Data Mining: Concepts and Techniques

— Chapter 6 —
Classification and Prediction

Slide credits:

Han and Kamber

Tan,Steinbach, Kumar

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods

Motivating Example – Fruit Identification

Skin	Color	Size	Flesh	Conclusion
Hairy	Brown	Large	Hard	safe
Hairy	Green	Large	Hard	Safe
Smooth	Red	Large	Soft	Dangerous
Hairy	Green	Large	Soft	Safe
Smooth	Red	Small	Hard	Dangerous
...				

Classification vs. Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction (Regression)
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

Example – Credit Approval

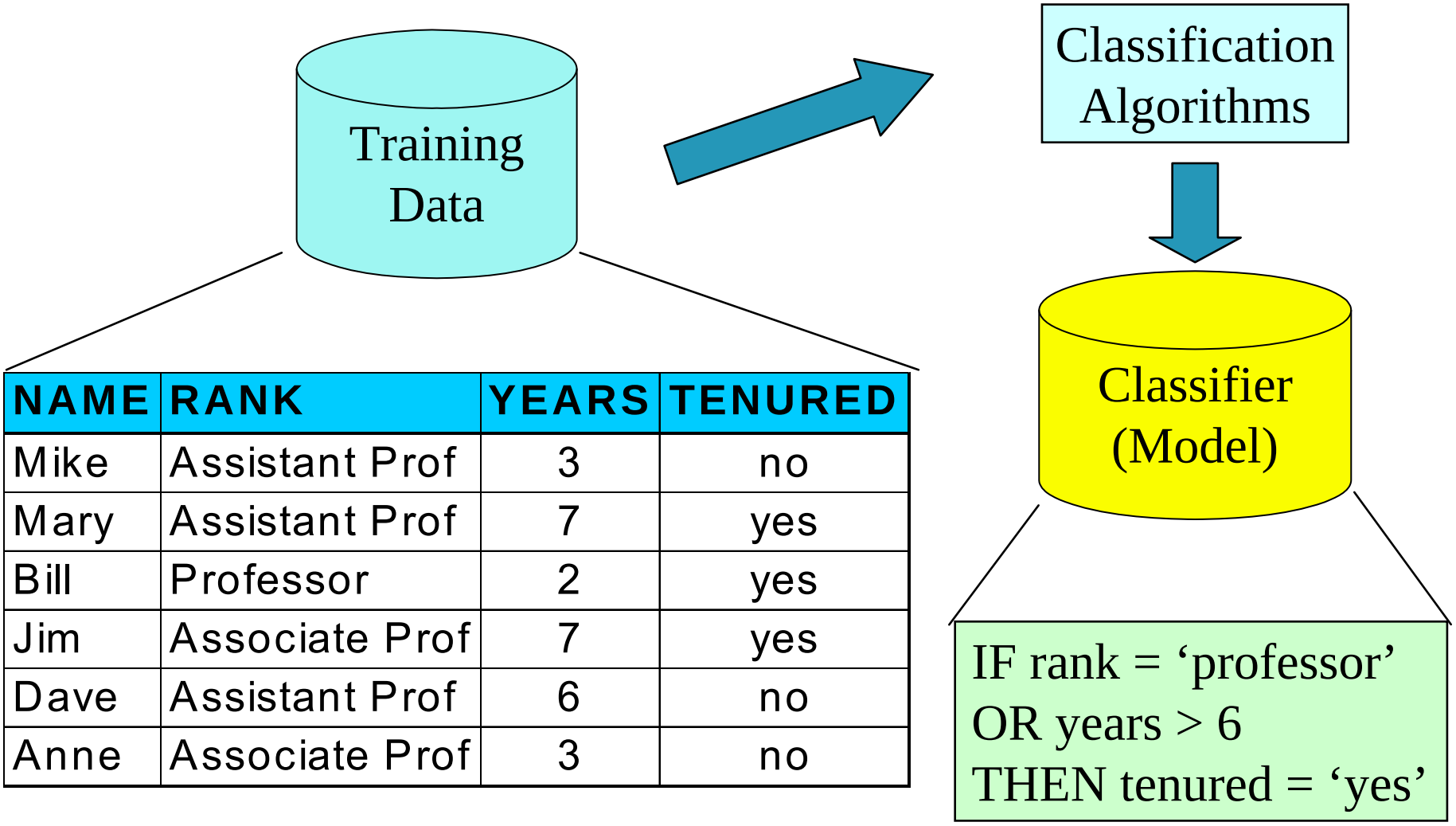
Name	Age	Income	...	Credit
Clark	35	High	...	Excellent
Milton	38	High	...	Excellent
Neo	25	Medium	...	Fair
...	...	...	...	...

- Classification rule:
  - If age = “31...40” and income = high then credit_rating = excellent
- Future customers
  - Paul: age = 35, income = high $\Rightarrow$ excellent credit rating
  - John: age = 20, income = medium $\Rightarrow$ fair credit rating

Classification—A Two-Step Process

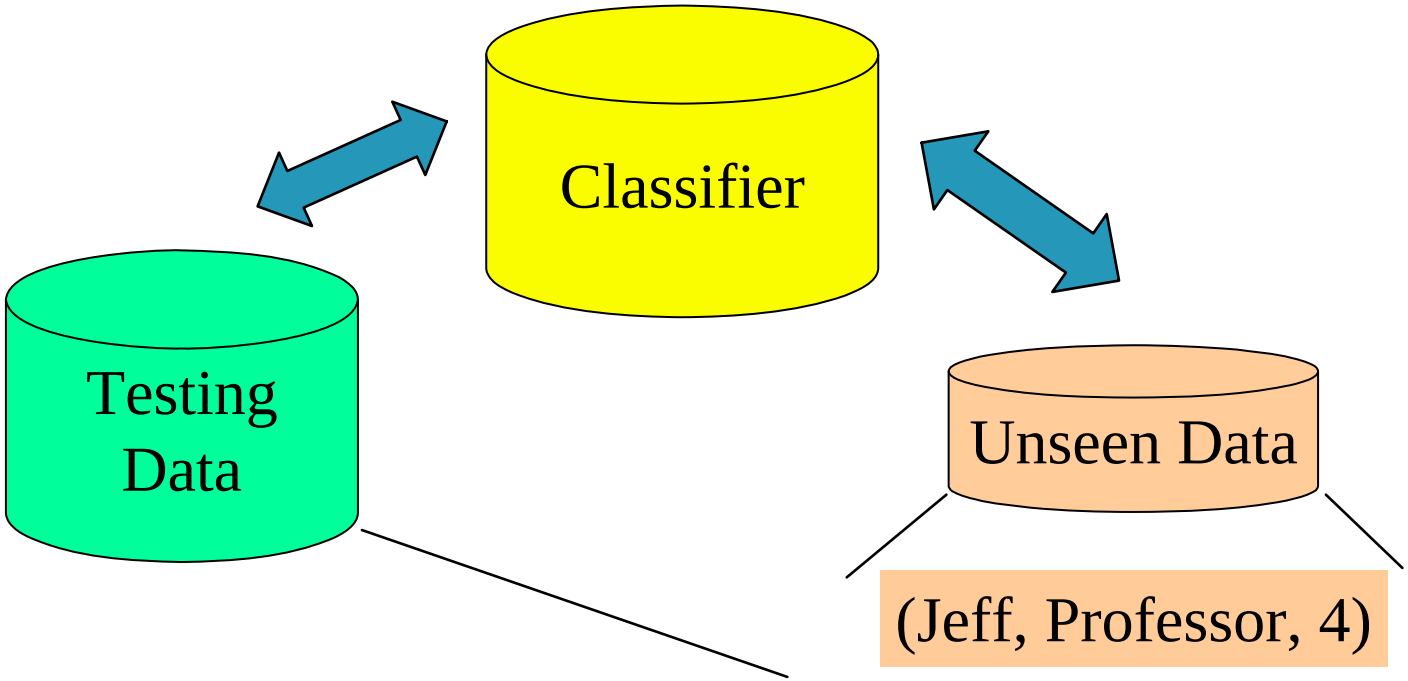
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision 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
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Issues: Data Preparation

- Data cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (feature selection)
  - Remove the irrelevant or redundant attributes
- Data transformation
  - Generalize and/or normalize data

Issues: Evaluating Classification Methods

- Accuracy
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, decision tree size or compactness of classification rules

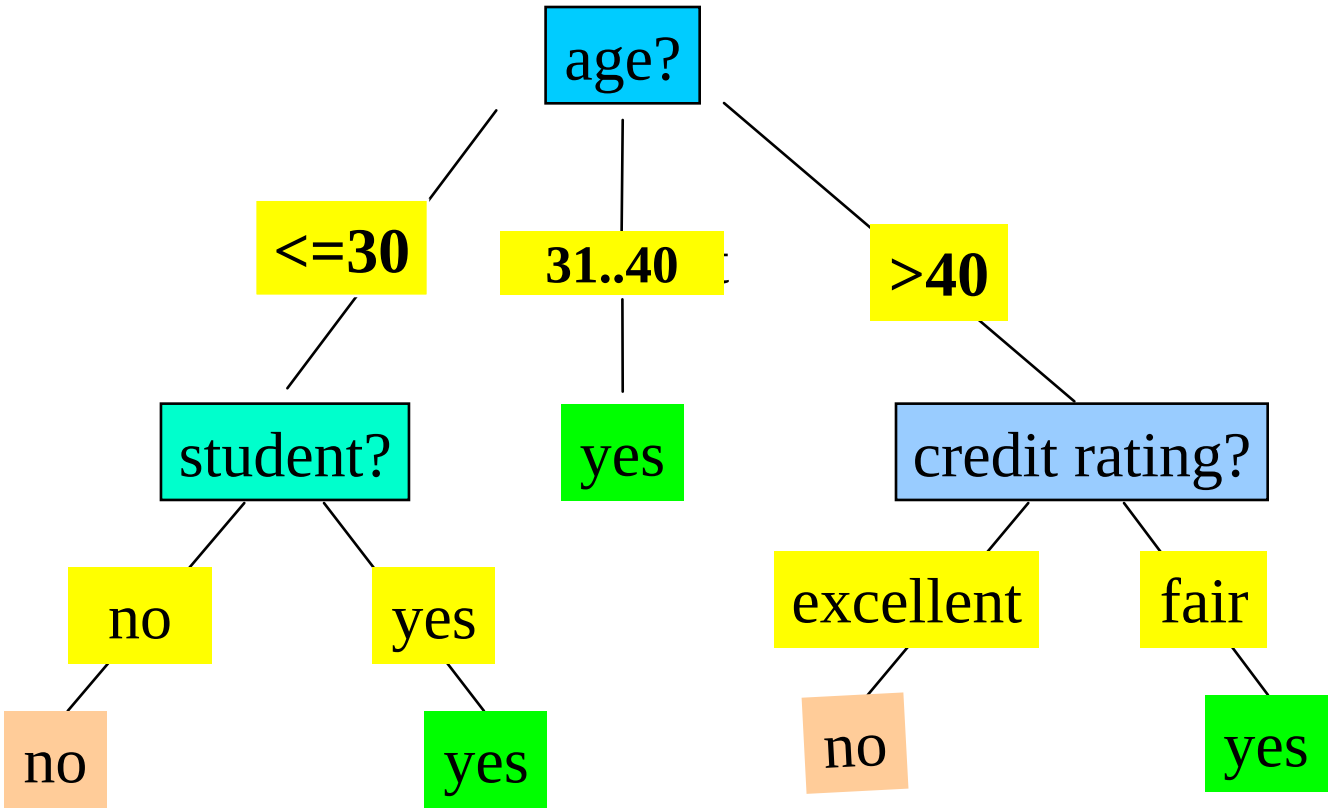
Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods

Training Dataset

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

Output: A Decision Tree for "*buys_computer*"



Algorithm for Decision Tree Induction

- ID3 (Iterative Dichotomiser), C4.5, by Quinlan
  - http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz
- CART (Classification and Regression Trees)
- 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
  - A test attribute is selected that “best” separate the data into partitions
    - Heuristic or statistical measure
  - Samples are partitioned recursively based on selected attributes
- 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 Measures

- Idea: select attribute that partition samples into homogeneous groups
- Measures
  - Information gain (ID3)
  - Gain ratio (C4.5)
  - Gini index (CART)

Attribute Selection Measure: Information Gain (ID3)

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gain – difference between original information requirement and the new information requirement by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Example: Information Gain

- Class P: buys_computer = “yes”,
- Class N: buys_computer = “no”

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

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Information-Gain for Continuous-Value Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Attribute Selection Measure: Gain Ratio (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

  - GainRatio(A) = Gain(A)/SplitInfo(A)
- Ex. $SplitInfo_A(D) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$
  - gain_ratio(income) = 0.029/0.926 = 0.031
- The attribute with the maximum gain ratio is selected as the splitting attribute

Attribute Selection Measure: Gini index (CART)

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

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

  where p_j is the relative frequency of class *j* in *D*
- If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node

Example: Gini index

- Ex. D has 9 tuples in buys_computer = “yes” and 5 in “no”

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in D_1: {low, medium} and 4 in D_2

$$gini_{income \in \{low, medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_1)$$

$$= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2)$$

$$= 0.450$$

$$= Gini_{income \in \{high\}}(D)$$

but $gini_{\{medium,high\}}$ is 0.30 and thus the best since it is the lowest

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

Comparing Attribute Selection Measures

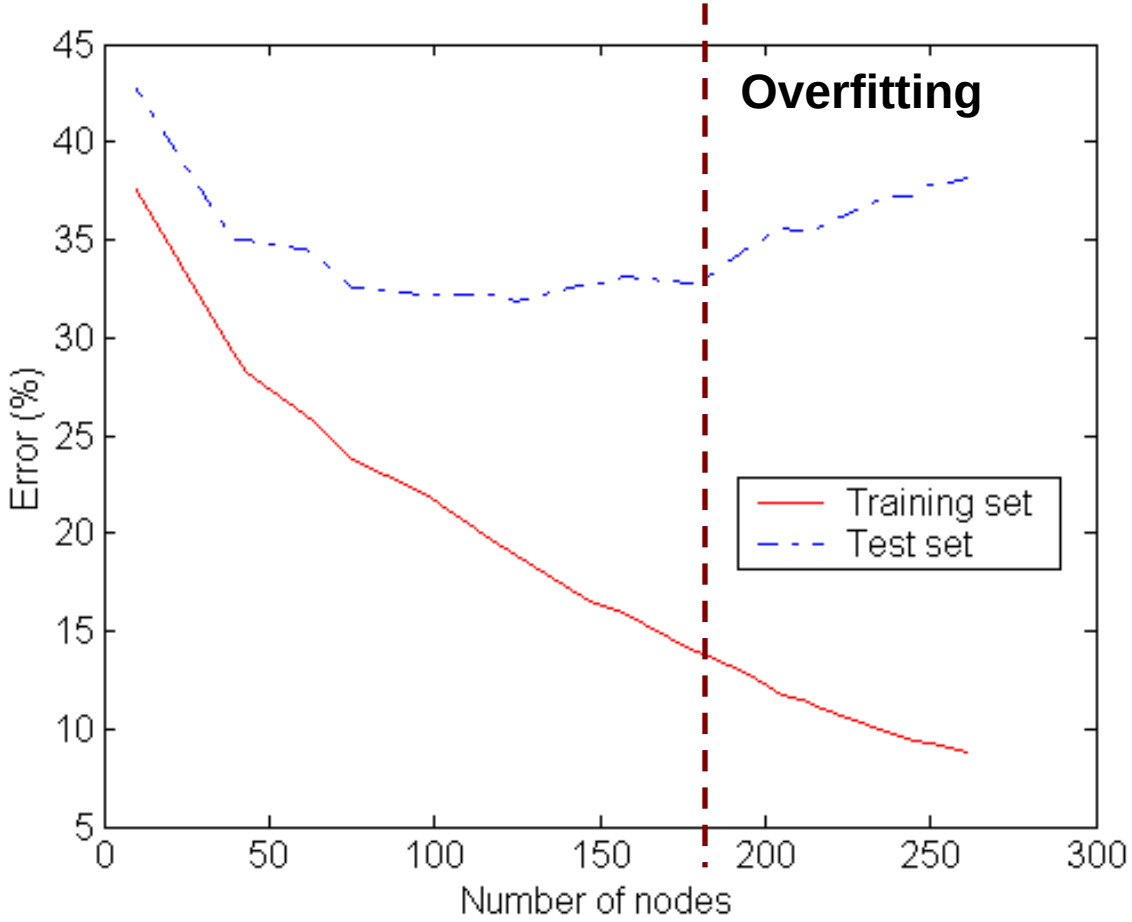
- The three measures, in general, return good results but
  - Information gain:
    - biased towards multivalued attributes
  - Gain ratio:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - Gini index:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
  - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
  - CART: finds multivariate splits based on a linear comb. of attrs.
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior than others

