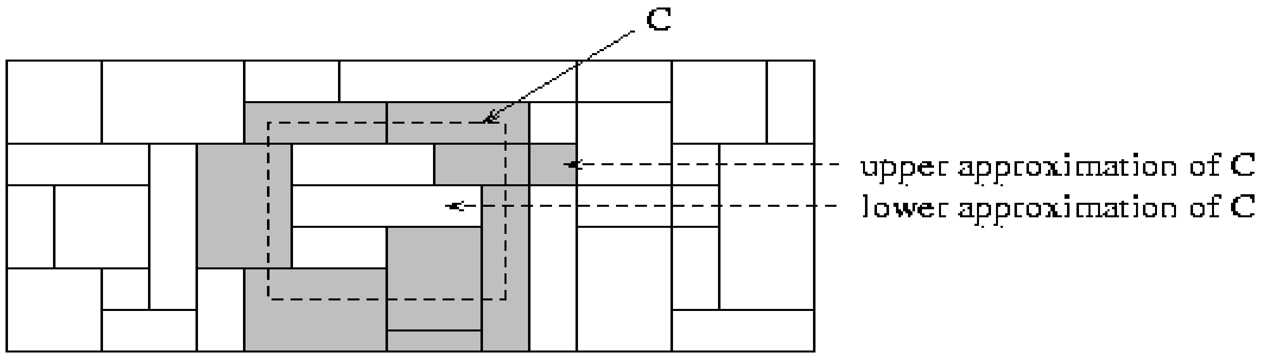
- Instance-based learning: lazy evaluation
- Decision-tree and Bayesian classification: eager evaluation
- Key differences
  - Lazy method may consider query instance *xq* when deciding how to generalize beyond the training data *D*
  - Eager method cannot since they have already chosen global approximation when seeing the query
- Efficiency: Lazy - less time training but more time predicting
- Accuracy
  - Lazy method effectively uses a richer hypothesis space since it uses many local linear functions to form its implicit global approximation to the target function
  - Eager: must commit to a single hypothesis that covers the entire instance space

## Genetic Algorithms

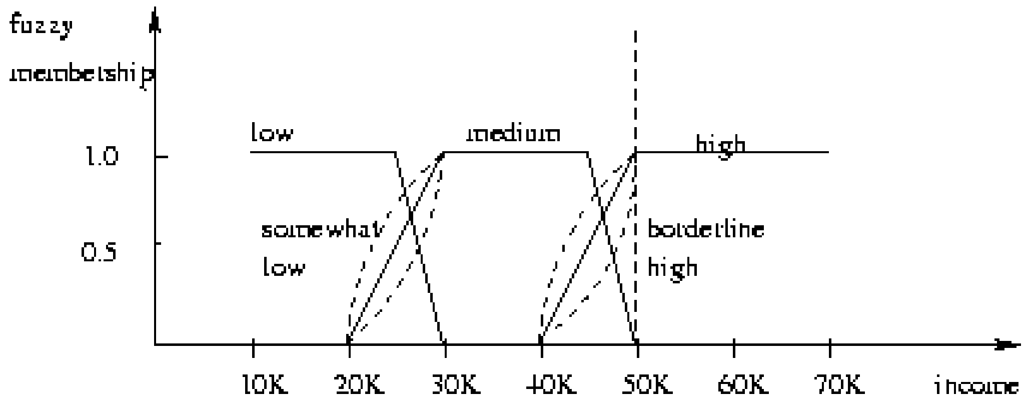
- GA: based on an analogy to biological evolution
- Each rule is represented by a string of bits
- An initial population is created consisting of randomly generated rules
  - e.g., IF $A_1$ and Not $A_2$ then $C_2$ can be encoded as 100
- Based on the notion of survival of the fittest, a new population is formed to consists of the fittest rules and their offsprings
- The fitness of a rule is represented by its classification accuracy on a set of training examples
- Offsprings are generated by crossover and mutation

# Rough Set Approach

- Rough sets are used to approximately or "roughly" define equivalent classes
- A rough set for a given class C is approximated by two sets: a lower approximation (certain to be in C) and an upper approximation (cannot be described as not belonging to C)
- Finding the minimal subsets (reducts) of attributes (for feature reduction) is NP-hard but a discernibility matrix is used to reduce the computation intensity



## Fuzzy Set Approaches



- Fuzzy logic uses truth values between 0.0 and 1.0 to represent the degree of membership (such as using fuzzy membership graph)
- Attribute values are converted to fuzzy values
  - e.g., income is mapped into the discrete categories {low, medium, high} with fuzzy values calculated
- For a given new sample, more than one fuzzy value may apply
- Each applicable rule contributes a vote for membership in the categories
- Typically, the truth values for each predicted category are summed

## Bayesian Classification: Why?

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

## Bayesian Theorem

- Given training data $D$, *posteriori probability of a hypothesis h*, $P(h|D)$ follows the Bayes theorem

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \underset{h \in H}{\arg\max}\, P(h|D) = \underset{h \in H}{\arg\max}\, P(D|h)P(h).$$

- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

## Estimating a-posteriori probabilities

- Bayes theorem:

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

- P(X) is constant for all classes
- P(C) = relative freq of class C samples
- C such that $P(C|X)$ is maximum =

  C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C)$ is unfeasible!

