
Università degli Studi di Milano
Master Degree in Computer Science

Information Management course

Teacher: Alberto Ceselli

Lecture 18: 12/12/2012

Data Mining: Concepts and Techniques


(3rd ed.)

— Chapter 8, 9 —

Jiawei Han, Micheline Kamber, and Jian Pei
University of Illinois at Urbana-Champaign &
Simon Fraser University

©2011 Han, Kamber & Pei. All rights reserved.

Classification methods

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Support Vector Machines 
- Model Evaluation and Selection
- *Rule-Based Classification*
- *Techniques to Improve Classification Accuracy: Ensemble Methods*

Classification: A Mathematical Mapping

- **Classification:** predicts categorical class labels

- E.g., Personal homepage classification

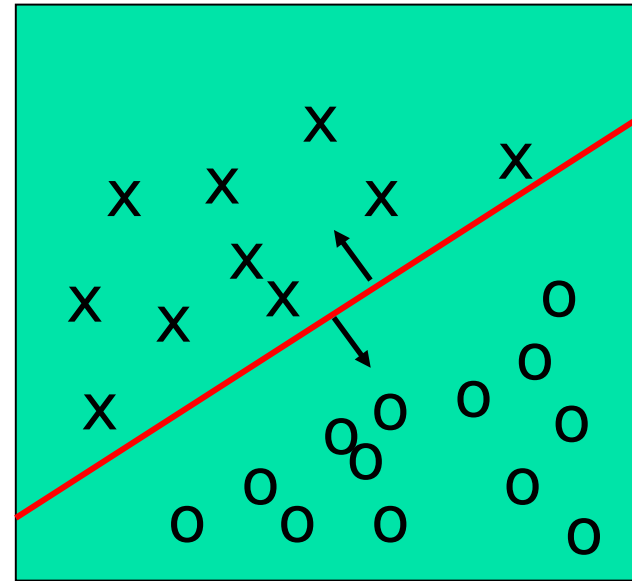
- $x_i = (x_1, x_2, x_3, \dots)$, $y_i = +1$ or -1
- x_1 : # of word “homepage”
- x_2 : # of word “welcome”

- Mathematically,

- $x \in X = \mathfrak{R}^n$, $y \in Y = \{+1, -1\}$,
- We want to derive a function $f: X \rightarrow Y$

- Linear Classification

- Binary Classification problem
- Data above the red line belongs to class ‘x’
- Data below red line belongs to class ‘o’
- Examples: SVM, Perceptron, Probabilistic Classifiers



Perceptron: finding a separating hyperplane

Hyperplane: $wx = b$

- Mathematical model:

find w

$$\text{s.t. } wx_k - b \geq 0 \quad (\text{forall } k: y_k = 1)$$

$$wx_k - b < 0 \quad (\text{forall } k: y_k = -1)$$

$$\|w\| = 1$$

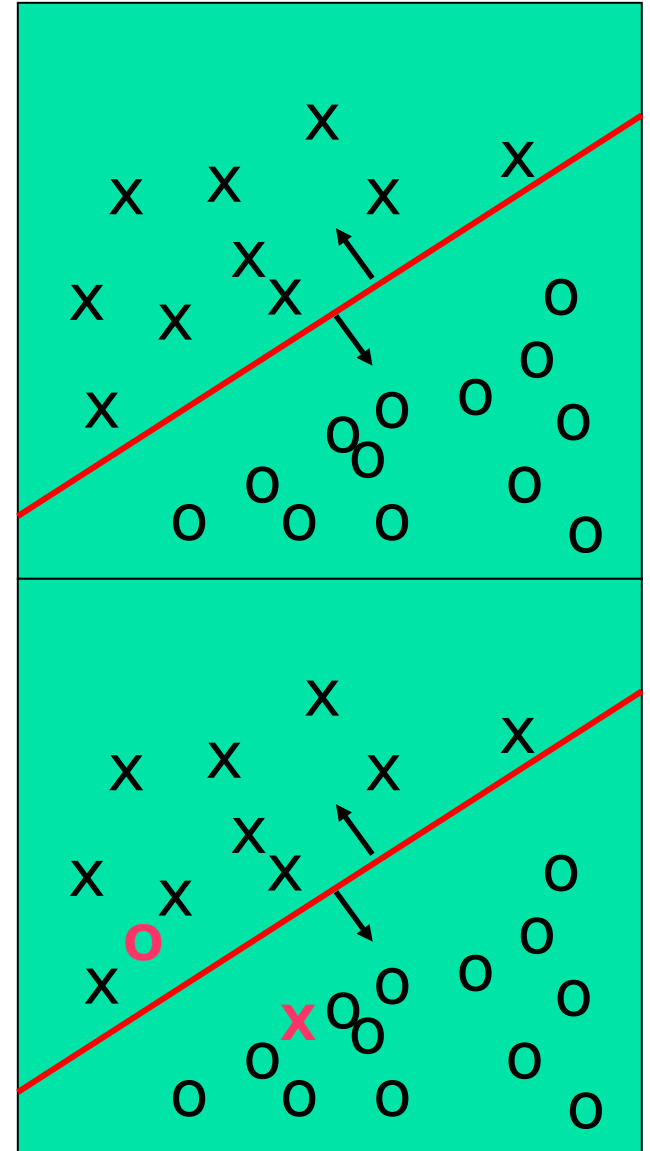
- Mathematical model:

$$\text{minimize } \sum_{i=1}^m d_k$$

$$\text{s.t. } wx_k - b + d_k \geq 0 \quad (\text{forall } k: y_k = 1)$$

$$wx_k - b - d_k < 0 \quad (\text{forall } k: y_k = -1)$$

$$\|w\| = 1$$



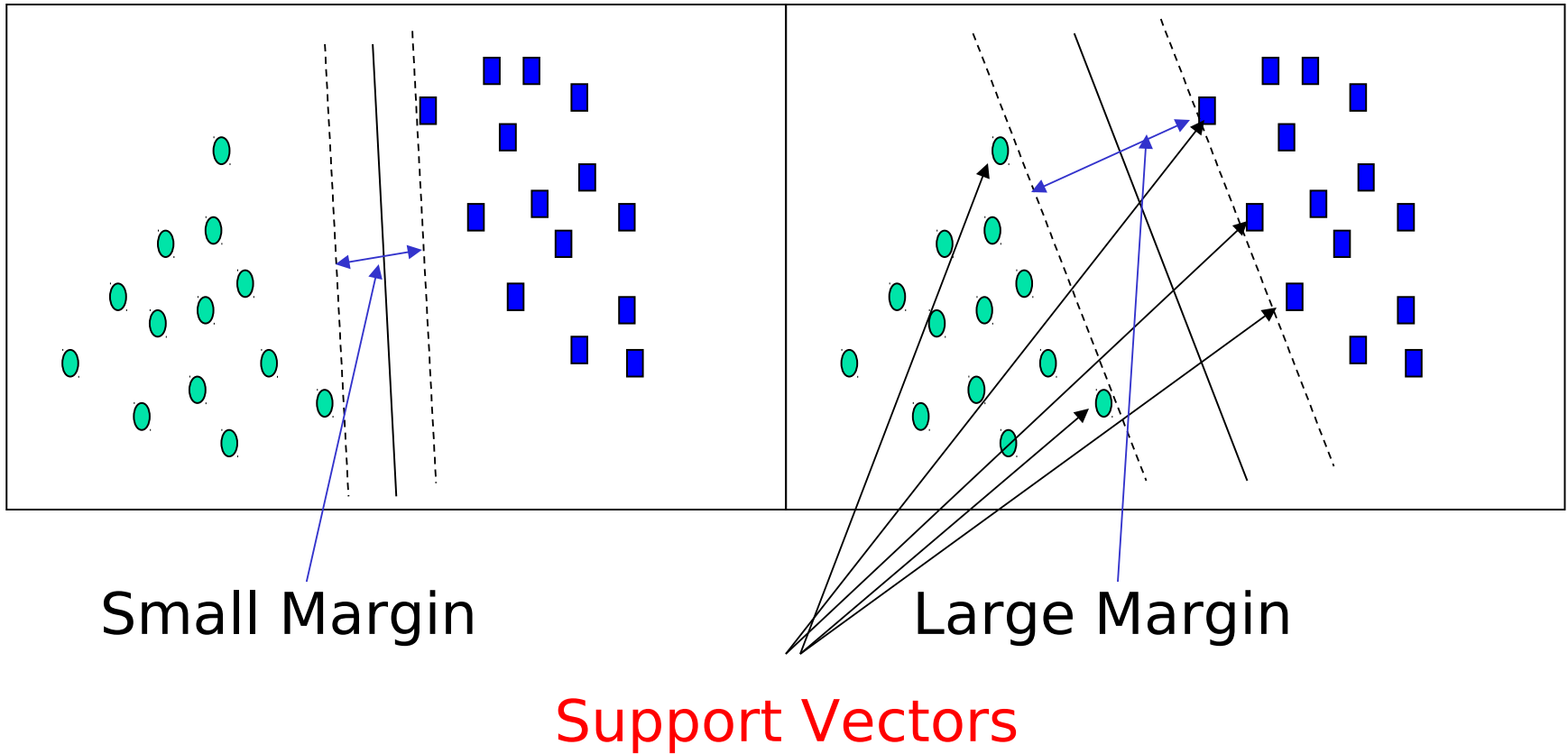
SVM—Support Vector Machines

- A relatively 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)

