Big Data:
From Theory to Practice

Xing Wu

xingwu@shu.edu.cn

Shanghai University

Classification and Prediction

- What is classification and regression (prediction)? ←
- Issues regarding classification and prediction
- Classification by decision tree induction and random forest
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

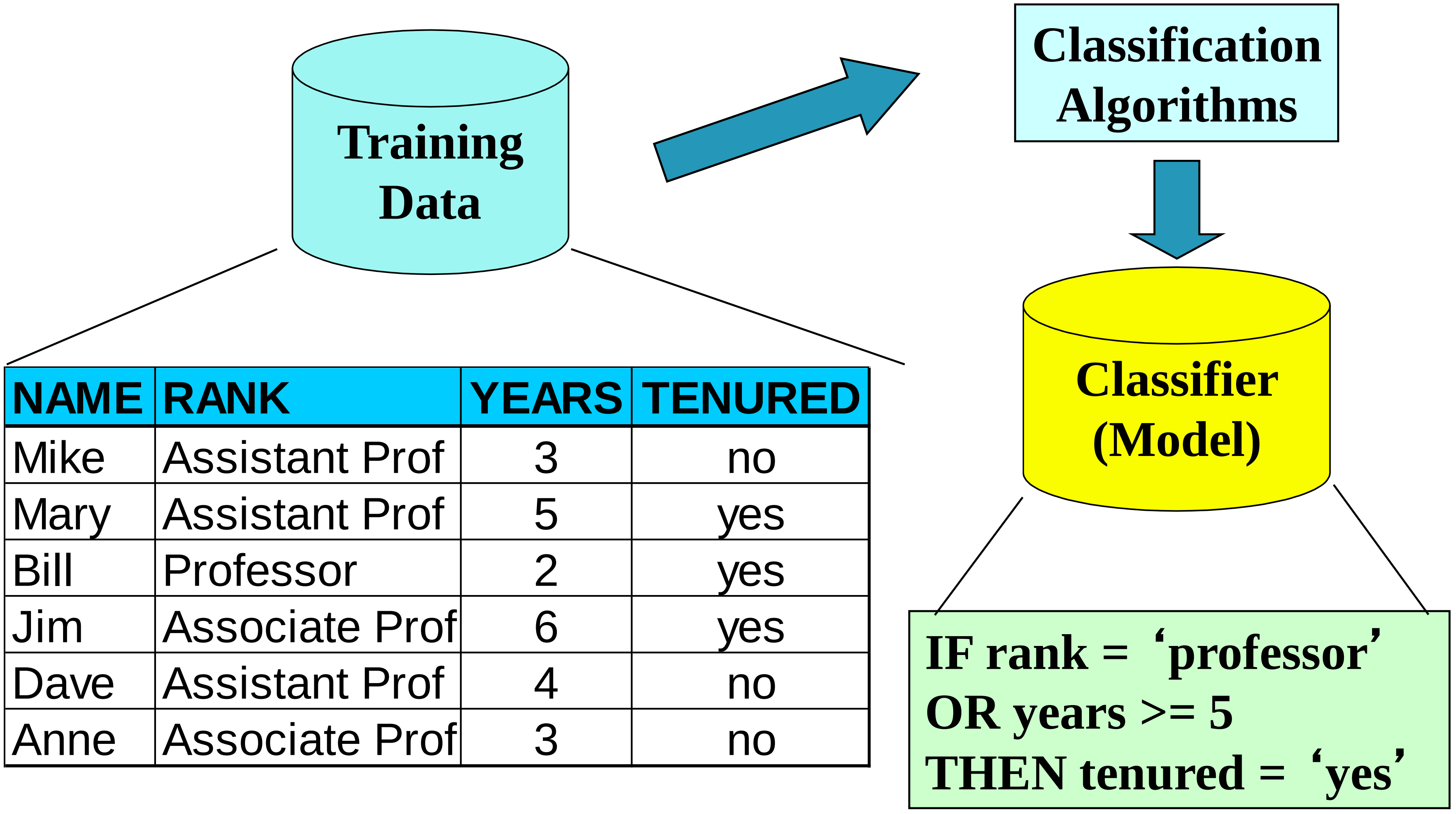
Prediction: Classification vs. Regression

- 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
- Regression
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection

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

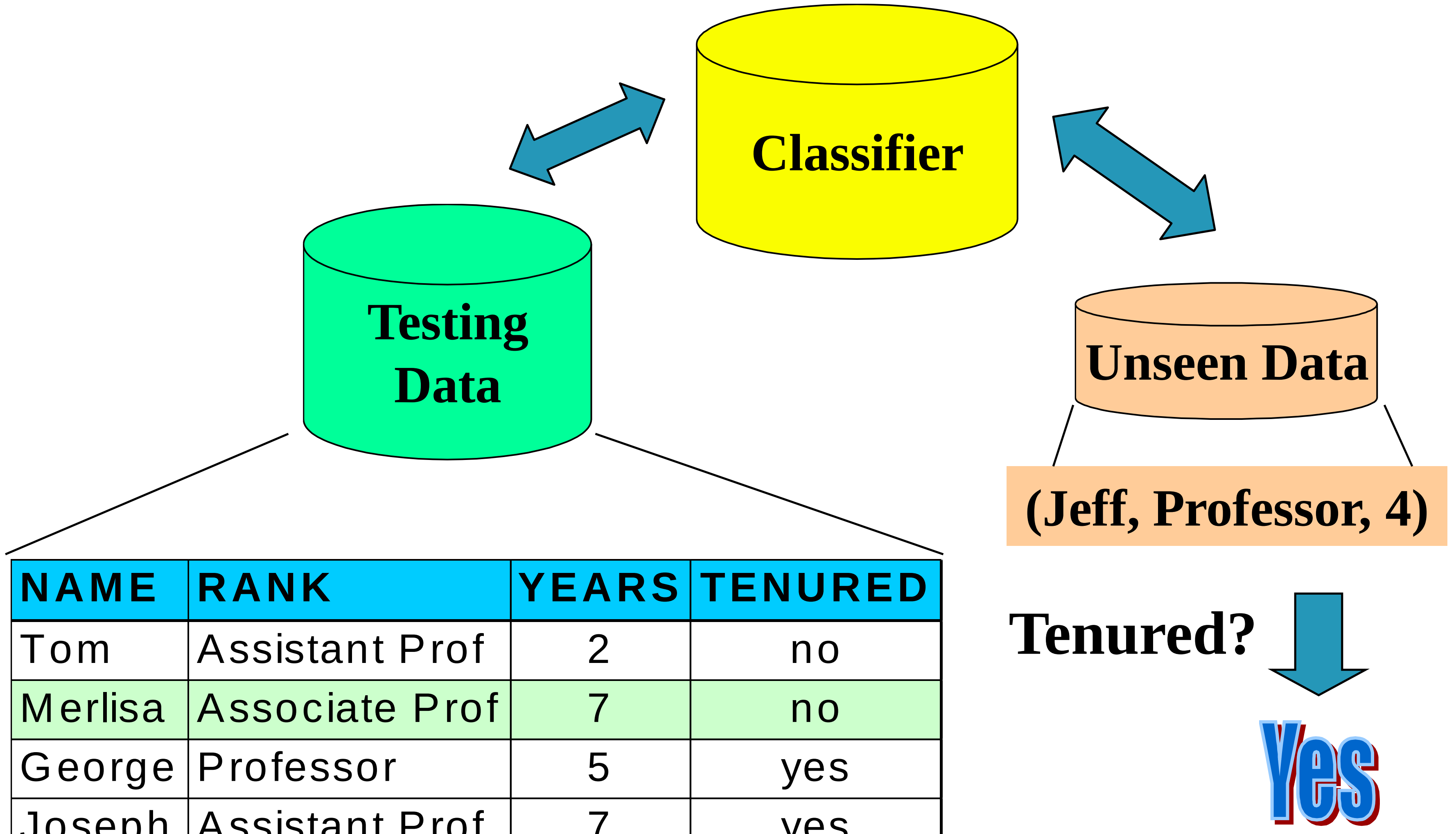
Process (1): Model Construction



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	5	yes
Bill	Professor	2	yes
Jim	Associate Prof	6	yes
Dave	Assistant Prof	4	no
Anne	Associate Prof	3	no

**IF rank = 'professor'
OR years >= 5
THEN tenured = 'yes'**

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 model built on 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

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

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
  - classifier accuracy: predicting class label
  - regression accuracy: guessing value of predicted attributes
- Speed
  - time to construct the model (training time)
  - time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency on large databases
- Interpretability
  - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

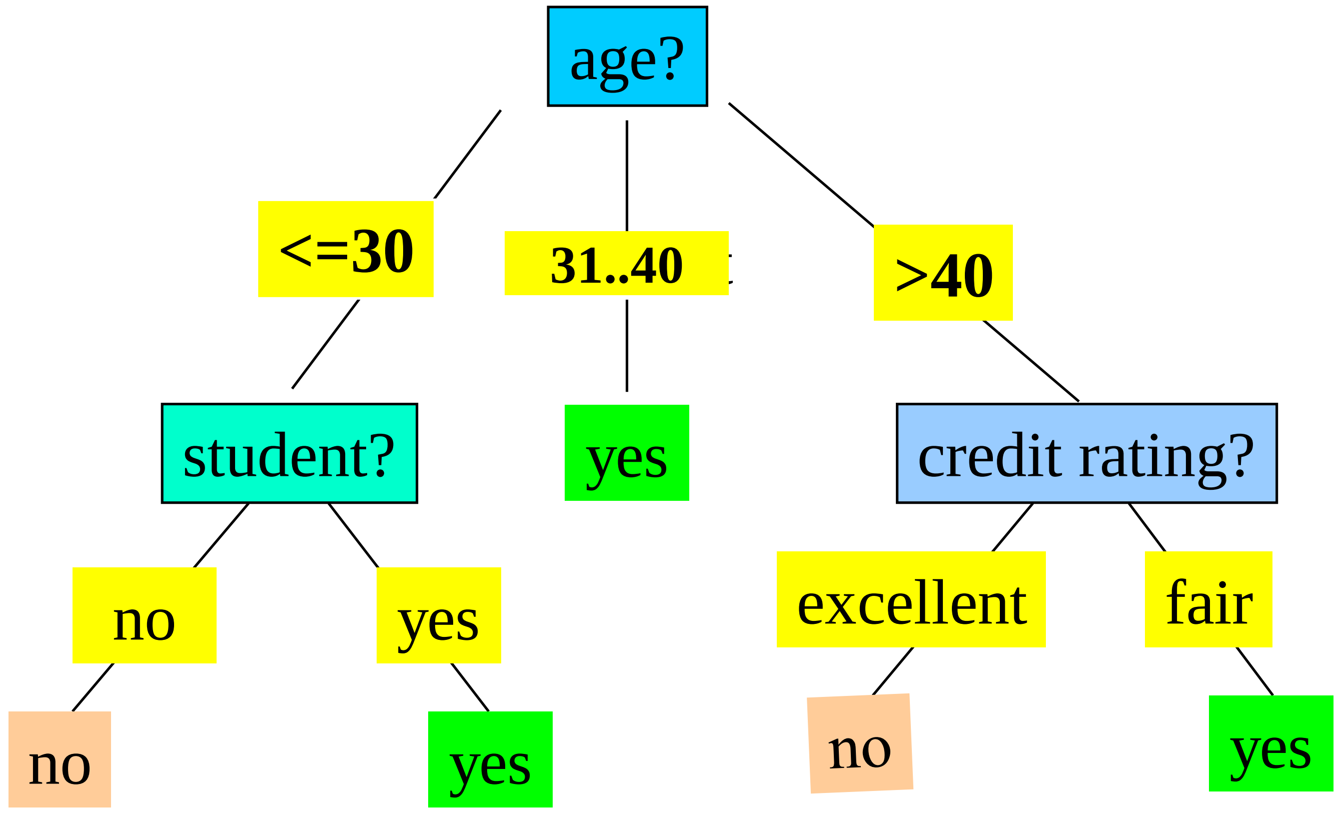
Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Decision Tree Induction: 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

- 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 (or most) 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

Attribute Selection Measure: Information Gain (ID3/C4.5)

- 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 $|C_{i,D}|/|D|$
- Initial entropy:

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

- New entropy (after using A to split D into v partitions) to classify D:

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

- Information gained by branching on attribute A

$$Gain(A) = Entropy(D) - Entropy_A(D)$$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

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

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

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

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

$\frac{5}{14}E(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Entropy(D) - Entropy_{age}(D) = 0.246$$

