Data Classification (a)

Lijun Zhang
zlj@nju.edu.cn
http://cs.nju.edu.cn/zlj

Outline

- **Introduction**
- Feature Selection for Classification
- Decision Trees
- Rule-Based Classifiers
- Probabilistic Classifiers
- Summary

Introduction

- Input
  - A data set of examples, that was partitioned into classes
- Process
  - Learn the structure of the above set
- Output
  - A model that can be used to predict the label of unseen examples
- Keywords
  - Training data, Class labels, Training model, Testing data, Learner

Classification

Definition

Given a set of training data points, each of which is associated with a class label, determine the class label of one or more previously unseen test instances.

Two phases

- **Training** phase: a training model is constructed from the training instances
- **Testing** phase: the training model is used to determine the class label of one or more unseen test instances

Applications

Application	Feature	Label
Customer target marketing	Demographic profiles	User interest in a particular product
Medical disease management	Patient medical tests and treatments	Treatment outcomes
Document categorization and filtering	Words in the document	Topics
Multimedia data analysis	Features of photos, videos, and audio	Activities of users

Formulation

- Training Data $\mathcal{D} = \{(\overline{X_1}, y_1), \ldots, (\overline{X_n}, y_n)\}$
  - Where $\overline{X_i} \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$
  - If $k = 2$, we may use $y_i \in \{0,1\}$ or $y_i \in \{\pm 1\}$
- Output of a classification algorithm
  - Label prediction
  - Numerical score: learner assigns a score to each instance–label combination
- Overfitting
  - Models accurately predict the labels of training instances, but they perform poorly on unseen test instances

Outline

- Introduction
- **Feature Selection for Classification**
- Decision Trees
- Rule-Based Classifiers
- Probabilistic Classifiers
- Summary

Feature Selection for Classification

- Motivations
  - Features of varying relevance for predicting class labels
- Three primary types of methods
  - **Filter** models: a mathematical criterion is used to filter out irrelevant features
  - **Wrapper** models: evaluate the feature by a classification algorithm
  - **Embedded** models: embed the problem of feature selection into the process of classification

Filter Models

The Process

- A feature or a *subset* of features is evaluated with the use of a *class-sensitive discriminative criterion*
- Evaluate a group of features can reduce redundancy but the search space is huge
  - ✓ 2^d possible subsets
- Some methods use a *linear combination* of the original features
  - ✓ Dimensionality reduction

Gini Index (1)

- Let $v_1, \ldots, v_r$ be r possible values of a particular categorical attribute
- Let p_{ij} be the fraction of data points containing v_i that belong to class j
- A value-specific Gini index

$$G(v_i) = 1 - \sum_{j=1}^{k} p_{ij}^2$$

  - $G(v_i) = 1 - 1/k$ if v_i distributed evenly
  - $G(v_i) = 0$ if v_i appears in one class
  - Lower values of the Gini index imply greater discrimination

Gini Index (2)

- Let $v_1, \ldots, v_r$ be r possible values of a particular categorical attribute
- Let p_{ij} be the fraction of data points containing v_i that belong to class j
- A value-specific Gini index

$$G(v_i) = 1 - \sum_{j=1}^{k} p_{ij}^2$$

- An attribute-wise Gini index

$$G = \frac{\sum_{i=1}^{r} n_i G(v_i)}{n}$$

Entropy (1)

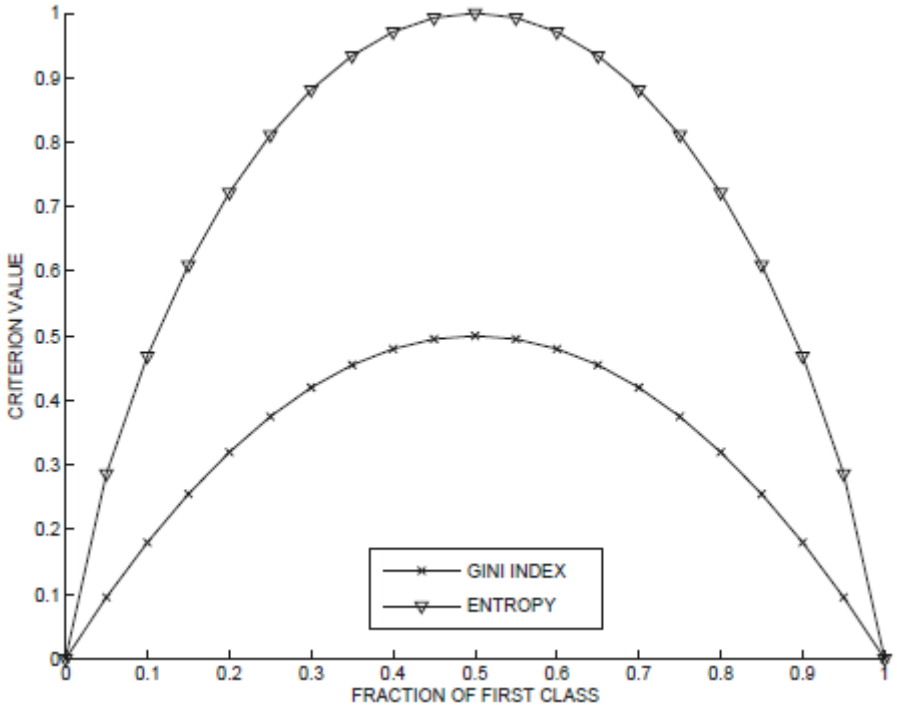
- Let $v_1, \ldots, v_r$ be r possible values of a particular categorical attribute
- Let p_{ij} be the fraction of data points containing v_i that belong to class j
- A value-specific entropy

$$E(v_i) = -\sum_{j=1}^{k} p_{ij} \log_2 p_{ij}$$

  - Lie in the interval $[0, \log_2 k]$
  - Lower values of the entropy imply greater discrimination