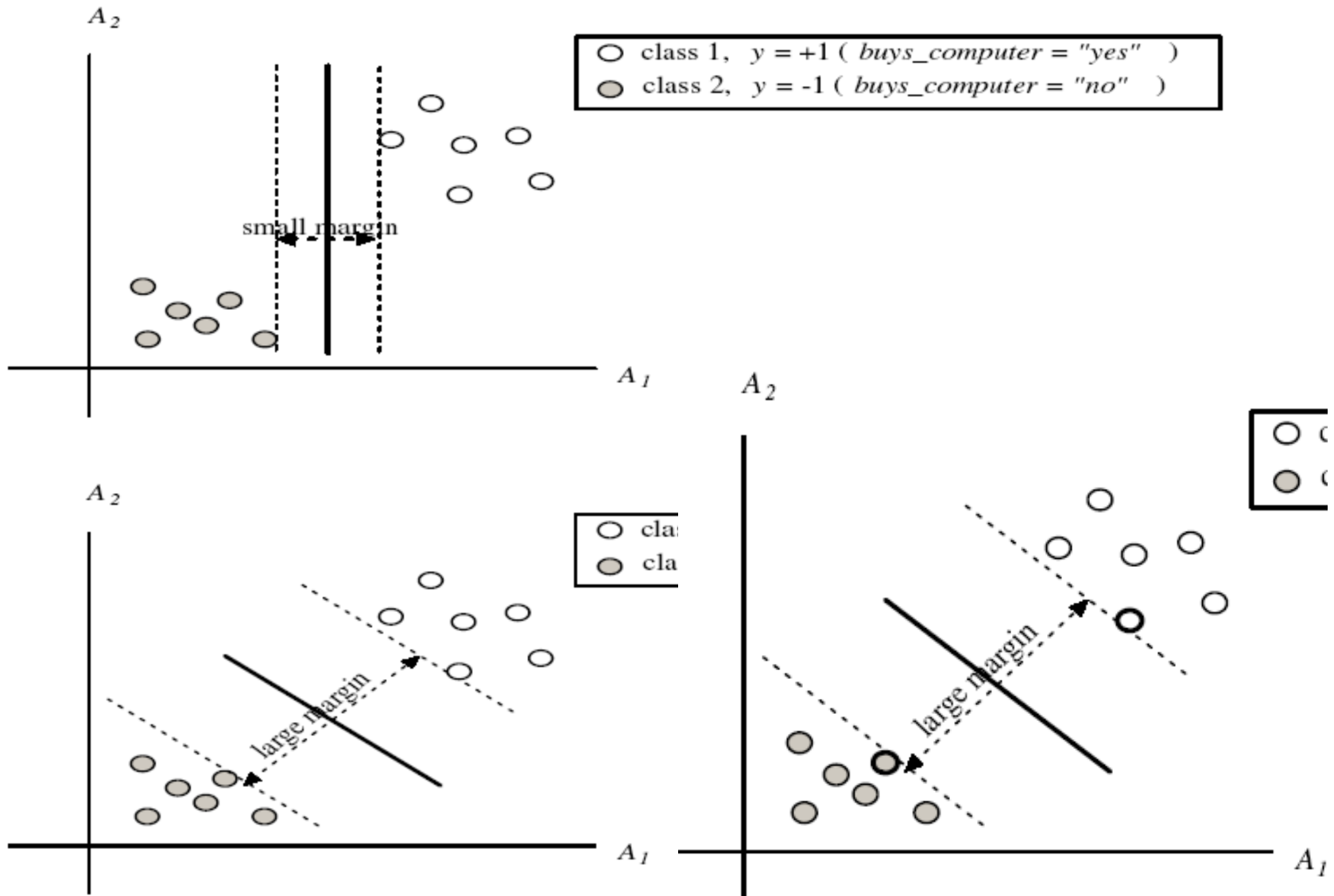
SVM—History and Applications

- 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 for: classification and numeric prediction
- Applications:
 - handwritten digit recognition, object recognition, speaker identification, benchmarking time-series prediction tests