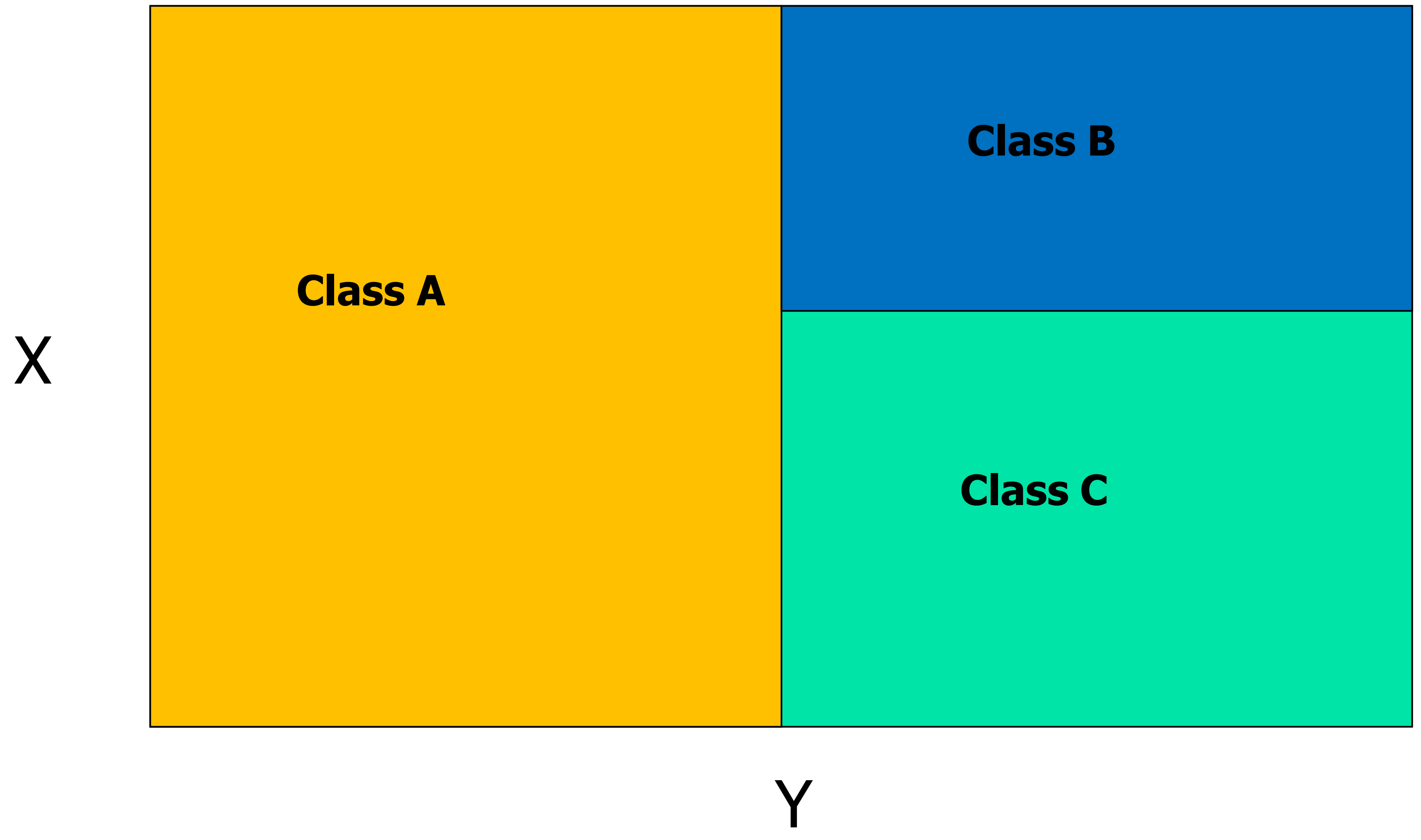
Similarly,

$$Gain(income) = 0.029$$

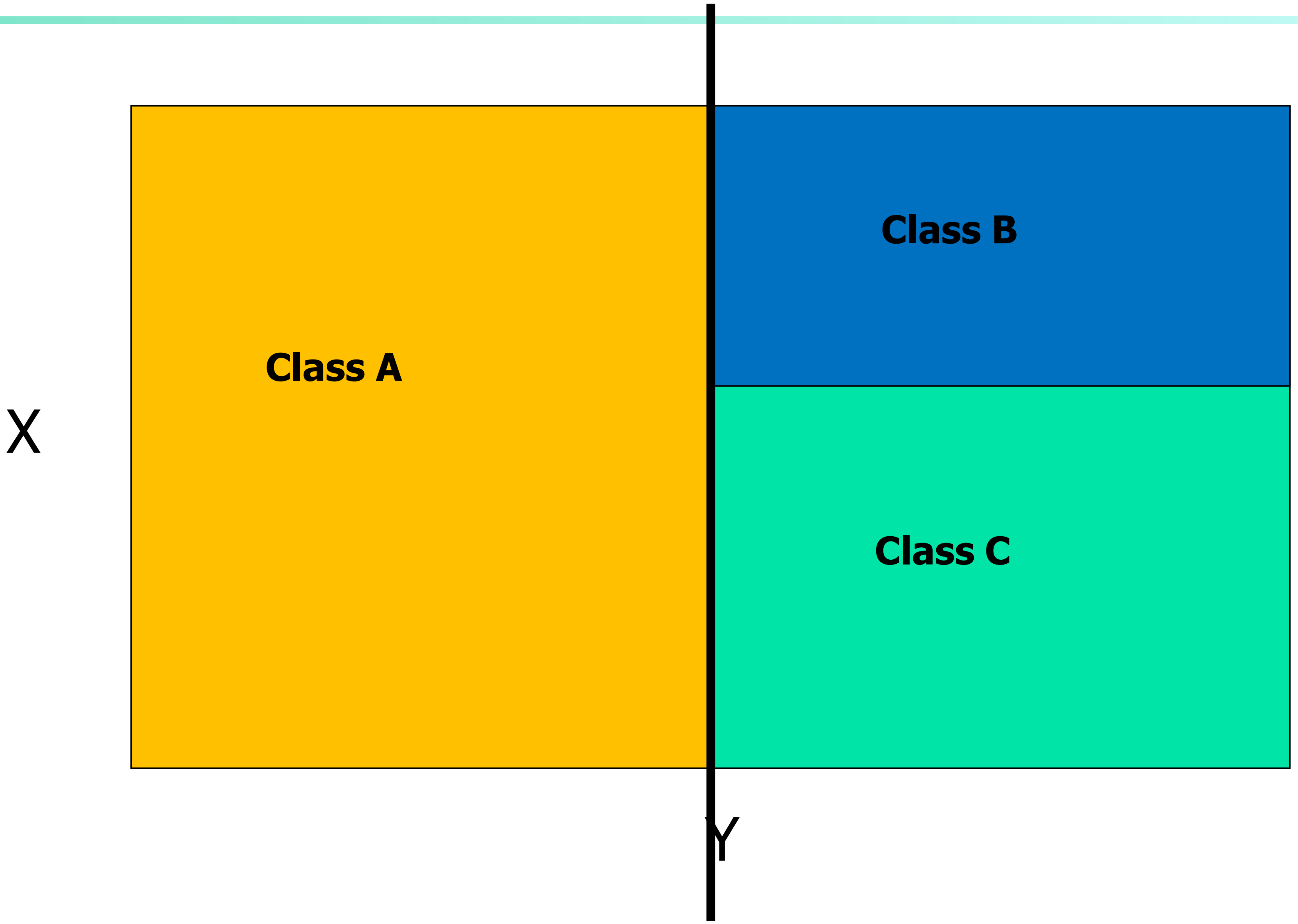
$$Gain(student) = 0.151$$

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

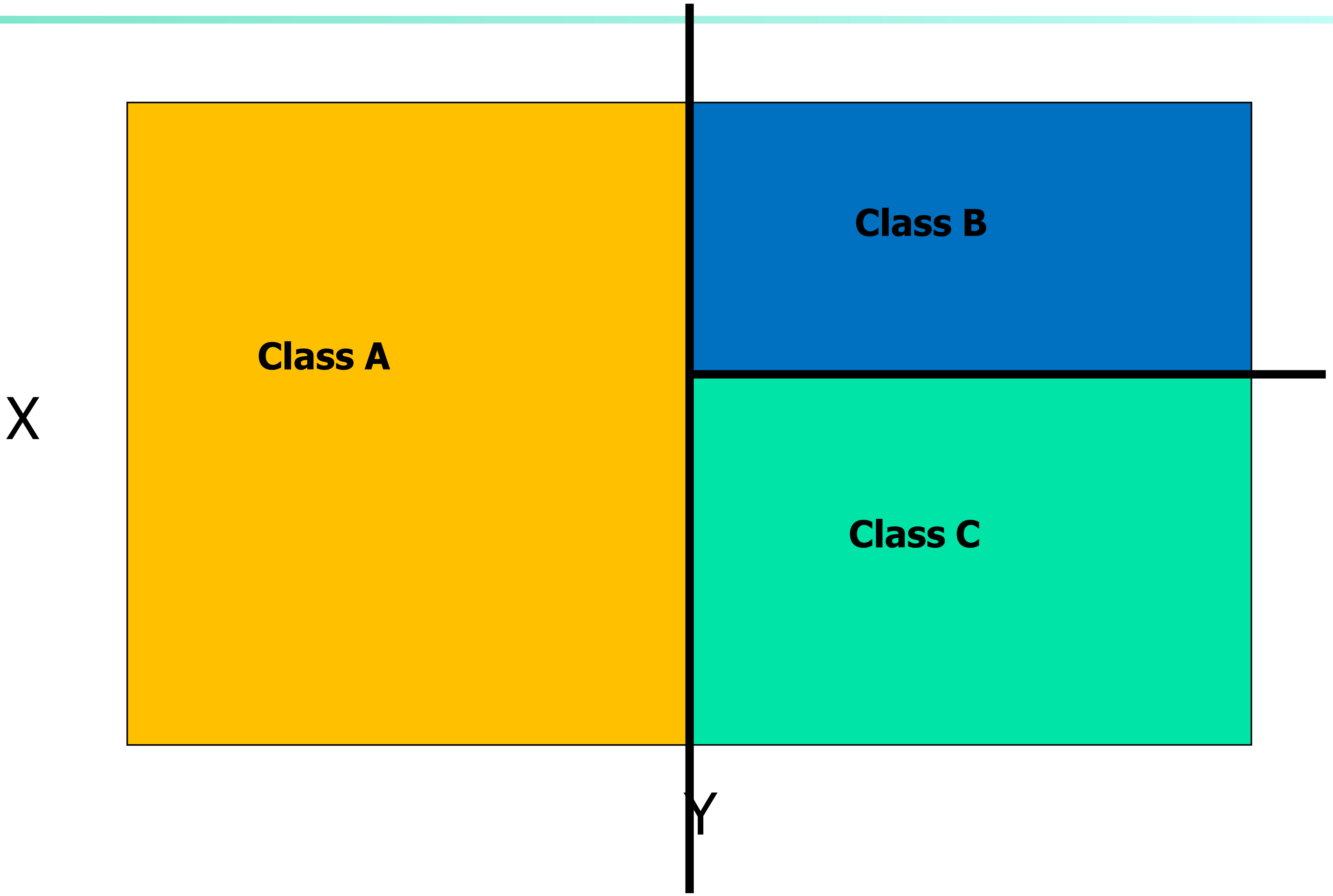
Decision Boundary ?



Decision Boundary ?



Decision Boundary ?



Computing 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 yielding maximum information gain for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying $A \leq$ split-point, and D2 is the set of tuples in D satisfying $A >$ split-point

Gain Ratio for Attribute Selection (C4.5)

- Information gain measure is biased towards attributes with a large number of values
- C4.5 (a successor of ID3) uses gain ratio to overcome the problem (normalization to information gain)

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

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

Gini index (CART, IBM IntelligentMiner)

- 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 (proportion) 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 largest reduction in impurity is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Gini index (CART, IBM IntelligentMiner)

- 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

Weka Demo of A Decision Tree

http://www.cs.waikato.ac.nz/ml/weka/

Breast Cancer Data

Attribute Information:

1. Class: no-recurrence-events, recurrence-events
2. age: 10-19, 20-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89, 90-99.
3. menopause: lt40, ge40, premeno.
4. tumor-size: 0-4, 5-9, 10-14, 15-19, 20-24, 25-29, 30-34, 35-39, 40-44, 45-49, 50-54, 55-59.
5. inv-nodes: 0-2, 3-5, 6-8, 9-11, 12-14, 15-17, 18-20, 21-23, 24-26, 27-29, 30-32, 33-35, 36-39.
6. node-caps: yes, no.
7. deg-malig: 1, 2, 3.
8. breast: left, right.
9. breast-quad: left-up, left-low, right-up, right-low, central.
10. irradiat: yes, no.

Class Distribution:

1. no-recurrence-events: 201 instances
2. recurrence-events: 85 instances

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

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - 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

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

- 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

Random Forest – creates a large number of trees with a random selection of features

Scalability Framework for RainForest

- 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

Rainforest: Training Set and Its AVC Sets

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

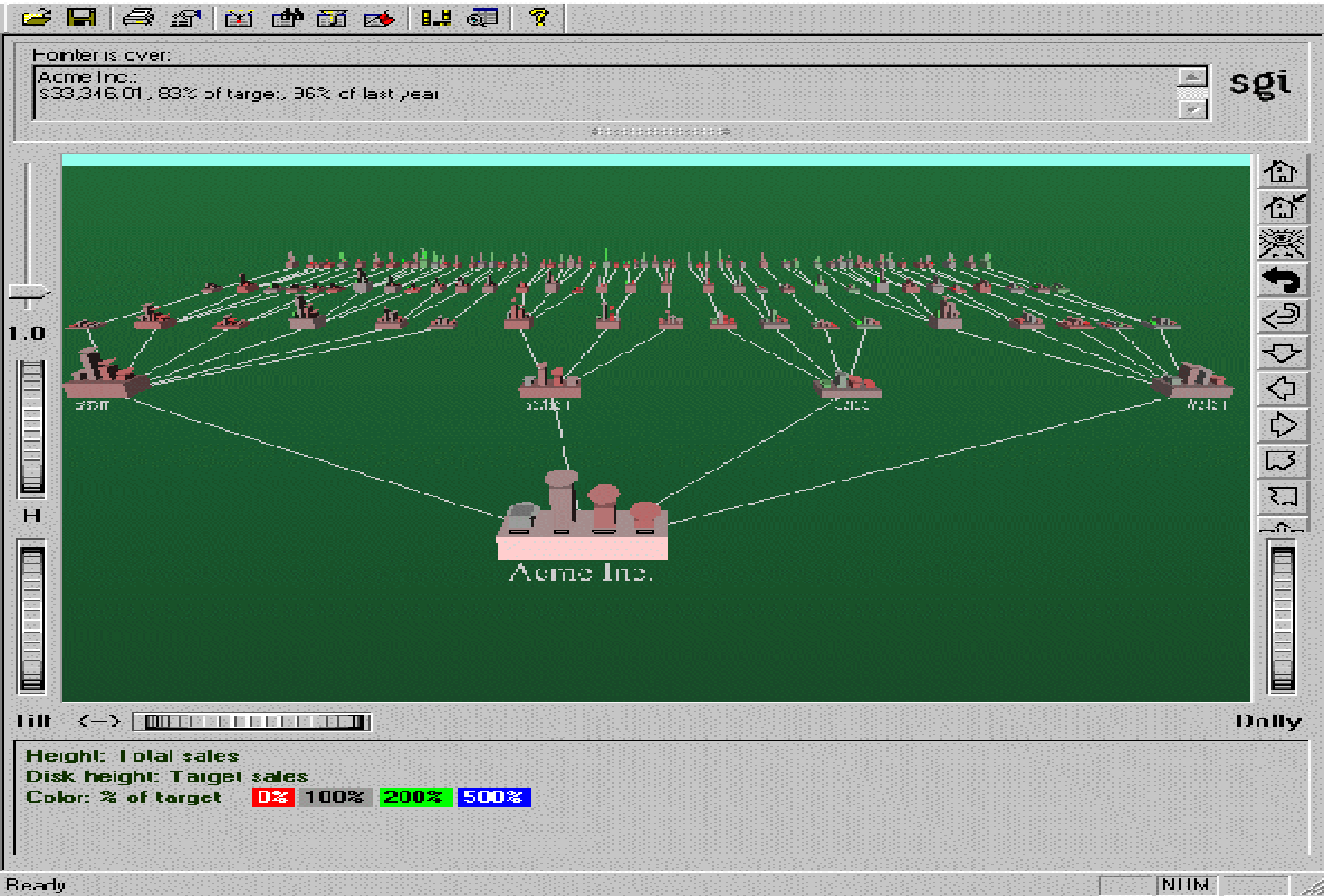
AVC-set on *credit_rating*

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 fewer scans of DB, an incremental alg.

Visualization of a Decision Tree in SGI/MineSet 3.0



Pointer is over:

Acme Inc.:
$33,346.01, 83% of target, 36% of last year

Height: Total sales
Disk height: Target sales
Color: % of target 0% 100% 200% 500%

Random Forest

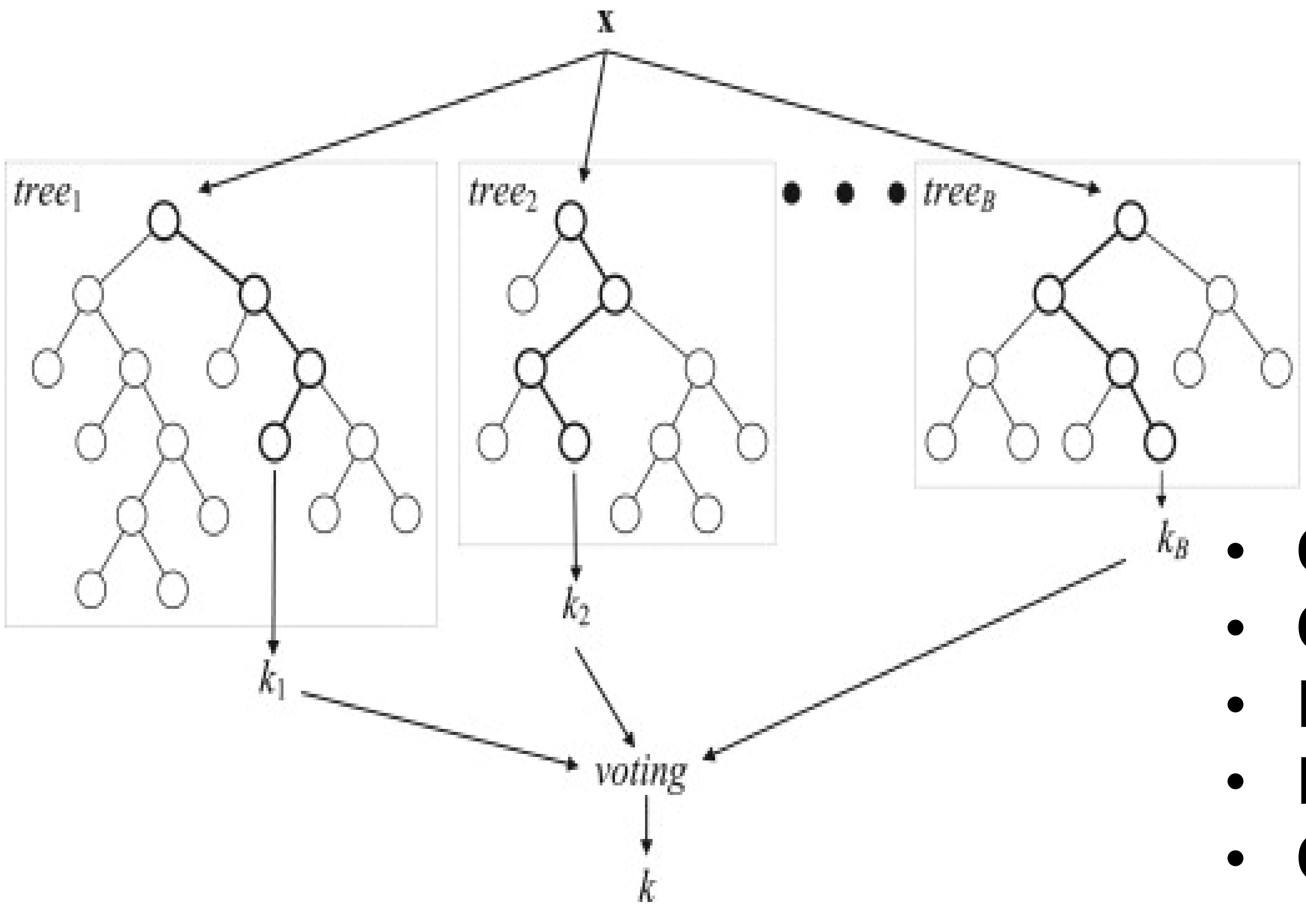
- Randomly select a sub set of features
- Create a fully grown decision tree on the n data points, with the sub feature set, sampled with replacement.
- Repeat the two steps to create a large number of trees forming a random forest.
- Apply each tree in the forest on test data and use majority vote of all trees as final prediction.
- The random forest may work better than a tree constructed from all features.
- Why does it work better? Reduce variance, bias, no feature selection, two parameters (#feature, #tree)

Advantages and Disadvantages

- It is one of the most accurate learning algorithms available.
- It runs efficiently on large databases.
- It can handle thousands of input variables without variable deletion.
- It gives estimates of what variables are important in the classification.
- It generates an internal unbiased estimate of the generalization error as the forest building progresses.
- It has an effective method for estimating missing data and maintains accuracy when a large proportion of the data are missing.
- It computes proximities between pairs of cases that can be used in clustering, locating outliers, or (by scaling) give interesting views of the data.
- The capabilities of the above can be extended to unlabeled data, leading to unsupervised clustering, data views and outlier detection.
- It offers an experimental method for detecting variable interactions.
- Disadvantages: overfitting

Wikipedia

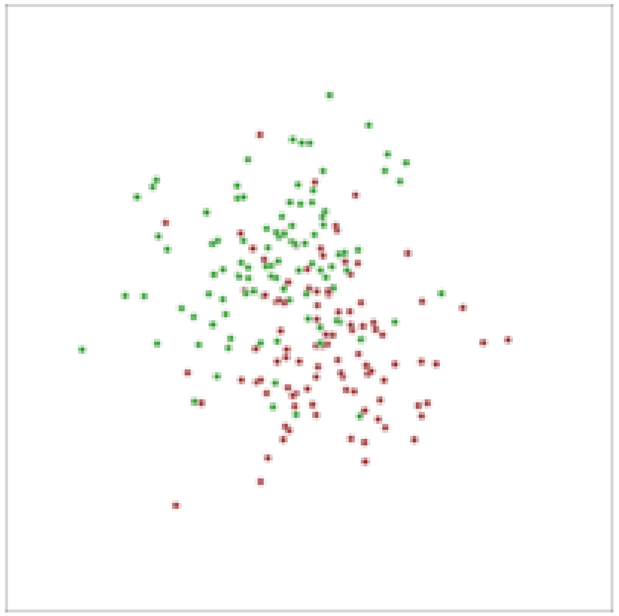
Mining Data with Random Forest



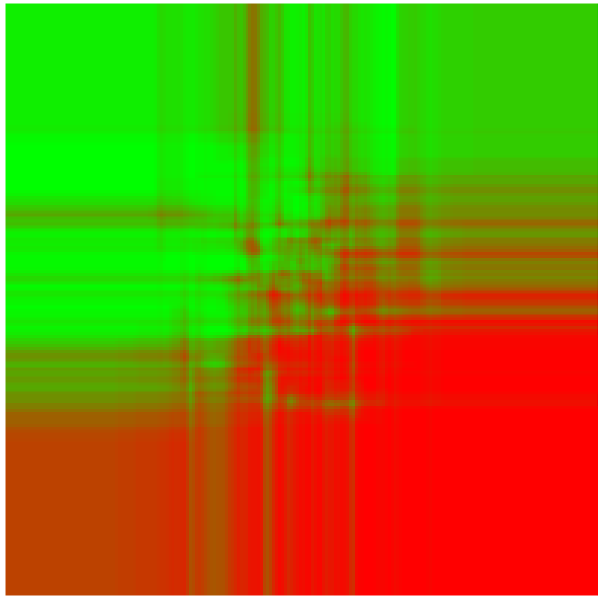
- **Classification**
- **Clustering**
- **Regression**
- **Feature Selection**
- **Outlier detection**

Review paper: http://www.sciencedirect.com/science/article/pii/S0031320310003973

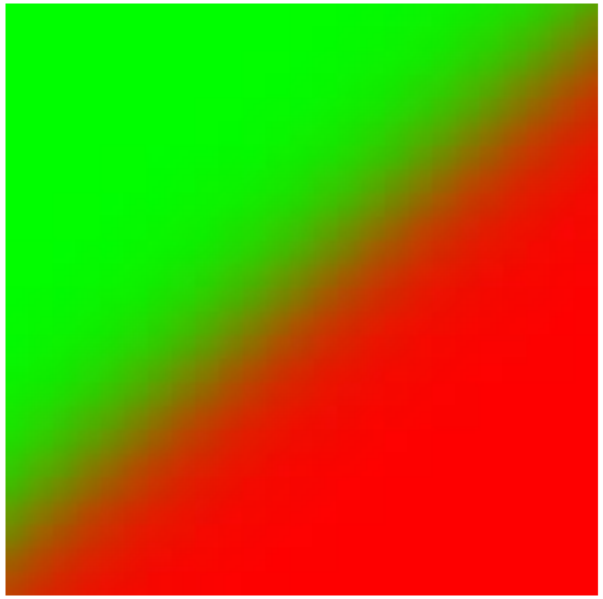
An Example



Two Gaussian point clouds



Random Forest



Logistic regression

Wikipedia

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification ←
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- 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: Basics

- Let **X** be a data sample ("*evidence*"), class label is unknown
- Let H be a *hypothesis* that X belongs to class C
- Classification is to determine P(C|**X**), the probability that the hypothesis holds given the observed data sample **X**
- P(C) (*prior probability*), the initial probability of C
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|C) (likelihood), the probability of observing the sample **X**, given that the hypothesis H holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- Given training data **X**, *posteriori probability of a hypothesis* H, P(C|**X**), follows the Bayes theorem

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

- Informally, this can be written as

  posteriori = prior x likelihood / evidence

- Predicts **X** belongs to C_i iff the probability $P(C_i|\mathbf{X})$ is the highest among all the $P(C_k|X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayesian Classifier

- 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 $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem (1<= *i* <= m)

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

- Since P(X) is constant for all classes, only

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

needs to be maximized

Derivation of Naïve Bayes Classifier

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

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

- This greatly reduces the computation cost: Only counts the class distribution
- 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 continuous-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

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

Naïve Bayesian Classifier: Training Dataset

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: An 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")

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)

Avoiding the 0-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be non-zero. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10),
- Use Laplacian correction (or Laplacian estimator)
  - Adding 1 to each case

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayesian Classifier: Comments

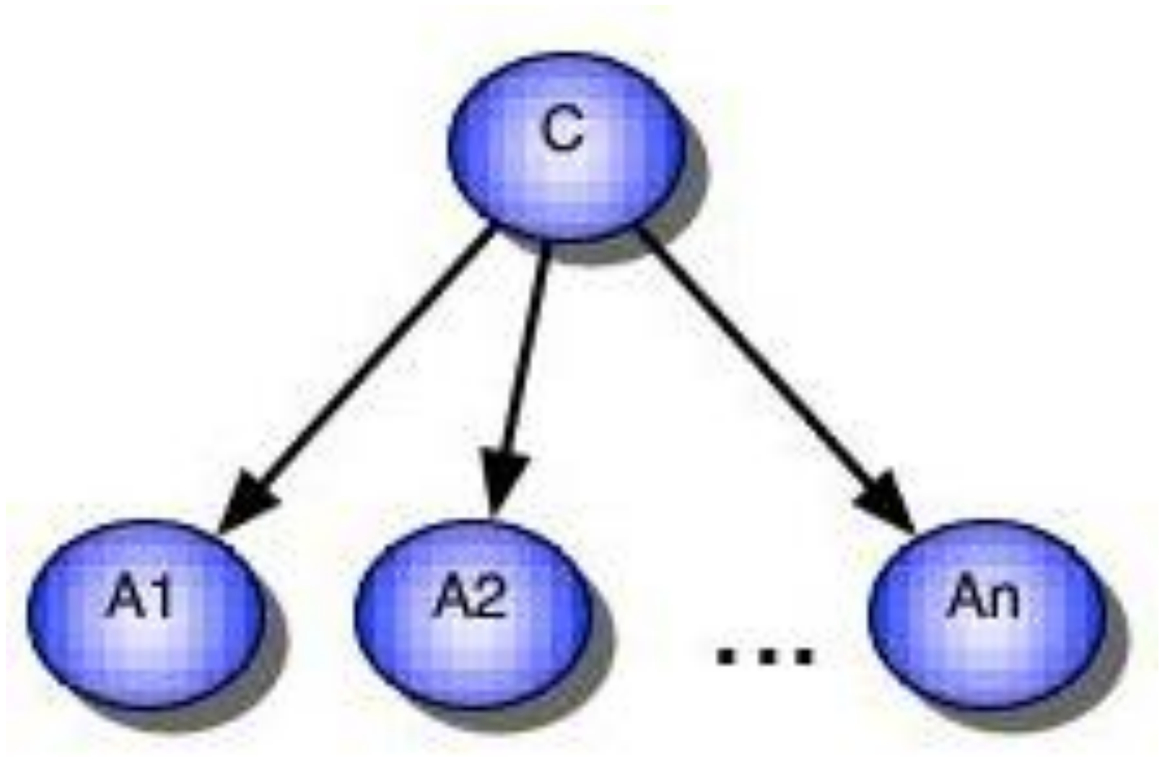
- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., salary and age.