Entropy (2)

- An example



- An attribute-wise entropy

$$E = \frac{\sum_{i=1}^{r} n_i E(v_i)}{n}$$

Fisher Score (1)

- μ be the mean of a numeric feature
- μ_j and σ_j be the mean and standard deviation of data points belong to class j for a numeric feature
- p_j be the fraction of data belong to class j
- The Fisher score

$$F = \frac{\sum_{j=1}^{k} p_j (\mu_j - \mu)^2}{\sum_{j=1}^{k} p_j \sigma_j^2}$$

  - Ratio of the interclass separation to intraclass separation
  - Larger values of imply greater discrimination

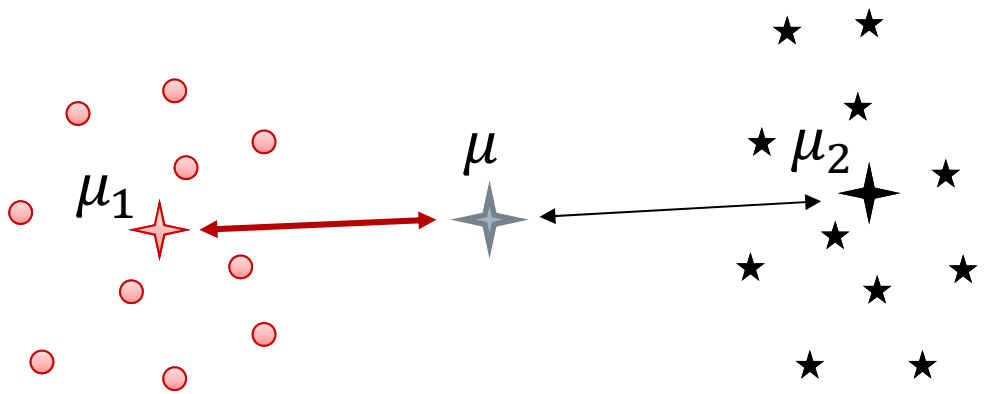
Fisher Score (2)

- An Example

Fisher Score (3)

- An Example
  - Interclass separation

$$F = \frac{\sum_{j=1}^{k} p_j (\mu_j - \mu)^2}{\sum_{J=1}^{k} p_j \sigma_j^2}$$



Fisher Score (4)

An Example

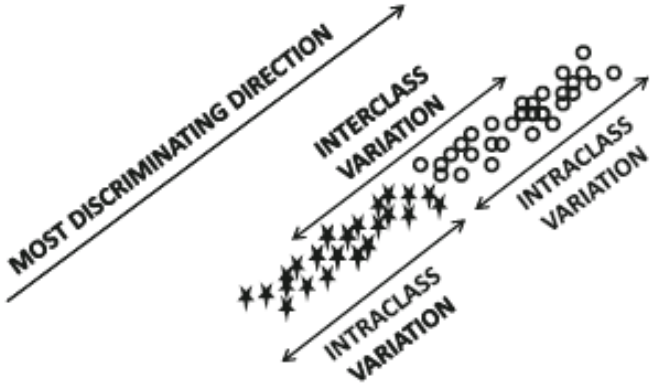
- Interclass separation

$$F = \frac{\sum_{j=1}^{k} p_j (\mu_j - \mu)^2}{\sum_{J=1}^{k} p_j \sigma_j^2}$$

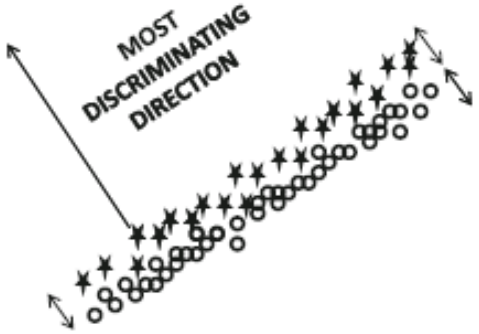
- Intraclass separation

Fisher's Linear Discriminant Analysis (LDA)

- A generalization of the Fisher score to find linear combinations of features
- A supervised dimensionality reduction method to maximize the class-specific discrimination



(a) Discriminating direction is aligned with high-variance direction



(b) Discriminating direction is aligned with low-variance direction

Two-class Scenario (1)