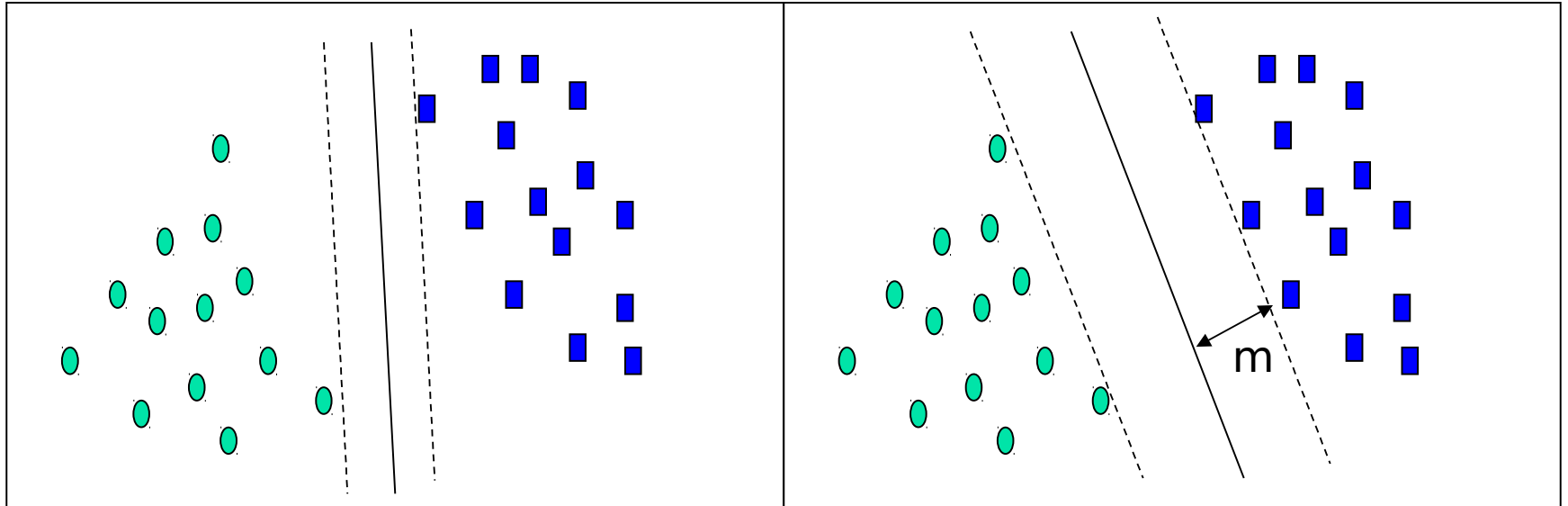
SVM—General Philosophy



SVM—Margins and Support Vectors



SVM—When Data Is Linearly Separable



Let data D be $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_{|D|}, y_{|D|})$, 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)*

SVM—Linearly Separable

A hyperplane: $\mathbf{w}\mathbf{x} = b$

where $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$ 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:

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

$$H_2: w_0 + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for } y_i = -1$$

- 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 \rightarrow *Quadratic Programming (QP)* \rightarrow Lagrangian multipliers

SVM - A QP model

A hyperplane: $\mathbf{w}\mathbf{x} = b$

where $\mathbf{w} = \{w_1, w_2, \dots, w_n\}$ is a weight vector and b a scalar (bias)

- Separating margin: $D = \frac{2}{\|\mathbf{w}\|}$ $\|\mathbf{w}\| = \sqrt{\sum_{i=1}^n (w_i)^2}$

- Find an optimal hyperplane (linearly separable):

$$\min \frac{1}{2} \|\mathbf{w}\|^2$$

$$s.t. y_k (w x_k - b) \geq 1 \quad \forall k = 1 \dots m$$

- Find an optimal hyperplane (general):

$$\min \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{k=1}^m d_k$$

$$s.t. y_k (w x_k - b) + d_k \geq 1 \quad \forall k = 1 \dots m$$

$$d_k \geq 0 \quad \forall k = 1 \dots m$$

SVM - A QP model

- Find an optimal hyperplane (general):

$$\min \frac{1}{2} \|w\|^2 + C \sum_{k=1}^m d_k$$

$$s.t. y_k (w x_k - b) + d_k \geq 1 \quad \forall k = 1 \dots m$$

$$d_k \geq 0 \quad \forall k = 1 \dots m$$

- Langrangean (dual) function:

$$L = \min \frac{1}{2} \|w\|^2 + C \sum_{k=1}^m d_k - \sum_{k=1}^m \alpha_k (y_k (w x_k - b) + d_k - 1) - \sum_{k=1}^m \mu_k d_k$$

- Derivatives:

$$\frac{\partial L}{\partial w} = w - \sum_{k=1}^m \alpha_k y_k x_k$$

$$\frac{\partial L}{\partial b} = \sum_{k=1}^m \alpha_k y_k$$

$$\frac{\partial L}{\partial d_k} = C - \alpha_k - \mu_k$$

SVM - A QP model

- Lagrangean (dual) function:

$$L = \min \frac{1}{2} \|w\|^2 + C \sum_{k=1}^m d_k - \sum_{k=1}^m \alpha_k (y_k (w x_k - b) + d_k - 1) - \sum_{k=1}^m \mu_k d_k$$

- Optimality conditions:

$$\frac{\partial L}{\partial w} = w - \sum_{k=1}^m \alpha_k y_k x_k = 0$$

$$\frac{\partial L}{\partial b} = \sum_{k=1}^m \alpha_k y_k = 0$$

$$\frac{\partial L}{\partial d_k} = C - \alpha_k - \mu_k = 0$$

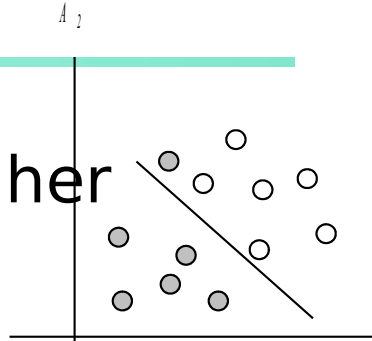
- Dual problem: ... (blackboard discussion)
- Interpretation of KKT conditions: ... (blackboard discussion)