      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

Weka Demo

Vote Classification

Graphical Model for Naïve Bayes Classifier



http://people.csail.mit.edu/kersting/profile/PROFILE_nb.html

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of IF-THEN rules

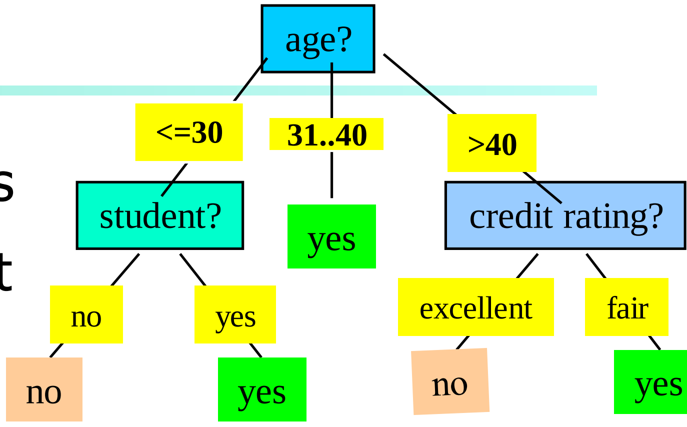
  R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

  - Rule antecedent/precondition vs. rule consequent
- Assessment of a rule: *coverage* and *accuracy*
  - n_{covers} = # of data points covered by R
  - $n_{correct}$ = # of data points correctly classified by R

  coverage(R) = n_{covers} /|D| /* D: training data set */

  accuracy(R) = $n_{correct}$ / n_{covers}
- If more than one rule is triggered, need **conflict resolution**
  - Size ordering: assign the highest priority to the triggering rules that has the "toughest" requirement (i.e., with the *most attribute test*)
  - Rule-based ordering (**decision list**): rules are organized into one long priority list, according to some measure of rule quality (accuracy) or by experts

Rule Extraction from a Decision Tree



- Rules are easier to understand than large trees
- 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 holds the class prediction
- Rules are mutually exclusive and exhaustive
- 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*

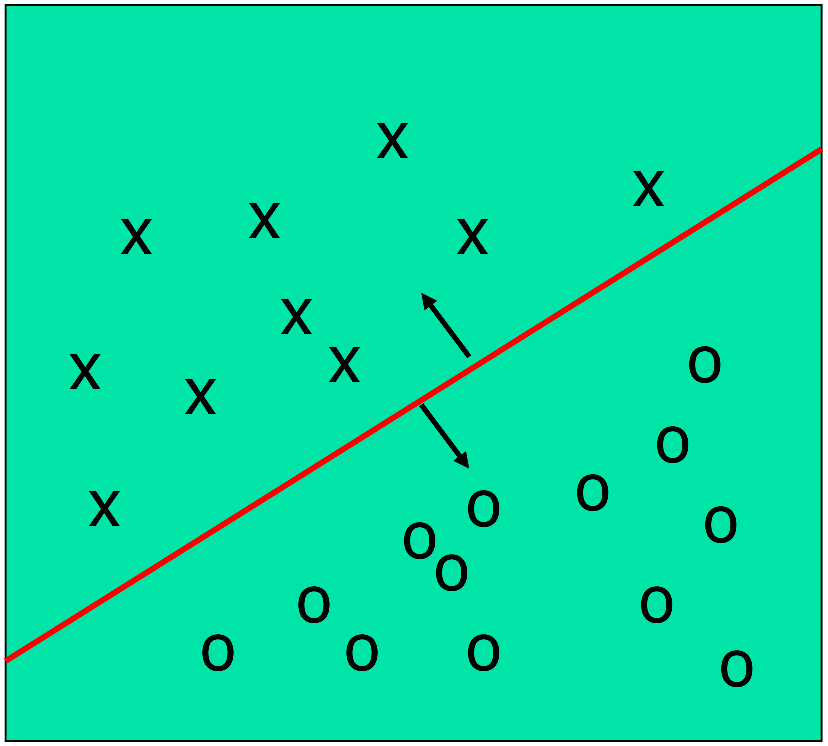
Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Classification: A Mathematical Mapping

- Classification:
  - predicts categorical class labels
- E.g., Personal homepage classification
  - $x_i = (x_1, x_2, x_3, \ldots)$, $y_i = +1$ or -1
  - x_1 : # of a word "homepage"
  - x_2 : # of a word "welcome"
- Mathematically
  - $x \in X = \Re^n$, $y \in Y = \{+1, -1\}$
  - We want a function $f: X \rightarrow Y$

Linear Classification



- Binary Classification problem
- The data above the red line belongs to class 'x'
- The data below red line belongs to class 'o'
- Examples: SVM, Perceptron

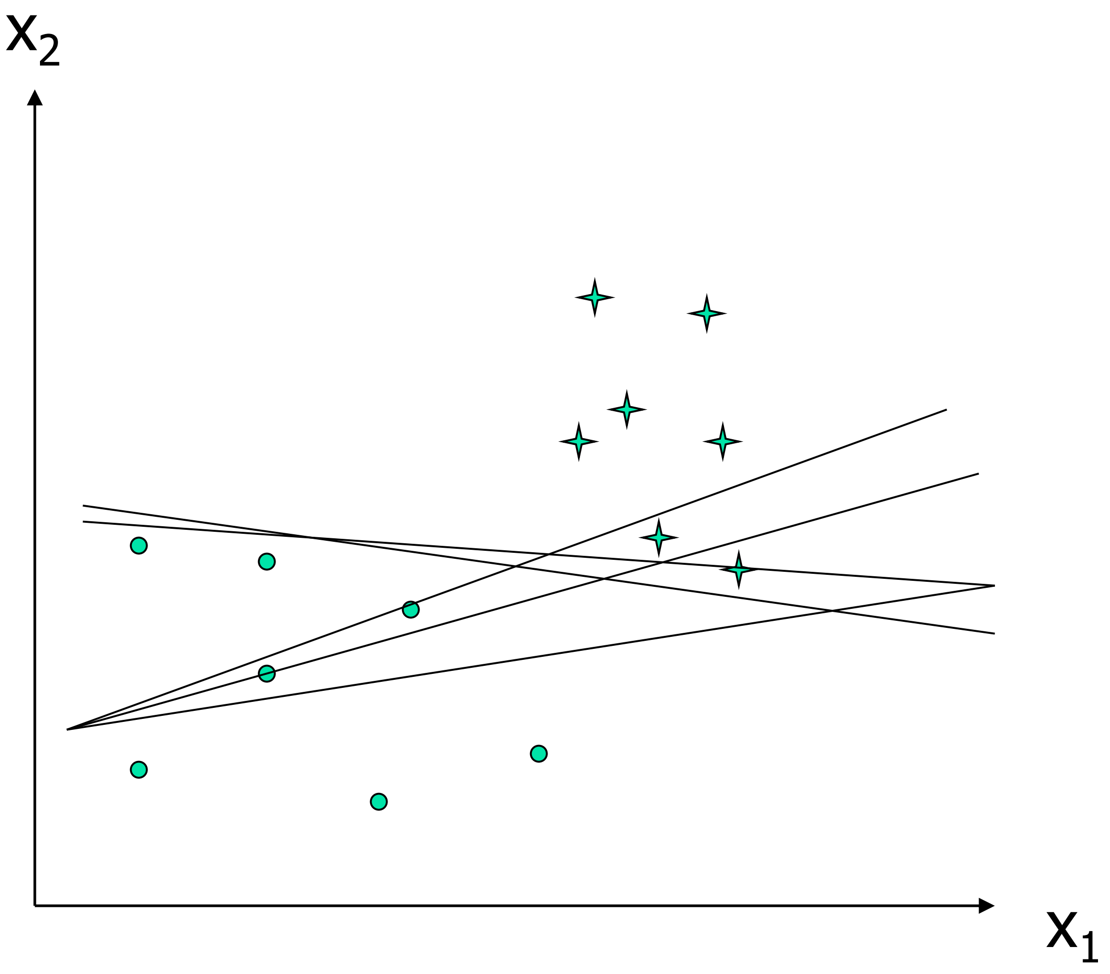
Generative VS Discriminative

Generative Classifier: develop a model to simulate the generation of data of each class (Naïve Bayes, HMM)
Discriminative Classifier: find a function to separate different groups of data
(decision tree, perceptron, neural networks, support vector machines, etc)

Discriminative Classifiers

- Advantages
  - prediction accuracy is generally high
    - As compared to Bayesian methods – in general
  - robust, works when training examples contain errors
  - fast evaluation of the learned target function
    - Bayesian networks are normally slow
- Criticism
  - Cannot simulate data generation
  - long training time
  - difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

Perceptron



- Vector: **x**, **w**
- Scalar: x, y, w

Input: $\{(\mathbf{x}_1, t_1), \ldots\}$

Output: classification function $f(\mathbf{x})$

$f(\mathbf{x}_i) > 0$ for $t_i = +1$

$f(\mathbf{x}_i) < 0$ for $t_i = -1$

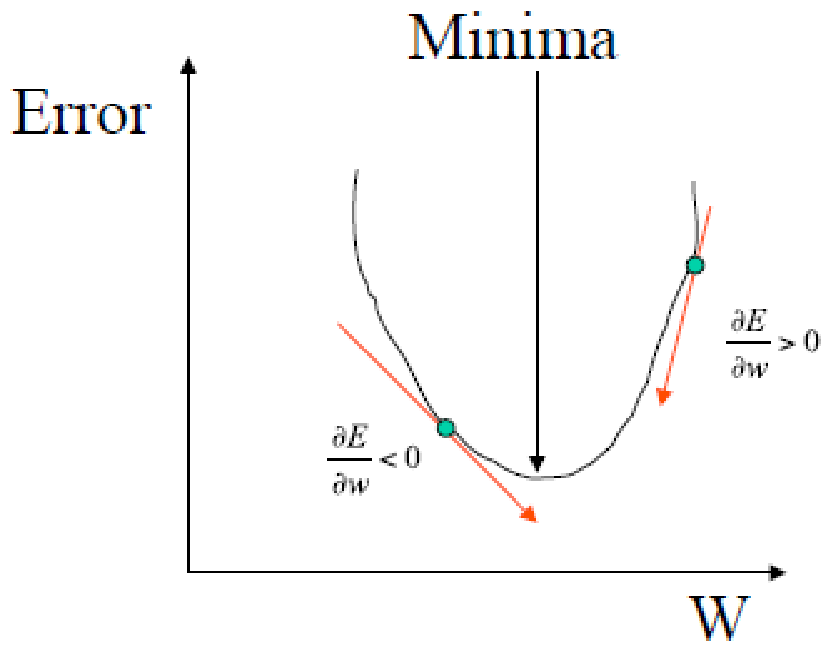
$f(x) = w_1x_1+w_2x_2+b = 0$

- Perceptron: update W additively

Perceptron Criterion

- Minimize classification error
- Input data (vector): $x^1, x^2, \ldots, x^N$ and corresponding target value $t^1, t^2, \ldots, t^N$.
- Goal: for all x in C_1 ($t = 1$), $w^T x > 0$, for all x in C_2 ($t = -1$), $w^T x < 0$. Or for all x: $w^T x t > 0$.
- Error: $E^{perc}(w) = -\sum_{x^n \in M} w^T x^n t^n$. M is the set of misclassified data points.

Gradient Descent


Minima
Error
W
$\frac{\partial E}{\partial w} > 0$
$\frac{\partial E}{\partial w} < 0$

For each misclassified data point, adjust weight as follows:

$$w = w - \frac{\partial E}{\partial w} \times \eta = w + \eta\, x^n t^n$$

Perceptron Algorithm

- Initialize weight w
- **Repeat**

  For each data point (x^n, t^n)

  Classify each data point using current w.

  If $w^T x^n t^n > 0$ (correct), do nothing

  If $w^T x^n t^n < 0$ (wrong), $w^{new} = w + \eta x^n t^n$

  $w = w^{new}$
- **Until** w is not changed (all the data will be separated correctly, if data is linearly separable) or error is below a threshold.

Rosenblatt, 1962

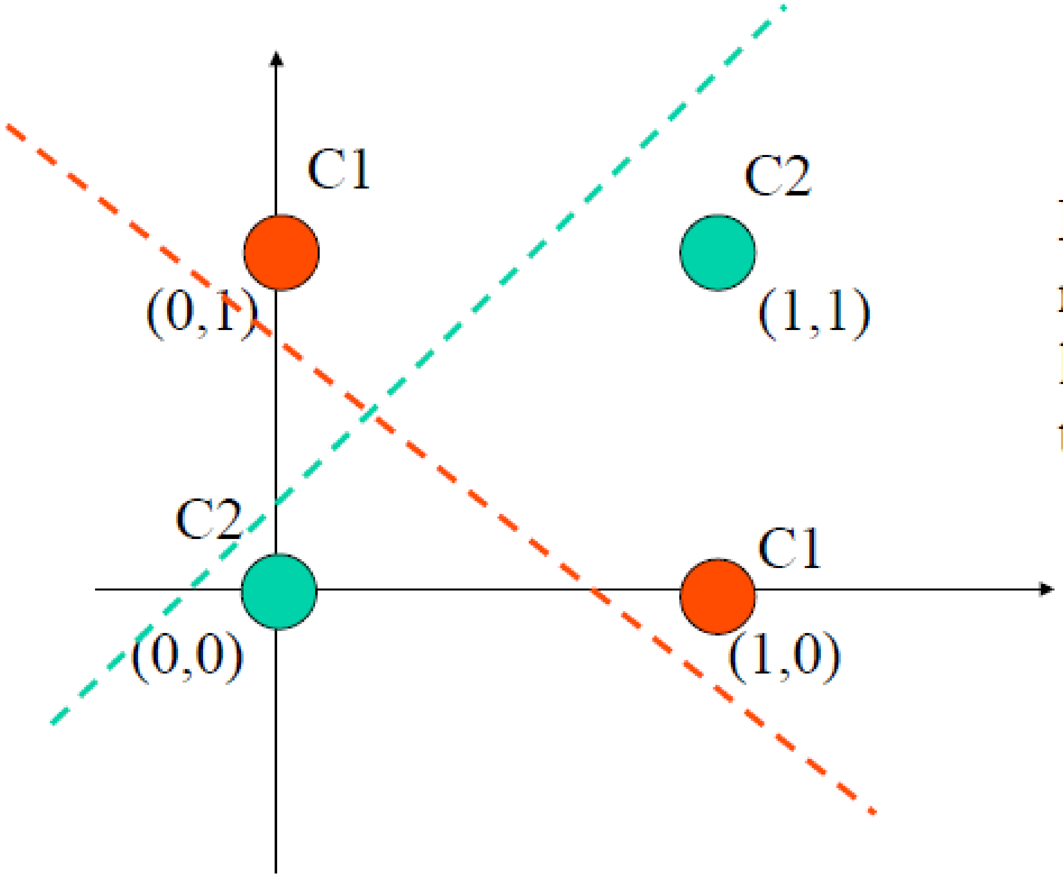
Perceptron Learning Movie

- https://www.youtube.com/watch?v=vGwemZhPlsA

Perceptron Convergence Theorem

- For any data set which is linearly separable, the algorithm is guaranteed to find a solution in a finite number of steps (Rosenblatt, 1962; Block 1962; Nilsson, 1965; Minsky and Papert 1969; Duda and Hart, 1973; Hand, 1981; Arbib, 1987; Hertz et al., 1991)

Exclusive OR Problem



Perceptron (or one-layer neural network) can not learn a function to separate the two classes perfectly.

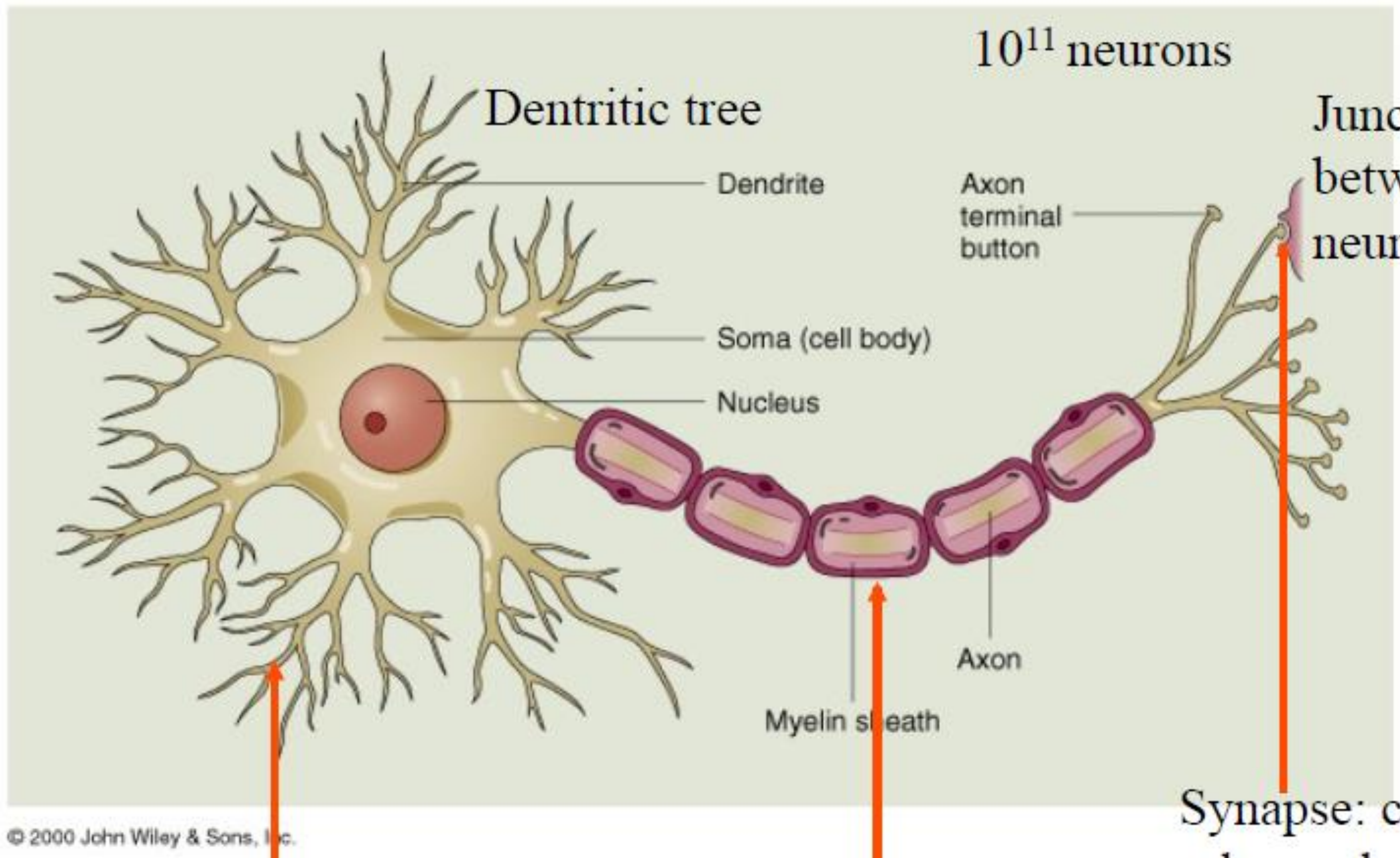
Extend to Neural Networks

- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- A neural network: A set of connected input/output units where each connection has a **weight** associated with it
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