- $\overline{\mu_0}$ and $\overline{\mu_1}$ be the mean vectors
- Σ_0 and Σ_1 be the covariance matrices
- p_0 and p_1 be the fractions of two classes
- Fisher score for a vector $\overline{W}$

$$FS(\overline{W}) = \frac{\text{Between Class Scatter along } \overline{W}}{\text{Within Class Scatter along } \overline{W}} \propto \frac{(\overline{W}\cdot\overline{\mu_1} - \overline{W}\cdot\overline{\mu_0})^2}{p_0[\text{Variance(Class 0)}] + p_1[\text{Variance(Class 1)}]}$$
$$= \frac{\overline{W}[(\overline{\mu_1}-\overline{\mu_0})^T(\overline{\mu_1}-\overline{\mu_0})]\overline{W}^T}{p_0[\overline{W}\Sigma_0\overline{W}^T] + p_1[\overline{W}\Sigma_1\overline{W}^T]} = \frac{[\overline{W}\cdot(\overline{\mu_1}-\overline{\mu_0})]^2}{\overline{W}(p_0\Sigma_0 + p_1\Sigma_1)\overline{W}^T}.$$

  - $\overline{W}\cdot\overline{\mu_0}$ and $\overline{W}\cdot\overline{\mu_1}$ are the means of two classes after projection by $\overline{W}$
  - $\overline{W}\Sigma_0\overline{W}^\top$ and $\overline{W}\Sigma_1\overline{W}^\top$ are variances of two classes after projection by $\overline{W}$

Two-class Scenario (2)

- Between-class scatter-matrix

$$S_b = (\overline{\mu_1} - \overline{\mu_0})^\top (\overline{\mu_1} - \overline{\mu_0})$$

- Within-class scatter matrix

$$S_w = p_0\Sigma_0 + p_1\Sigma_1$$

- The objective

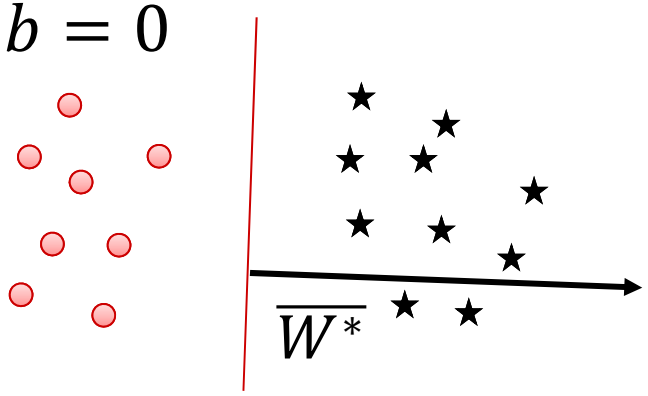
$$\max_{\bar{W}\in\mathbb{R}^d} \frac{\bar{W}S_b\bar{W}^\top}{\bar{W}S_w\bar{W}^\top} \Longleftrightarrow \max_{\bar{W}\in\mathbb{R}^d} \bar{W}S_b\bar{W}^\top \quad \text{s.t.} \quad \bar{W}S_w\bar{W}^\top = 1$$

  - Generalized Eigenproblem

$$S_b\bar{W}^\top = \lambda S_w\bar{W}^\top \Longrightarrow \overline{W^*} \propto (\overline{\mu_1} - \overline{\mu_0})\,(p_0\Sigma_0 + p_1\Sigma_1)^{-1}$$

Discussions

- Can be used as used as a stand-alone classifier
  - $\overline{W^*} \cdot \bar{X} + b = 0$



- A special case of least-square regression
- Extension to multi-class is straightforward

Wrapper Models

- Leverage classification algorithms to select features
- A general approach

1. Create an augmented set of features F by adding one or more features to the current feature set.
2. Use a classification algorithm $\mathcal{A}$ to evaluate the accuracy of the set of features F. Use the accuracy to either accept or reject the augmentation of F.

  - A greedy strategy
  - Random sampling

Embedded Models

- Knowledge about the features is embedded within the solution to the classification problem
- Consider a linear classifier

$$y_i = \text{sign}\{\overline{W} \cdot \overline{X} + b\}.$$

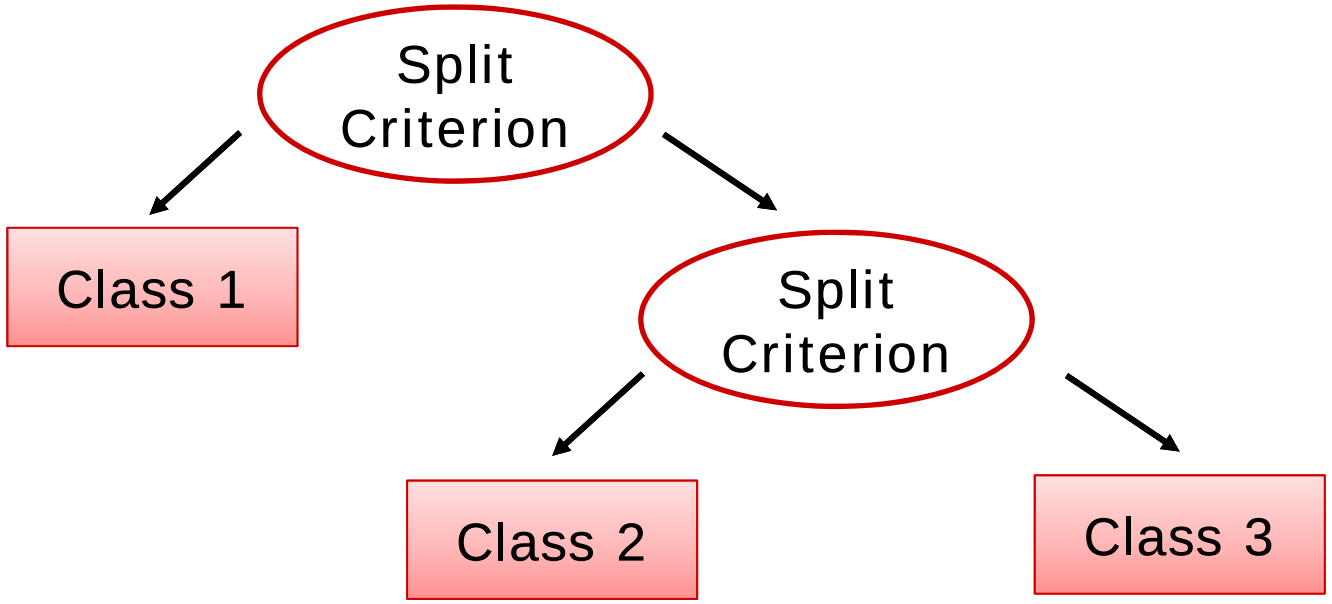
  - $\overline{W} = \{w_1, \ldots, w_d\}$ is a d-dimensional vector of coefficients
  - Remove the i-th feature if $|w_i|$ is small
  - Retrain the model with the remaining features
  - Repeat the above process