Overfitting

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies and noises



Tree Pruning

- 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
    - Use a set of data different from the training data to decide which is the “best pruned tree”
    - Occam's razor: prefers smaller decision trees (simpler theories) over larger ones

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

Scalable Decision Tree Induction Methods

- 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)
  - Builds an AVC-list (attribute, value, class label)
- BOAT (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples

RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list**: AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X*)
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n*)
  - Set of AVC-sets of all predictor attributes at the node *n*

Rainforest Illustration

Training Examples

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

AVC-set on *Age*

Age	Buy_Computer	
	yes	no
<=30	3	2
31..40	4	0
>40	3	2

AVC-set on *income*

income	Buy_Computer	
	yes	no
high	2	2
medium	4	2
low	3	1

AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

- Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory
- Each subset is used to create a tree, resulting in several trees
- These trees are examined and used to construct a new tree *T'*
  - It turns out that *T'* is very close to the tree that would be generated using the whole data set together
- Adv: requires only two scans of DB, an incremental alg.

Decision Tree: Comments

- 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

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods

Bayesian Classification

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Naïve Bayesian
  - Independence assumption
- Bayesian network
  - Concept
  - Using Bayesian network
  - Training/learning Bayesian network

Bayes' theorem

- Bayes' theorem/rule/law relates the conditional and marginal probabilities of stochastic events
  - P(*H*) is the prior probability of H.
  - P(H|X) is the conditional probability (posteriori probability) of H given *X*.
  - P(*X*|H) is the conditional probability of *X* given H.
  - P(*X*) is the prior probability of *X*

$$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{P(\mathbf{X})}$$

- Cookie example:
  - Bowl 1: 10 chocolate + 30 plain
  - Bowl 2: 20 chocolate + 20 plain
  - Plain cookie -> the probability the cookie is picked out of bowl 1?

Naïve Bayesian Classifier

- Naïve Bayesian / idiot Bayesian / simple Bayesian
- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector **X** = (x_1, x_2, ..., x_n)
- Suppose there are *m* classes C_1, C_2, ..., C_m.
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$

$$P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$$

- Since P(X) is constant for all classes, maximal $P(\mathbf{X}|C_i)P(C_i)$

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$P(\mathbf{X}|C_i) = \prod_{k=1}^{n} P(x_k|C_i) = P(x_1|C_i) \times P(x_2|C_i) \times ... \times P(x_n|C_i)$$

- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i,D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

and $P(x_k|C_i)$ is

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

Naïve Bayesian Classifier: Example

Class:
C1:buys_computer = ‘yes’
C2:buys_computer = ‘no’

Data sample
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

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

Naïve Bayesian Classifier: Example

- $P(C_i)$: P(buys_computer = "yes") = 9/14 = 0.643
  P(buys_computer = "no") = 5/14= 0.357

- Compute $P(X|C_i)$ for each class
  P(age = "<=30" | buys_computer = "yes") = 2/9 = 0.222
  P(age = "<= 30" | buys_computer = "no") = 3/5 = 0.6
  P(income = "medium" | buys_computer = "yes") = 4/9 = 0.444
  P(income = "medium" | buys_computer = "no") = 2/5 = 0.4
  P(student = "yes" | buys_computer = "yes) = 6/9 = 0.667
  P(student = "yes" | buys_computer = "no") = 1/5 = 0.2
  P(credit_rating = "fair" | buys_computer = "yes") = 6/9 = 0.667
  P(credit_rating = "fair" | buys_computer = "no") = 2/5 = 0.4

- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = "yes") = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = "no") = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
$P(X|C_i)*P(C_i)$: P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

Therefore, X belongs to class ("buys_computer = yes")

Naïve Bayesian Classifier: Comments

- Advantages
  - Fast to train and use
  - Can be highly effective in most of the cases
- Disadvantages
  - Based on a false assumption: class conditional independence - practically, dependencies exist among variables
- Idiot's Bayesian, not so stupid after all? David J. Hand, Keming Yu, International Statistical Review, 2001

- How to deal with dependencies?
  - Bayesian Belief Networks

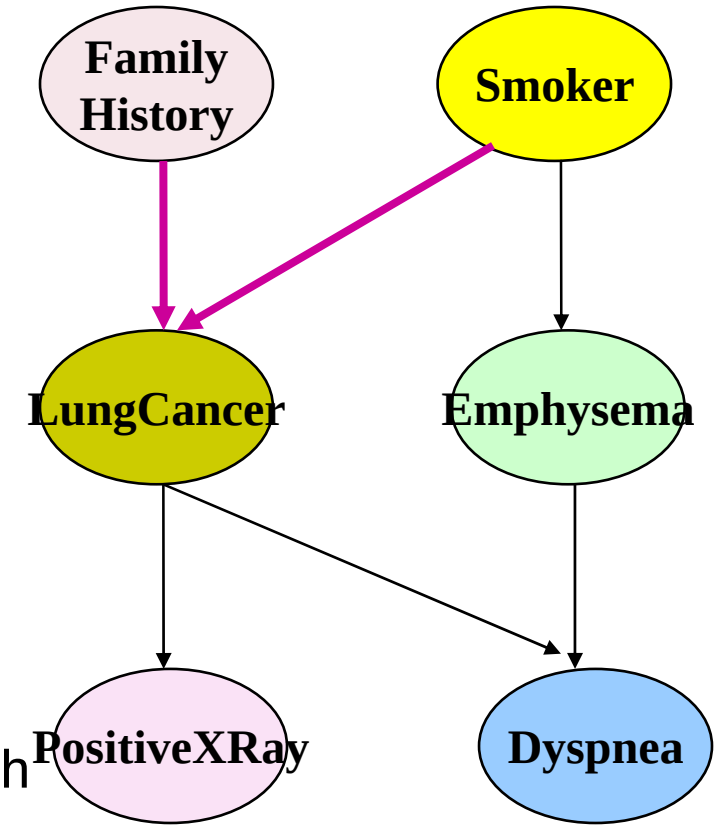
Bayesian Belief Networks – Motivating Example

- Symptoms: difficult to breath
- Patient profile: smoking? age?
- Family history?
- XRay?

Lung Cancer?

Bayesian Belief Networks

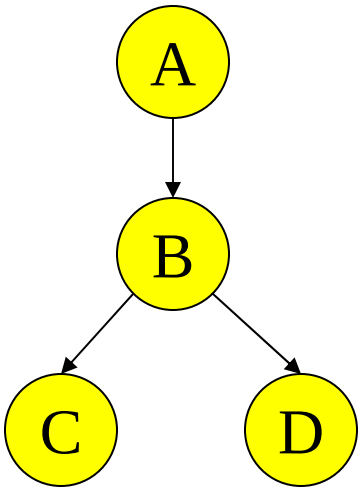
- Bayesian belief networks (belief networks, Bayesian networks, probabilistic networks) is a graphical model that represents a set of variables and their probabilistic independencies
- One of the most significant contribution in AI
- Trained Bayesian networks can be used for classification and reasoning
- Many applications: spam filtering, speech recognition, diagnostic systems



Bayesian Network: Definition

A Bayesian network is made up of:

1. A Directed Acyclic Graph



2. A conditional probability table for each node in the graph

A	P(A)
false	0.6
true	0.4

A	B	P(B|A)
false	false	0.01
false	true	0.99
true	false	0.7
true	true	0.3

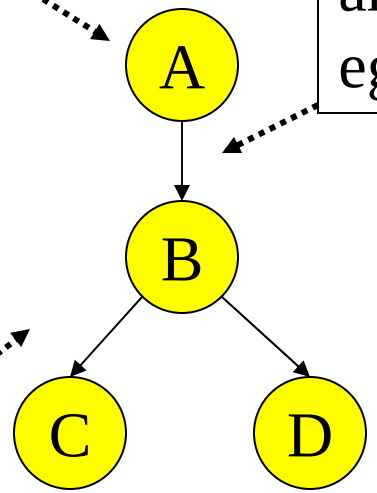
B	D	P(D|B)
false	false	0.02
false	true	0.98
true	false	0.05
true	true	0.95

B	C	P(C|B)
false	false	0.4
false	true	0.6
true	false	0.9
true	true	0.1

Directed Acyclic Graph

Each node in the graph is a random variable

A node *X* is a parent of another node *Y* if there is an arrow from node *X* to node *Y* eg. *A* is a parent of *B*



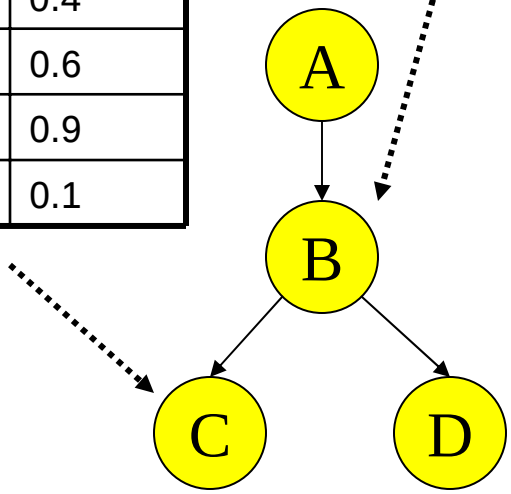
Informally, an arrow from node *X* to node *Y* means *X* has a direct influence on *Y*

Conditional Probability Table

A	P(A)
false	0.6
true	0.4

A	B	P(B|A)
false	false	0.01
false	true	0.99
true	false	0.7
true	true	0.3

B	C	P(C|B)
false	false	0.4
false	true	0.6
true	false	0.9
true	true	0.1



B	D	P(D|B)
false	false	0.02
false	true	0.98
true	false	0.05
true	true	0.95

add up to 1

Each node X_i has a conditional probability distribution $P(X_i \mid \text{Parents}(X_i))$ that quantifies the effect of the parents on the node

For a Boolean variable with k Boolean parents, how many probabilities need to be stored?

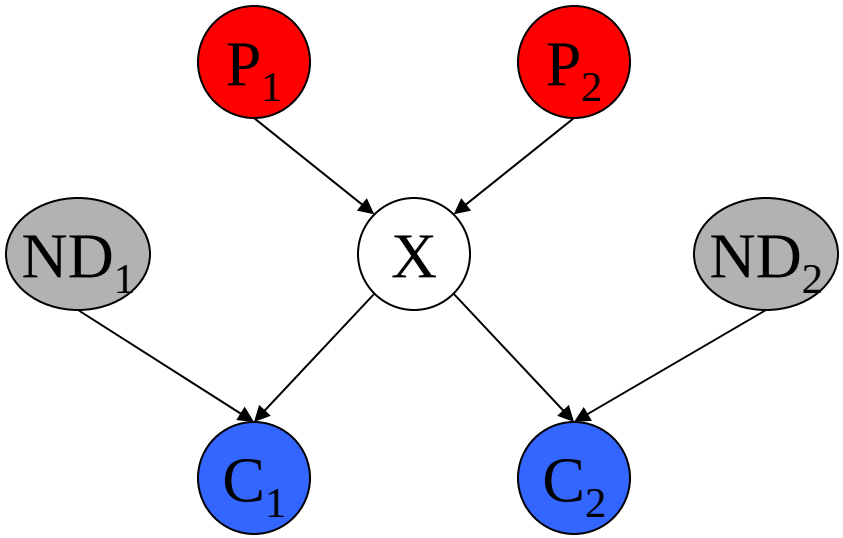
Bayesian Networks: Important Properties

1. Encodes the conditional independence relationships between the variables in the graph structure
2. Is a compact representation of the joint probability distribution over the variables

Conditional Independence

The Markov condition: given its parents (P_1, P_2),

a node (X) is conditionally independent of its non-descendants (ND_1, ND_2)



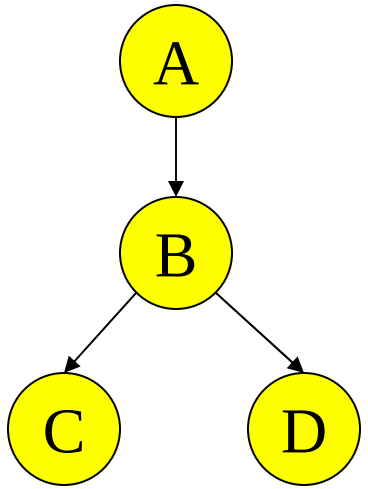
Joint Probability Distribution

Due to the Markov condition, we can compute the joint probability distribution over all the variables X_1, …, X_n in the Bayesian net using the formula:

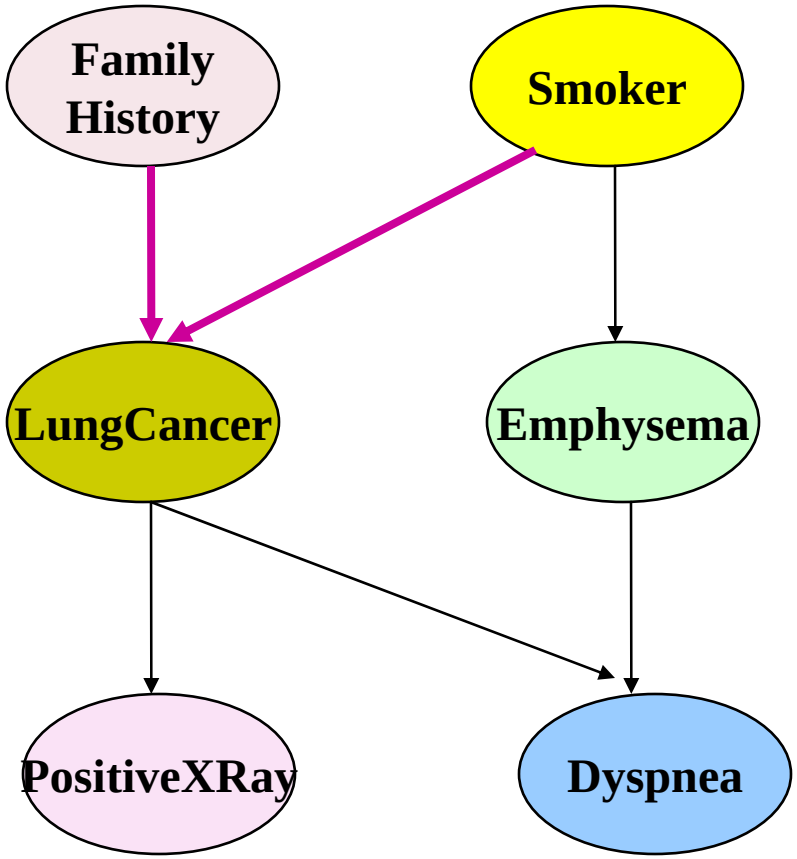
$$P(X_1 = x_1, ..., X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$

Example:

P(A = true, B = true, C = true, D = true)
= P(A = true) * P(B = true | A = true) *
  P(C = true | B = true) P(D = true | B = true)
= (0.4)*(0.3)*(0.1)*(0.95)



Bayesian Networks: Example



Bayesian Belief Networks

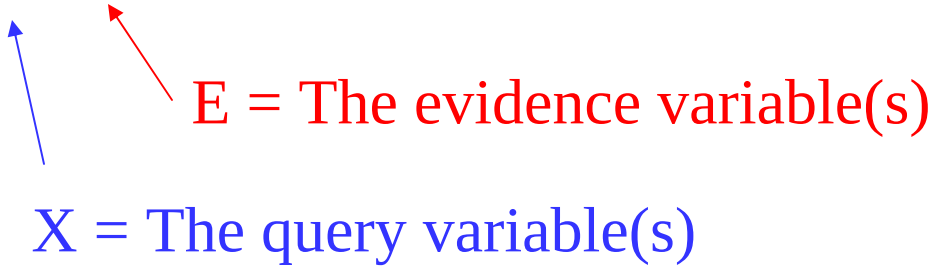
The **conditional probability table** (**CPT**) for variable LungCancer:

	(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

Using the Bayesian Network:
P(LungCancer | Smoker, PXRay, Dyspnea)?