Brain

Images.google.com

A Typical Cortical Neuron

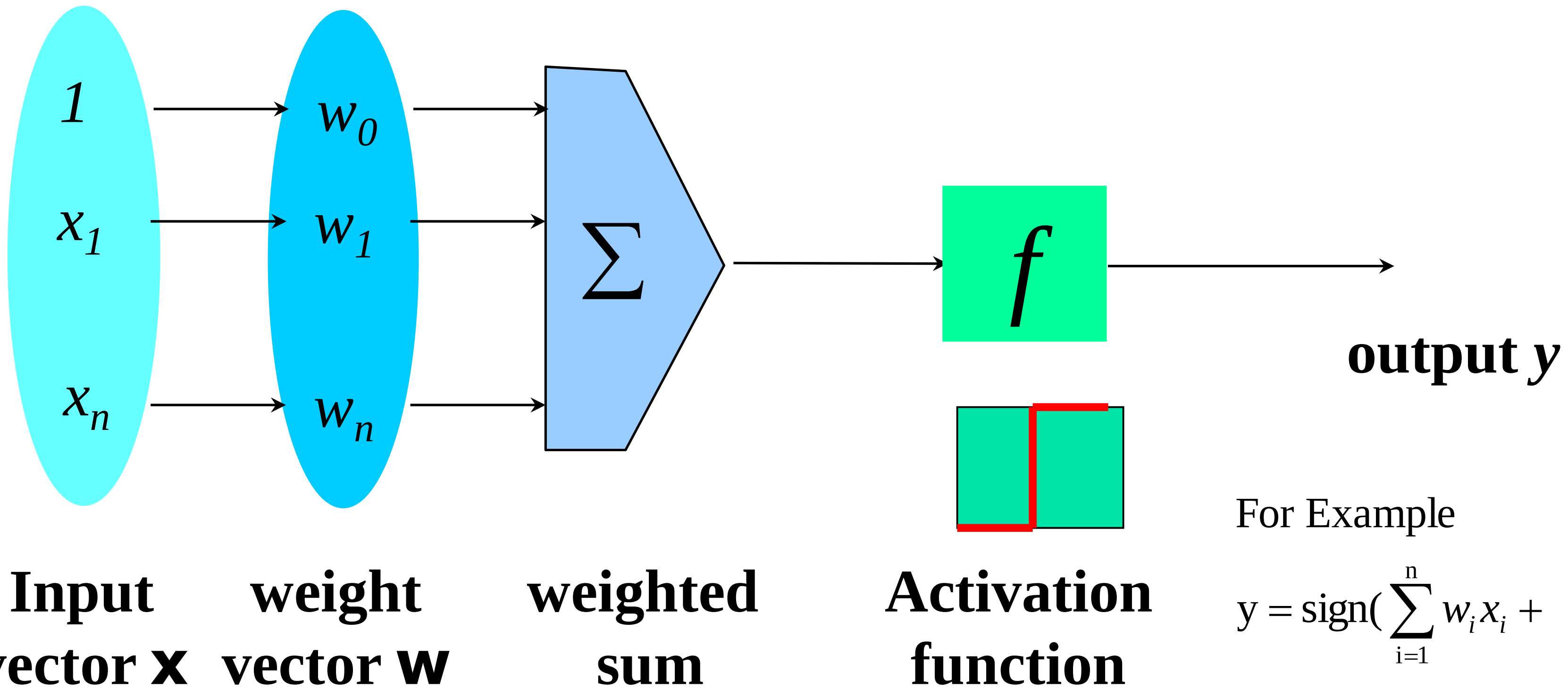


Collect chemical signals

Axon: generate Potentials (Fire/not Fire)

Synapse: control release chemical transmitters.

A Neuron (= a perceptron)

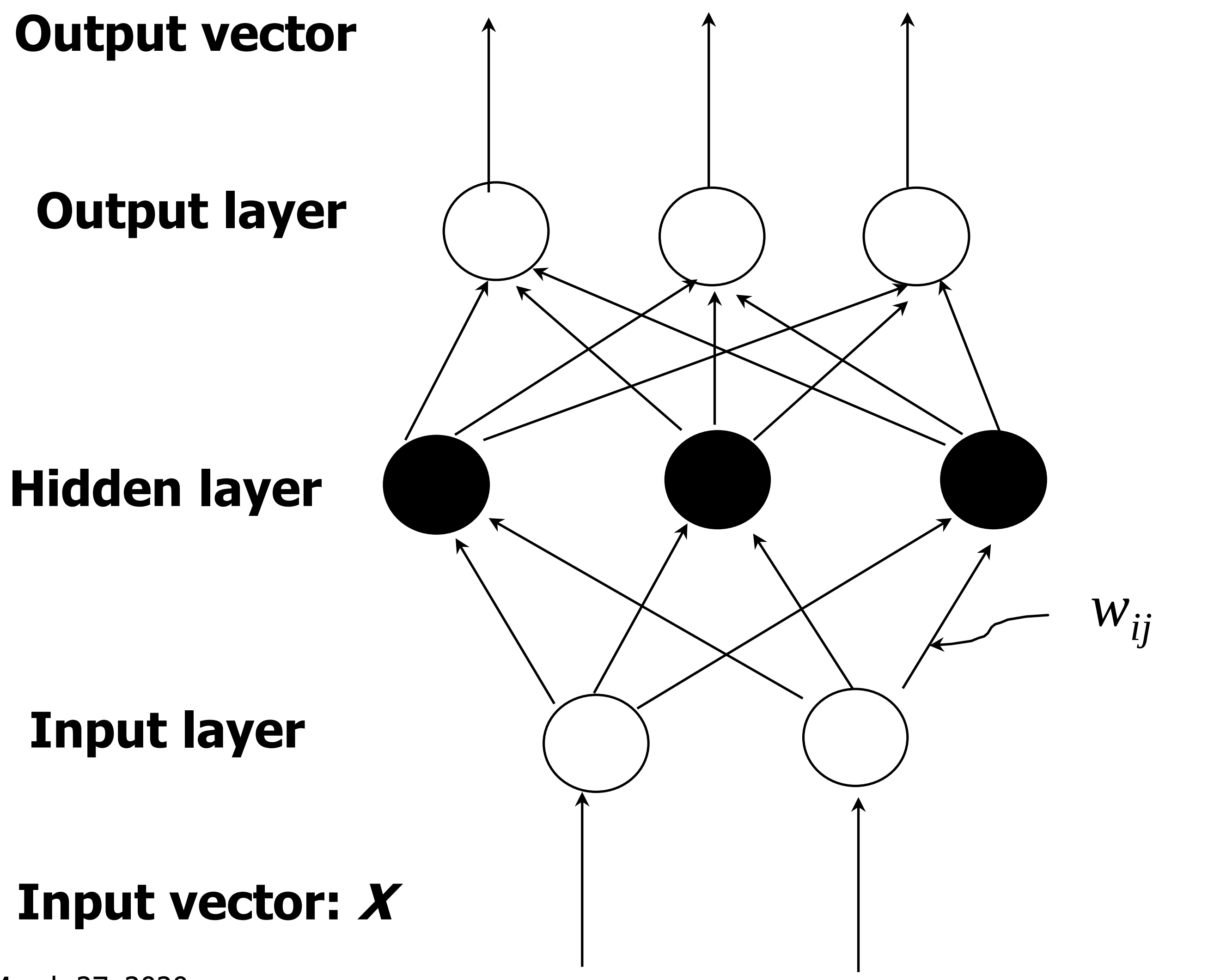


For Example

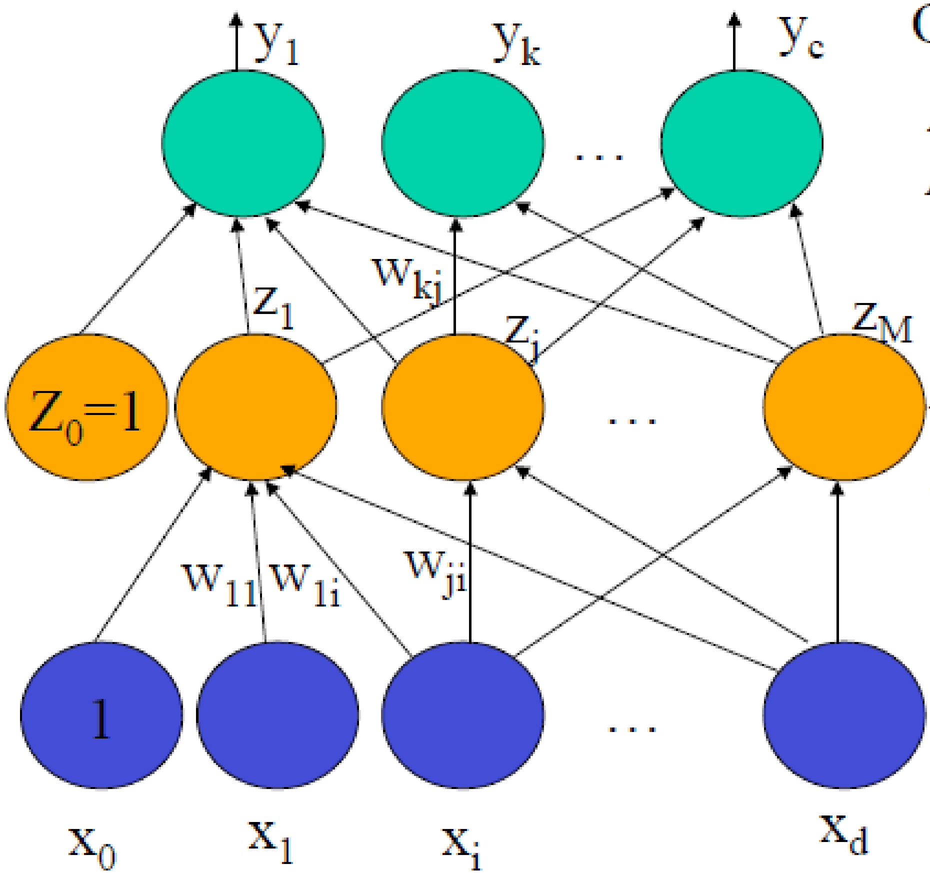
$$y = \mathrm{sign}\left(\sum_{i=1}^{n} w_i x_i + w_0\right)$$

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

A Multi-Layer Feed-Forward Neural Network



Two-Layer Neural Network



Output

Activation function: f (linear,sigmoid, softmax)

Activation of unit a_k:

$$\sum_{j=0}^{M} w_{kj} z_j$$

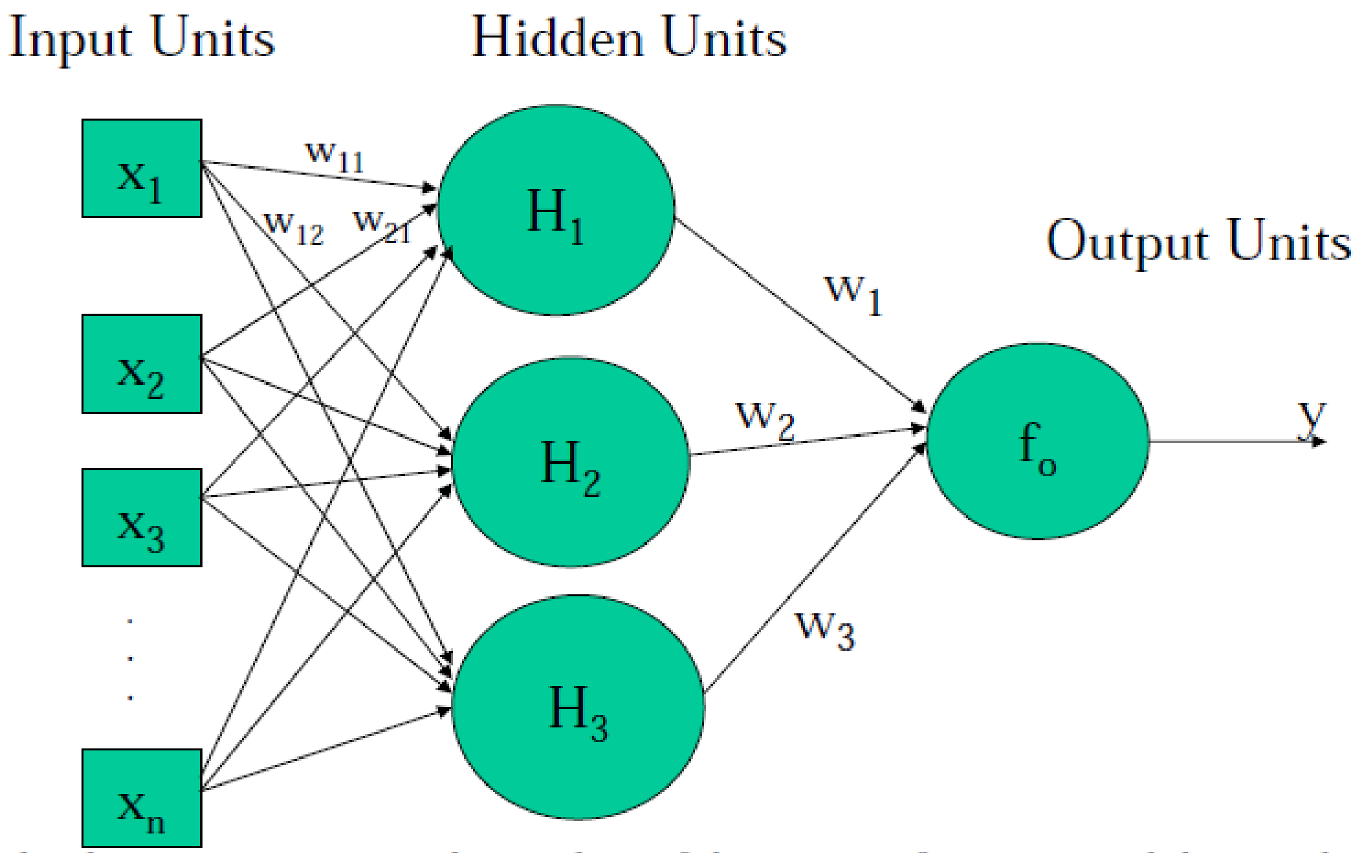
Activation function: g (linear, tanh, sigmoid)

Activation of unit a_j:

$$\sum_{i=0}^{d} w_{ji} x_i$$

$$y_k = f(\sum_{j=0}^{M} w_{kj} \times g(\sum_{i=0}^{d} w_{ji} x_i))$$

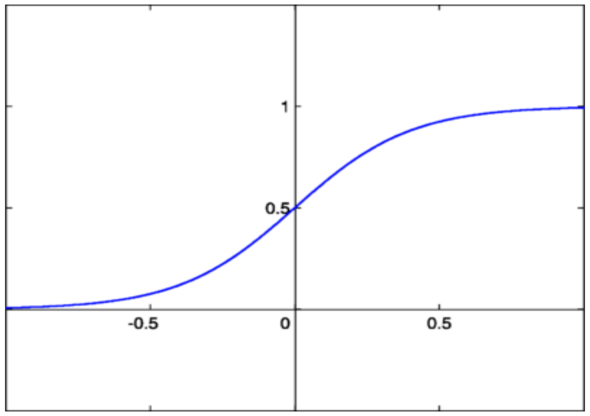
A General Neural Network



Each weighted connection means the product of the output of one unit and the weight is sent to another unit as input. Each hidden unit and output unit have a transfer function to convert the sum of inputs into an output. Let transfer function of hidden unit be f_h (e.g., identity function) output unit to be f_o (e.g., sigmoid function, $1/(1+e^{-x})$).

Activation / Transfer Function

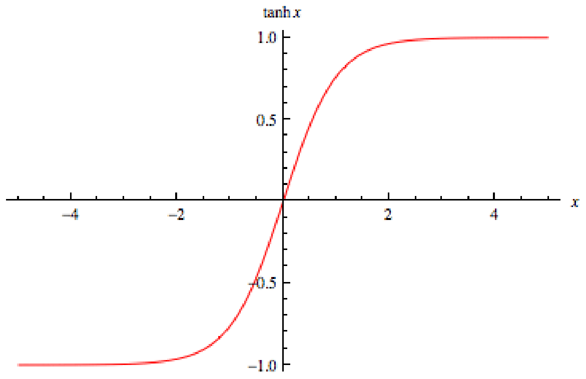
- Sigmoid function:



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

- Tanh function

What is derivative of f(x)?



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

Neural Network is a Universal Function Approximator

We can represent neural network as an function:

$$y = f_o\left(\sum_i w_i f_h\left(\sum_j x_j w_{ij}\right)\right)$$

This function is universal, which means that any function y=*f*(x) can be approximated by this function accurately, given a set of appropriate weights W.

So, the key is to adjust weights W to make neural network to approximate the function of our interest. e.g., given input of sequence features, tell if it is a gene or not (1: yes, 0: no)?

Adjust Weights by Training

- How to adjust weights?
- Adjust weights using known examples (training data) $(x_1,x_2,x_3,\ldots x_n,y)$. This process is called training or learning
- Try to adjust weights so that the difference between the output of the neural network and y (called target) becomes smaller and smaller.
- Goal is to minimize Error (difference)

Adjust Weights using Gradient Descent (back-propagation)

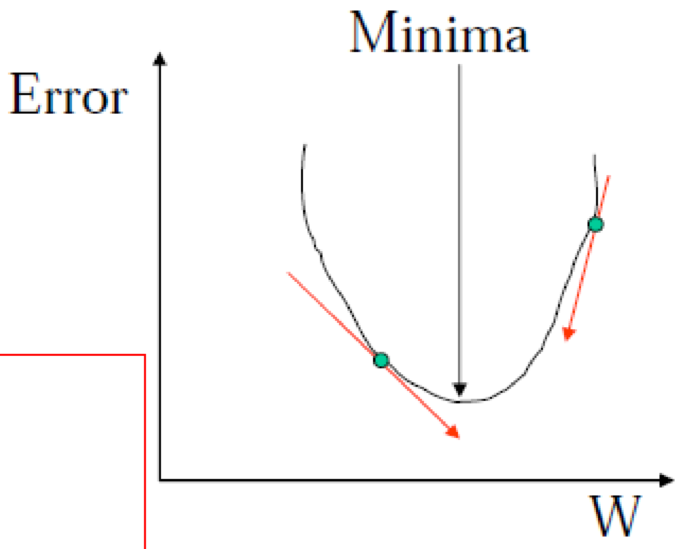
Known:

Data: $(x_1, x_2, x_3, \dots, x_n)$ (y)

Unknown weights w:

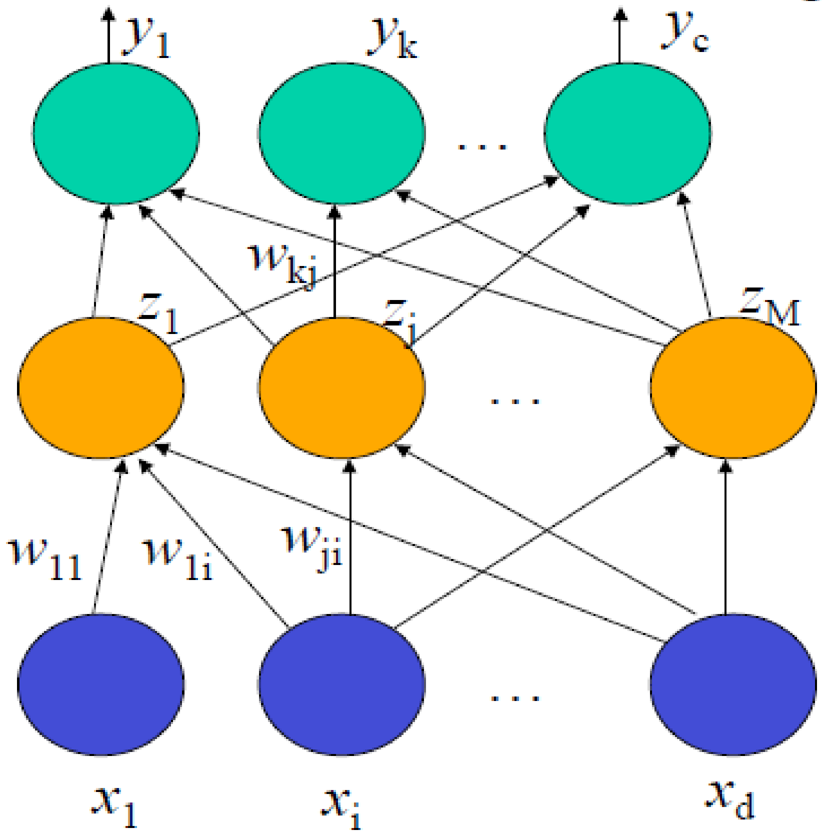
$w_{11}, w_{12}, \dots$

```
Randomly initialize weights
Repeat
    for each example, compute output $o$
        calculate error $E = (o-y)^2$
        compute the derivative of $E$ over $w$: $dw = \frac{\partial E}{\partial w}$
        $w_{new} = w_{prev} - \eta * dw$
Until error doesn't decrease or max num of iterations
```



Note: η is learning rate or step size.