Outline

- Introduction
- Feature Selection for Classification
- **Decision Trees**
- Rule-Based Classifiers
- Probabilistic Classifiers
- Summary

Decision Trees

- Tree-like structure



- Split criterion: a condition on one or more feature variables

$$Age \leq 30$$

Decision Tree Construction (1)

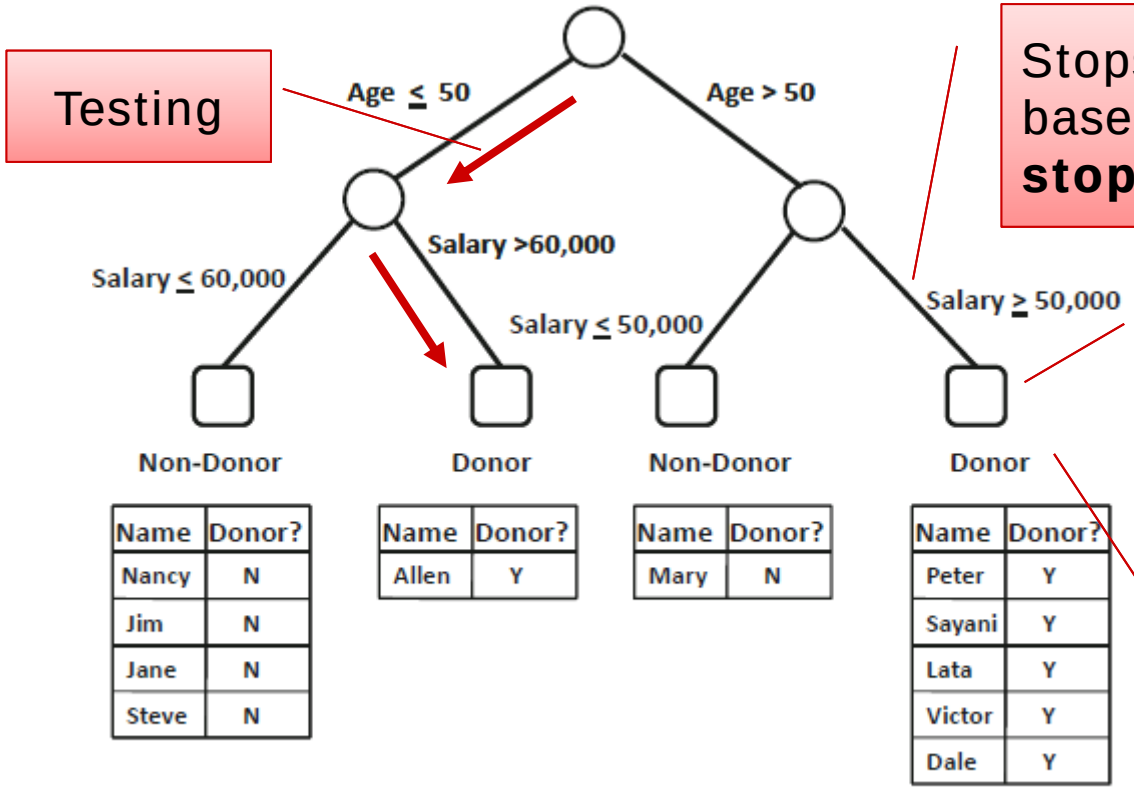
- Training data

- Identify a split criterion so that the level of "mixing" of the class variables in each branch of the tree is low

Name	Age	Salary	Donor?
Nancy	21	37,000	N
Jim	27	41,000	N
Allen	43	61,000	Y
Jane	38	55,000	N
Steve	44	30,000	N
Peter	51	56,000	Y
Sayani	53	70,000	Y
Lata	56	74,000	Y
Mary	59	25,000	N
Victor	61	68,000	Y
Dale	63	51,000	Y

Decision Tree Construction (2)

- Create nodes at the leaf level in which the donors and nondonors are separated well