Using Bayesian Network for Inference

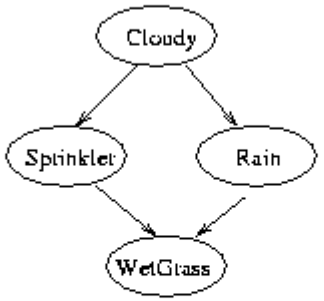
- Using a Bayesian network to compute probabilities is called inference
- General form: P(X | E)



- Exact inference is feasible in small to medium-sized networks
- Exact inference in large networks takes a very long time
  - Approximate inference techniques which are much faster and give pretty good results

Inference Example

P(C=F)	P(C=T)
0.5	0.5



C	P(S=F)	P(S=T)
F	0.5	0.5
T	0.9	0.1

C	P(R=F)	P(R=T)
F	0.8	0.2
T	0.2	0.8

S	R	P(W=F)	P(W=T)
F	F	1.0	0.0
T	F	0.1	0.9
F	T	0.1	0.9
T	T	0.01	0.99

Joint probability:
P(C, S, R, W) = P(C) * P(S|C) * P(R|C) * P(W|S,R)

Suppose the grass is wet, which is more likely?

$$\Pr(S=1|W=1) = \frac{\Pr(S=1,W=1)}{\Pr(W=1)} = \frac{\sum_{c,r}\Pr(C=c,S=1,R=r,W=1)}{\Pr(W=1)} = 0.2781/0.6471 = 0.430$$

$$\Pr(R=1|W=1) = \frac{\Pr(R=1,W=1)}{\Pr(W=1)} = \frac{\sum_{c,s}\Pr(C=c,S=s,R=1,W=1)}{\Pr(W=1)} = 0.4581/0.6471 = 0.708$$

where

$$\Pr(W=1) = \sum_{c,r,s}\Pr(C=c,S=s,R=r,W=1) = 0.6471$$

Training Bayesian Networks

- Several scenarios:
  - Given both the network structure and all variables observable: *learn only the CPTs*
  - Network structure known, some hidden variables: *gradient descent* (greedy hill-climbing) method, analogous to neural network learning
  - Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
  - Unknown structure, all hidden variables: No good algorithms known for this purpose
- Ref. D. Heckerman: Bayesian networks for data mining

Dynamic Bayesian Network

- Dynamic Bayesian network: a Bayesian network that models sequences of variables
  - Hidden Markov Model
- Applications:
  - Temporal data
    - speech recognition
  - Sequential data
    - Natural language text
    - Protein sequences

Related Graphical Models

- Bayesian networks (directed graphical model)
- Markov networks (undirected graphical model)
  - Conditional random field
- Applications:
  - Sequential data
    - Natural language text
    - Protein sequences

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods

Lazy vs. Eager Learning

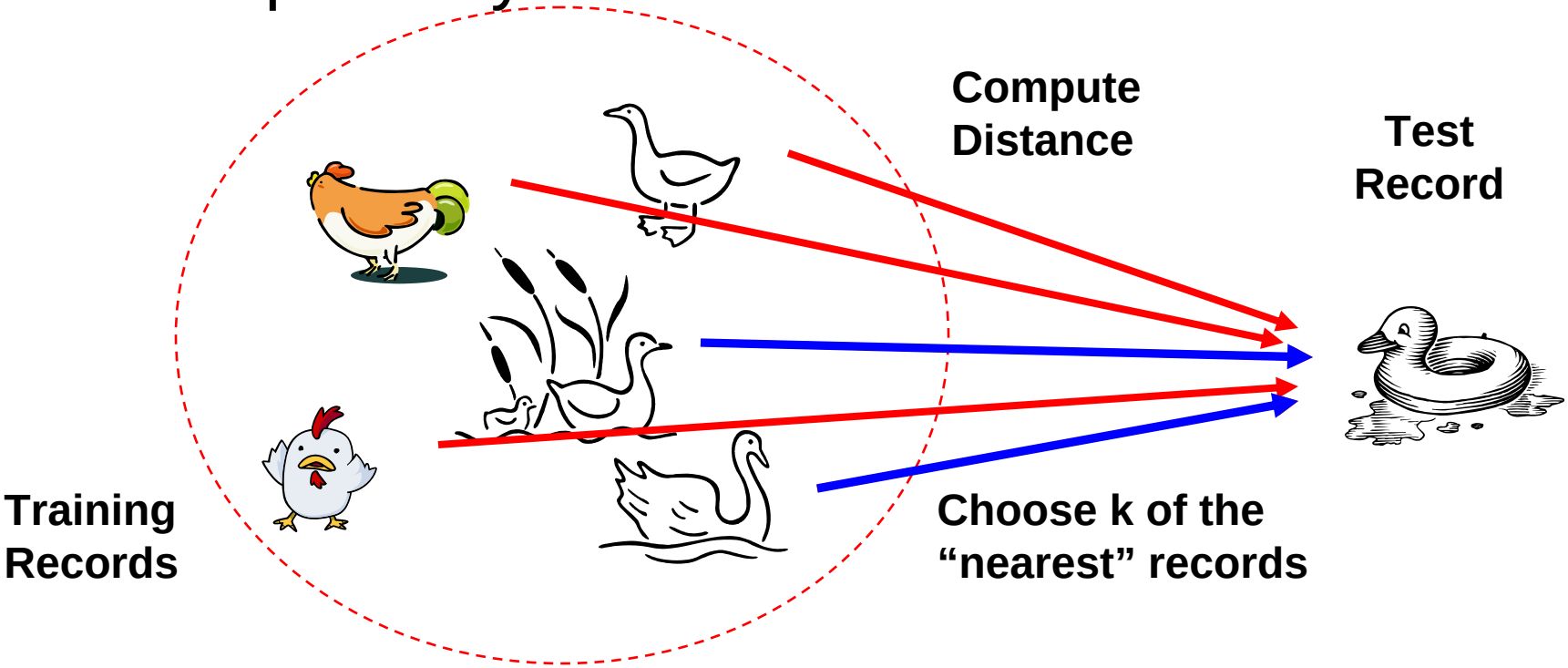
- Lazy vs. eager learning
  - Lazy learning (e.g. instance-based learning): stores training data (or only minor processing) and waits till receiving test data
  - Eager learning (e.g. decision tree, Bayesian): constructs a classification model before receiving test data
- Efficiency
  - Lazy learning: less time in training but more in predicting
  - Eager learning: more time in training but less in predicting
- Accuracy
  - Lazy learning: effectively uses a richer hypothesis space by using many local linear functions to form its implicit global approximation to the target function
  - Eager learning: must commit to a single hypothesis that covers the entire instance space

Lazy Learner: Instance-Based Methods

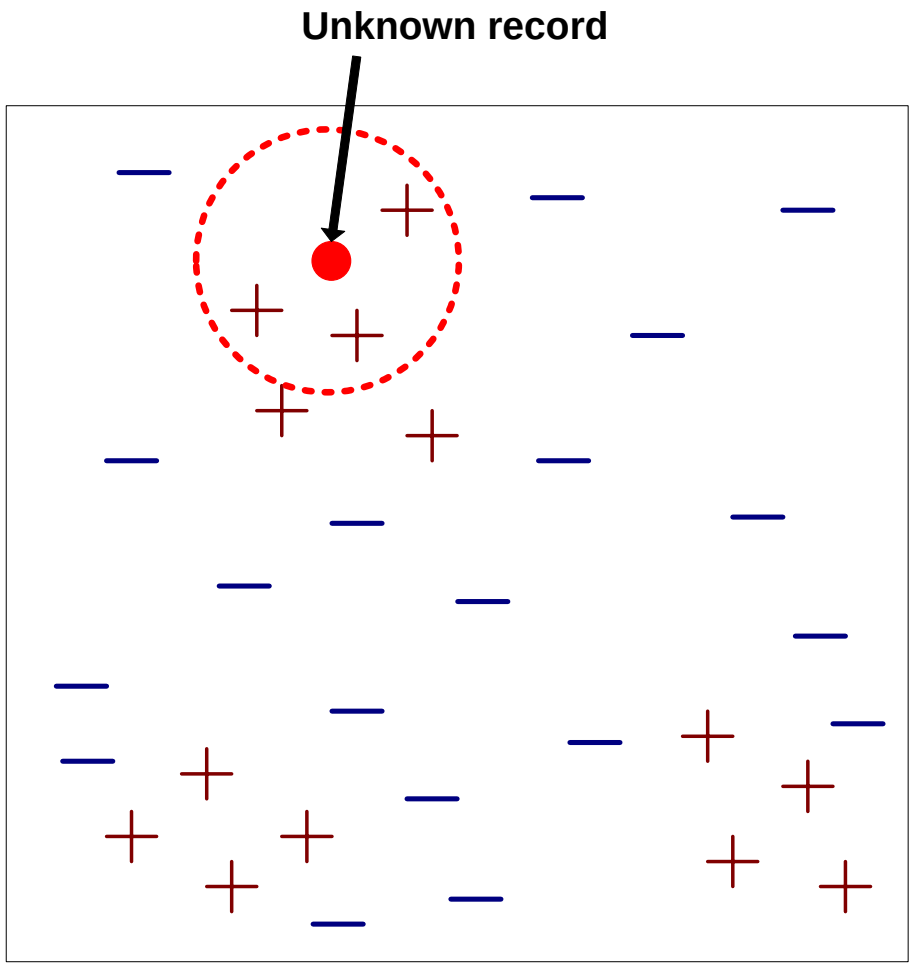
- Typical approaches
  - *k*-nearest neighbor approach
    - Instances represented as points in a Euclidean space.
  - Locally weighted regression
    - Constructs local approximation
  - Case-based reasoning
    - Uses symbolic representations and knowledge-based inference

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



Nearest-Neighbor Classifiers



- Algorithm
  - Compute distance from test record to training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Nearest Neighbor Classification

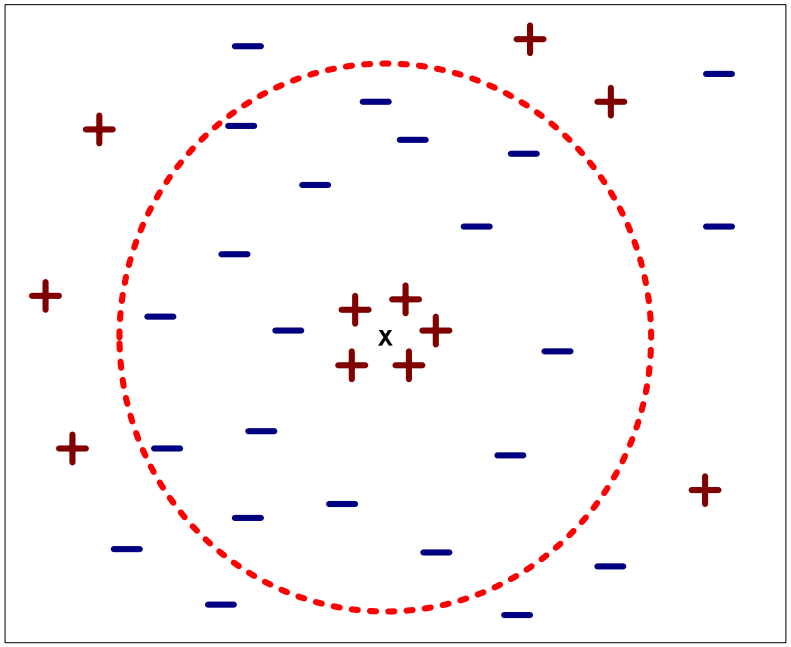
- Compute distance between two points:
  - Euclidean distance

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

- Determine the class from nearest neighbor list
  - take the majority vote of class labels among the k-nearest neighbors
  - Weigh the vote according to distance
    - weight factor, $w = 1/d^2$

Nearest Neighbor Classification

- Choosing the value of k:
  - If k is too small, sensitive to noise points
  - If k is too large, neighborhood may include points from other classes



Nearest Neighbor Classification

- Scaling issues
  - Attributes may have to be scaled to prevent distance measures from being dominated by one of the attributes
  - Example:
    - height of a person may vary from 1.5m to 1.8m
    - weight of a person may vary from 90lb to 300lb
    - income of a person may vary from $10K to $1M
  - Solution?
- Real-valued prediction for a given unknown tuple
  - Returns the mean values of the *k* nearest neighbors

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods

Support Vector Machines: Overview

- A relatively new classification method for both separable and non-separable data
- Features
  - Sound mathematical foundation
  - Training time can be slow but efficient methods are being developed
  - Robust and accurate, less prone to overfitting
- Applications: handwritten digit recognition, speaker identification, ...