Forward Propagation



Output y_k

Activation function: f (linear,sigmoid, softmax)

Activation of unit a_k: $\sum_{j=1}^{M} w_{kj} z_j$

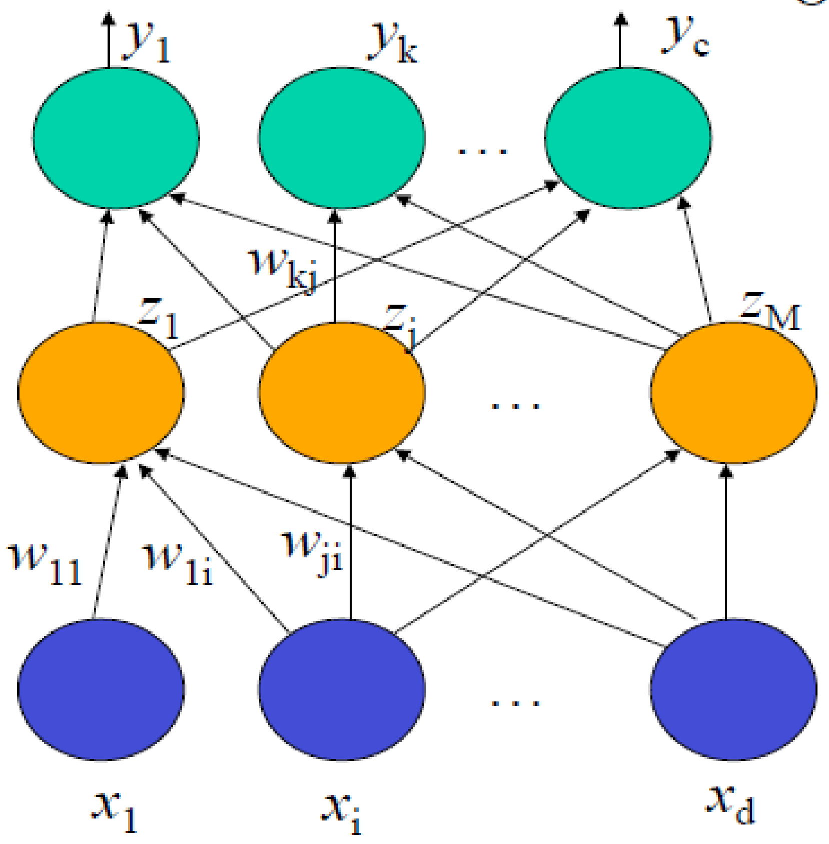
z_j

Activation function: g (linear, tanh, sigmoid)

Activation of unit a_j: $\sum_{i=1}^{d} w_{ji} x_i$

Time complexity?

Forward Propagation



Output y_k

Activation function: f (linear,sigmoid, softmax)

Activation of unit a_k: $\sum_{j=1}^{M} w_{kj} z_j$

z_j

Activation function: g (linear, tanh, sigmoid)

Activation of unit a_j: $\sum_{i=1}^{d} w_{ji} x_i$

Time complexity?
O(dM + MC) = O(W)

Matrix View of Propagation

w_{11}	**w_{12}**	**w_{13}**	**…**	**w_{1d}**
W_{21}	W_{22}	…	….	w_{2d}
….				
W_{m1}	W_{m2}			w_{md}

$* \quad (X_1, x_2, \ldots, X_d) \qquad = (a_1, a_2, \ldots, a_m)$

$\downarrow$ **Apply Transfer function**

w_{11}	**w_{12}**	**w_{13}**	**…**	**w_{1m}**
W_{21}	W_{22}	…	….	w_{2c}
….				
W_{c1}	W_{m2}			w_{cm}

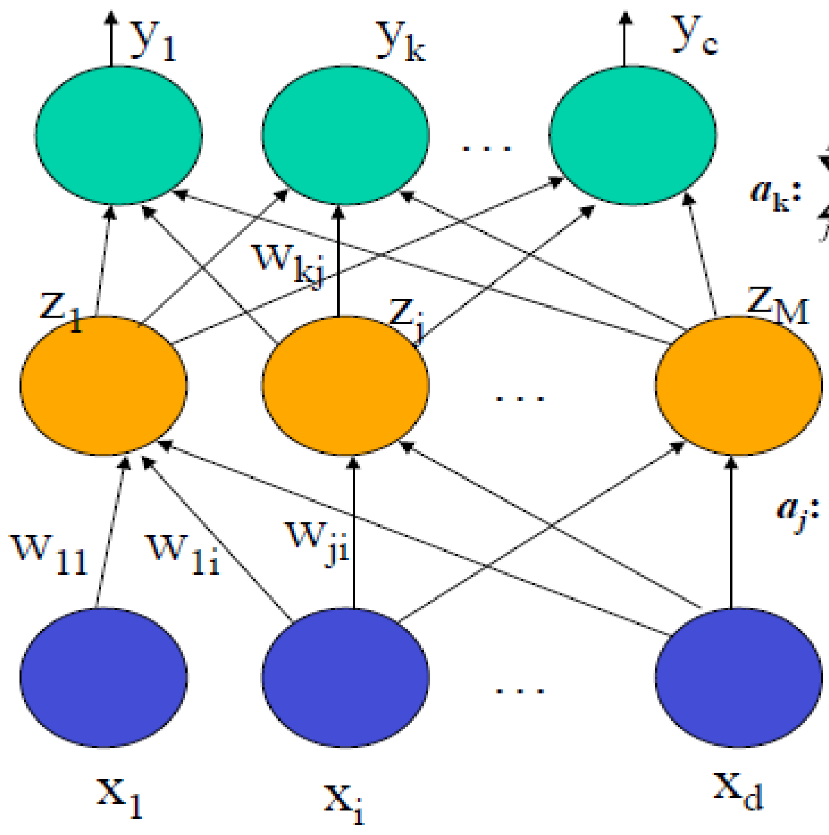
$* \quad (z_1, z_2, \ldots, z_m)$

$(b_1, b_2, \ldots, b_c) =$

Apply transfer function $\swarrow$

$(O_1, O_2, \ldots, O_c)$

Backward Propagation



f

a_k: $\sum_{j=1}^{M} w_{kj} z_j$

g

a_j: $\sum_{i=1}^{d} w_{ji} x_i$

$$E = \frac{1}{2}\sum_{k=1}^{C}(y_k - t_k)^2$$

$$\frac{\partial E}{\partial y_k} = y_k - t_k$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial a_k} = (y_k - t_k) f'(a_k) = \delta_k$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = \delta_k z_j$$

$$\frac{\partial E}{\partial a_j} = \sum_{k=1}^{c}\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial a_k}\frac{\partial a_k}{\partial z_j}\frac{\partial z_j}{\partial a_j} = \sum_{k=1}^{C}\delta_k w_{kj} g'(a_j) = \delta_j$$

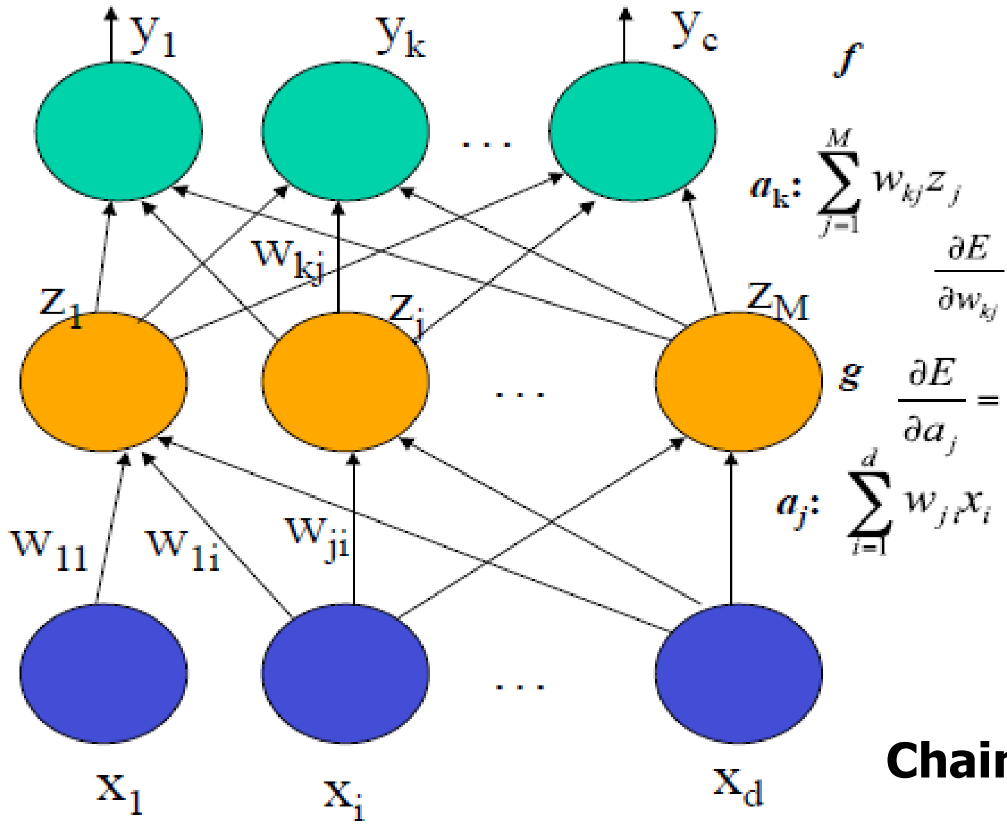
$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial wj\,i} = \delta_j x_i$$

Chain Rule of Calculating Derivatives

Time complexity?

If no back-propagation, time complexity is: (MdC+CM)

Backward Propagation



$$E = \frac{1}{2}\sum_{k=1}^{C}(y_k - t_k)^2$$

$$\frac{\partial E}{\partial y_k} = y_k - t_k$$

$$\frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial a_k} = (y_k - t_k) f'(a_k) = \delta_k$$

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} = \delta_k z_j$$

$$\frac{\partial E}{\partial a_j} = \sum_{k=1}^{c}\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial a_k}\frac{\partial a_k}{\partial z_j}\frac{\partial z_j}{\partial a_j} = \sum_{k=1}^{C}\delta_k w_{kj} g'(a_j) = \delta_j$$

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial wj\,i} = \delta_j x_i$$

Chain Rule of Calculating Derivatives

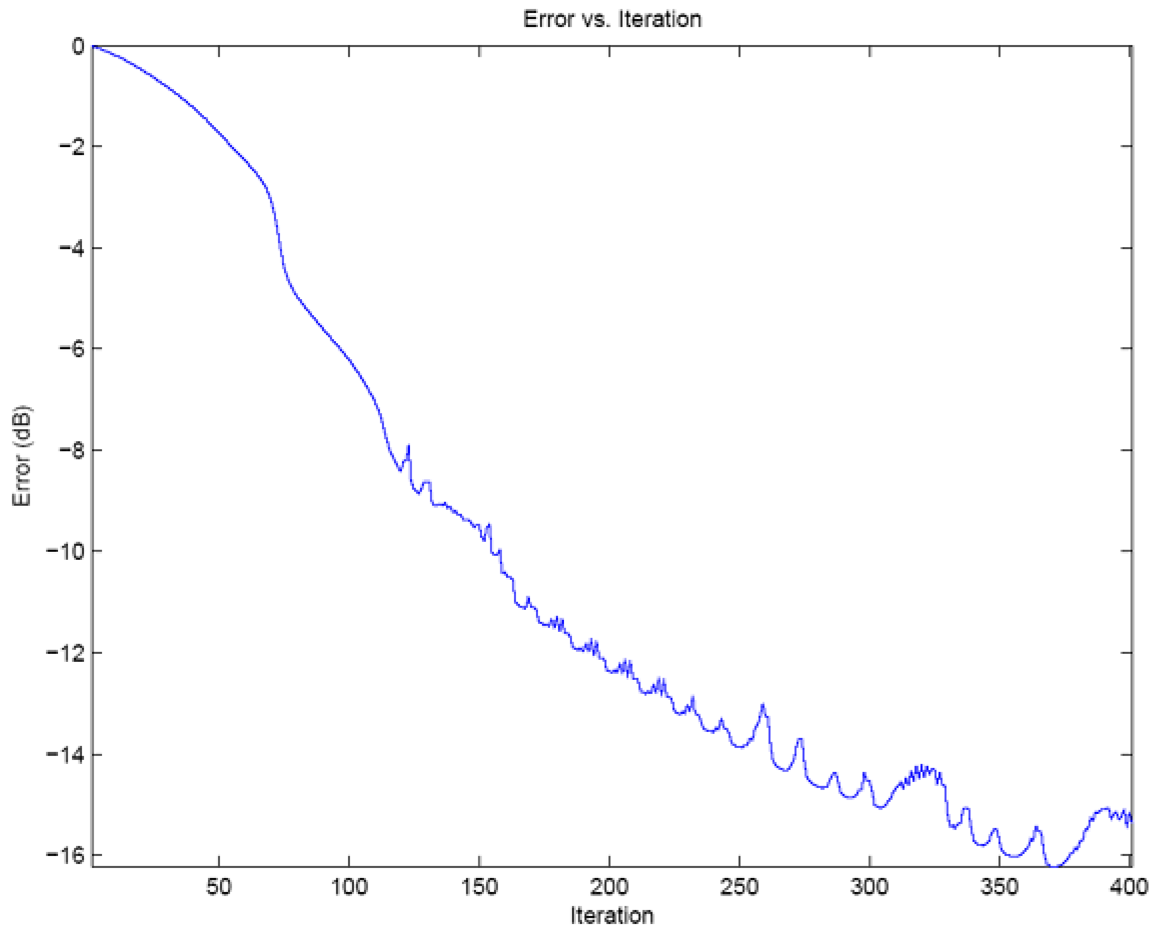
If no back-propagation, time complexity is: (MdC+CM)

Time complexity?
O(CM+Md) = O(W)

Backpropagation

- Iteratively present a set of training tuples & compare the network's prediction with the actual known target value.
- An iteration on all tuples is called an Epoch.
- For each training tuple, the weights are modified to **minimize the mean squared error** between prediction and the actual target value
- Modifications are made in the "**backwards**" direction **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)
  - Repeat propagation and back-propagation until terminating condition is reached (when error is very small, etc.)

Training Error



Defining a Network Topology

- First decide the **network topology:** # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalizing the input values for each attribute measured in the training tuples to [0.0—1.0]
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a *different network topology* or a *different set of initial weights*

Test Phase and Prediction

- Weights are known
- Given an input x, neural network generates an output O
- Binary classification: O > 0.5 → positive, O <= 0.5 → negative
- Multi-classification: Choose the class of the output node with highest value
- Regression: O is the predicted value

Neural Network as a Classifier

- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology
  - 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

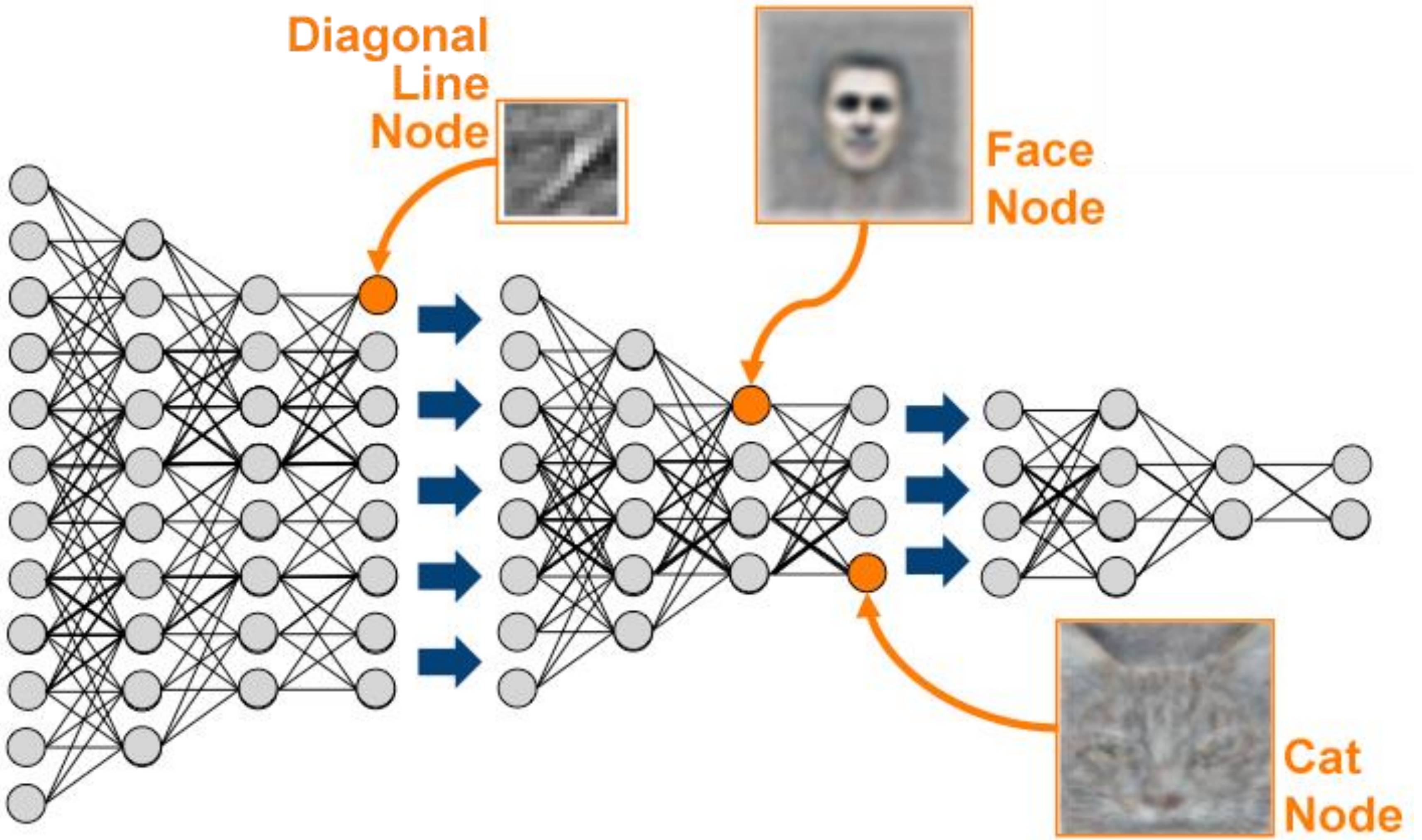
Neural Network Demo - NNClass

On sysbio.rnet.missouri.edu server
In /home/chengji/nnrank1.2

Neural Network Software

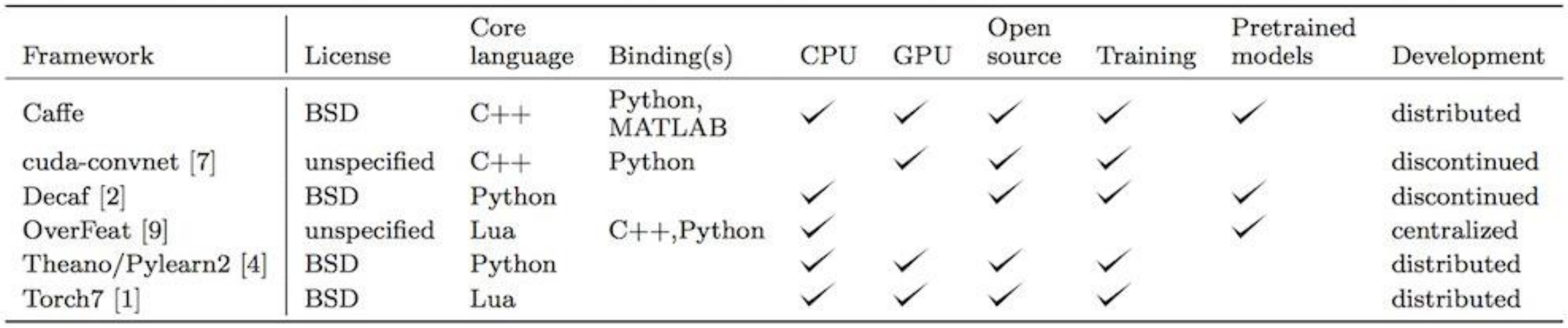
- Weka (Java)
- NNClass in NNRank package (C++, good performance, very fast): http://sysbio.rnet.missouri.edu/multicom_toolbox/tools.html
- Matlab