(a) Univariate split

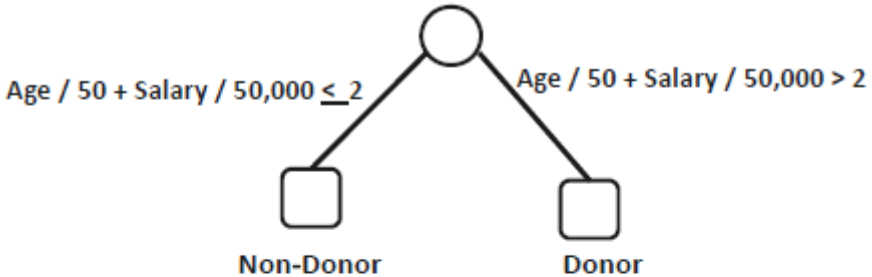
Stops the growth based on a **stopping criterion**

Each leaf node is labeled with the **dominant** class at that node

Pruning if necessary

Decision Tree Construction (3)

- Multivariate splits lead to shallower trees



Name	Donor?
Nancy	N
Jim	N
Jane	N
Steve	N
Mary	N

Name	Donor?
Allen	Y
Peter	Y
Sayani	Y
Lata	Y
Victor	Y
Dale	Y

(b) Multivariate split

The Overall Procedure

Algorithm *GenericDecisionTree*(Data Set: $\mathcal{D}$)
begin
  Create root node containing $\mathcal{D}$;
  repeat
    Select an eligible node in the tree;
    Split the selected node into two or more nodes
      based on a pre-defined split criterion;
  until no more eligible nodes for split;
  Prune overfitting nodes from tree;
  Label each leaf node with its dominant class;
end

Ways of Splits

- Binary attribute: one type of split
- Categorical attribute
  - r-way split for r possible values
  - Binary split by testing each of the $2^r - 1$ groupings of attributes
  - Convert to binary data
- Numeric attribute
  - r-way split for r possible values
  - Binary condition: $x \leq a$
    - ✓ If there are m data points, then there are m possible split points

Split Criteria (1)

- Error rate
  - Let p be the fraction of the dominant class in $\mathcal{S}$
  - Error rate of $\mathcal{S}$

$$E(\mathcal{S}) = 1 - p$$

  - Error rate of a split

$$\text{Error\_Rate}(\mathcal{S} \Longrightarrow \mathcal{S}_1 \ldots \mathcal{S}_r) = \sum_{i=1}^{r} \frac{|\mathcal{S}_i|}{|\mathcal{S}|} E(\mathcal{S}_i)$$

  - The smaller the better

Split Criteria (2)

Gini index

- Let $p_1, \ldots, p_k$ be the class distribution of $\mathcal{S}$
- Gini index of $\mathcal{S}$

$$G(S) = 1 - \sum_{j=1}^{k} p_j^2$$

- Gini index of a split

$$\text{Gini-Split}(S \Rightarrow S_1 \ldots S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} G(S_i)$$

- The smaller the better

Split Criteria (3)

Entropy

- Let $p_1, \ldots, p_k$ be the class distribution of $\mathcal{S}$
- Entropy of $\mathcal{S}$

$$E(S) = -\sum_{j=1}^{k} p_j \log_2(p_j)$$

- Entropy of a split

$$\text{Entropy-Split}(S \Rightarrow S_1 \ldots S_r) = \sum_{i=1}^{r} \frac{|S_i|}{|S|} E(S_i)$$

- The smaller the better

Split Criteria (4)

Information Gain (IG)

$$E(S) - \text{Entropy-Split}(S \Rightarrow S_1 \dots S_r)$$

- Large values of the reduction are desirable

A limitation

- Both entropy and information gain are biased in favor of splits with larger degree

Normalized Information Gain

- Dividing IG by $-\sum_{i=1}^{r} \frac{|S_i|}{|S|} \log_2(\frac{|S_i|}{|S|})$

Stopping Criterion and Pruning (1)

- It is easy to construct a decision tree that achieves 100% accuracy on training data
  - However, it often generalizes poorly
- In general, simpler models are preferable to more complex models
  - If they produce the same error on the training data

Stopping Criterion and Pruning (2)

- Stop the growth of the tree early
  - Difficult to decide the stopping point
- Pruning the overfitting portion
  - Minimum Description Length principle (MDL)
    - ✓ Cost = a weighted sum of its training error and its complexity
    - ✓ Construct tree to optimize the cost
  - Cross-validation
    - ✓ Build the tree on part of training data
    - ✓ Prune based on the performance on the remaining/validation/holdout data

Outline

- Introduction
- Feature Selection for Classification
- Decision Trees
- **Rule-Based Classifiers**
- Probabilistic Classifiers
- Summary

Rule-Based Classifiers (1)

- A rule: IF *Condition* THEN *Conclusion*.