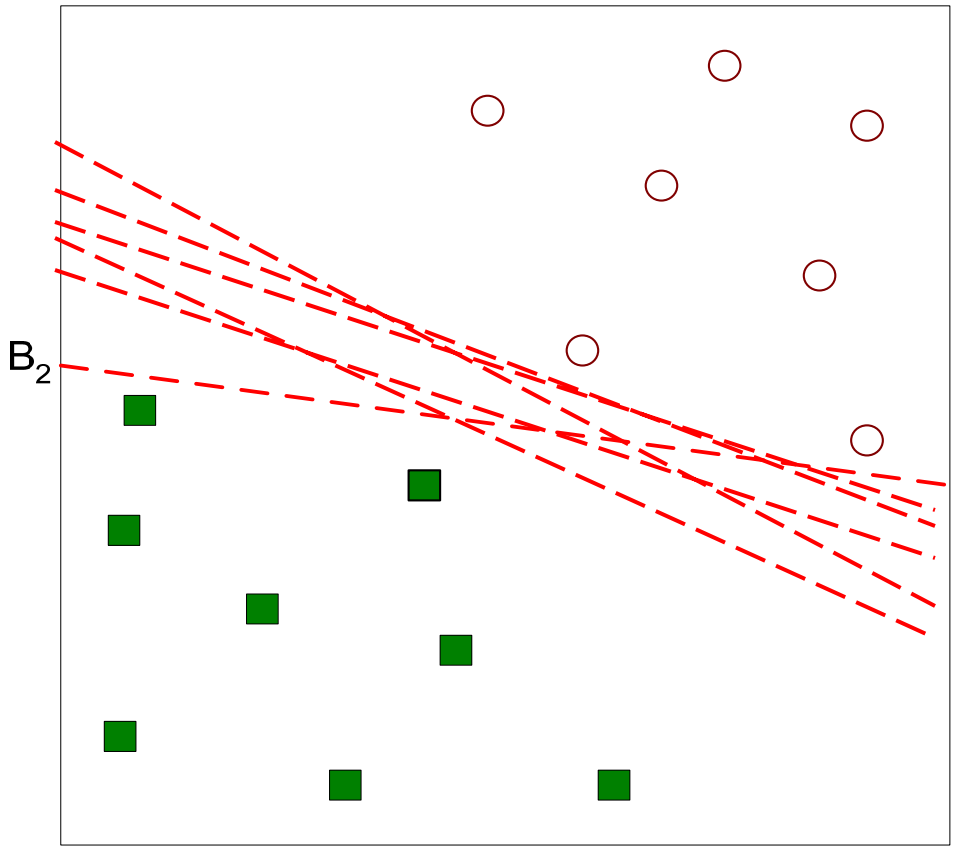
Support Vector Machines: History

- Vapnik and colleagues (1992)
  - Groundwork from Vapnik-Chervonenkis theory (1960 – 1990)
- Problems driving the initial development of SVM
  - Bias variance tradeoff, capacity control, overfitting
  - Basic idea: accuracy on the training set vs. capacity

- A Tutorial on Support Vector Machines for Pattern Recognition, Burges, Data Mining and Knowledge Discovery,1998

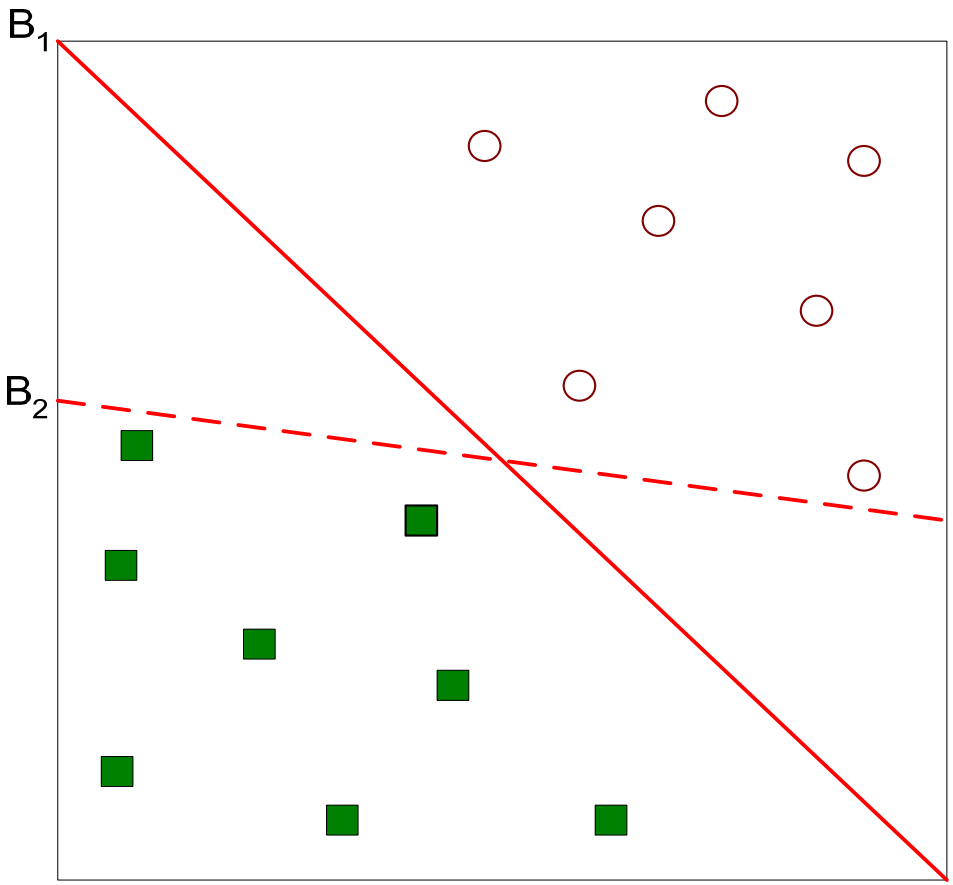
Linear Support Vector Machines

- Problem: find a linear hyperplane (decision boundary) that best separate the data

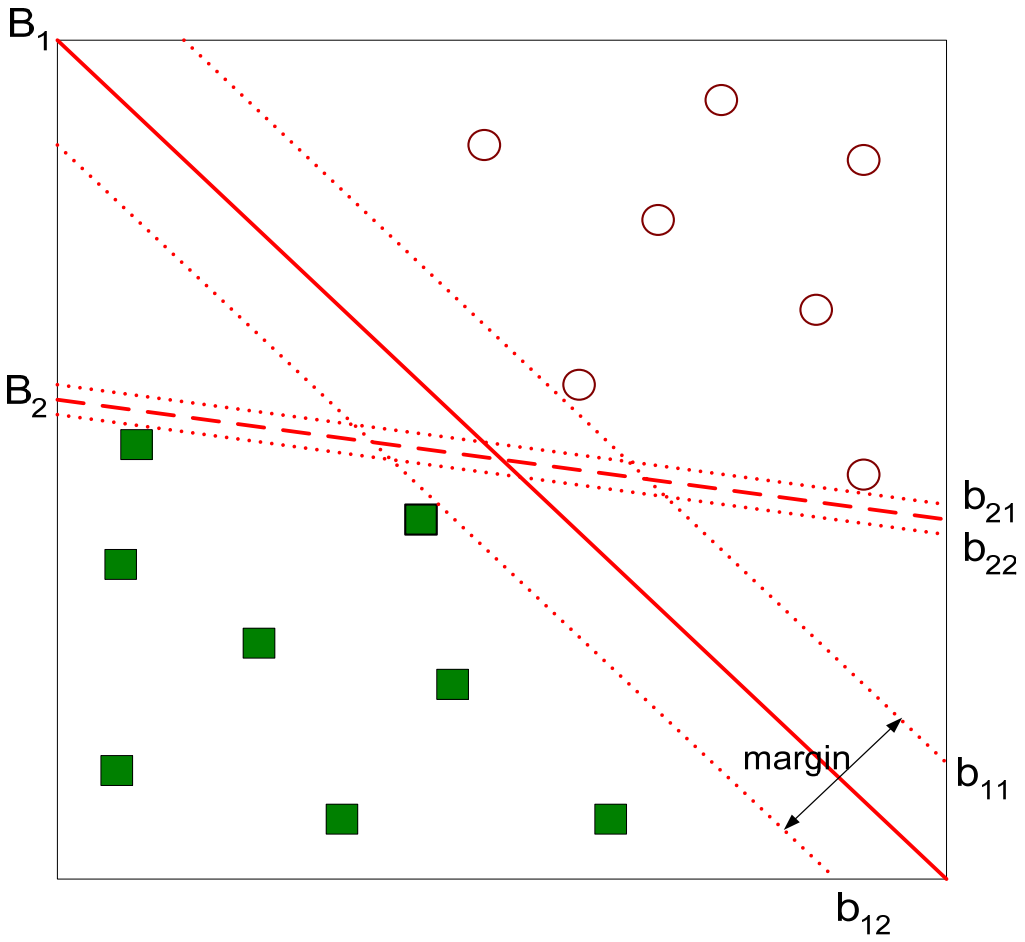


Linear Support Vector Machines

- Which line is better? B1 or B2?
- How do we define better?



Support Vector Machines



- Find hyperplane maximizes the margin

Support Vector Machines Illustration

- A separating hyperplane can be written as

  $$\mathbf{W} \bullet \mathbf{X} + b = 0$$

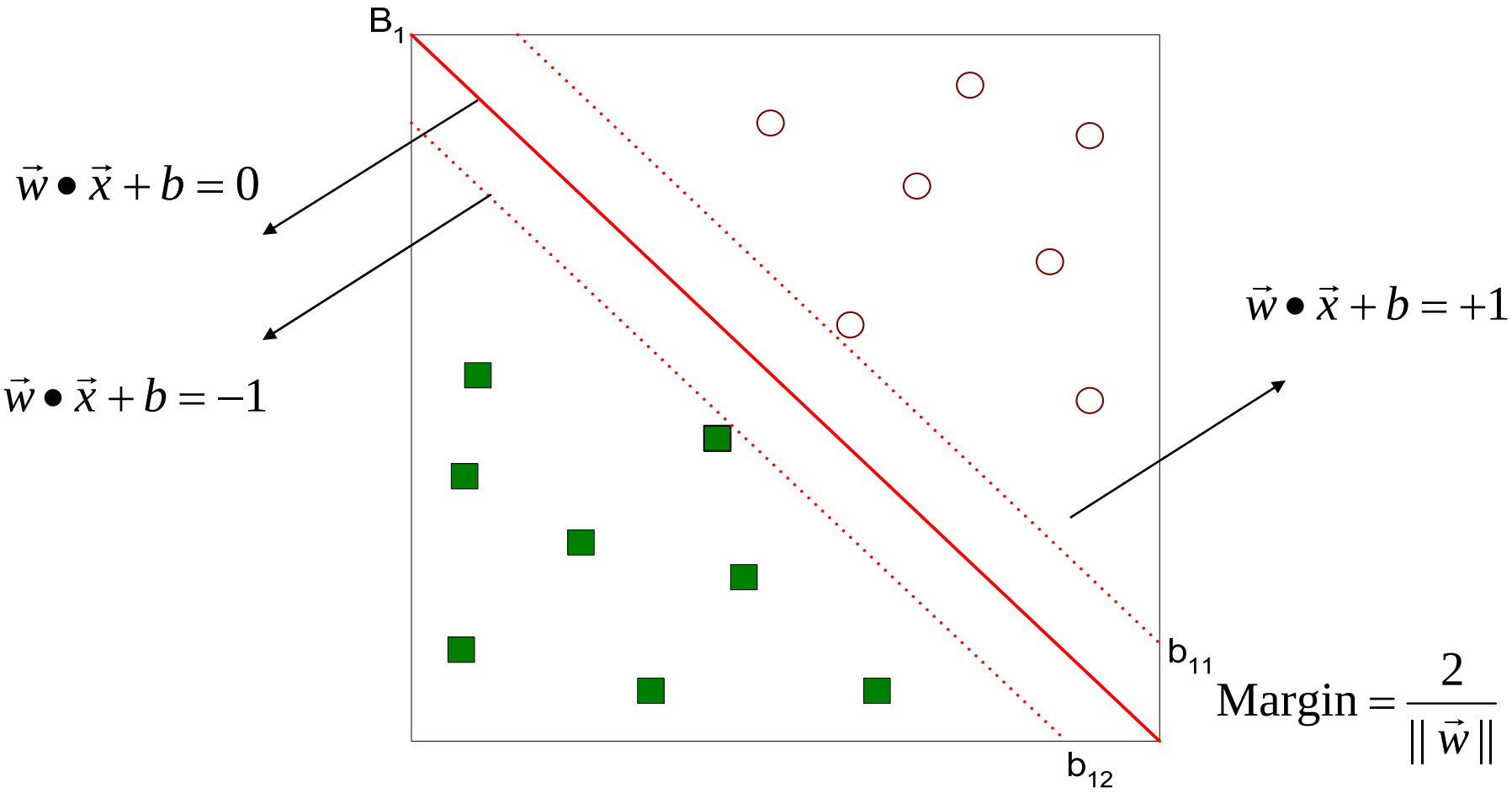
  where $\mathbf{W}=\{w_1, w_2, ..., w_n\}$ is a weight vector and b a scalar (bias)
- For 2-D it can be written as

  $$w_0 + w_1 x_1 + w_2 x_2 = 0$$
- The hyperplane defining the sides of the margin:

  $$H_1: w_0 + w_1 x_1 + w_2 x_2 = 1$$

  $$H_2: w_0 + w_1 x_1 + w_2 x_2 = -1$$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**

Support Vector Machines



For all training points:

$$\vec{w} \bullet \vec{x}_i + b \geq +1 \text{ for } y_i = +1$$
$$\vec{w} \bullet \vec{x}_i + b \leq -1 \text{ for } y_i = -1$$

$$\Rightarrow y_i(\vec{w} \bullet \vec{x}_i + b) - 1 \geq 0$$

Support Vector Machines

- We want to maximize: $\text{Margin} = \frac{2}{\| \vec{w} \|}$
  - Equivalent to minimizing: $\| \vec{w} \|^2$
  - But subjected to the constraints: $y_i(\vec{w} \bullet \vec{x}_i + b) - 1 \geq 0$
- Constrained optimization problem
  - Lagrange reformulation

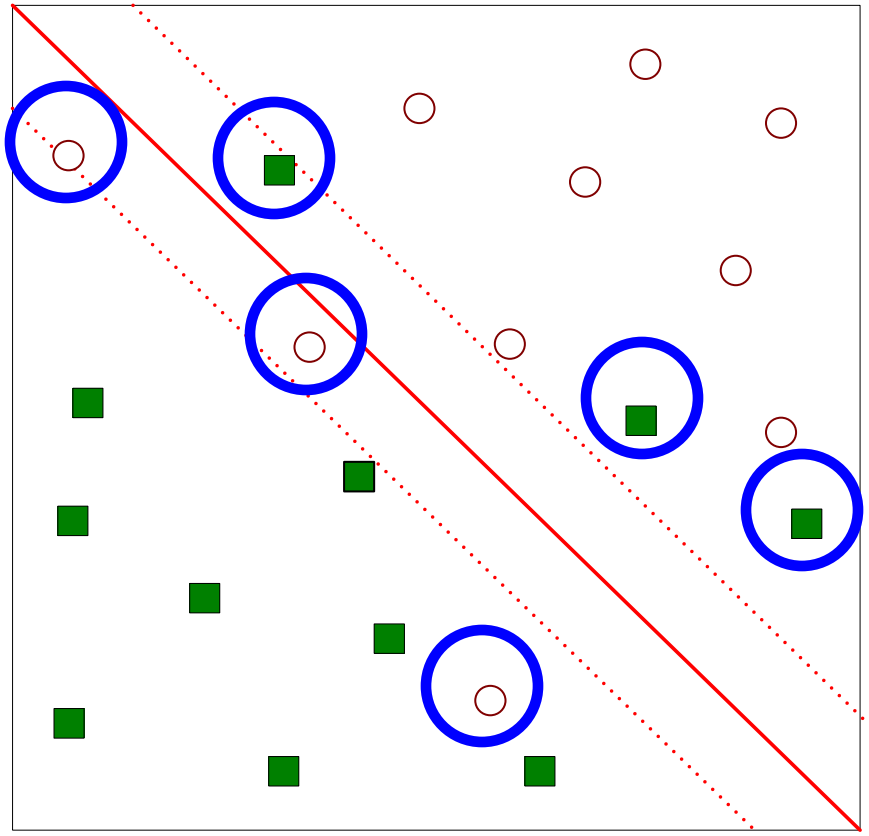
$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{l} \alpha_i$$