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

SVM—Linearly Inseparable

- Transform the original input data into a higher dimensional space



Example 6.8 Nonlinear transformation of original input data into a higher dimensional space. Consider the following example. A 3D input vector $\mathbf{X} = (x_1, x_2, x_3)$ is mapped into a 6D space Z using the mappings $\phi_1(\mathbf{X}) = x_1, \phi_2(\mathbf{X}) = x_2, \phi_3(\mathbf{X}) = x_3, \phi_4(\mathbf{X}) = (x_1)^2, \phi_5(\mathbf{X}) = x_1x_2$, and $\phi_6(\mathbf{X}) = x_1x_3$. A decision hyperplane in the new space is $d(\mathbf{Z}) = \mathbf{WZ} + b$, where \mathbf{W} and \mathbf{Z} are vectors. This is linear. We solve for \mathbf{W} and b and then substitute back so that we see that the linear decision hyperplane in the new (\mathbf{Z}) space corresponds to a nonlinear second order polynomial in the original 3-D input space,

$$\begin{aligned} d(\mathbf{Z}) &= w_1x_1 + w_2x_2 + w_3x_3 + w_4(x_1)^2 + w_5x_1x_2 + w_6x_1x_3 + b \\ &= w_1z_1 + w_2z_2 + w_3z_3 + w_4z_4 + w_5z_5 + w_6z_6 + b \end{aligned} \quad \blacksquare$$

- Search for a linear separating hyperplane in the new space

SVM: Different Kernel functions

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

Polynomial kernel of degree h : $K(\mathbf{X}_i, \mathbf{X}_j) = (\mathbf{X}_i \cdot \mathbf{X}_j + 1)^h$

Gaussian radial basis function kernel : $K(\mathbf{X}_i, \mathbf{X}_j) = e^{-\|\mathbf{X}_i - \mathbf{X}_j\|^2 / 2\sigma^2}$

Sigmoid kernel : $K(\mathbf{X}_i, \mathbf{X}_j) = \tanh(\kappa \mathbf{X}_i \cdot \mathbf{X}_j - \delta)$

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

“geometric” 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
 - 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

SVM vs. Neural Network

■ SVM

- 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

- Nondeterministic algorithm
- Generalizes well but doesn't have strong mathematical foundation
- Can easily be learned in incremental fashion
- To learn complex functions—use multilayer perceptron (nontrivial)

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 in C
 - **SVM-torch**: another recent implementation also written in C

Classification methods

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Support Vector Machines
- Model Evaluation and Selection 
- *Rule-Based Classification*
- *Techniques to Improve Classification Accuracy: Ensemble Methods*

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
 - Holdout method, random subsampling
 - Cross-validation
 - Bootstrap
- Comparing classifiers:
 - Confidence intervals
 - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

| Actual class \ Predicted class | C_1 | $\neg C_1$ |
|--------------------------------|-----------------------------|-----------------------------|
| C_1 | True Positives (TP) | False Negatives (FN) |
| $\neg C_1$ | False Positives (FP) | True Negatives (TN) |

Example of Confusion Matrix:

| Actual class \ Predicted class | buy_computer = yes | buy_computer = no | Total |
|--------------------------------|--------------------|-------------------|-------|
| buy_computer = yes | 6954 | 46 | 7000 |
| buy_computer = no | 412 | 2588 | 3000 |
| Total | 7366 | 2634 | 10000 |

- Given m classes, an entry, \mathbf{CM}_{ij} in a **confusion matrix** indicates # of tuples in class i that were labeled by the classifier as class j
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics: Accuracy, Error Rate, Sensitivity and Specificity

| | | | |
|----------|-----------|-----------|------------|
| A\P | C | $\neg C$ | |
| C | TP | FN | P |
| $\neg C$ | FP | TN | N |
| | P' | N' | All |

- **Classifier Accuracy**, or recognition rate: percentage of test set tuples that are correctly classified

$$\text{Accuracy} = (\text{TP} + \text{TN})/\text{All}$$

- **Error rate**: $1 - \text{accuracy}$, or
Error rate = $(\text{FP} + \text{FN})/\text{All}$

- **Class Imbalance Problem:**
 - One class may be *rare*, e.g. fraud, or HIV-positive
 - Significant *majority of the negative class* and minority of the positive class
 - **Sensitivity**: True Positive recognition rate
 - **Sensitivity** = TP/P
 - **Specificity**: True Negative recognition rate
 - **Specificity** = TN/N

