Classification Methods I

READING

Ch 10 from Hand

Ch 7 from Han

Ack: Slides from Ch 7 (Han)+Figures from Duda&Hart

Course outline

- Introduction* (1)
- Data Warehouse (1)
- Data Preprocessing** (2/1)
- Classification Methods
- Clustering Methods (4)
- Pattern finding (2)
- Applications (1-2)
- Multimedia Mining (2)
- Survey of recent research (2)
  - Presentations
- Course project (2)
  - Presentations

* Number of lectures in (.)
** Actual/Original Estimate

DM Methods: Summary

- Data mining components
- Models and patterns
- Curse of dimensionality
- Scoring functions

Next: Classification Methods

Classification Algorithms

- READING: C10 (Hand), C7 (Han)
- We will be covering the following
  - Linear discriminants and Perceptrons
  - Decision tree induction
  - Bayesian Classification
  - Multilayered Perceptrons (MLP)

Simple case: A Perceptron

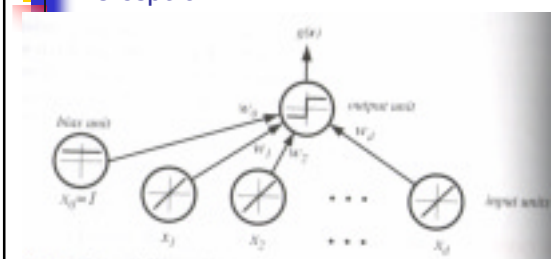
- A discriminative rule—learning the decision boundary surface
- Simplest form: a linear combination of the measurements (**x**).

$$g(\mathbf{x}) = \sum w_j x_j, 0 \le j \le p \text{ are the weights}$$

Goal of learning is to adjust the weights such that g(x) > 0 for class 1 samples and g(x)<0 for class 2 samples.

Duda&HartFig 5.1

Perceptron



Perceptron: adjusting the weights

- Start with an initial set of (random) weights
- Classify the first training sample
  - Correct classification: weights are unchanged
  - Incorrect: $\mathbf{W}=\mathbf{W}+\lambda\mathbf{X}_j$ --the second term is a small correction that is added; \lambda is a small constant
  - This is repeated for all the training samples
  - If the two classes are linearly separable, then this method will eventually find a separating surface.

Discriminant functions

- Pattern classifiers are characterized by discriminant functions $g_i(\mathbf{x})$, i=1, 2, …, *c*.
- The classifier is said to assign a feature vector $\mathbf{x}$ to class i if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$.
- The classifier is viewed as a machine that computes *c* discriminant functions and selects the category corresponding to the largest discriminant
- Linear discriminant: linear combination of the components of $\mathbf{x}$.

Two category case

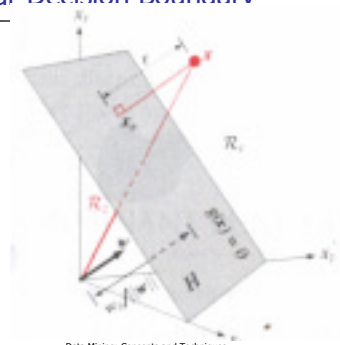
$$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

$$\text{class 1 if } g(\mathbf{x}) > 0;\ \text{class 2 if } g(\mathbf{x}) < 0$$

- $g(\mathbf{x})$ =0 defines the decision surface that separates points assigned to class 1 from those assigned to class 2.
- When $g(\mathbf{x})$ is linear, this decision surface is a *hyperplane.*
- The weight vector is normal to any vector lying on the hyperplane (to see this,, consider any two points on the hyperplane; $g(\mathbf{x1})$=g($\mathbf{x2}$)=0, and take the difference.)

Duda&HartFig 5.2

Linear Decision Boundary



Linear decision boundaries

$\mathbf{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$ (x1 and x2 on the hyperplane)

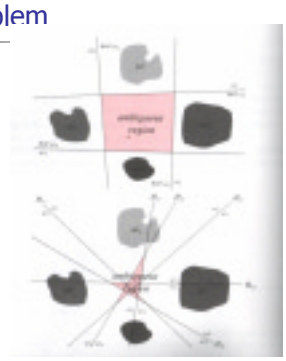
$\mathbf{x} = \mathbf{x}_p + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$, $\mathbf{x}_p$ is the normal projection of $\mathbf{x}$ onto H

$g(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0 = r\|\mathbf{w}\|$

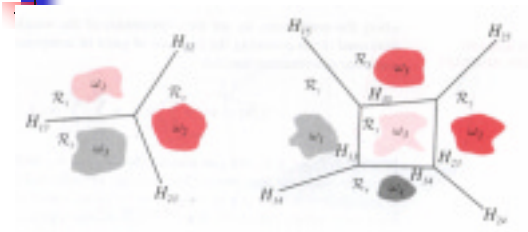
$r = \frac{g(\mathbf{x})}{\|\mathbf{w}\|}$ Distance from the hyperplane

Multiclass problem

c class to (*c*-1) –two class problems.