Support Vector Machines

- What if the problem is not linearly separable?
- Introduce slack variables to the constraints:

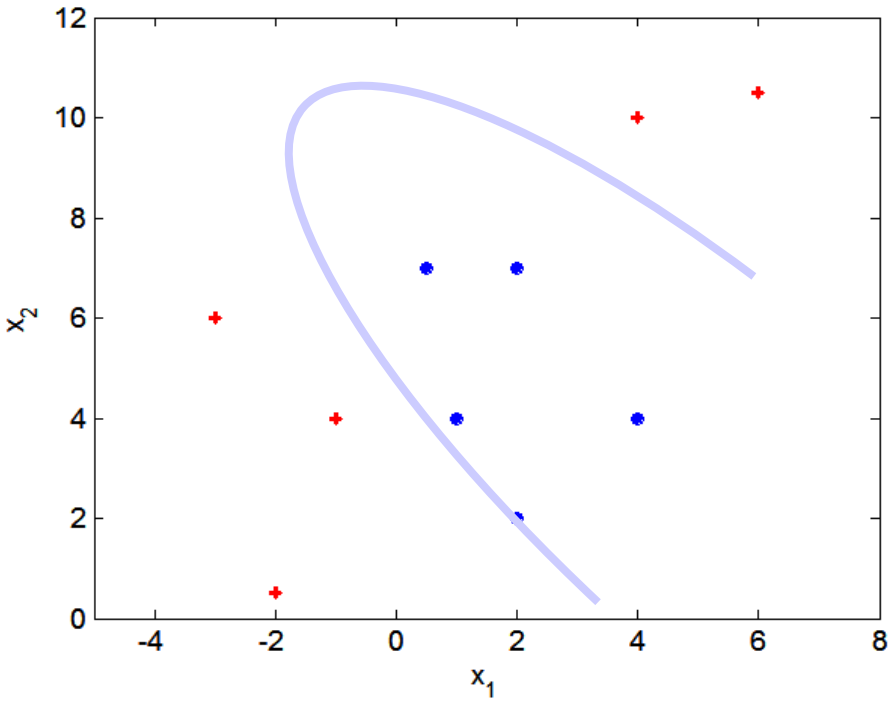
$$\vec{w} \bullet \vec{x}_i + b \geq +1 - \xi_i \text{ for } y_i = +1$$
$$\vec{w} \bullet \vec{x}_i + b \leq -1 + \xi_i \text{ for } y_i = -1$$

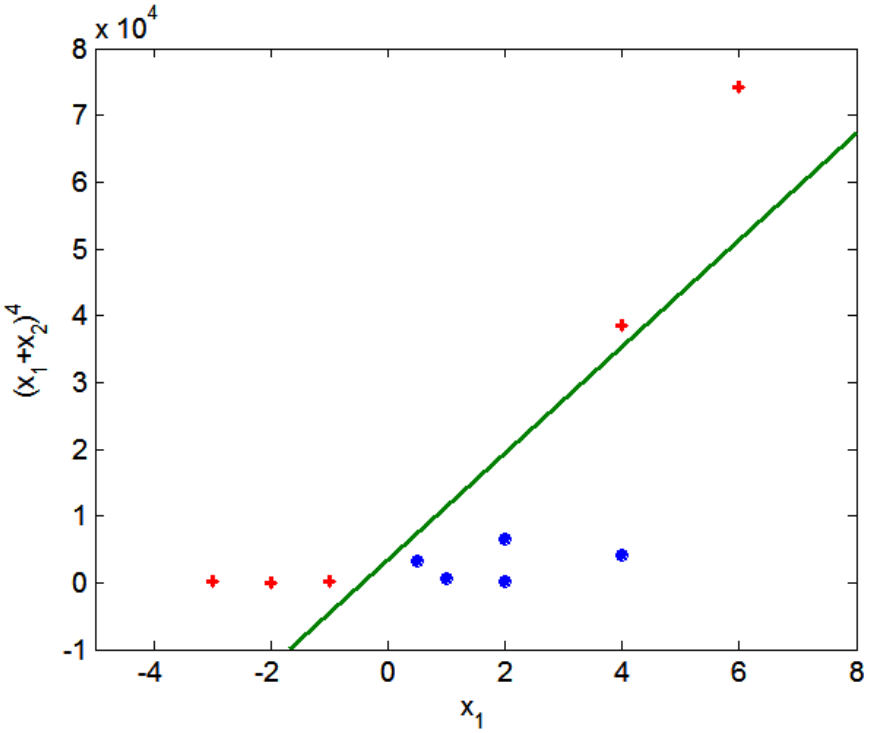
- Upper bound on the training errors: $\sum_i \xi_i$



Nonlinear Support Vector Machines

- What if decision boundary is not linear?
- Transform the data into higher dimensional space and search for a hyperplane in the new space
- Convert the hyperplane back to the original space





SVM—Kernel functions

- Instead of computing the dot product on the transformed data tuples, it is mathematically equivalent to instead applying a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$ to the original data, i.e., $K(\mathbf{X}_i, \mathbf{X}_j) = \Phi(\mathbf{X}_i)\ \Phi(\mathbf{X}_j)$
- Typical Kernel Functions

$$\text{Polynomial kernel of degree } h: \quad K(X_i, X_j) = (X_i \cdot X_j + 1)^h$$

$$\text{Gaussian radial basis function kernel}: \quad K(X_i, X_j) = e^{-\|X_i - X_j\|^2/2\sigma^2}$$

$$\text{Sigmoid kernel}: \quad K(X_i, X_j) = \tanh(\kappa X_i \cdot X_j - \delta)$$

- SVM can also be used for classifying multiple (> 2) classes and for regression analysis (with additional user parameters)

Support Vector Machines: Comments and Research Issues

- Robust and accurate with nice generalization properties
- Effective (insensitive) to high dimensions
  - Complexity characterized by # of support vectors rather than dimensionality
- Scalability in training
- Extension to regression analysis
- Extension to multiclass SVM

SVM Related Links

- SVM web sites
  - www.kernel-machines.org
  - www.kernel-methods.net
  - www.support-vector.net
  - www.support-vector-machines.org
- Representative implementations
  - LIBSVM: an efficient implementation of SVM, multi-class classifications
  - SVM-light: simpler but performance is not better than LIBSVM, support only binary classification and only C language
  - SVM-torch: another recent implementation also written in C.

SVM—Introduction Literature

- "Statistical Learning Theory" by Vapnik: extremely hard to understand, containing many errors too
- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Knowledge Discovery and Data Mining*, 2(2), 1998.
  - Better than the Vapnik's book, but still written too hard for introduction, and the examples are not-intuitive
- The book "An Introduction to Support Vector Machines" by N. Cristianini and J. Shawe-Taylor
  - Also written hard for introduction, but the explanation about the mercer's theorem is better than above literatures
- The neural network book by Haykins
  - Contains one nice chapter of SVM introduction

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Evaluation and measures
- Ensemble methods

Other Classification Methods

- Rule based classification
- Neural networks
- Genetic algorithms
- Rough set approaches
- Fuzzy set approaches

Rule-Based Classifier

- Classify records by a collection of IF-THEN rules
- Basic concepts
  - IF (*Condition*) THEN *y*
  - (*Condition*) $\rightarrow$ *y*
    - LHS: rule antecedent or condition
    - RHS: rule consequent
- Using the rules
- Learning the rules

Rule-Based Classifiers: Concepts

- Rules:
  - IF (*Condition*) THEN *y*
  - (*Condition*) $\rightarrow$ *y*
    - LHS: rule antecedent or condition
    - RHS: rule consequent
  - E.g. IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

Rule-based Classifier: Example

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammals
python	cold	no	no	no	reptiles
salmon	cold	no	no	yes	fishes
whale	warm	yes	no	yes	mammals
frog	cold	no	no	sometimes	amphibians
komodo	cold	no	no	no	reptiles
bat	warm	yes	yes	no	mammals
pigeon	warm	no	yes	no	birds
cat	warm	yes	no	no	mammals
leopard shark	cold	yes	no	yes	fishes
turtle	cold	no	no	sometimes	reptiles
penguin	warm	no	no	sometimes	birds
porcupine	warm	yes	no	no	mammals
eel	cold	no	no	yes	fishes
salamander	cold	no	no	sometimes	amphibians
gila monster	cold	no	no	no	reptiles
platypus	warm	no	no	no	mammals
owl	warm	no	yes	no	birds
dolphin	warm	yes	no	yes	mammals
eagle	warm	no	yes	no	birds

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds

R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes

R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals

Assessment of a Rule

- Coverage of a rule:
  - Fraction of records that satisfy the antecedent of a rule
  - coverage(R) = n_{covers} /|D| where n_{covers} = # of tuples covered by R and D is the training data set
- Accuracy of a rule:
  - Fraction of records that satisfy both the antecedent and consequent of a rule
  - accuracy(R) = $n_{correct}$ / n_{covers} where $n_{correct}$ = # of tuples correctly classified by R

Characteristics of Rule-Based Classifier

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

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

Using the Rules

- Rules that are mutually exclusive and exhaustive
- Rules that are not mutually exclusive
  - A record may trigger more than one rules
  - Solution? – Conflict resolution
    - Rule Ordering
    - Unordered rule set – use voting schemes
- Rules that are not exhaustive
  - A record may not trigger any rules
  - Solution?
    - Use a default class

Rule-Based Ordering

- Rule-based ordering
  - Individual rules are ranked based on their quality
  - Rule set is known as a decision list
- Class-based ordering
  - Classes are sorted in order of decreasing importance
  - Rules are sorted by the classes

R1: (Give Birth = no) $\wedge$ (Can Fly = yes) $\rightarrow$ Birds
R2: (Give Birth = no) $\wedge$ (Live in Water = yes) $\rightarrow$ Fishes
R3: (Give Birth = yes) $\wedge$ (Blood Type = warm) $\rightarrow$ Mammals
R4: (Give Birth = no) $\wedge$ (Can Fly = no) $\rightarrow$ Reptiles
R5: (Live in Water = sometimes) $\rightarrow$ Amphibians

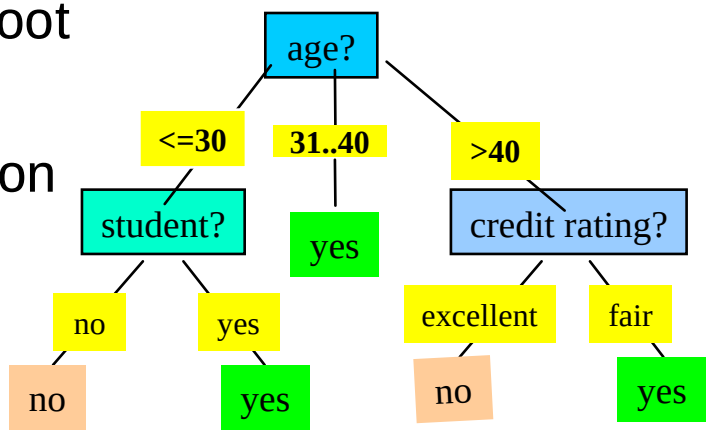
Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
turtle	cold	no	no	sometimes	?

Building Classification Rules

- Indirect Method: Extract rules from other classification models
  - Decision trees. E.g. C4.5 Rules
- Direct Method: Extract rules directly from data
  - Sequential Covering. E.g.: CN2, RIPPER
  - Associative Classification.

Rule Extraction from a Decision Tree

- One rule is created for each path from the root to a leaf - each attribute-value pair forms a conjunction, the leaf holds the class prediction
- Rules are mutually exclusive and exhaustive
- Pruning (C4.5): class-based ordering



- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = *no* THEN *buys_computer* = *no*

IF *age* = young AND *student* = *yes* THEN *buys_computer* = *yes*

IF *age* = mid-age THEN *buys_computer* = *yes*

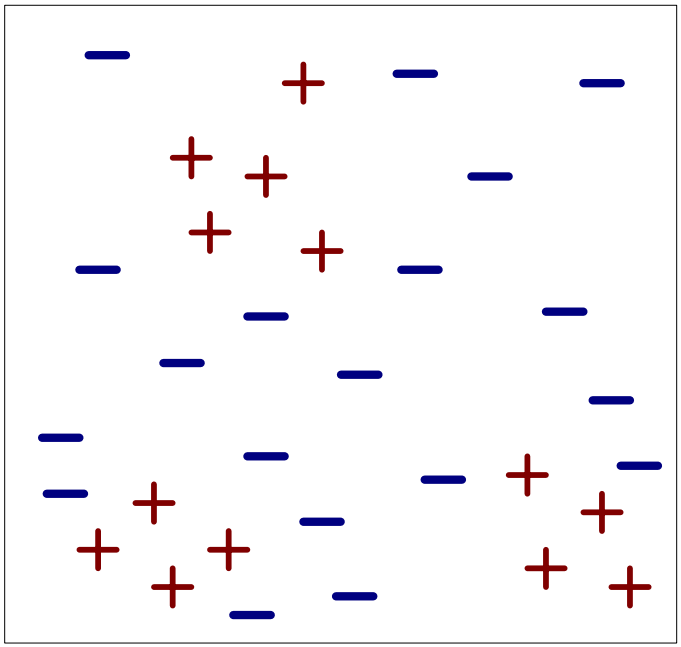
IF *age* = old AND *credit_rating* = *excellent* THEN *buys_computer* = *yes*