Most Recent Development – Deep Learning



Deep Learning Tools

- Pylearn2
- Theano
- Caffe
- Torch
- Cuda-convnet
- Deeplearning4j



Framework	License	Core language	Binding(s)	CPU	GPU	Open source	Training	Pretrained models	Development
Caffe	BSD	C++	Python, MATLAB	✓	✓	✓	✓	✓	distributed
cuda-convnet [7]	unspecified	C++	Python		✓	✓	✓		discontinued
Decaf [2]	BSD	Python		✓		✓	✓	✓	discontinued
OverFeat [9]	unspecified	Lua	C++,Python	✓				✓	centralized
Theano/Pylearn2 [4]	BSD	Python		✓	✓	✓	✓		distributed
Torch7 [1]	BSD	Lua		✓	✓	✓	✓		distributed

Six Top General Data Mining Tools

- RapidMiner
- Weka
- R-programming
- Orange (python)
- KNIME
- NLTK

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM) ⟸
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

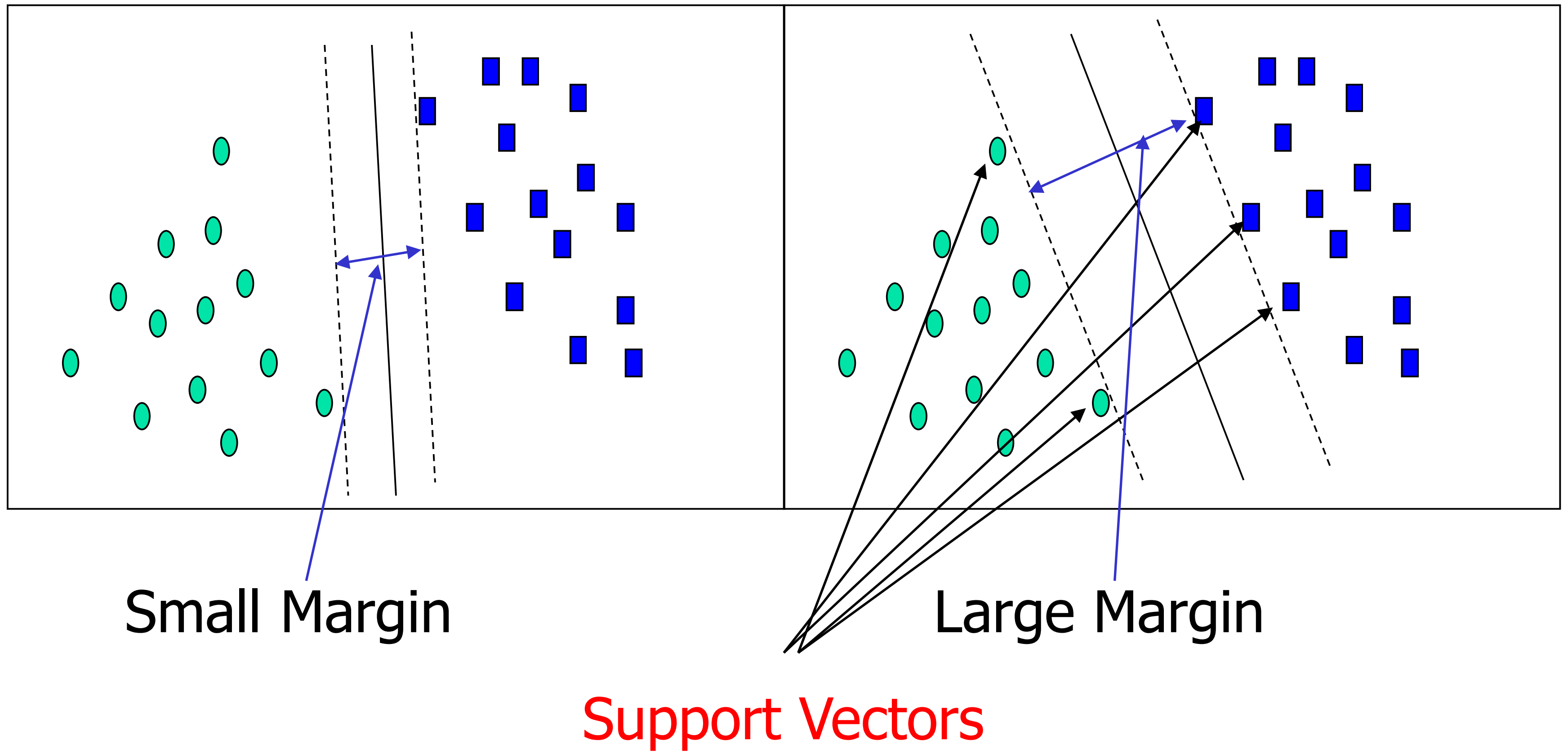
SVM—Support Vector Machines

- A new classification method for both linear and nonlinear data
- It uses a nonlinear mapping to transform the original training data into a higher dimension
- With the new dimension, it searches for the linear optimal separating hyperplane (i.e., "decision boundary")
- With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane
- SVM finds this hyperplane using support vectors ("essential" training tuples) and margins (defined by the support vectors)

SVM—History and Applications

- Vapnik and colleagues (1992)—groundwork from Vapnik & Chervonenkis' statistical learning theory in 1960s
- Features: training can be slow but accuracy is high owing to their ability to model complex nonlinear decision boundaries (margin maximization)
- Used both for classification and prediction
- Applications:
    - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

SVM—General Philosophy



SVM—Linearly Separable

- A separating hyperplane can be written as

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

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

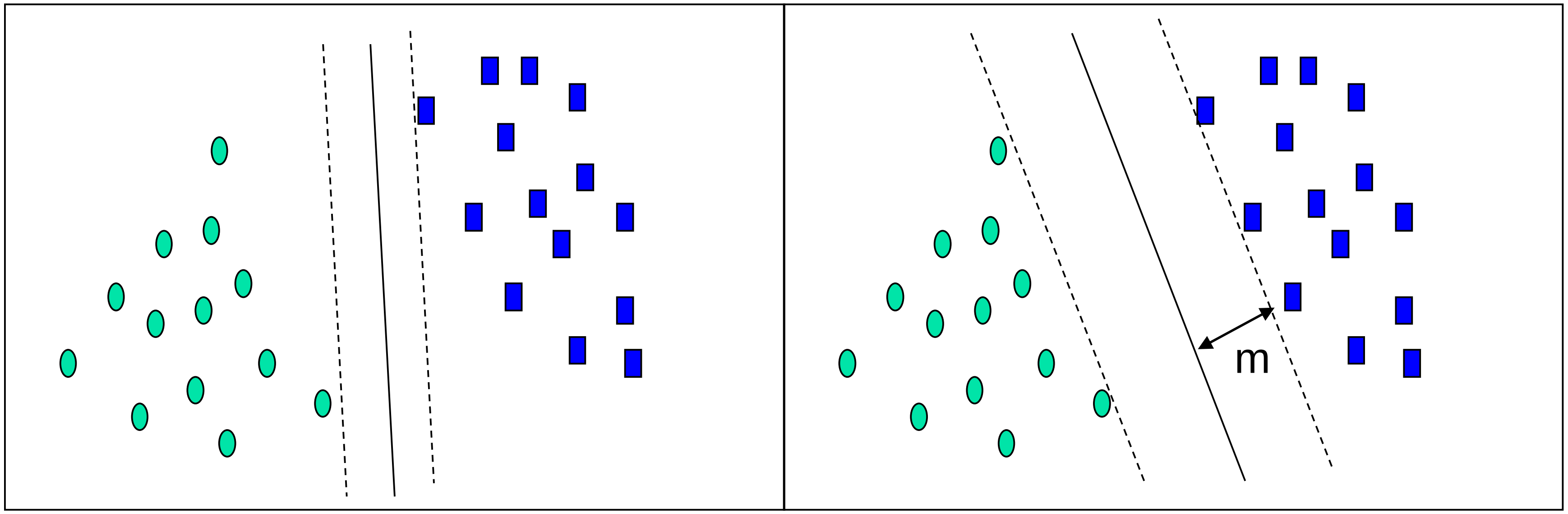
  $$w_0 + w_1 x_1 + w_2 x_2 = 0$$

- The hyperplane defining the sides of the margin:

  H_1: $w_0 + w_1 x_1 + w_2 x_2 \geq 1$ for $y_i = +1$, and

  H_2: $w_0 + w_1 x_1 + w_2 x_2 \leq -1$ for $y_i = -1$
- Any training tuples that fall on hyperplanes H_1 or H_2 (i.e., the sides defining the margin) are **support vectors**
- This becomes a **constrained (convex) quadratic optimization** problem: Quadratic objective function and linear constraints → *Quadratic Programming (QP)* → Lagrangian multipliers

SVM—When Data Is Linearly Separable

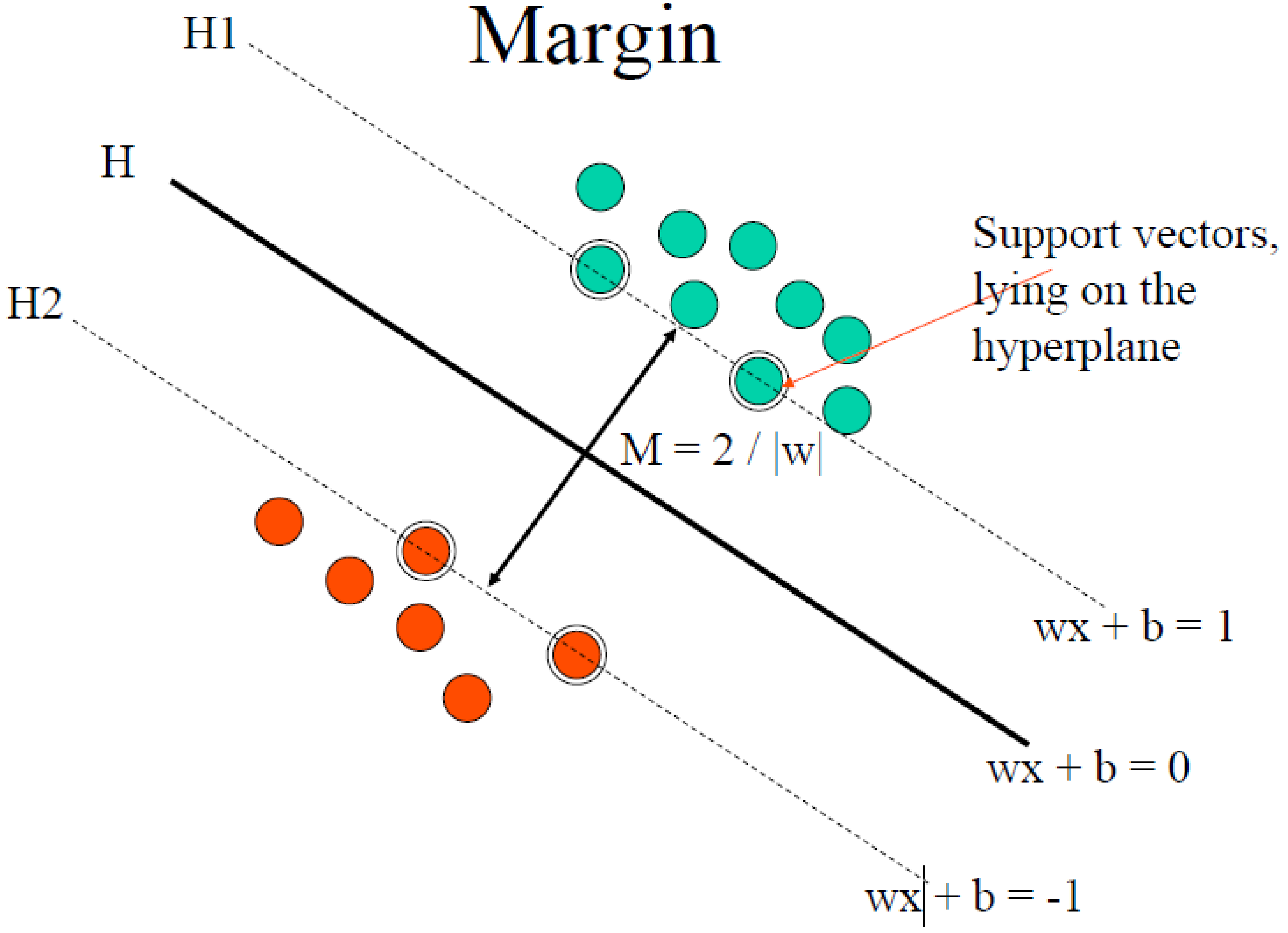


Let data D be ($\mathbf{X}_1$, y_1), …, ($\mathbf{X}_{|D|}$, $y_{|D|}$), where $\mathbf{X}_i$ is the set of training tuples associated with the class labels y_i

There are infinite lines (hyperplanes) separating the two classes but we want to find the best one (the one that minimizes classification error on unseen data)

SVM searches for the hyperplane with the largest margin, i.e., **maximum marginal hyperplane** (MMH)

Margin


H1
H
H2
Support vectors, lying on the hyperplane
M = 2 / |w|
wx + b = 1
wx + b = 0
wx + b = -1

Quadratic Optimization

- Maximize 2 / |w|, subject to the linear constraints
- $x_i . w + b >= +1$, for $y_i = +1$
- $x_i . w + b <= -1$, for $y_i = -1$
- Combined into one set of inequalities

  $y_i(x_i.w + b) – 1 >= 0$, for all i.
- How many constraints are there?
- How to solve the constraint optimization problem? (Lagrangian)

Primal Optimization

- Constraints: $y_i(x_i.w + b) - 1 >= 0$, for all i. (constraints set C1)
- Introduce a Lagrange multiplier for each inequalites: a_i.

$$L_P = \frac{1}{2}|w|^2 - \sum_{i=1}^{l} a_i y_i (x_i.w + b) + \sum_{i=1}^{l} a_i$$

Dual Optimization

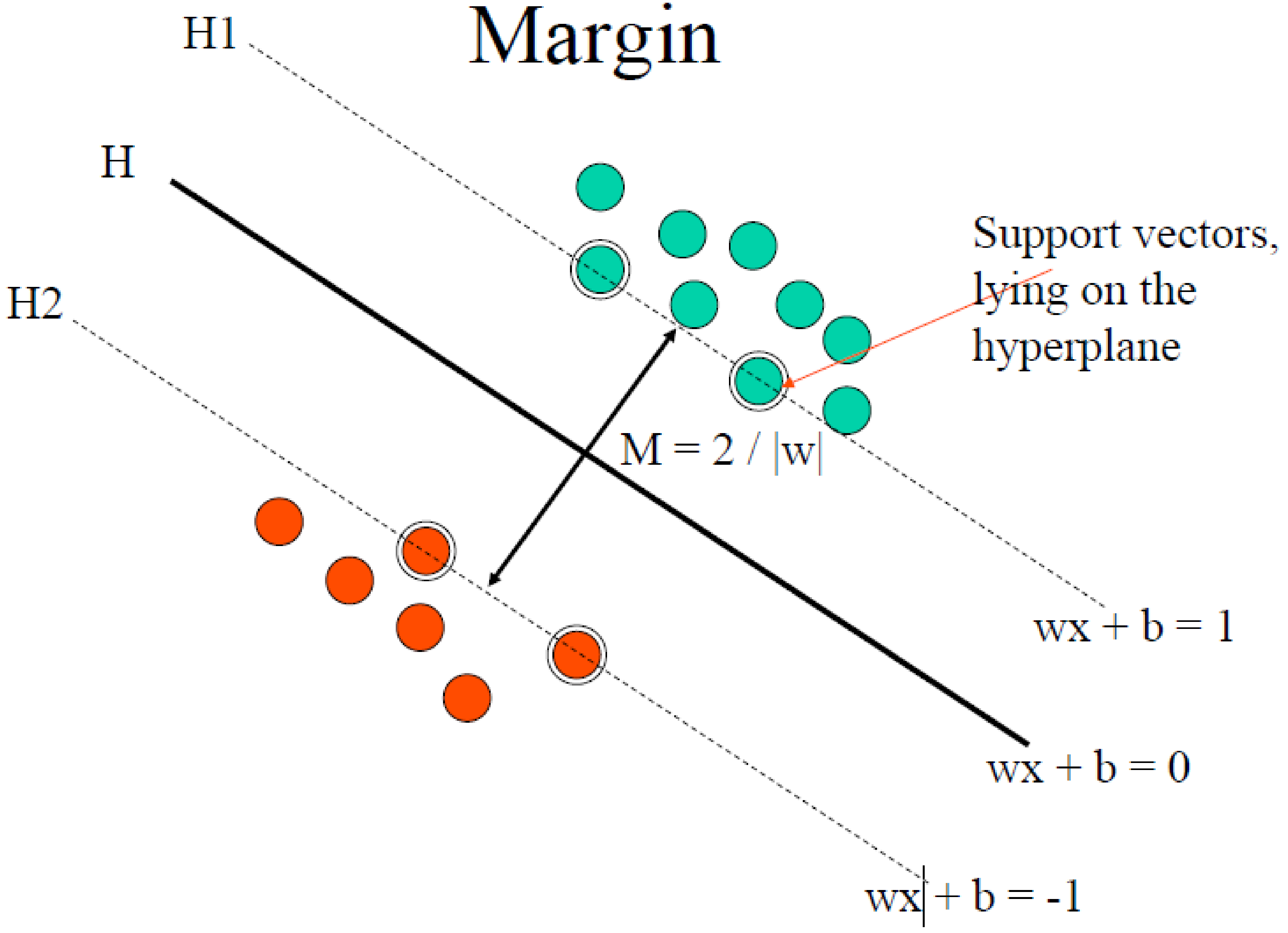
$$\text{maximize} \quad \mathcal{L}_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to} \quad \sum_i \alpha_i y_i = 0$$

Solution of w and b

$$w = \sum_{i=1}^{Ns} a_i y_i x_i$$

Margin


H1
H
H2
Support vectors, lying on the hyperplane
M = 2 / |w|
wx + b = 1
wx + b = 0
wx + b = -1

How to calculate b?

How to calculate b?

- Take a positive (resp. negative) support vector
- Given $\mathbf{W} \bullet \mathbf{X} + b = 1$
- $b = 1 - \mathbf{W} \bullet \mathbf{X}$
- Good practice is to use all the support vectors to caculate b and take the average

Prediction

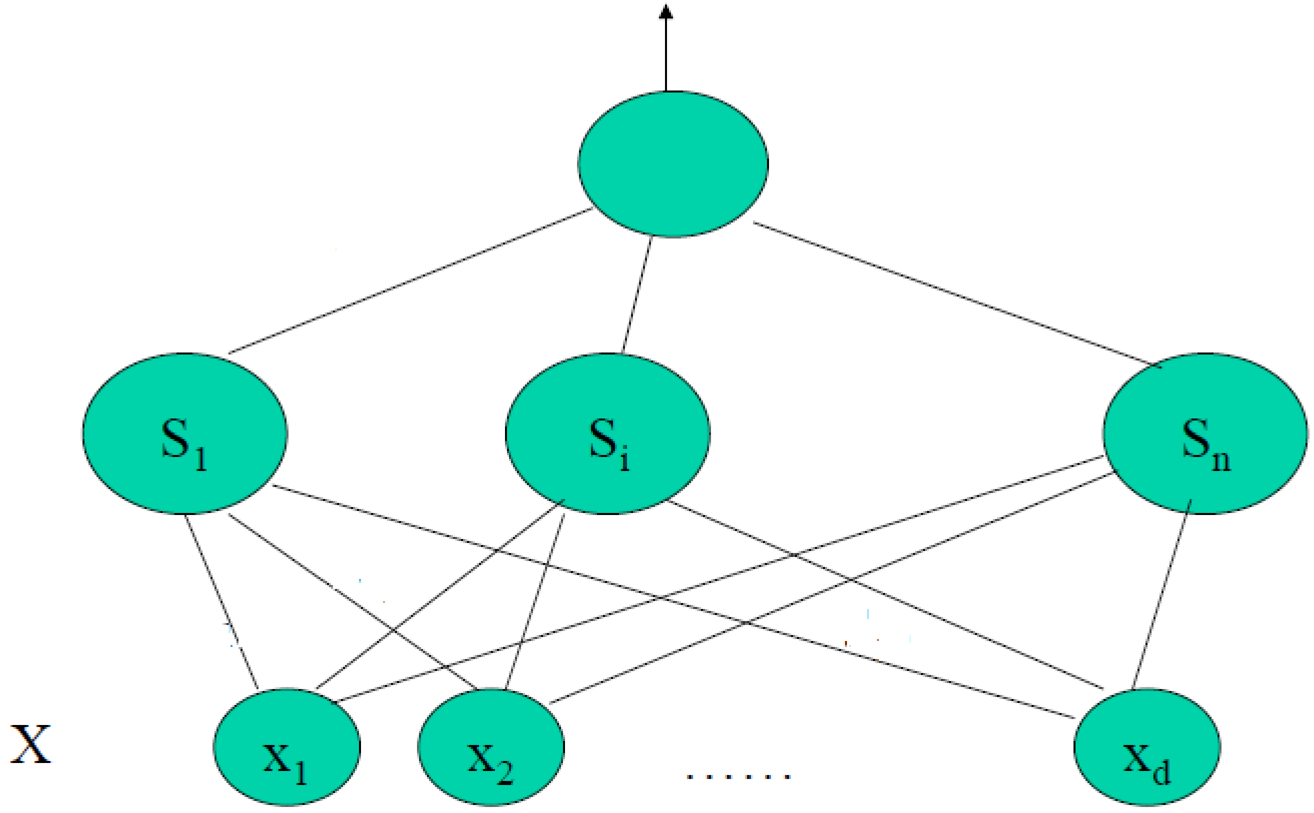
- $f(x) = wx + b = \sum_{i=1}^{Ns} (a_i y_i x_i x) + b$

How to interpret it?