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision:** coherence – what % of tuples that the classifier labeled as positive are actually positive

$$\textit{precision} = \frac{TP}{TP + FP}$$

- **Recall:** completeness – what % of positive tuples did the classifier label as positive?

$$\textit{recall} = \frac{TP}{TP + FN}$$

- Perfect score is 1.0

- Inverse relationship between precision & recall

- **F measure (F_1 or F-score):** harmonic mean of precision and recall,

$$F = \frac{2 \times \textit{precision} \times \textit{recall}}{\textit{precision} + \textit{recall}}$$

- F_β : weighted measure of precision and recall

- assigns β times as much weight to recall as to precision

$$F_\beta = \frac{(1 + \beta^2) \times \textit{precision} \times \textit{recall}}{\beta^2 \times \textit{precision} + \textit{recall}}$$

Classifier Evaluation Metrics: Example

| Actual Class\Predicted class | cancer = yes | cancer = no | Total | Recognition(%) |
|------------------------------|--------------|-------------|-------|-------------------------------|
| cancer = yes | 90 | 210 | 300 | 30.00 <i>(sensitivity)</i> |
| cancer = no | 140 | 9560 | 9700 | 98.56 <i>(specificity)</i> |
| Total | 230 | 9770 | 10000 | 96.40 <i>(accuracy)</i> |

- $Precision = 90/230 = 39.13\%$
 $Recall = 90/300 = 30.00\%$

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

■ 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 D_i 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

Evaluating Classifier Accuracy: Bootstrap

- **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

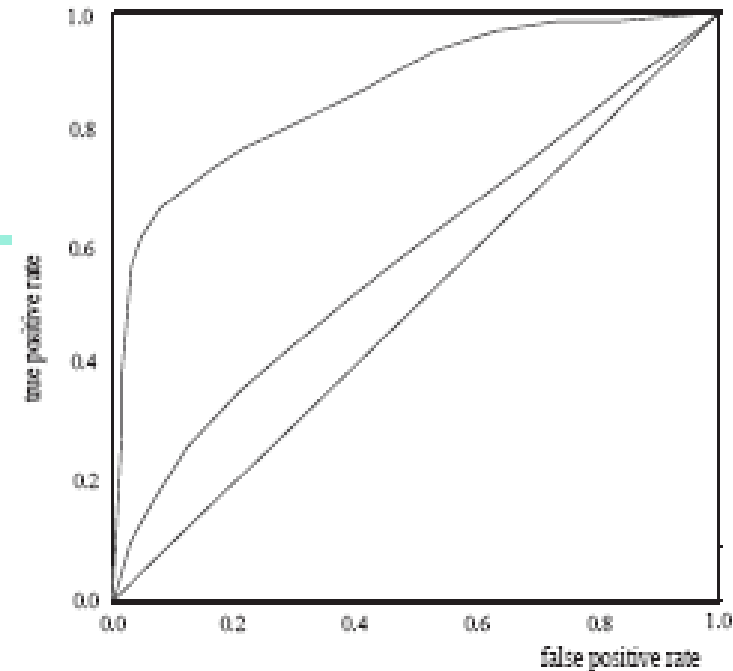
- Several bootstrap methods, and a common one is **.632 bootstrap**

- A data set with d tuples 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 end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 - 1/d)^d \approx e^{-1} = 0.368$)
- Repeat the sampling procedure k times, overall accuracy of the model:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set})$$

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
- Rank the test subsets in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- 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

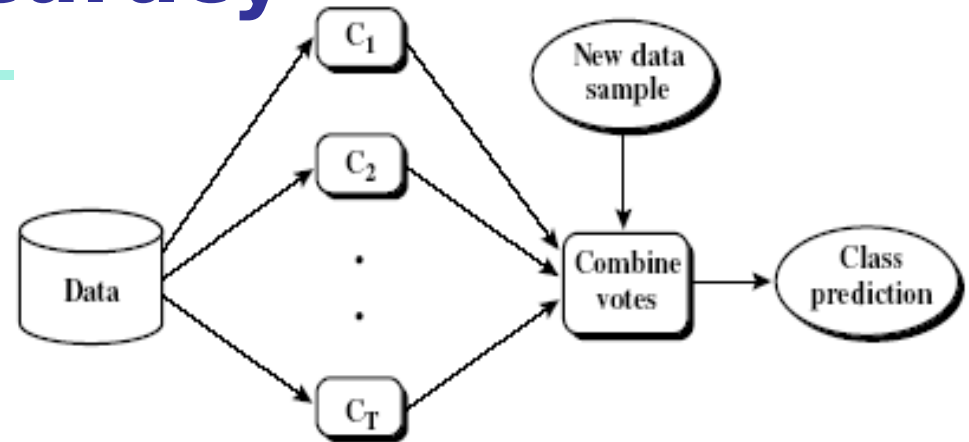
Issues Affecting Model Selection

- **Accuracy**
 - classifier accuracy: predicting class label
- **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, such as decision tree size or compactness of classification rules