## Bayesian classification

- The classification problem may be formalized using a-posteriori probabilities:
- $P(C|X)$ = prob. that the sample tuple $X=<x_1,\ldots,x_k>$ is of class C.
- E.g. P(class=N | outlook=sunny,windy=true,…)
- Idea: assign to sample X the class label C such that $P(C|X)$ is maximal

## Naïve Bayesian Classification

- Naïve assumption: attribute independence

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

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

## Example

A patient takes a lab test and the result comes back positive. The test returns a correct positive result in 98% of the cases in which the disease is actually present, and a correct negative result in 97% of the cases in which the disease is not present. Furthermore, .008 of all people have this cancer.

| | | | |
|---|---|---|---|
| $P(\text{cancer}) =$ | .008 | $P(\neg\text{cancer}) =$ | .992 |
| $P(+ \mid \text{cancer}) =$ | .98 | $P(- \mid \text{cancer}) =$ | .02 |
| $P(+ \mid \neg\text{cancer}) =$ | .03 | $P(- \mid \neg\text{cancer}) =$ | .97 |

$$P(+) = P(+ \mid c'r)\,P(c'r) + P(+ \mid \neg c'r)\,P(\neg c'r) = .0376$$

$$P(\text{cancer} \mid +) = \frac{P(+ \mid cancer)\,P(cancer)}{P(+)} \quad \mathbf{= .209}$$

# Exercise

Suppose a second test for the same patient returns a positive result as well. What are the posterior probabilities for cancer?

| | | | |
|---|---|---|---|
| P(cancer) = | .008 | P(¬cancer) = | .992 |
| P(+ \| cancer) = | .98 | P(– \| cancer) = | .02 |
| P(+ \| ¬cancer) = | .03 | P(– \| ¬cancer) = | .97 |

$$P(+_1+_2) = P(+_1+_2 \mid c'r)\, P(c'r) + P(+_1+_2 \mid \neg c'r)\, P(\neg c'r) \quad = .00858$$

$$P(\text{cancer} \mid +_1+_2) = \frac{P(+_1+_2 \mid cancer)\, P(cancer)}{P(+_1+_2)} \quad \mathbf{= .896}$$

# Play-tennis example: estimating $P(x_i|C)$

| 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 |

| | |
|---|---|
| **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) = 2/5** |
| windy | |
| **P(true\|p) = 3/9** | **P(true\|n) = 3/5** |
| **P(false\|p) = 6/9** | **P(false\|n) = 2/5** |

## Play-tennis example: classifying X

- An unseen sample X = <rain, hot, high, false>

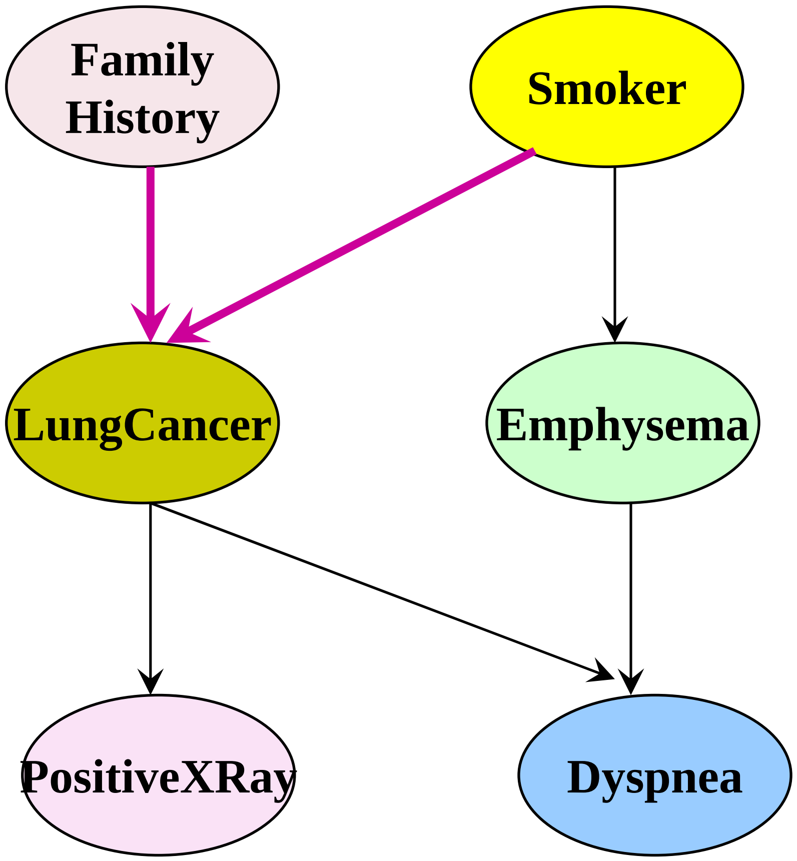
- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286