  Antecedents — Consequents

  - Antecedents may be any arbitrary condition on the feature variables
    - ✓ E.g. a pattern
  - Consequents are class labels

- A decision tree may be viewed as a special case of a rule-based classifier

$$Age \leq 50 \text{ AND } Salary \leq 60,000 \Rightarrow \neg Donor$$
$$Age \leq 50 \text{ AND } Salary > 60,000 \Rightarrow Donor$$

Rule-Based Classifiers (2)

- Training: create a set of rules
- Testing: discover rules that are triggered by the test instance
  - Multiple rules may be triggered
- Properties of rule sets
  - Mutually exclusive rules: at most one rule can be triggered
  - Exhaustive rules: at least one rule can be triggered
- Solve conflicts: ordering, voting

Rule Generation from Decision Trees

- Rules can be extracted from the different paths in a decision tree.
  - Conjunctive form $Age \leq 50$ AND $Salary \leq 60,000 \Rightarrow \neg Donor$
  - Exhaustive and mutually exclusive
- Rule-pruning
  - Conjuncts are pruned from the rules
- Rule ordering
  - Group rules by consequents/classes
  - Rank rule sets by description length, or false-positive errors on a holdout set

Sequential Covering Algorithms

The Procedure

1. (*Learn-One-Rule*) Select a particular class label and determine the "best" rule from the current training instances $\mathcal{D}$ with this class label as the consequent. Add this rule to the bottom of the ordered rule list.

2. (*Prune training data*) Remove the training instances in $\mathcal{D}$ that are covered by the rule learned in the previous step. All training instances matching the *antecedent* of the rule must be removed, whether or not the class label of the training instance matches the consequent.

How to order rules?

- Class-based ordering
  - ✓ Rare classes are ordered first
  - ✓ Rules for a particular class are generated contiguously
  - ✓ A stopping criterion to terminate addition
    - MDL, Error rate on a validation set, Threshold

Sequential Covering Algorithms

The Procedure

1. (*Learn-One-Rule*) Select a particular class label and determine the "best" rule from the current training instances $\mathcal{D}$ with this class label as the consequent. Add this rule to the bottom of the ordered rule list.

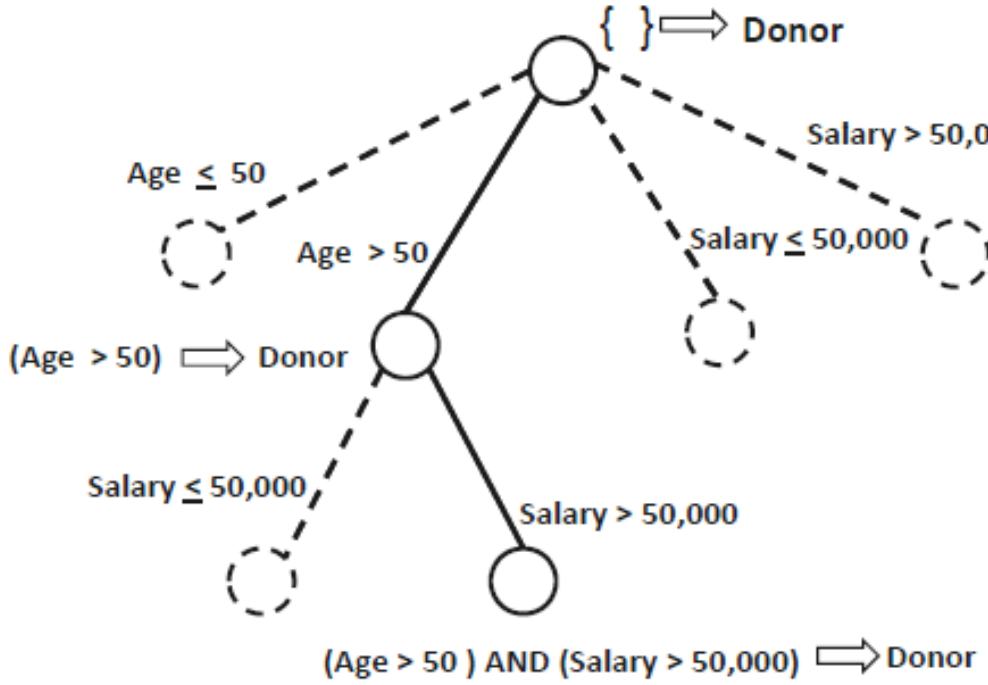
2. (*Prune training data*) Remove the training instances in $\mathcal{D}$ that are covered by the rule learned in the previous step. All training instances matching the *antecedent* of the rule must be removed, whether or not the class label of the training instance matches the consequent.

How to order rules?

- Class-based ordering
- Quality-based ordering
  - ✓ A quality measure is used to select next rule
  - ✓ E.g., one might generate the rule with the highest confidence

Learn-One-Rule

- Successively add conjuncts to the left-hand side of the rule based on a quality criterion until stop



Quality Criterion (1)

- The general idea is to combine accuracy and coverage criteria
- Laplacian Smoothing

$$\text{Laplace}(\beta) = \frac{n^+ + \beta}{n^+ + n^- + k\beta}$$

  - n^+ and n^- represents the number of correctly and incorrectly classified examples
  - It becomes accuracy if $\beta = 0$
  - $n^+ + n^-$ is the coverage
  - Penalize low coverage

Quality Criterion (2)

Likelihood ratio statistic

- $p_1, \ldots p_k$ be the fraction of examples belonging to each class in the full data
- The expected number of each class for a rule covers n examples

$$n_1^e = p_i n \quad \ldots \quad n_k^e = p_k n$$

- The true number of each class for a rule convers n examples: $n_1, \ldots, n_k$
- The likelihood ration statistic

$$R = 2\sum_{j=1}^{k} n_j \log(n_j / n_j^e)$$

Quality Criterion (3)

Likelihood ratio statistic (cont.)