Why Is SVM Effective on High Dimensional Data?

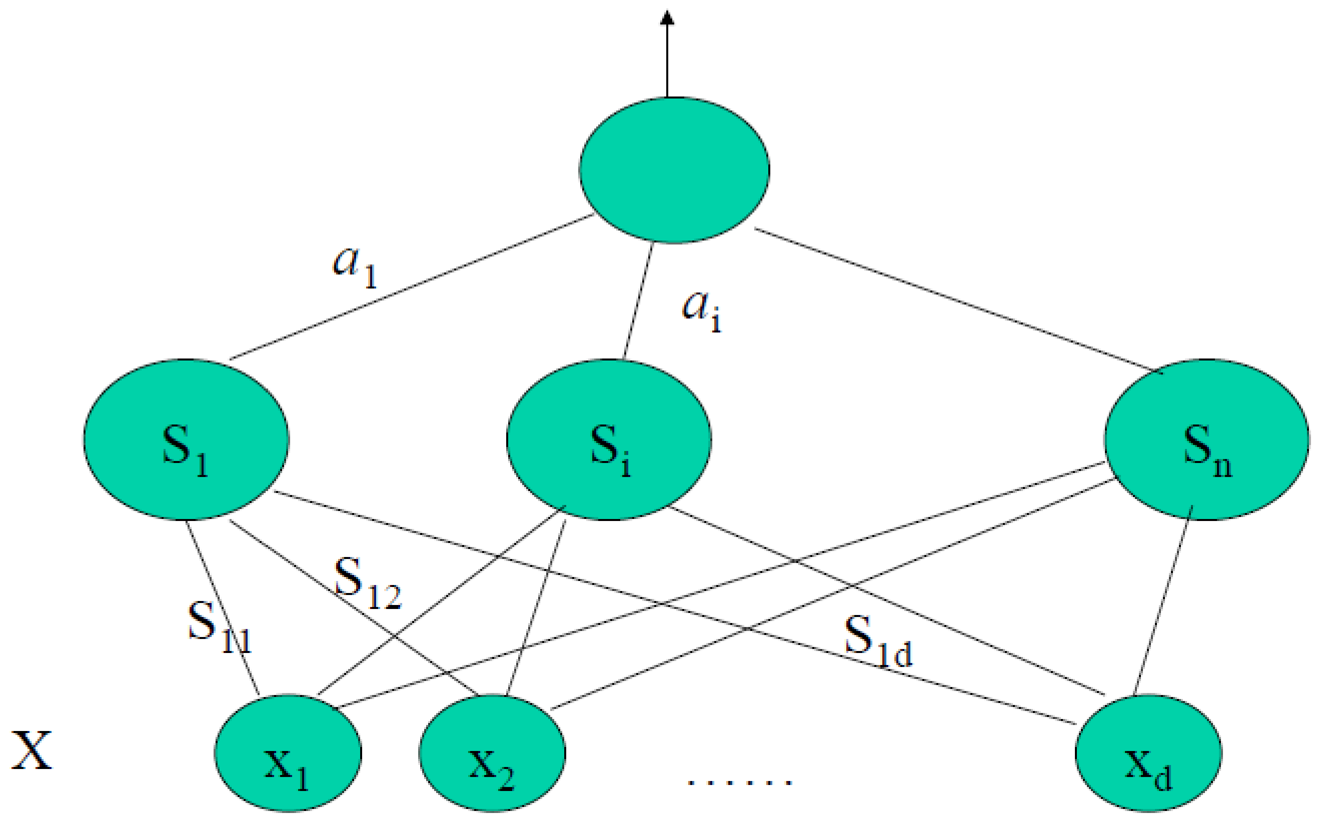
- The complexity of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The support vectors are the essential or critical training examples — they lie closest to the decision boundary (MMH)
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

Connection between SVM and NN?

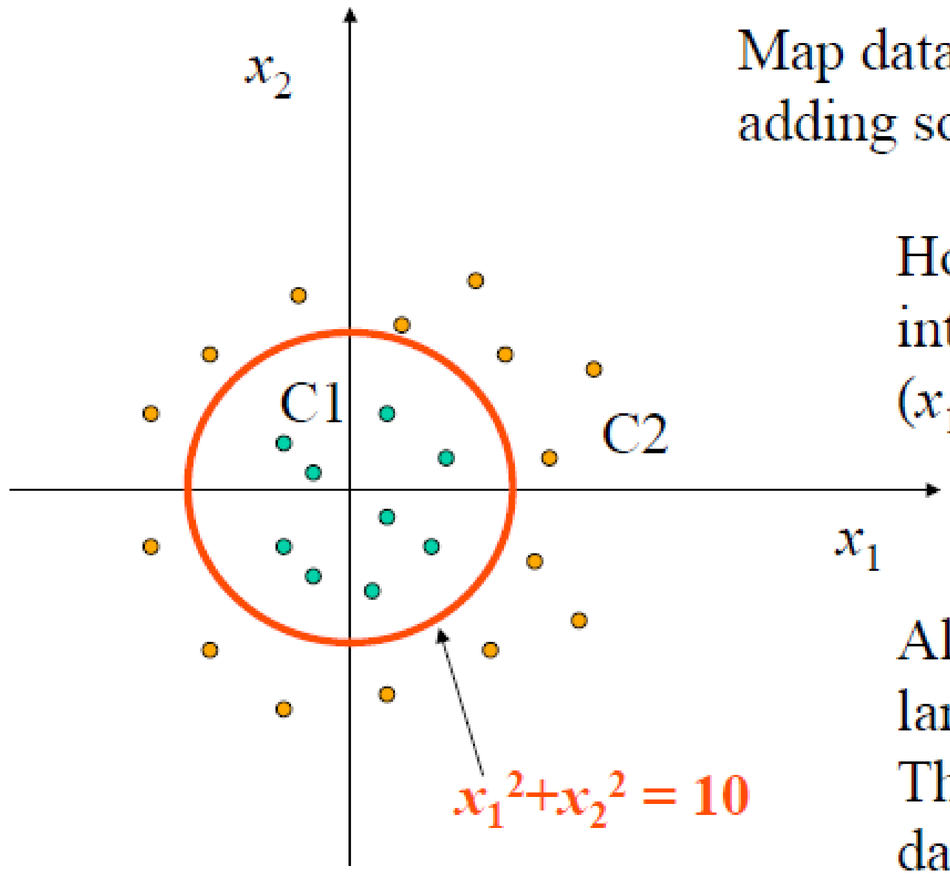


$$f(x) = wx + b = \sum_{i=1}^{Ns}(a_i y_i x_i x) + b$$

Connection Between SVM and NN



Support Vector Machine Approach



Map data point into high dimension, e.g. adding some non-linear features.

How about we augument feature into three dimension $(x_1, x_2, x_1^2+x_2^2)$.

All data points in class C2 have a larger value for the third feature Than data points in C1. Now data is linearly separable.

SVM Kernel Mapping Demo

- http://www.youtube.com/watch?v=3liCbRZPrZA

Dual Optimization

$$\text{maximize} \quad \mathcal{L}_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

$$\text{subject to} \quad \sum_i \alpha_i y_i = 0$$

Nonlinear Support Vector Machines

- In the L_D function, what really matters is dot products: $x_i.x_j$.
- Idea: map the data to some other (possibly infinite dimensional) Euclidean space *H*, using a mapping.

$$\Phi : R^d \mapsto H$$

Then the training algorithm would only depend on the data through dot products in *H*, i.e. $\Phi(x_i).\ \Phi(x_j)$.

Kernel Trick

- If there were a kernel function *K* such that $K(x_i,x_j) = \Phi(x_i).\Phi(x_j)$, we would only need to use *K* in the training algorithm and would never need to explicitly do the mapping Φ.
- One example Gaussian kernel: $K(x_i,x_j) = e\^(-|x_i-x_j|^2/2\sigma^2)$. In this example, H is infinite dimensional.
- So we simply replace $x_i.x_j$ with $K(x_i,x_j)$ in the training algorithm, the algorithm will happily produce a support vector machine which lives in an infinite dimensional space, and furthermore do so in roughly the same amount of time it would take to train on the un-mapped data.

How to Use the Machine?

- We can’t get w if we do not do explicit mapping.
- Once again we use kernel trick.

$$f(x) = (\sum_{i=1}^{Ns} a_i y_i \Phi(s_i))\Phi(x) + b = \sum_{i=1}^{Ns} a_i y_i K(s_i, x) + b$$

What’s the problem from a computational point of view?

Common Kernels

(1) $K(x,y) = (x.y + 1)^p$.
p is degree. p = 1,
linear kernel.

(2) Gaussian radial basis kernel $K(x,y) = e^{-|x-y|^2/2\sigma^2}$

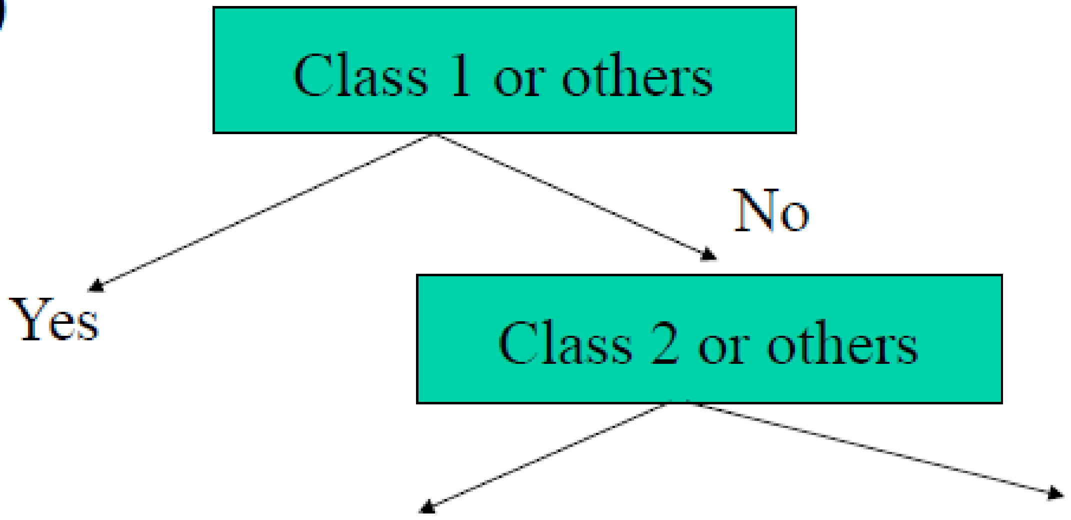
(3) Hyperbolic Tanh kernel $K(x,y) = \tanh(kx.y - \delta)$

Note: RBF kernel, the weights (a_i) and centers (S_i) are automatically Learned.
Tanh kernel is equivalent to two-layer neural network, where Number of hidden units is number of support vectors. a_i corresponds To the weights of the second layer.

Multi-Class SVM

- Most widely used method: one versus all
- Also direct multi-classification using SVM.

(K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs, JMLR, 2001)



Global Solutions and Uniqueness

- For SVM optimization, every local solution is global due to the property to the convex objective function.
- The solution is guaranteed to be unique.
- SVM training always finds a global solution is in contrast to the case of neural networks, where many local minima usually exist.

A Bound from Leave-One-Out

- E[P(error)] = N_s / number of training samples, where N_s is the number of support vectors
- What does this tells us?

SVM vs. Neural Network

- SVM
  - Hot in late 1990s and early 2000
  - Deterministic algorithm
  - Nice Generalization properties
  - Hard to learn – learned in batch mode using quadratic programming techniques
  - Using kernels can learn very complex functions

- Neural Network
  - Relatively old, but hot again
  - Nondeterministic algorithm
  - Generalizes well
  - Can easily be learned in incremental fashion
  - To learn complex functions—use multilayer perceptron (not that trivial)
  - Local minima

SVM Tools

- SVM-light: http://svmlight.joachims.org/
- LIBSVM: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Gist: http://bioinformatics.ubc.ca/gist/
- Weka

SVM Related Links

- SVM Website
  - http://www.kernel-machines.org/
- Representative implementations
  - LIBSVM: an efficient implementation of SVM, multi-class classifications, nu-SVM, one-class SVM, including also various interfaces with java, python, etc.
  - 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 Demo (Weka)

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.
- The book "An Introduction to Support Vector Machines" by N. Cristianini and J. Shawe-Taylor
- The neural network book by Haykins
  - Contains one nice chapter of SVM introduction

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification ←
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Associative Classification

- Associative classification
  - Association rules are generated and analyzed for use in 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

$$P_1 \wedge p_2 \ldots \wedge p_l \rightarrow \text{“}A_{class} = C\text{”} \text{ (conf, sup)}$$

- 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) $\rightarrow$ 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

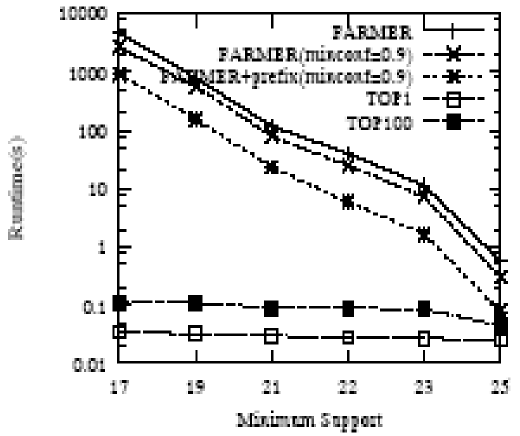
A Closer Look at CMAR

- CMAR (Classification based on Multiple Association Rules: Li, Han, Pei, ICDM' 01)
- Efficiency: Uses an enhanced FP-tree that maintains the distribution of class labels among tuples satisfying each frequent itemset
- Rule pruning whenever a rule is inserted into the tree
  - Given two rules, R_1 and R_2, if the antecedent of R_1 is more general than that of R_2 and $\mathrm{conf}(R_1) \geq \mathrm{conf}(R_2)$, then R_2 is pruned
  - Prunes rules for which the rule antecedent and class are not positively correlated, based on a χ^2 test of statistical significance
- Classification based on generated/pruned rules
  - If only one rule satisfies tuple X, assign the class label of the rule
  - If a rule set S satisfies X, CMAR
    - divides S into groups according to class labels
    - uses a weighted χ^2 measure to find the strongest group of rules, based on the statistical correlation of rules within a group
    - assigns X the class label of the strongest group

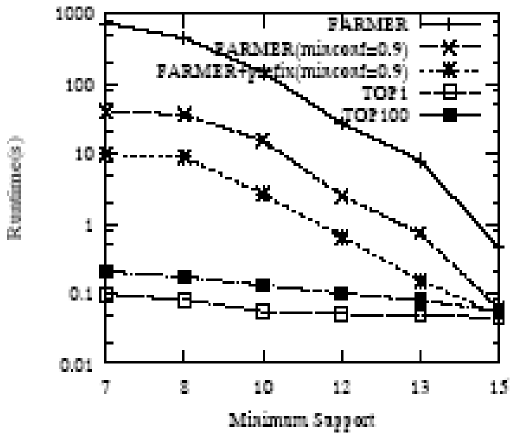
Associative Classification May Achieve High Accuracy and Efficiency (Cong et al. SIGMOD05)

Dataset	RCBT	CBA	IRG Classifier	C4.5 family			SVM
				single tree	bagging	boosting	
AML/ALL (ALL)	91.18%	91.18%	64.71%	91.18%	91.18%	91.18%	97.06%
Lung Cancer(LC)	97.99%	81.88%	89.93%	81.88%	96.64%	81.88%	96.64%
Ovarian Cancer(OC)	97.67%	93.02%	-	97.67%	97.67%	97.67%	97.67%
Prostate Cancer(PC)	97.06%	82.35%	88.24%	26.47%	26.47%	26.47%	79.41%
Average Accuracy	95.98%	87.11%	80.96%	74.3%	77.99%	74.3%	92.70%

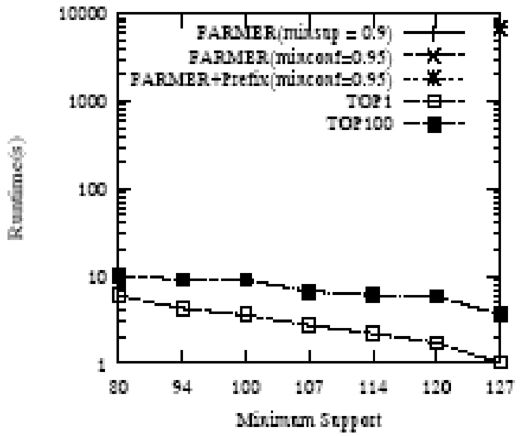
Table 2: Classification Results



(a) ALL-AML leukemia



(b) Lung Cancer



(c) Ovarian Cancer

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors) ⇦
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Lazy vs. Eager Learning

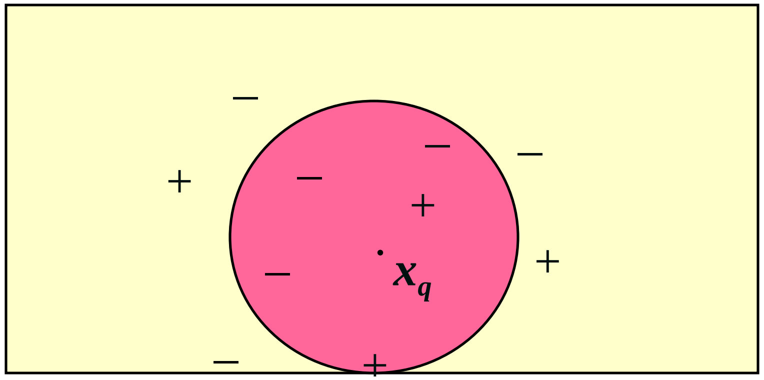
- Lazy vs. eager learning
  - Lazy learning (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple
  - Eager learning (the above discussed methods): Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify
- Lazy: less time in training but more time in 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

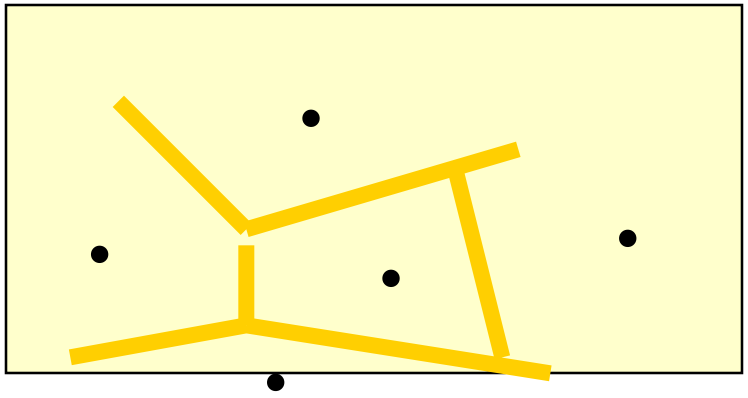
Lazy Learner: Instance-Based Methods

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

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, dist($\mathbf{X_1}$, $\mathbf{X_2}$)
- Target function could be discrete- or real- valued
- For discrete-valued, *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





Discussion on the *k*-NN Algorithm

- k-NN for real-valued prediction for a given unknown tuple
  - Returns 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 x_q
    - Give greater weight to closer neighbors

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

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

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Regression $\leftarrow$
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

What Is Regression?