Classification methods

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Support Vector Machines
- Model Evaluation and Selection
- *Rule-Based Classification*
- *Techniques to Improve Classification* 
- *Accuracy: Ensemble Methods*

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, \dots, M_k , with the aim of creating an improved model M^*
- Popular ensemble methods
 - Bagging: averaging the prediction over a collection of classifiers
 - Boosting: weighted vote with a collection of classifiers
 - Ensemble: combining a set of heterogeneous classifiers

Bagging: Bootstrap 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., bootstrap)
 - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample \mathbf{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 \mathbf{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 significantly 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
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have 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, $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- 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 misclassified, 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:

$$error(M_i) = \sum_j^d w_j \times err(X_j)$$

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


Random Forest (Breiman 2001)

- Random Forest:
 - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
 - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
 - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
 - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
 - **Oversampling**: re-sampling of data from positive class
 - **Under-sampling**: randomly eliminate tuples from negative class
 - **Threshold-moving**: moves the decision threshold, t , so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
 - Ensemble techniques: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary 

Summary (I)

- **Classification** is a form of data analysis that extracts **models** describing important data classes.
- Effective and scalable methods have been developed for **decision tree induction**, **Naive Bayesian classification**, **rule-based classification**, and many other classification methods.
- **Evaluation metrics** include: accuracy, sensitivity, specificity, precision, recall, F measure, and F_{β} measure.
- **Stratified k-fold cross-validation** is recommended for accuracy estimation. **Bagging** and **boosting** can be used to increase overall accuracy by learning and

Summary (II)

- **Significance tests** and **ROC curves** are useful for model selection.
- There have been numerous **comparisons of the different classification** methods; 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, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

Reference: Books on Classification

- E. Alpaydin. *Introduction to Machine Learning*, 2nd ed., MIT Press, 2011
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, 1984.
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*, 2ed. John Wiley, 2001
- T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001
- H. Liu and H. Motoda (eds.). *Feature Extraction, Construction, and Selection: A Data Mining Perspective*. Kluwer Academic, 1998
- T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997
- S. Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman and Hall/CRC, 2009.
- J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993
- J. W. Shavlik and T. G. Dietterich. *Readings in Machine Learning*. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and A. Karim. *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*. McGraw-Hill, 1991

Reference: Decision-Trees

- M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: An interactive approach to decision tree construction. *KDD'99*
- C. Apte and S. Weiss. Data mining with decision trees and decision rules. *Future Generation Computer Systems*, 13, 1997
- C. E. Brodley and P. E. Utgoff. Multivariate decision trees. *Machine Learning*, 19:45–77, 1995.
- P. K. Chan and S. J. Stolfo. Learning arbiter and combiner trees from partitioned data for scaling machine learning. *KDD'95*
- U. M. Fayyad. Branching on attribute values in decision tree generation. *AAAI'94*
- M. Mehta, R. Agrawal, and J. Rissanen. SLIQ : A fast scalable classifier for data mining. *EDBT'96*.
- 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*.
- 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. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, Mar. 1989
- S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2:345–389, 1998.
- R. Rastogi and K. Shim. Public: A decision tree classifier that integrates building and

Reference: Neural Networks

- C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford University Press, 1995
- Y. Chauvin and D. Rumelhart. *Backpropagation: Theory, Architectures, and Applications*. Lawrence Erlbaum, 1995
- J. W. Shavlik, R. J. Mooney, and G. G. Towell. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111-144, 1991
- S. Haykin. *Neural Networks and Learning Machines*. Prentice Hall, Saddle River, NJ, 2008
- J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison Wesley, 1991.
- R. Hecht-Nielsen. *Neurocomputing*. Addison Wesley, 1990
- B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996

Reference: Support Vector Machines

- C. J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge Univ. Press, 2000.
- H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. N. Vapnik. Support vector regression machines, NIPS, 1997
- J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schoelkopf, C. J. C. Burges, and A. Smola, editors, *Advances in Kernel Methods| Support Vector Learning*, pages 185–208. MIT Press, 1998
- B. Schölkopf, P. L. Bartlett, A. Smola, and R. Williamson. Shrinking the tube: A new support vector regression algorithm. NIPS, 1999.
- H. Yu, J. Yang, and J. Han. Classifying large data sets using SVM with hierarchical clusters. KDD'03.