IF *age* = young AND *credit_rating* = *fair* THEN *buys_computer* = *no*

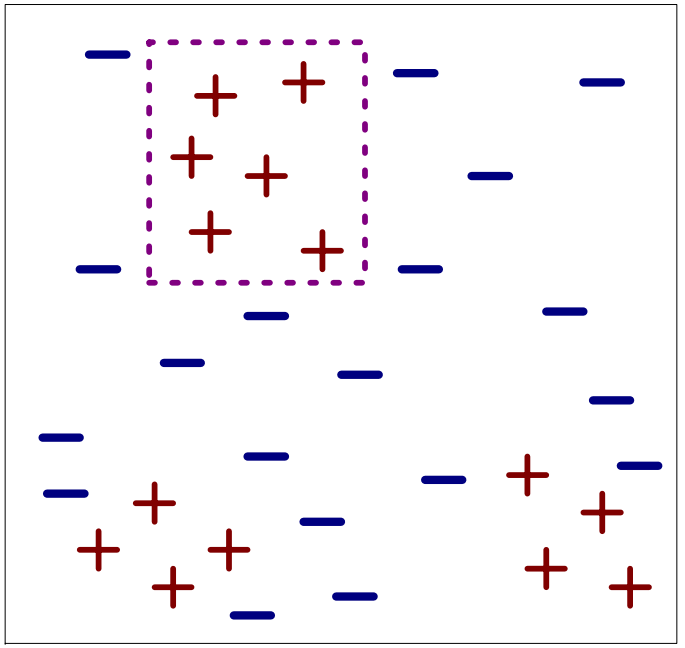
Direct Method: Sequential Covering

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

Example of Sequential Covering

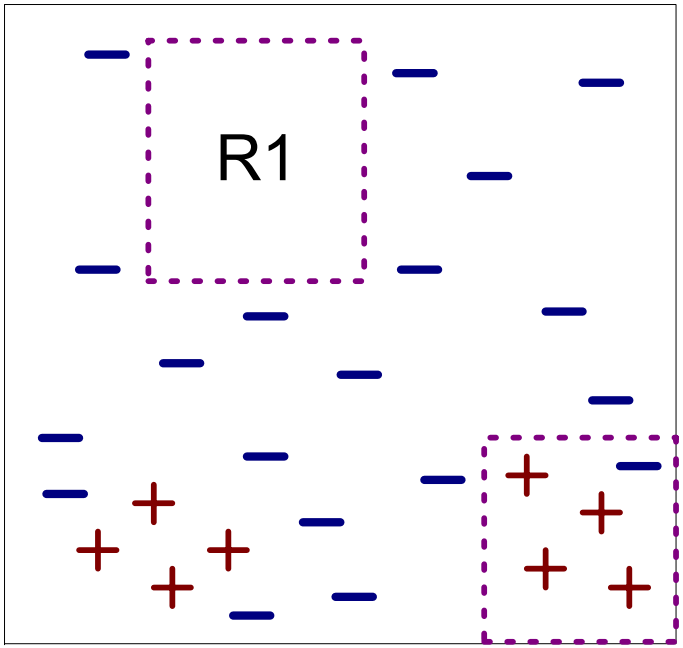


(i) Original Data

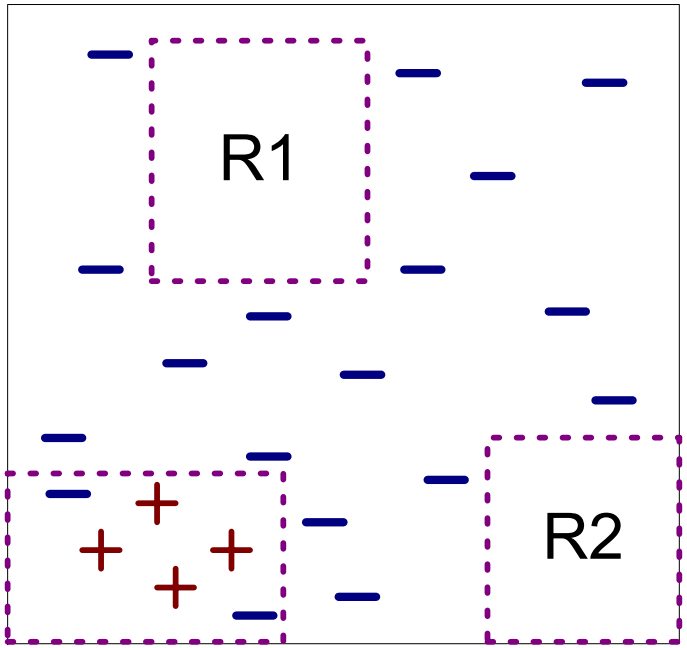


(ii) Step 1

Example of Sequential Covering



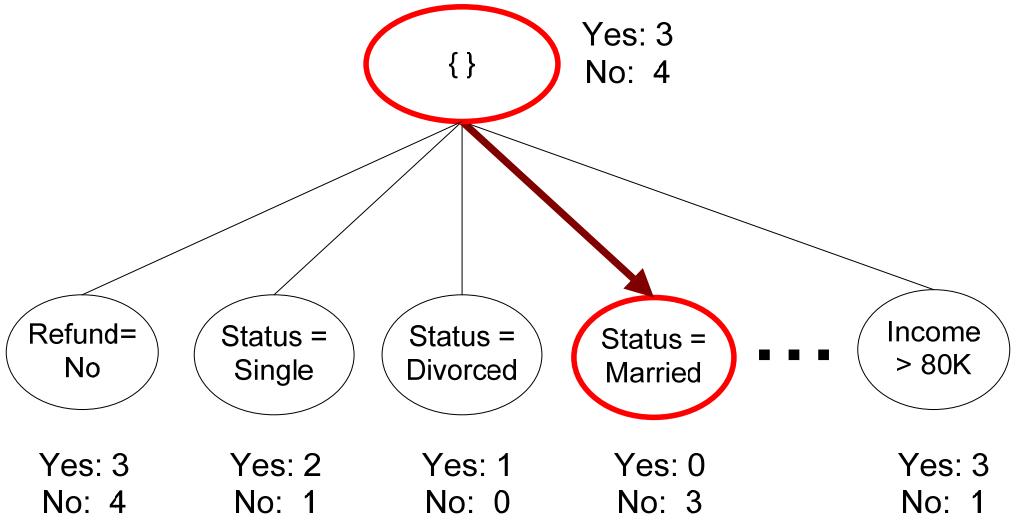
(iii) Step 2



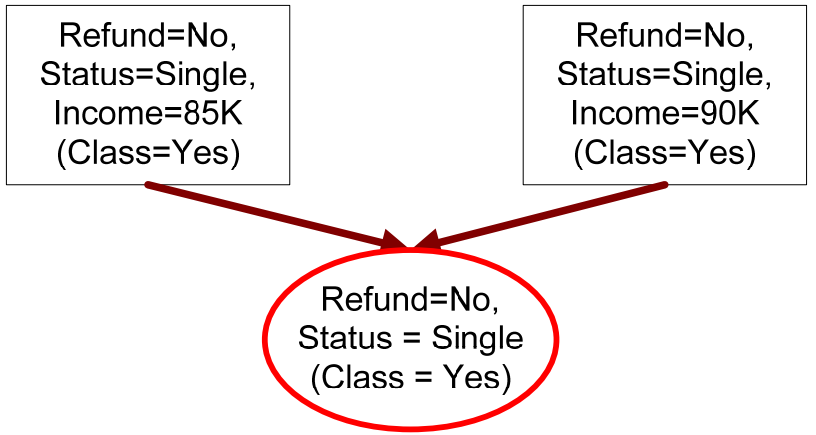
(iv) Step 3

Rule Growing

- Two common strategies



(a) General-to-specific



(b) Specific-to-general

Learn-One-Rule

- Star with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
  - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
  - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$FOIL\_Gain = pos' \times (\log_2 \frac{pos'}{pos'+neg'} - \log_2 \frac{pos}{pos+neg})$$

It favors rules that have high accuracy and cover many positive tuples

- Rule pruning based on an independent set of test tuples

$$FOIL\_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

If *FOIL_Prune* is higher for the pruned version of R, prune R

Direct Method: Multi-Class

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

Associative Classification

- Associative classification
  - Search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels
  - Classification: Based on evaluating a set of rules in the form of

    P_1 ^ p_2 ... ^ p_l → "A_{class} = C" (conf, sup)
- Why effective?
  - It explores highly confident associations among multiple attributes and may overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time
  - In many studies, associative classification has been found to be more accurate than some traditional classification methods, such as C4.5

Typical Associative Classification Methods

- CBA (Classification By Association: Liu, Hsu & Ma, KDD'98)
  - Mine association possible rules in the form of
    - Cond-set (a set of attribute-value pairs) → class label
  - Build classifier: Organize rules according to decreasing precedence based on confidence and then support
- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM'01)
  - Classification: Statistical analysis on multiple rules
- CPAR (Classification based on Predictive Association Rules: Yin & Han, SDM'03)
  - Generation of predictive rules (FOIL-like analysis)
  - High efficiency, accuracy similar to CMAR
- RCBT (Mining top-*k* covering rule groups for gene expression data, Cong et al. SIGMOD'05)
  - Explore high-dimensional classification, using top-k rule groups
  - Achieve high classification accuracy and high run-time efficiency

Rule-Based Classifiers: Comments

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

Other Classification Methods

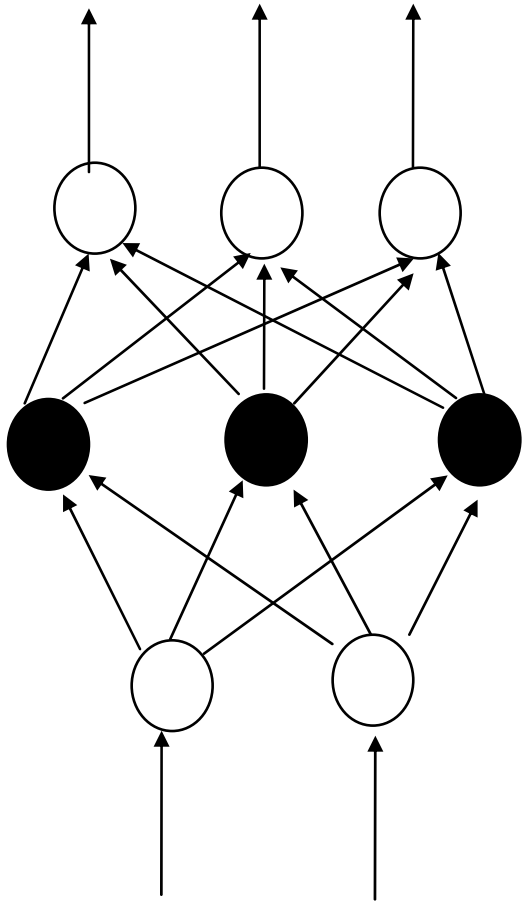
- Rule based classification
- Neural networks
- Genetic algorithms
- Rough set approaches
- Fuzzy set approaches

Classification: A Mathematical Mapping

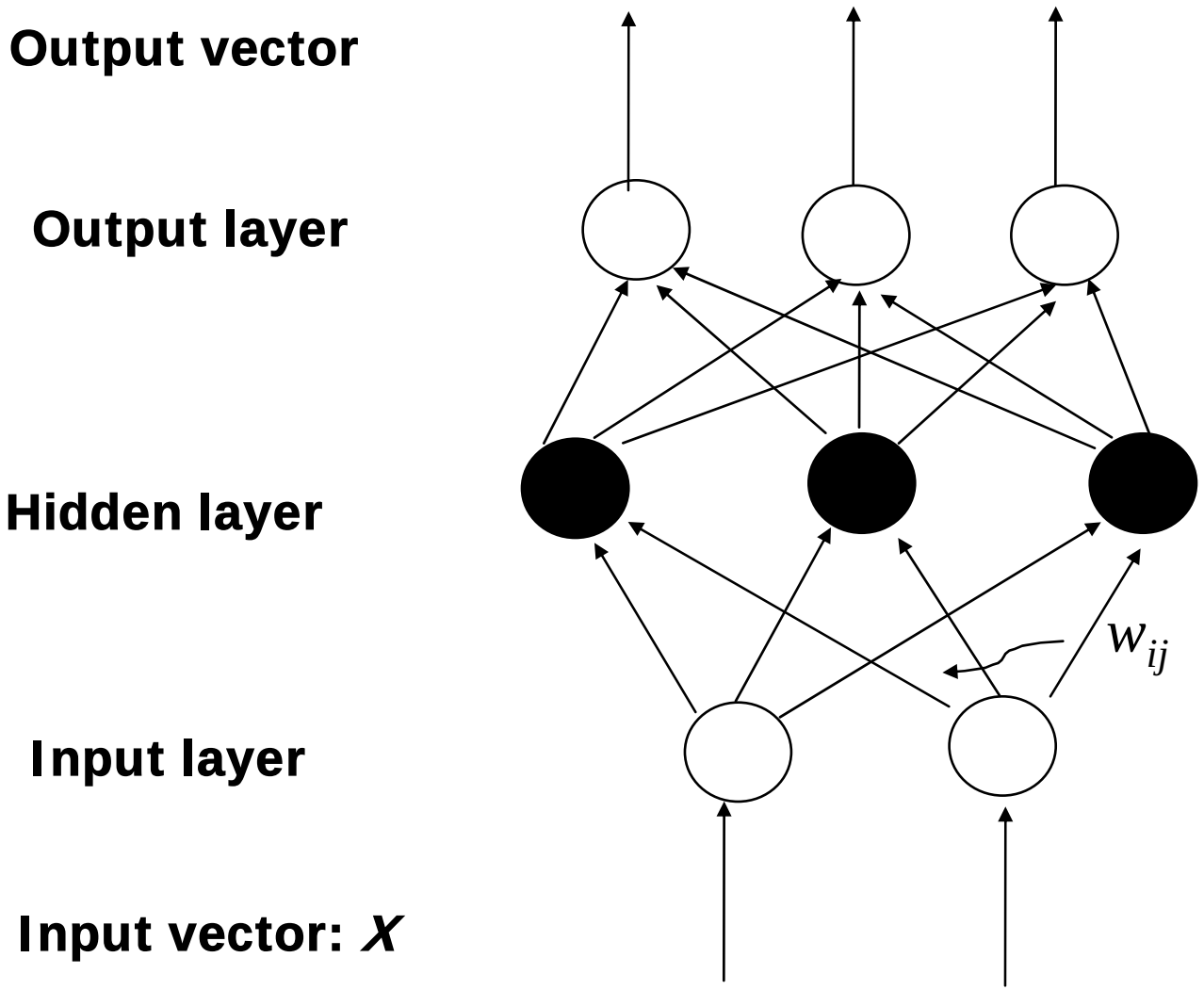
- Mathematically
  - $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$
  - We want a function f: X → Y
- Linear classifiers
  - Probabilistic Classifiers (Naive Bayesian)
  - SVM
  - Perceptron

Neural Networks

- A neural network: A set of connected input/output units where each connection is associated with a **weight**
- Learning phase: adjusting the weights so as to predict the correct class label of the input tuples
  - Backpropagation
- From a statistical point of view, networks perform **nonlinear regression**



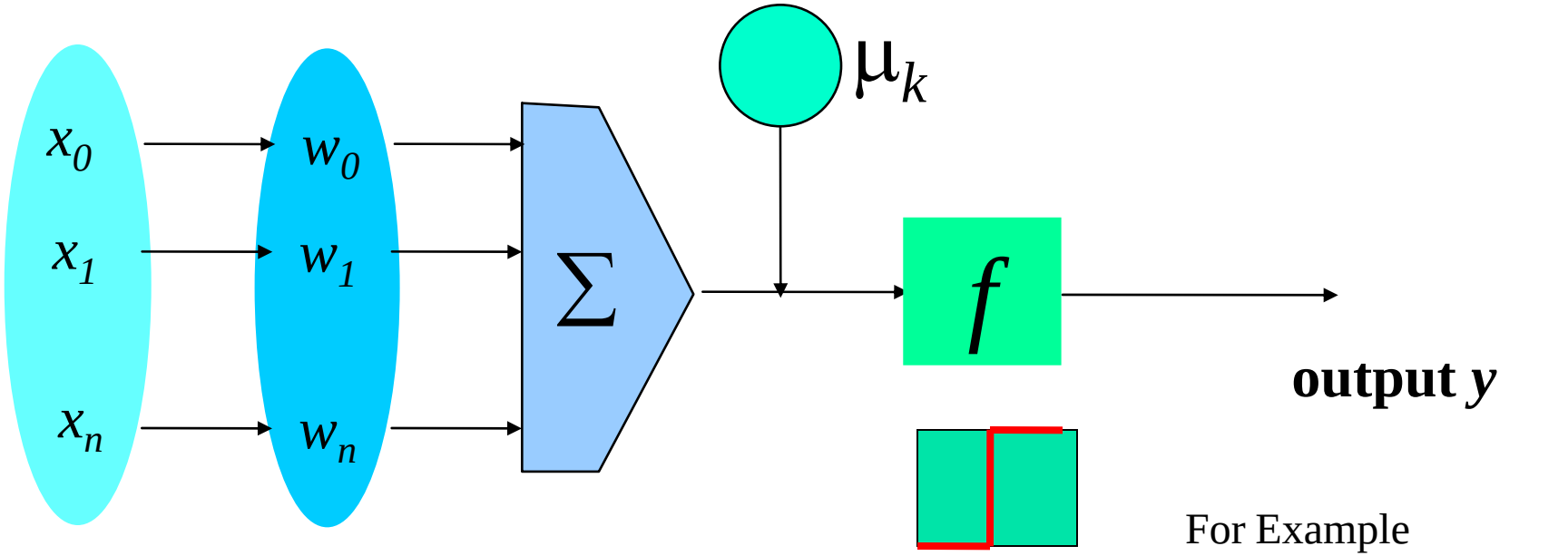
A Multi-Layer Feed-Forward Neural Network



A Multi-Layer Neural Network

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward** in that none of the weights cycles back to an input unit or to an output unit of a previous layer
- From a statistical point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any function

A Neuron (= a perceptron)



- The *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

Backpropagation

- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"
- Steps
  - Initialize weights (to small random #s) and biases in the network
  - Propagate the inputs forward (by applying activation function)
  - Backpropagate the error (by updating weights and biases)
  - Terminating condition (when error is very small, etc.)