Multiclass: Decision boundaries



$$g_i(\mathbf{x}) = \mathbf{w}_i^T \mathbf{x} + w_{i0}, \; i = 1, \ldots, c$$

assign $\mathbf{x}$ to class i if $g_i(\mathbf{x}) > g_j(\mathbf{x}), \;$ for all $j \neq i$

Fisher Linear discriminant

- Finding the *best* direction of W to project—so as to achieve a maximum separation between samples.
- What should be a measure of separation? Consider difference of sample means:

$\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$ --mean of the sample vectors

$\tilde{m}_i = \frac{1}{n_i} \sum_{y \in Y_i} y = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{w}^T \mathbf{x}$ --mean of projected samples

= mean of $\mathbf{m}_i$

Fisher

Projected mean diff: $|\tilde{m}_1 - \tilde{m}_2| = |\mathbf{W}^T (\mathbf{m}_1 - \mathbf{m}_2)|$

Scatter for projected samples: $\tilde{s}_i^2 = \sum_{y \in Y_i} (y - \tilde{m}_i)^2$

Total within-class scatter: $\tilde{s}_1^2 + \tilde{s}_2^2$

FLD: $J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$

Scatter Matrices

Let $\mathbf{S}_i = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$

Within-class scatter Matrix: Symmetric, positive definite, Usually non-singular (n>p)

$\mathbf{S}_W = \mathbf{S}_1 + \mathbf{S}_2$

$\tilde{s}_i^2 = \sum_{\mathbf{x} \in D_i} \mathbf{w}^T (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T \mathbf{w}$

$= \mathbf{w}^T \mathbf{S}_i \mathbf{w}$

$\tilde{s}_1^2 + \tilde{s}_2^2 = \mathbf{w}^T \mathbf{S}_W \mathbf{w}$

between-class scatter matrix (rank 1 as it is an outer-product Matrix)

$\mathbf{S}_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$

$(\tilde{m}_1 - \tilde{m}_2)^2 = \mathbf{w}^T \mathbf{S}_B \mathbf{w}$

Classification in reduced dimensions

Score function: $J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$

Optimal weight direction: $\mathbf{w}_{\text{LDA}} = \mathbf{S}_W^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$

Classify the vectors by projecting them along **w.** The classification problem is thus reduced to a one dimensional problem.

Threshold selection: once the samples are projected in the direction of the weight vector, a threshold determines to which one one of the two classes they belong to.

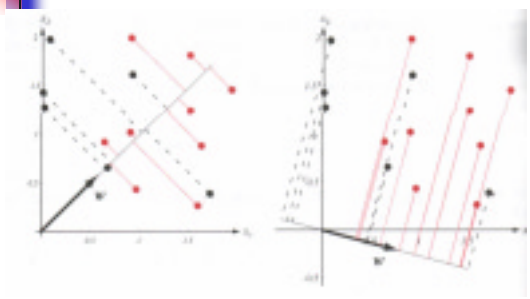
Fisher: thereshold selection

- Specification of threshold: for Gaussian distributions with equal covariance matrix, one can calculate the optimal threshold.
- However, Fisher's criterion is derived without assuming normality. We can still use the same threshold (though may not be optimal)

assign $\mathbf{x}$ to class 1 if

$$\left\{ \mathbf{x} - \frac{1}{2}(\mathbf{m}_1 + \mathbf{m}_2) \right\}^T \mathbf{w} > \log\left(\frac{p(c_1)}{p(c_2)} \right)$$

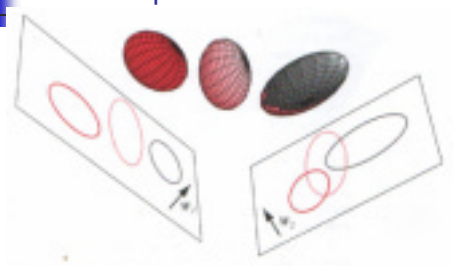
FDS: Example



FDA: Complexity

- $O(c.p^2.n)$: c—number of classes; p—dimensions; n–# samples.
- Since $n>>p$, main cost is in estimating the class covariance matrices S_i. These can be generated in at most two passes of the database, one for get the means and the other to generate the $O(p^2)$ covariance terms.