- The likelihood ration statistic

$$R = 2\sum_{j=1}^{k} n_j \log(n_j / n_j^e)$$

- Favor covered examples whose distributions are very different from the original training data

- Favor large coverage

Quality Criterion (4)

- FOIL's Information Gain
  - First order inductive learner
  - A rule covers n_1^+ positive examples and n_1^- negative examples
  - After addition of a conjunct, it covers n_2^+ positive examples and n_2^- negative examples

$$FG = n_2^+ \left(\log_2 \frac{n_2^+}{n_2^+ + n_2^-} - \log_2 \frac{n_1^+}{n_1^+ + n_1^-} \right)$$

  - High coverage and high accuracy
- Minimum description length

Rule Pruning

- Minimum Description Length
- Pessimistic Error Rate
  - Add a penalty term δ to each growth
- Error rate on a validation set

- The order of conjuncts for pruning
  - Reverse order
  - Greedy
  - Any order
- Duplicate rules are removed

Associative Classifiers

- The basic strategy
  1. Mine all class-based association rules at a given level of minimum support and confidence
  2. For a given test instance, use the mined rules for classification
- The 1st step
  - Mine all association rules and then filter out
    - ✓ Wasteful, Redundancy
  - Classification based on associations
    - ✓ Create 1-rule-items, extend, stop

Associative Classifiers

- The basic strategy
  1. Mine all class-based association rules at a given level of minimum support and confidence
  2. For a given test instance, use the mined rules for classification
- The 2nd step
  - Ordered strategy
    - ✓ By support and confidence
  - Unordered strategy

Outline

- Introduction
- Feature Selection for Classification
- Decision Trees
- Rule-Based Classifiers
- **Probabilistic Classifiers**
- Summary

Naive Bayes Classifier (1)

Bayes theorem

Example 10.5.1 *A charitable organization solicits donations from individuals in the population of which* $6/11$ *have age greater than 50. The company has a success rate of* $6/11$ *in soliciting donations, and among the individuals who donate, the probability that the age is greater than 50 is* $5/6$*. Given an individual with age greater than 50, what is the probability that he or she will donate?*

- E is the event $(Age > 50)$
- D is the event of being a donor
- Posterior probability $P(D|E)$
- Bayes theorem

$$P(D|E) = \frac{P(E|D)P(D)}{P(E)} = \frac{(5/6)(6/11)}{6/11} = 5/6.$$

Naive Bayes Classifier (2)

Model for Classification

- The goal is to predict

$$P(C = c|\bar{X} = (a_1 \dots a_d))$$

- Can it be estimate directly?
  - ✓ Training data may not contain $(a_1 \dots a_d)$
- Bayes theorem

$$P(C = c|x_1 = a_1, \dots x_d = a_d) = \frac{P(C = c)P(x_1 = a_1, \dots x_d = a_d|C = c)}{P(x_1 = a_1, \dots x_d = a_d)}$$
$$\propto P(C = c)P(x_1 = a_1, \dots x_d = a_d|C = c).$$

- Naive Bayes approximation

$$P(x_1 = a_1, \dots x_d = a_d|C = c) = \prod_{j=1}^{d} P(x_j = a_j|C = c)$$

Conditional Independence

Naive Bayes Classifier (3)

- Bayes Probability

$$P(C = c|x_1 = a_1, \ldots x_d = a_d) \propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c).$$

Naive Bayes Classifier (3)

Bayes Probability

$$P(C = c|x_1 = a_1, \ldots x_d = a_d) \propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c).$$

- $P(C = c)$: estimated as the fraction of the training data points belonging to class c

$$P(C = c) = \frac{r(c)}{n}$$

  - ✓ $r(c)$ is the number of examples in class c
  - ✓ n is the number of total training examples

Naive Bayes Classifier (3)

Bayes Probability

$$P(C = c|x_1 = a_1, \ldots x_d = a_d) \propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c).$$

- $P(C = c)$: $P(C = c) = \frac{r(c)}{n}$
- $P(x_j = a_j|C = c)$: estimated as the fraction of training examples taking on value a_j, conditional on the fact, that they belong to class c

$$P(x_j = a_j|C = c) = \frac{q(a_j, c)}{r(c)}$$

Naive Bayes Classifier (4)

Bayes Probability

$$P(C = c|x_1 = a_1, \ldots x_d = a_d) \propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j|C = c).$$

- $P(C = c)$: $P(C = c) = \frac{r(c)}{n}$
- $P(x_j = a_j|C = c)$:

$$P(x_j = a_j|C = c) = \frac{q(a_j, c)}{r(c)}$$

- Laplacian smoothing
  - ✓ To avoid overfitting for rare classes

$$P(x_j = a_j|C = c) = \frac{q(a_j, c) + \alpha}{r(c) + \alpha \cdot m_j}$$

  - ✓ m_j is the number of distinct values

Ranking Model

- Rank a set of testing data for class c
  - $P(x_1 = a_1, \ldots, x_d = a_d)$ is lost in

$$P(C = c | x_1 = a_1, \ldots x_d = a_d) \propto P(C = c) \prod_{j=1}^{d} P(x_j = a_j | C = c).$$

  - Estimating it directly is difficult
  - An alternative way

$$P(C = c | x_1 = a_1, \ldots x_d = a_d) = \frac{P(C = c) \prod_{j=1}^{d} P(x_j = a_j | C = c)}{\sum_{c=1}^{k} P(C = c) \prod_{j=1}^{d} P(x_j = a_j | C = c)}.$$