Neural Network as a Classifier: Comments

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or ``structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of ``hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Ability to classify untrained patterns
  - Well-suited for continuous-valued inputs and outputs
  - Successful on a wide array of real-world data
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Other Classification Methods

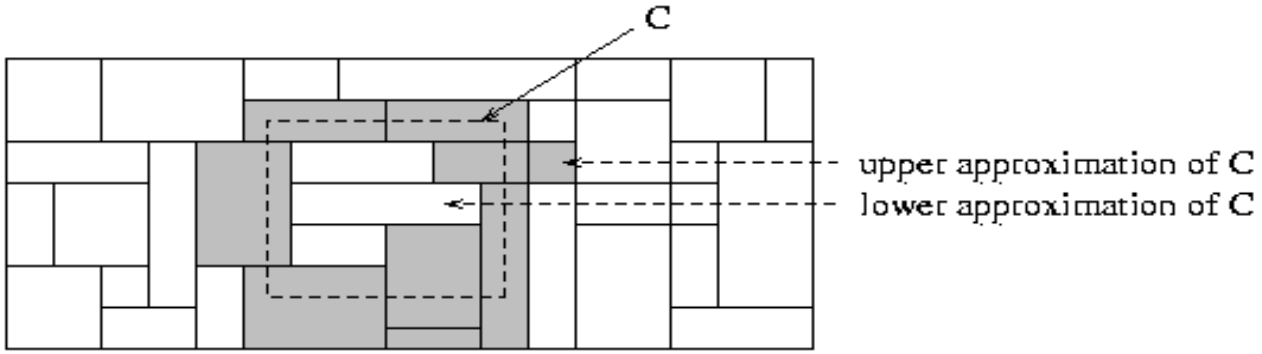
- Rule based classification
- Neural networks
- Genetic algorithms
- Rough set approaches
- Fuzzy set approaches

Genetic Algorithms (GA)

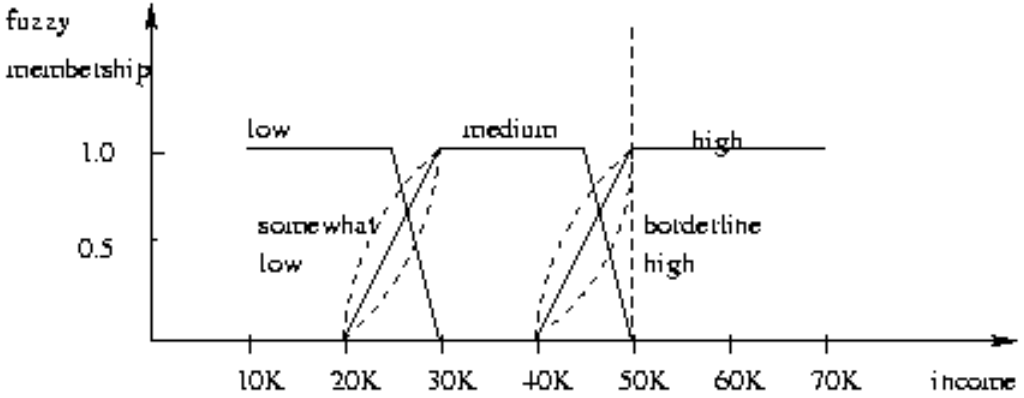
- Genetic Algorithm: based on an analogy to biological evolution
- An initial **population** is created consisting of randomly generated rules
  - Each rule is represented by a string of bits
  - E.g., if A_1 and $\neg A_2$ then C_2 can be encoded as 100
- Based on the notion of survival of the **fittest**, a new population is formed to consist 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*
- The process continues until a population P evolves *when each rule in P satisfies a prespecified threshold*
- Slow but easily parallelizable

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** (which stores the differences between attribute values for each pair of data tuples) 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, and these sums are combined

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Prediction methods
- Evaluation metrics and methods
- Ensemble methods

Prediction

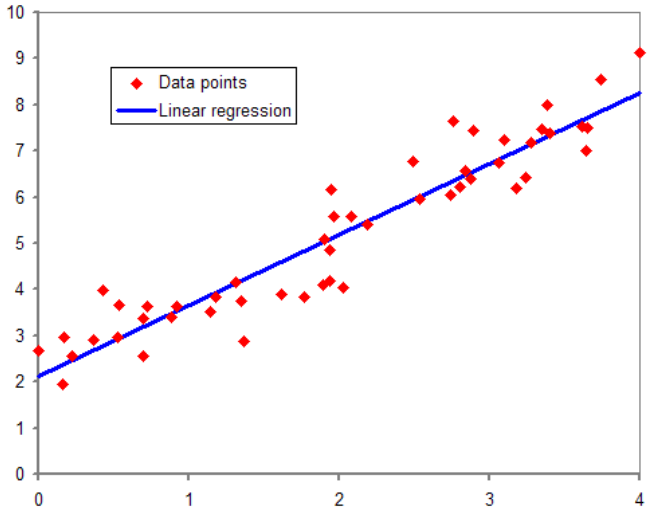
- Prediction vs. classification
  - Classification predicts categorical class label
  - Prediction predicts continuous-valued attributes
- Major method for prediction: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear regression
  - Other regression methods: generalized linear model, logistic regression, Poisson regression, regression trees

Linear Regression

- Linear regression: $Y = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_P X_P$
  - Line fitting: $y = w_0 + w_1 x$
  - Polynomial fitting: $Y = b_2 x^2 + b_1 x + b_0$
  - Many nonlinear functions can be transformed
- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2}$$

$$w_0 = \bar{y} - w_1 \bar{x}$$



Other Regression-Based Models

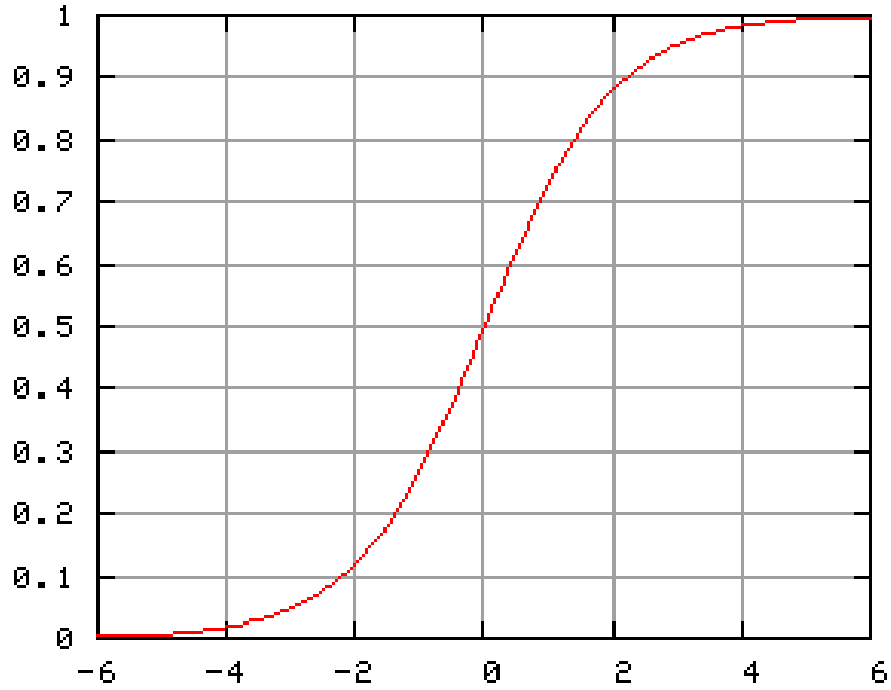
- General linear model
  - Logistic regression: models the probability of some event occurring as a linear function of a set of predictor variables
    - vs. Bayesian classifier
    - Assumes logistic model
  - Poisson regression (log-linear model): models the data that exhibit a Poisson distribution
    - Assumes Poisson distribution for response variable
- Maximum likelyhood method

Logistic Regression

- Logistic regression: models the probability of some event occurring as a linear function of a set of predictor variables
- Logistic function

$$f(z) = \frac{1}{1 + e^{-z}}$$

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \cdots + \beta_k x_k,$$



Poisson Regression

- Poisson regression (log-linear model): models the data that exhibit a Poisson distribution
  - Assumes Poisson distribution for response variable
  - Assumes logarithm of its expected value follows a linear model
  - Simplest case: $\log(\mathrm{E}(Y)) = a + bx.$

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Prediction methods
- Evaluation metrics and methods
- Ensemble methods

Model Evaluation

- Metrics for Performance Evaluation
- Methods for Model Comparison
- Methods for Performance Evaluation

Metrics for Performance Evaluation

- Focus on the predictive capability of a model
- Accuracy of a classifier: percentage of test set tuples that are correctly classified by the model – limitations?
  - Binary classification: $\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$
  - Error rate (misclassification rate) = 1 – accuracy
- Confusion matrix: given *m* classes, *CMi,j*, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
  - Binary classification confusion matrix

		PREDICTED CLASS	
		positive	negative
ACTUAL CLASS	positive	TP	FN
	negative	FP	TN

TP (true positive)

FN (false negative)

FP (false positive)

TN (true negative)

Cost-Sensitive Measures

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

$$\text{False Negative Rate (FNR)} = \frac{FN}{TP + FN}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{FP + TN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F - measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Recall} + \text{Recall}}$$

		PREDICTED CLASS	
		positive	negative
ACTUAL CLASS	positive	TP	FN
	negative	FP	TN

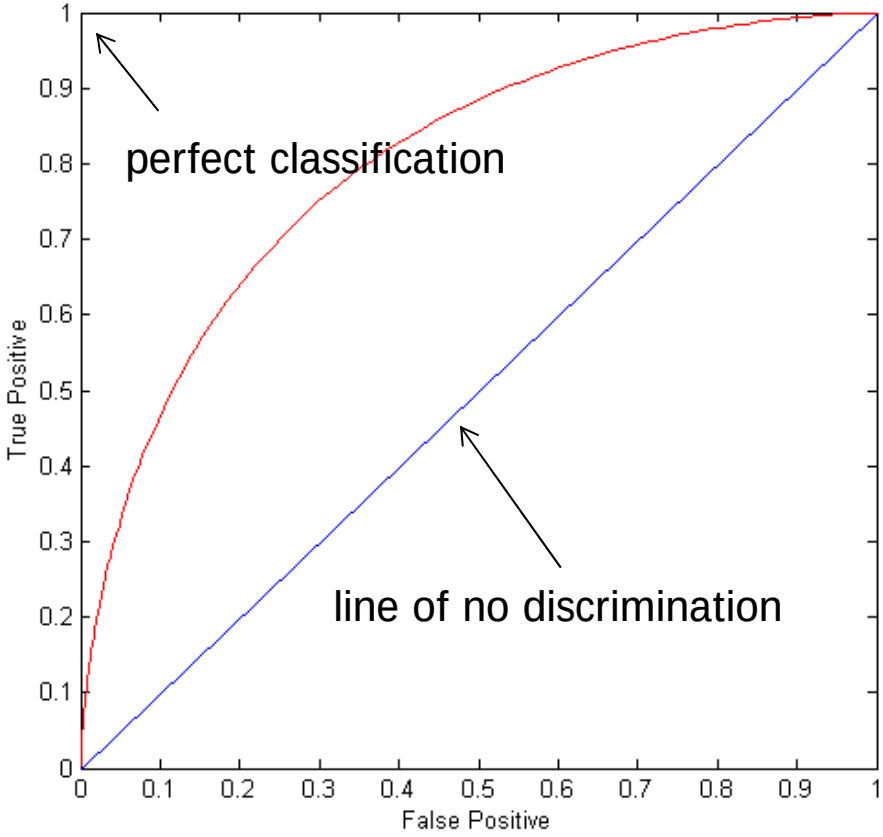
→ sensitivity/recall/true positive rate

→ specificity/true negative rate

↓ precision

Model Comparison: ROC (Receiver Operating Characteristic)

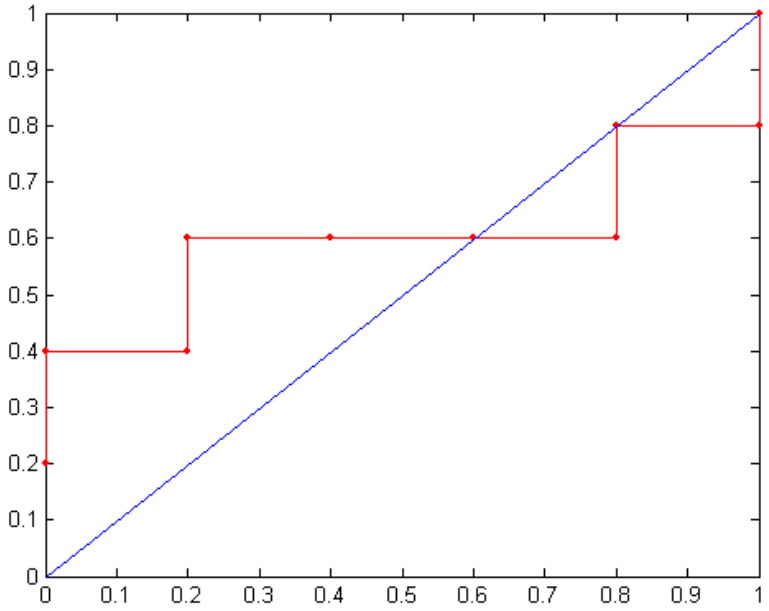
- From signal detection theory
- True positive rate vs. false positive rate
- Sensitivity vs (1 - specificity)
- Each prediction result represents one point (varying threshold, sample distribution, etc)