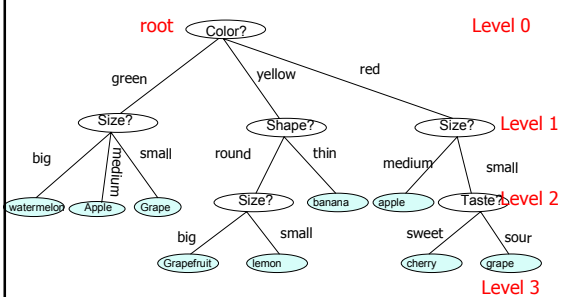
Multiclass problem



Classification Algorithms

- Linear discriminants and Perceptrons
- Decision tree induction
- Bayesian Classification
- Multilayered Networks

Classification and Regression Trees



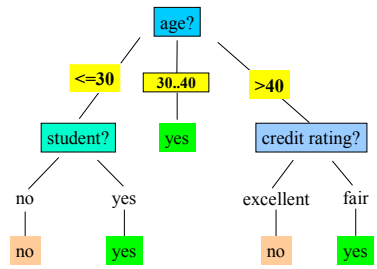
Training Dataset

This follows an example from Quinlan's ID3

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

http://www.cse.unsw.edu.au/~quinlan

Output: A Decision Tree for "*buys_computer*"



Classification by Decision Tree Induction

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

CART: General framework

- Split: binary- or multi-valued? Binary trees simple to train, and can "express" any multi-valued tree.
- Which property should be tested at each node?
- When should the node be declared a leaf?
- How can a tree be pruned?
- How are the labels assigned to an impure node?
- How should the missing data be handled?

ALGORITHM Generate_decision_tree

1. Create a node *N*;
2. IF samples are all of the same class, *C* THEN
   1. RETURN *N* as a leaf node with label *C*
3. IF attribute-list is empty THEN
   1. RETURN N as a leaf-node labeled with the most common class in samples; //majority voting
4. SELECT test-attribute, the attribute among the attribute list with the highest information gain;
5. Label node N with test-attribute;
6. For each known value a_i of test-attribute //partition samples
   1. Grow branch from node N for the condition test_attribute= a_i
   2. Let s_i be the set of samples for which test-attribute=ai//a partition
   3. IF s_i is empty THEN
      1. Attach a leaf labeled with the most common class in samples
   4. ELSE attach the node returned by Generate_decision_tree (s_i , attribute-list – test-attribute)

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 (step 1)
  - 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) (step 4)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class (step 2)
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf (step 3)
  - There are no samples left (step 6.3)

Attribute Selection Measure

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

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain==greatest entropy reduction
- A popular measure is entropy impurity, $I(C) = -\sum_j P(c_j)\log_2 P(c_j)$
- Assume that at a node C there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

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

Information Gain in Decision Tree Induction

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

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

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

$$Gain(C, A) = I(C) - E(A)$$

Attribute Selection by Information Gain Computation

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

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

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

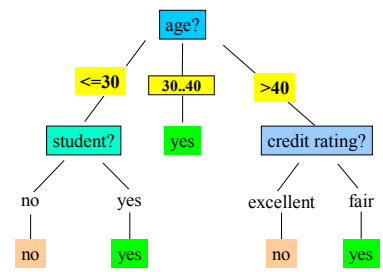
Hence

$$Gain(age) = I(p,n) - E(age) = 0.246$$

Similarly

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

Output: A Decision Tree for "*buys_computer*"



Gini Index (IBM IntelligentMiner)

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

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

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

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

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

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

When to stop splitting?

- Use validation/cross validation;
  - the tree is trained using a subset of data (e.g., 90%), with the remaining (10%) kept as a validation set.
  - Continue splitting nodes until the error in validation data is minimized.
  - Cross validation relies on *several* independently chosen subsets.
- Hypothesis testing (use all data for training)—NEXT SLIDE
  - but apply a statistical test (e.g., chi-square) to estimate whether expanding a node is better than a random split
- Use minimum description length (MDL) principle:
  - halting growth of the tree when the encoding complexity is minimized

Hypothesis testing

- Suppose n patterns survive at node N (with n1 in c1 and n2 in c2)
- We want to decide whether a candidate split s differs significantly from a random one
- Suppose a candidate split s sends P.n patterns to the left branch and (1-P).n to the right
  - A random split having this probability would have sent P.n1 of c1 patterns and P.n2 of c2 patterns to the left, remaining to the right
- Chi-aquare statitistic: to quantify the deviation from a random split

Chi-square statistic

$$\chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{ie})^2}{n_{ie}}$$

n_{iL} : # patterns in class i sent to the left under decision s

$n_{ie} = Pn_i$: # expected by the random rule

Chi-square statistic is larger the more s differs from the random one → if this is larger than a critical value, we can reject the null hypothesis; critical values depend on the number of degrees of freedom – is 1 in the above case.

Pruning

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

Extracting Classification Rules from Trees

- Represent the knowledge in the form of IF-THEN rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class prediction
- Rules are easier for humans to understand
- Example

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

Enhancements to basic decision tree induction

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

Classification in Large Databases

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

Scalable Decision Tree Induction Methods in Data Mining Studies

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

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.

Classification Algorithms

- Linear discriminants and Perceptrons
- Decision tree induction
- Bayesian Classification
- Multilayered Networks