Reference: Pattern-Based Classification

- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, Direct Discriminative Pattern Mining for Effective Classification, ICDE'08
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. Mining top-k covering rule groups for gene expression data. SIGMOD'05
- G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. KDD'99
- H. S. Kim, S. Kim, T. Weninger, J. Han, and T. Abdelzaher. NDPMine: Efficiently mining discriminative numerical features for pattern-based classification. *ECMLPKDD'10*
- W. Li, J. Han, and J. Pei, CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules, ICDM'01
- B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. *KDD'98*
- J. Wang and G. Karypis. HARMONY: Efficiently mining the best rules for classification. *SDM'05*

References: Rule Induction

- P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261-283, 1989.
- W. Cohen. Fast effective rule induction. *ICML'95*
- S. L. Crawford. Extensions to the CART algorithm. *Int. J. Man-Machine Studies*, 31:197-217, Aug. 1989
- J. R. Quinlan and R. M. Cameron-Jones. FOIL: A midterm report. *ECML'93*
- P. Smyth and R. M. Goodman. An information theoretic approach to rule induction. *IEEE Trans. Knowledge and Data Engineering*, 4:301-316, 1992.
- X. Yin and J. Han. CPAR: Classification based on predictive association rules. *SDM'03*

References: K-NN & Case-Based Reasoning

- A. Aamodt and E. Plazas. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Comm.*, 7:39–52, 1994.
- T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Information Theory*, 13:21–27, 1967
- B. V. Dasarathy. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991
- J. L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann, 1993
- A. Veloso, W. Meira, and M. Zaki. Lazy associative classification. *ICDM'06*

ences: Bayesian Method & Statistical M

- A. J. Dobson. An Introduction to Generalized Linear Models. Chapman & Hall, 1990.
- D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 1995.
- G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992
- A. Darwiche. Bayesian networks. *Comm. ACM*, 53:80–90, 2010
- A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society, Series B*, 39:1–38, 1977
- D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995
- F. V. Jensen. *An Introduction to Bayesian Networks*. Springer Verlag, 1996.
- D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 2009
- J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan

Tasks: Semi-Supervised & Multi-Class Learning

- O. Chapelle, B. Schoelkopf, and A. Zien. *Semi-supervised Learning*. MIT Press, 2006
- T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artificial Intelligence Research*, 2:263–286, 1995
- W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. ICML'07
- S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. on Knowledge and Data Engineering*, 22:1345–1359, 2010
- B. Settles. Active learning literature survey. In *Computer Sciences Technical Report 1648*, Univ. Wisconsin-Madison, 2010

Refs: Genetic Algorithms & Rough/Fuzzy Sets

- D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989
- S. A. Harp, T. Samad, and A. Guha. Designing application-specific neural networks using the genetic algorithm. NIPS, 1990
- Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1992.
- M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996
- Z. Pawlak. *Rough Sets, Theoretical Aspects of Reasoning about Data*. Kluwer Academic, 1991
- S. Pal and A. Skowron, editors, *Fuzzy Sets, Rough Sets and Decision Making Processes*. New York, 1998
- R. R. Yager and L. A. Zadeh. *Fuzzy Sets, Neural Networks and Soft Computing*. Van Nostrand Reinhold, 1994

References: Model Evaluation, Ensemble Methods

- L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.
- C. Elkan. The foundations of cost-sensitive learning. *IJCAI'01*
- B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, 1993.
- J. Friedman and E. P. Bogdan. Predictive learning via rule ensembles. *Ann. Applied Statistics*, 2:916–954, 2008.
- 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.
- J. R. Quinlan. Bagging, boosting, and c4.5. *AAAI'96*.
- G. Seni and J. F. Elder. *Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions*. Morgan and Claypool, 2010.

Surplus Slides

Issues: Evaluating Classification Methods

- Accuracy
 - classifier accuracy: predicting class label
 - predictor 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 in disk-resident databases
- Interpretability
 - understanding and insight provided by the model
- Other measures, e.g., goodness of rules, such as

Gain Ratio for Attribute Selection (C4.5) (MK:contains errors)

- 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)

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

- $\text{GainRatio}(A) = \frac{\text{Gain}(A)}{\text{SplitInfo}(A)} = \frac{0.029}{0.926} = 0.031$

- Ex.

- $\text{gain_ratio}(\text{income}) = 0.029/0.926 = 0.031$

- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini index (CART, IBM IntelligentMiner)

- 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

$$\begin{aligned} gini_{income \in \{low, medium\}}(D) &= \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2) \\ &= \frac{10}{14} \left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14} \left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\ &= 0.450 \\ &= Gini_{income \in \{high\}}(D) \end{aligned}$$

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

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

| | |
|---|---|
| <ul style="list-style-type: none"> Mean absolute error: $\frac{\sum_{i=1}^d y_i - y_i' }{d}$ Relative absolute error: $\frac{\sum_{i=1}^d y_i - y_i' }{\sum_{i=1}^d y_i - \bar{y} }$ | <ul style="list-style-type: none"> Mean squared error: $\frac{\sum_{i=1}^d (y_i - y_i')^2}{d}$ 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

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 &

Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al.'97)
- Classification at primitive concept levels
 - E.g., precise temperature, humidity, outlook, etc.
 - Low-level concepts, scattered classes, bushy classification-trees
 - Semantic interpretation problems
- Cube-based multi-level classification
 - Relevance analysis at multi-levels
 - Information-gain analysis with dimension + level