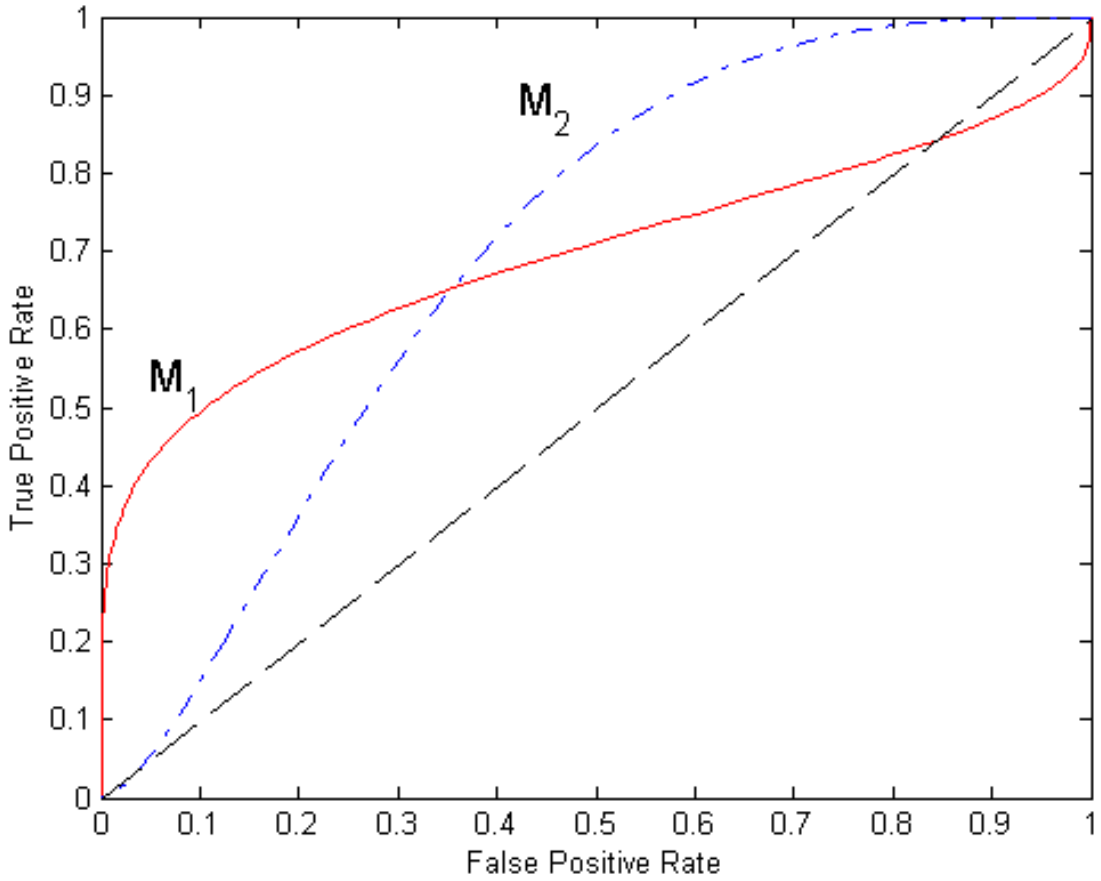
How to Construct an ROC curve

Instance	P(+|A)	True Class
1	0.95	+
2	0.93	+
3	0.87	-
4	0.85	-
5	0.85	-
6	0.85	+
7	0.76	-
8	0.53	+
9	0.43	-
10	0.25	+

- Sort instances according to posterior probability P(+|A) in decreasing order
- Apply threshold at each unique value of P(+|A)
- Compute and plot TPR and FPR



Using ROC for Model Comparison



- Area Under the ROC curve
  - Ideal: Area = 1
  - Diagonal: Area = 0.5
- M1 vs. M2?

Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function**: measures the error betw. y_i and the predicted value y_i'
  - Absolute error: $| y_i - y_i' |$
  - Squared error: $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set
  - Mean absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{d}$ Mean squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{d}$
  - Relative absolute error: $\frac{\sum_{i=1}^{d} | y_i - y_i' |}{\sum_{i=1}^{d} | y_i - \bar{y} |}$ Relative squared error: $\frac{\sum_{i=1}^{d} (y_i - y_i')^2}{\sum_{i=1}^{d} (y_i - \bar{y})^2}$

  The mean squared-error exaggerates the presence of outliers

  Popularly use (square) root mean-square error, similarly, root relative squared error

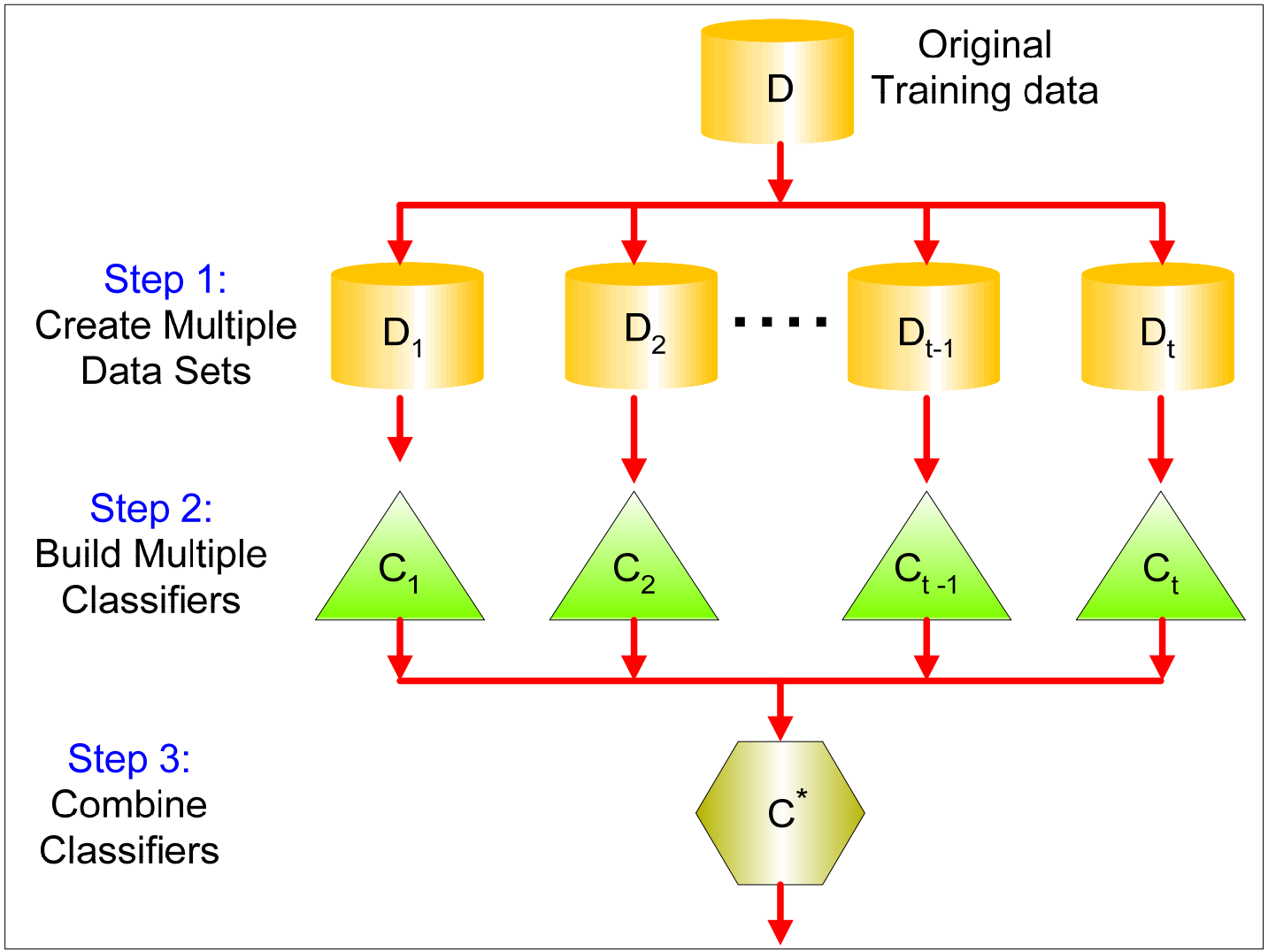
Methods of Evaluation

- Holdout method
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- Cross-validation (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use Di as test set and others as training set
  - Leave-one-out: k folds where k = # of tuples, for small sized data
  - Stratified cross-validation: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Chapter 6. Classification and Prediction

- Overview
- Classification algorithms and methods
  - Decision tree induction
  - Bayesian classification
  - Lazy learning and kNN classification
  - Support Vector Machines (SVM)
  - Others
- Prediction methods
- Evaluation metrics and methods
- Ensemble methods

Ensemble Methods



Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

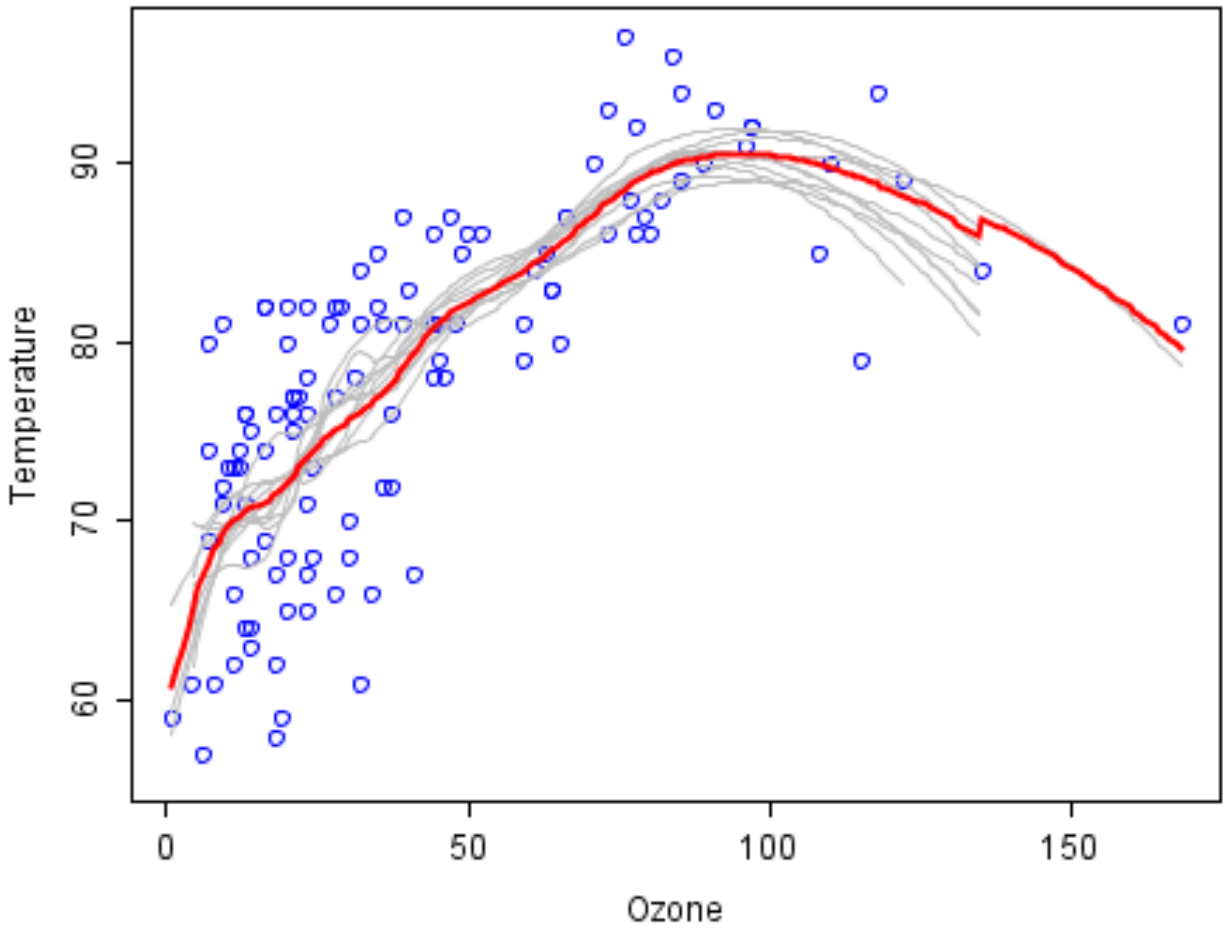
Ensemble Methods

- Bagging (Bootstrap aggregating): averaging the prediction over a collection of classifiers
- Boosting: weighted vote with a collection of classifiers
  - AdaBoost

Bagging: Bootstrap Aggregating (Breiman 1994)

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Create bootstrap sample D_i by sampling with replacement
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: the average value of each prediction
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Bagging Example for Prediction



Boosting

- Analogy: Diagnosis based on a combination of weighted votes
- How boosting works?
  - A series of k classifiers is iteratively learned
  - Subsequent classifiers pay more attention to the training tuples that were misclassified by previous classifiers
  - Combine the votes of each individual classifier based on their accuracy
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Initial weights are assigned to each tuple as the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples are sampled (with replacement) to form a training set D_i based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclssified, its weight is increased, o.w. it is decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

Adaboost: Classifier Weight

- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is

$$\log \frac{1 - error(M_i)}{error(M_i)}$$

Summary (I)

- Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends.
- Effective and scalable methods have been developed for decision trees induction, Naive Bayesian classification, Bayesian belief network, rule-based classifier, Backpropagation, Support Vector Machine (SVM), associative classification, nearest neighbor classifiers, and case-based reasoning, and other classification methods such as genetic algorithms, rough set and fuzzy set approaches.
- Linear, nonlinear, and generalized linear models of regression can be used for prediction. Many nonlinear problems can be converted to linear problems by performing transformations on the predictor variables. Regression trees and model trees are also used for prediction.

Summary (II)

- Stratified k-fold cross-validation is a recommended method for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.
- Significance tests and ROC curves are useful for model selection
- There have been numerous comparisons of the different classification and prediction methods, and the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, interpretability, and scalability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997.
- C. M. Bishop, **Neural Networks for Pattern Recognition**. Oxford University Press, 1995.
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. **Classification and Regression Trees**. Wadsworth International Group, 1984.
- C. J. C. Burges. **A Tutorial on Support Vector Machines for Pattern Recognition**. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998.
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning**. KDD'95.
- W. Cohen. **Fast effective rule induction**. ICML'95.
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data**. SIGMOD'05.
- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman and Hall, 1990.
- G. Dong and J. Li. **Efficient mining of emerging patterns: Discovering trends and differences**. KDD'99.

References (2)

- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley and Sons, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. Machine Learning, 1995.
- M. Kamber, L. Winstone, W. Gong, S. Cheng, and J. Han. **Generalization and decision tree induction: Efficient classification in data mining**. RIDE'97.
- B. Liu, W. Hsu, and Y. Ma. **Integrating Classification and Association Rule**. KDD'98.
- W. Li, J. Han, and J. Pei, **CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules**, ICDM'01.

References (3)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000.
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection**. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining**. EDBT'96.
- T. M. Mitchell. **Machine Learning**. McGraw Hill, 1997.
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey**, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. **FOIL: A midterm report**. ECML'93.
- J. R. Quinlan. **C4.5: Programs for Machine Learning**. Morgan Kaufmann, 1993.
- J. R. Quinlan. **Bagging, boosting, and c4.5**. AAAI'96.

References (4)

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB'98.
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB'96.
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005.
- 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.
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. **CPAR: Classification based on predictive association rules**. SDM'03
- H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters**. KDD'03.