- Sample X is classified in class n (don't play)

# 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

# Bayesian Belief Networks (I)



Bayesian Belief Networks

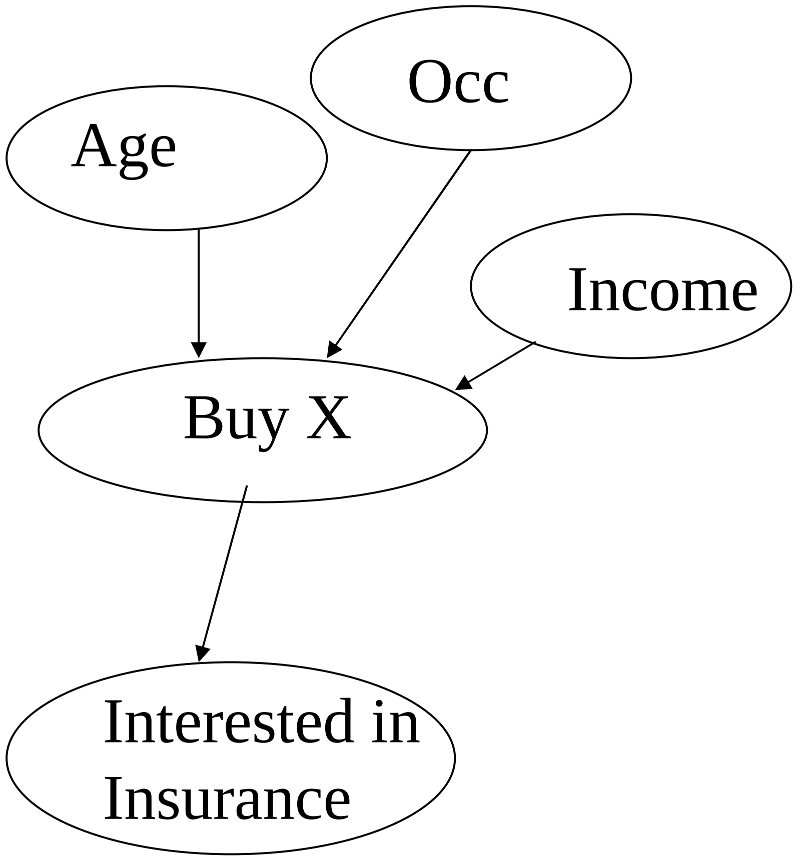
|  | (FH, S) | (FH, ~S) | (~FH, S) | (~FH, ~S) |
|---|---|---|---|---|
| LC | 0.8 | 0.5 | 0.7 | 0.1 |
| ~LC | 0.2 | 0.5 | 0.3 | 0.9 |

**The conditional probability table for the variable LungCancer**

## Bayesian Belief Networks (II)

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal relationships
- Several cases of learning Bayesian belief networks
  - Given both network structure and all the variables: easy
  - Given network structure but only some variables
  - When the network structure is not known in advance

## Example



- Age, Occupation and Income decides whether a costumer buys a product
- If a costumer buys a product then his Interest in insurance is independent with Age, Occupation, Income.

P(Age, Occ, Inc, Buy, Ins ) =
= P(Age)P(Occ)P(Inc)
P(Buy|Age,Occ,Inc)P(Int|Buy)

## Basic formula

$$P(x_1,\ldots.x_n \mid M) = \prod_{i=1}^{n} P(x_i \mid Pa_i, M)$$

$$Pa_i = parent(x_i)$$

## The case of „naive Bayes”

$$\begin{tikzcd}
 & & h \arrow[dll] \arrow[dl] \arrow[dr] & \\
e1 & e2 & \cdots\cdots\cdots & en
\end{tikzcd}$$

$$P(e1, e2, \ldots\ldots en, h) = P(h)\, P(e1 \mid h) \ldots\ldots P(en \mid h)$$

## Classification Accuracy: Estimating Error Rates

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

## Boosting and Bagging

- Boosting increases classification accuracy
  - Applicable to decision trees or Bayesian classifier
- Learn a series of classifiers, where each classifier in the series pays more attention to the examples misclassified by its predecessor
- Boosting requires only linear time and constant space

## Boosting Technique (II) — Algorithm

- Assign every example an equal weight *1/N*
- *For t = 1, 2, …, T Do*
  - Obtain a hypothesis (classifier) $h^{(t)}$ under $w^{(t)}$
  - Calculate the error of *h(t)* and re-weight the examples based on the error
  - Normalize $w^{(t+1)}$ to sum to 1
- Output a weighted sum of all the hypothesis, with each hypothesis weighted according to its accuracy on the training set