- (Numerical) prediction / regression is similar to classification
  - construct a model
  - use model to predict continuous or ordered value for a given input
- Regression is different from classification
  - Classification refers to predict categorical class label
  - Regression models continuous-valued functions
- Major method for prediction of real value: regression
  - model the relationship between one or more *independent* or **predictor** variables and a *dependent* or **response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression (neural networks, support vector machines, KNN)
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees

Linear Regression

- Linear regression: involves a response variable y and a single predictor variable x

  $y = w_0 + w_1 x$

  where w_0 (y-intercept) and w_1 (slope) are regression coefficients
- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2} \qquad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
  - Training data is of the form $(\mathbf{X_1}, y_1), (\mathbf{X_2}, y_2), \ldots, (\mathbf{X_{|D|}}, y_{|D|})$
  - Solvable by extension of least square method or using R, SAS, S-Plus

Matrix Representation

n **data points,** ***d*** **dimension**

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1d} \\ 1 & x_{21} & \dots & x_{2d} \\ \dots & . & \dots & . \\ 1 & x_{n1} & \dots & x_{nd} \end{pmatrix} \times \begin{pmatrix} w_0 \\ w_1 \\ \dots \\ w_d \end{pmatrix} + \varepsilon$$

n*1 n*(d+1) (d+1)*1

Matrix Representation: $\mathbf{Y} = \mathbf{XW} + \boldsymbol{\varepsilon}$

Multivariate Linear Regression

- Goal: minimize square error = $(Y\text{-}XW)^T(Y\text{-}XW) = Y^TY - 2X^TWY + W^TX^TXW$
- Derivative: $-2X^TY + 2X^TXW = 0$
- $W = (X^TX)^{-1}X^TY$
- Thus, we can solve linear regression using matrix inversion, transpose, and multiplication.

Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

  $$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

  convertible to linear with new variables: $x_2 = x^2$, $x_3 = x^3$

  $$y = w_0 + w_1 x + w_2 x_2 + w_3 x_3$$

Other Regression-Based Models

- Generalized linear model:
  - Foundation on which linear regression can be applied to modeling categorical response variables
  - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
- Regression trees and model trees
  - Trees to predict continuous values rather than class labels

Logistic Regression

- Estimate posterior distribution: $P(C_1|x)$
- Dose – response estimation: in bioassay, the relation between dose level and death rate P(death | x).
- We can not use 0/1 hard classification.
- We can not use unconstrained linear regression because P(death | x) must be in [0,1]?

Logistic Regression

X1	X2	…	Die / Live
0.01	0.004		1
0.001	0.02		0
…			
…			
0.003	0.005		1

Binomial distribution:

For one dose level x_i, $y_i = P(die|x_i)$, likelihood = $y_i^{t_i}(1-y_i)^{(1-t_i)}$

Maximize likelihood is equivalent to minimize negative log-likelihood:

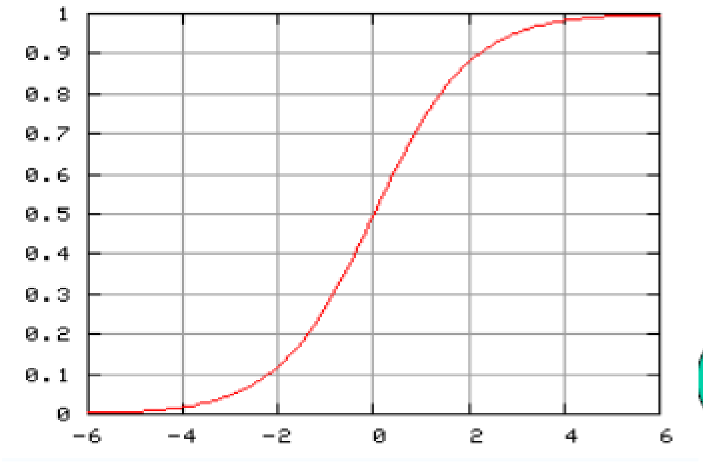
$$-t_i\log(y_i) - (1 - t_i)\log(1 - y_i)$$

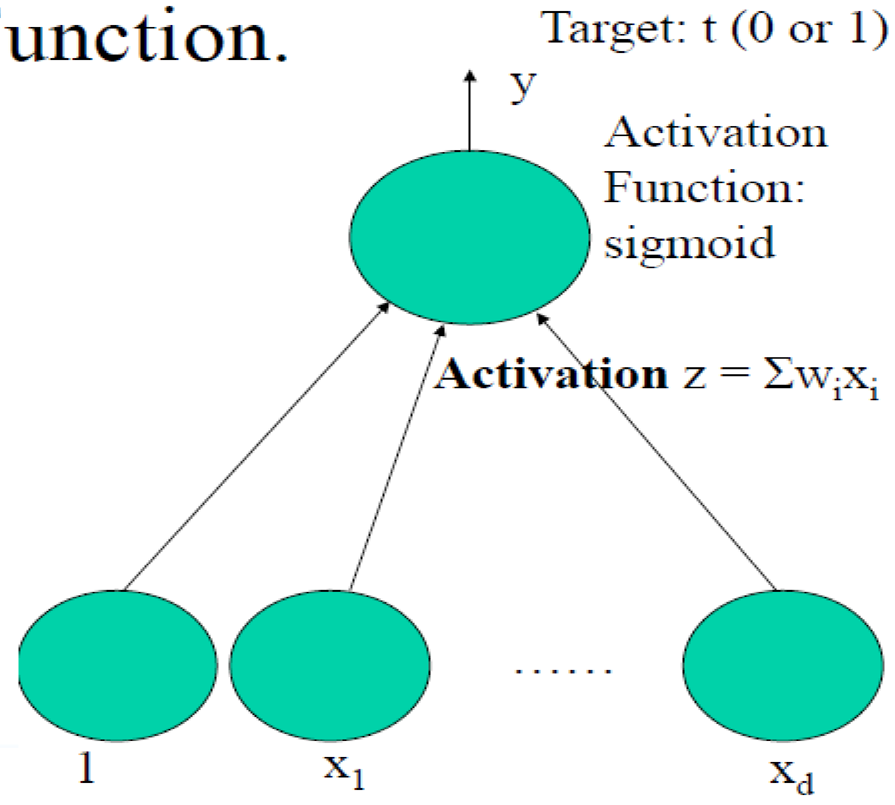
Logistic Regression As One Layer Neural Network – Gradient Descent

Logistic Regression and One Layer Neural Network With Sigmoid Function.

$$P(\text{death} \mid x) = \frac{1}{1+e^{-wx}}$$

(Sigmoid function)





How to Adjust Weights?

- Minimize error E= -tlog*y* - (1-*t*)log(1-*y*). For simplicity, we derive the formula for one data point. For multiple data points, just add the gradients together.

$$\frac{\partial E}{\partial y} = -\frac{t}{y} - \frac{1-t}{1-y}(-1) = -\frac{t}{y} - \frac{t-1}{1-y} = \frac{y-t}{y(1-y)}$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial z}\frac{\partial z}{\partial w_i} = \frac{y-t}{y(1-y)} y(1-y)x_i = (y-t)x_i$$

Update rule: $w_i^{(t+1)} = w_i^t + \eta(t-y)x_i$

Regression Trees and Model Trees

- Regression tree: proposed in CART system (Breiman et al. 1984)
  - CART: Classification And Regression Trees
  - Each leaf stores a *continuous-valued prediction*
  - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
  - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
  - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

How to Partition Data

- Minimize Square Errors

$$S = \sum_{c \in \text{leaves}(T)} \sum_{i \in C} (y_i - m_c)^2$$

where $m_c = \frac{1}{n_c} \sum_{i \in C} y_i$, the prediction for leaf c. Just as with clustering, we can re-write this as

$$S = \sum_{c \in \text{leaves}(T)} n_c V_c$$

where V_c is the within-leave variance of leaf c. So we will make our splits so as to minimize S.

www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf

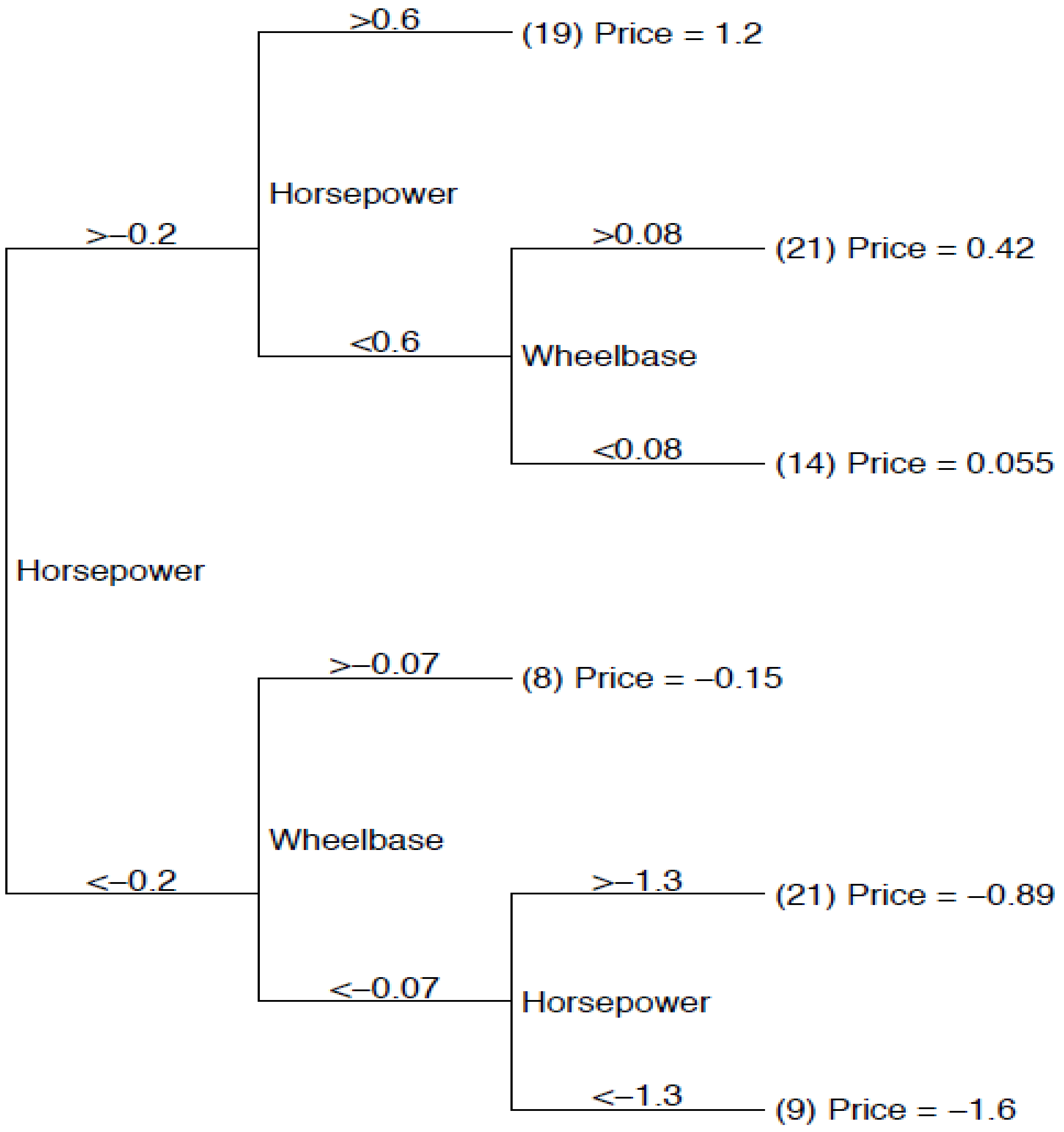
Algorithm

1. Start with a single node containing all points. Calculate m_c and S.

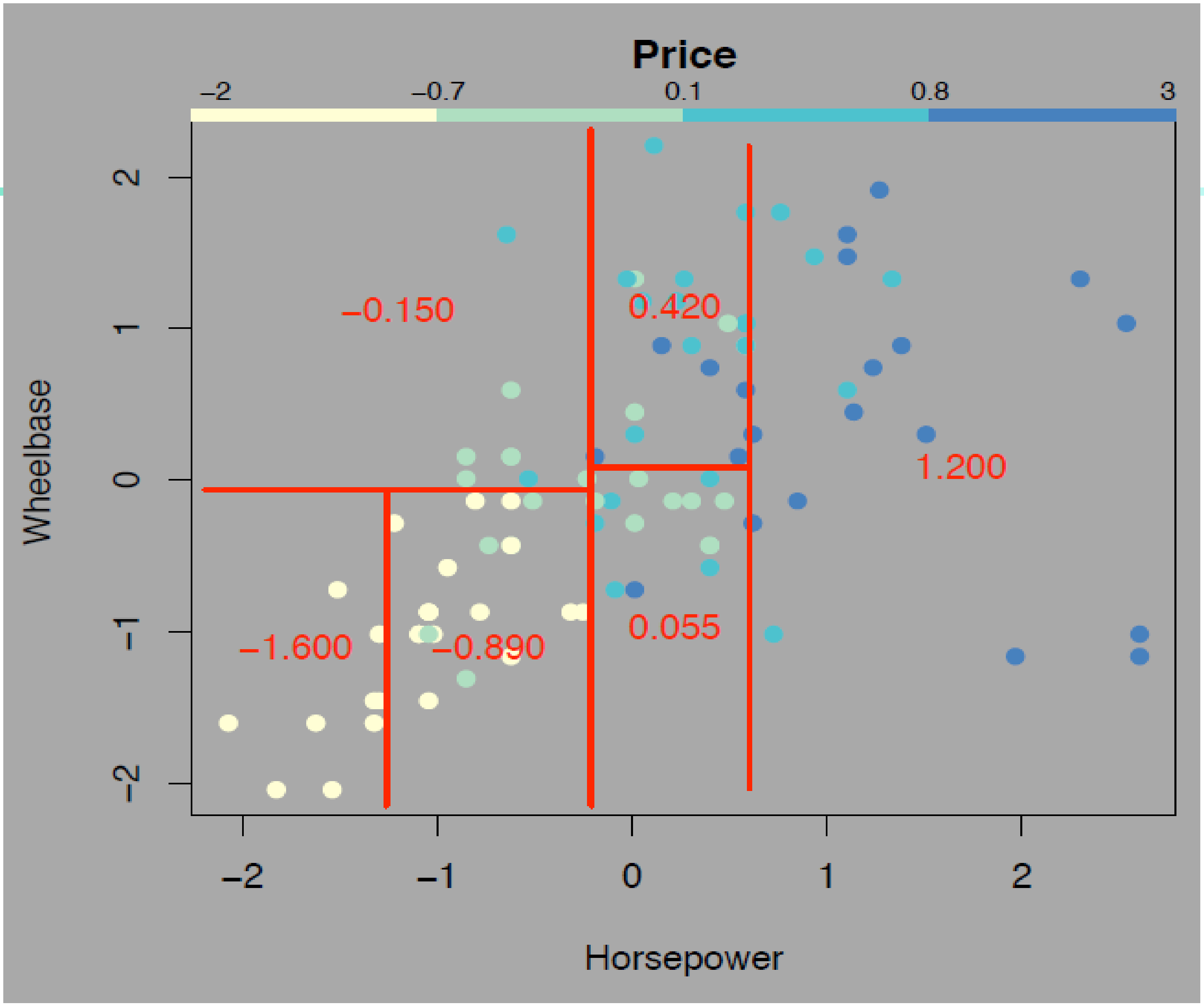
2. If all the points in the node have the same value for all the independent variables, stop. Otherwise, search over the splits of all variables for the one which will reduce S as much as possible. If the largest decrease in S would be less than some threshold , or one of the resulting nodes would contain less than q points, stop. Otherwise, take that split, creating two new nodes.

3. In each new node, go back to step 1.

www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf



Regression tree for prices of 1993 model cars. All values are standardized in [-1, 1]



www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

	$C_{1(Pre)}$	$C_{2(pre)}$
C_1	True positive	False negative
C_2	False positive	True negative

Classifier Accuracy Measures

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.52

- Accuracy of a classifier M, acc(M): percentage of test set tuples that are correctly classified by the model M
  - Error rate (misclassification rate) of M = 1 – acc(M)
  - Given *m* classes, $CM_{i,j}$, an entry in a **confusion matrix**, indicates # of tuples in class *i* that are labeled by the classifier as class *j*
- Alternative accuracy measures (e.g., for cancer diagnosis)

  sensitivity = t-pos/pos /* true positive recognition rate */

  specificity = t-neg/neg /* true negative recognition rate */

  precision = t-pos/(t-pos + f-pos)

  accuracy = sensitivity * pos/(pos + neg) + specificity * neg/(pos + neg)
  - This model can also be used for cost-benefit analysis

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

Evaluating the Accuracy of a Classifier or Predictor (I)

- <u>Holdout method</u>
  - 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
- <u>Cross-validation</u> (*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 D_i as test set and others as training set
  - <u>Leave-one-out</u>: k folds where k = # of tuples, for small sized data
  - <u>Stratified cross-validation</u>: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Evaluating the Accuracy of a Classifier or Predictor (II)

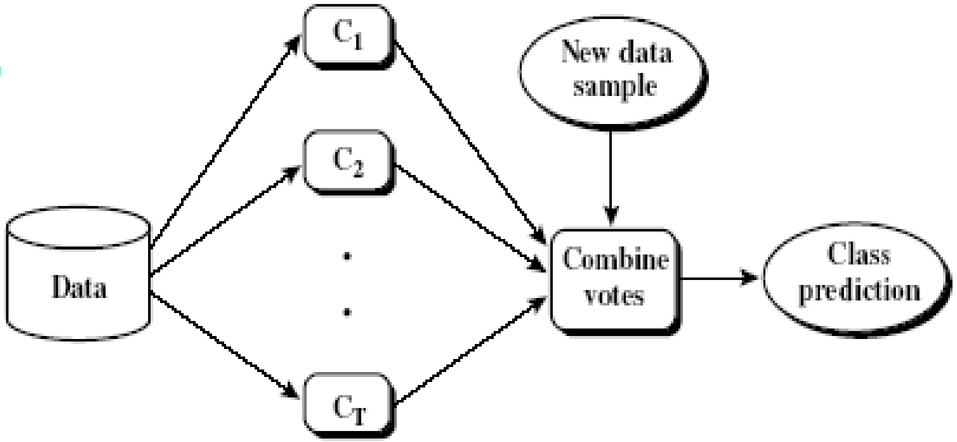
- Bootstrap
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several boostrap methods, and a common one is **.632 boostrap**
  - Suppose we are given a data set of d tuples. The data set is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data will end up in the bootstrap, and the remaining 36.8% will form the test set (since $(1 – 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedue k times, overall accuracy of the model:

$$acc(M) = \sum_{i=1}^{k} (0.632 \times acc(M_i)_{test\_set} + 0.368 \times acc(M_i)_{train\_set})$$

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers (e.g. Random Forest)
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., boostrap)
  - 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: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significant better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - Weights are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to pay more attention to the training tuples that were misclassified by M_i
  - The final M* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is 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
- 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:

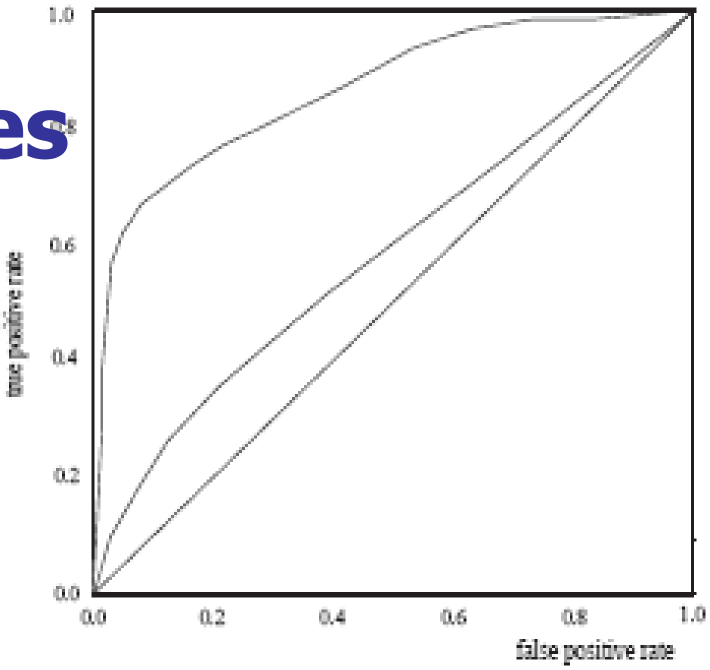
$$error(M_i) = \sum_{j}^{d} w_j \times err(\mathbf{X_j})$$

- The weight of classifier M_i's vote is $\log \frac{1-error(M_i)}{error(M_i)}$

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Model Selection: ROC Curves



- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Summary (I)

- Classification and regression 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.
- 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 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