  - Ranking by the above probability

Discussions

- Binary or Bernoulli model
  - When each feature has two outcomes
- Generalized Bernoulli model
  - When each feature has more than two outcomes
- Multinomial model
  - To address sparse frequencies
- Gaussian model
  - For numerical features
- All are mixture-based generative models
  - Training step is similar to the M-step in EM

Logistic Regression (1)

- Binary class $\{-1,1\}$
  - A discriminative model
  - Given $\bar{X} = (x_1 \ldots x_d)$, assume

$$P(C = +1|\bar{X}) = \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$

$$P(C = -1|\bar{X}) = \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$

  - θ_i is the coefficient for the i-th feature
  - θ_0 is the offset parameter
  - Can be treated as a linear classifier

$$\theta_0 + \sum_{i=1}^{d} \theta_i x_i \geq 0 \Rightarrow P(C = +1|\bar{X}) \geq 0.5 \geq P(C = -1|\bar{X}) \longrightarrow \bar{X} \in \text{class} + 1$$

Logistic Regression (1)

- Binary class $\{-1,1\}$
  - A discriminative model
  - Given $\bar{X} = (x_1 \ldots x_d)$, assume

$$P(C = +1|\bar{X}) = \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$

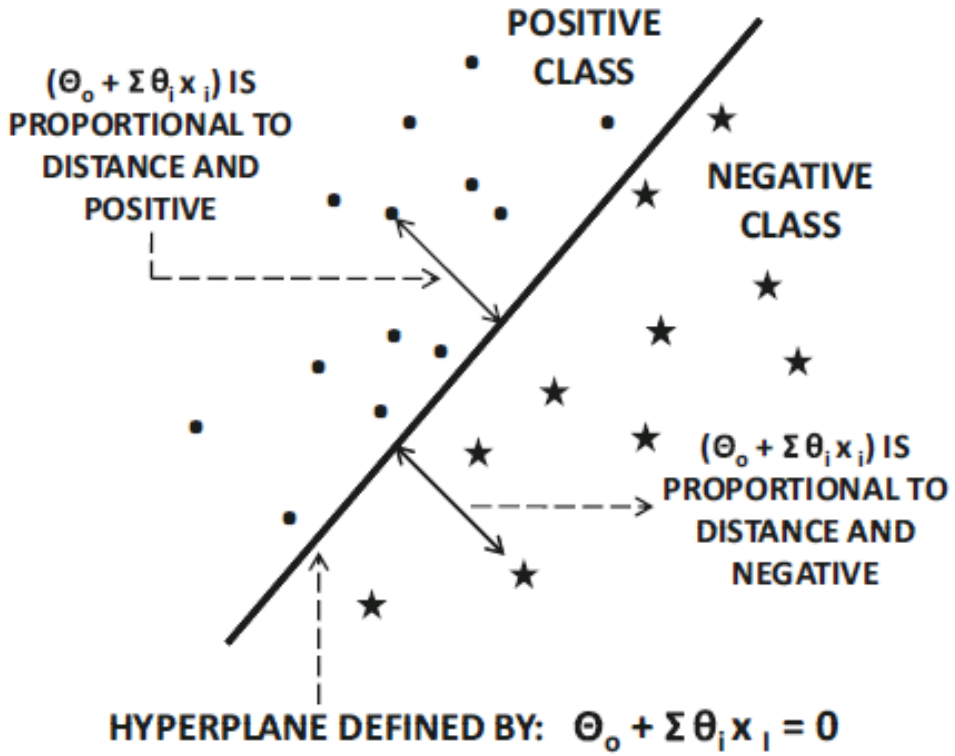
$$P(C = -1|\bar{X}) = \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_i)}}$$

  - θ_i is the coefficient for the i-th feature
  - θ_0 is the offset parameter
  - Can be treated as a linear classifier

$$\theta_0 + \sum_{i=1}^{d} \theta_i x_i \leq 0 \Rightarrow P(C = +1|\bar{X}) \leq 0.5 \leq P(C = -1|\bar{X}) \longrightarrow \bar{X} \in \text{class} - 1$$

Logistic Regression (2)

- An Example



Training a Logistic Regression Classifier (1)

- Maximum Likelihood Approach
  - D_+ be the data in positive classes
  - $D\_$ be the data in negative classes
  - Likelihood function

$$\mathcal{L}(\overline{\Theta}) = \prod_{\overline{X_k} \in \mathcal{D}_+} \frac{1}{1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_k^i)}} \prod_{\overline{X_k} \in \mathcal{D}_-} \frac{1}{1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_k^i)}}$$

  - Log Likelihood

$$\mathcal{LL}(\overline{\Theta}) = \log(\mathcal{L}(\overline{\Theta})) = -\sum_{\overline{X_k} \in \mathcal{D}_+} \log(1 + e^{-(\theta_0 + \sum_{i=1}^{d} \theta_i x_k^i)}) - \sum_{\overline{X_k} \in \mathcal{D}_-} \log(1 + e^{(\theta_0 + \sum_{i=1}^{d} \theta_i x_k^i)})$$

Training a Logistic Regression Classifier (2)

- The Optimization Problem

$$\max_{\bar{\Theta}} \mathcal{LL}(\bar{\Theta})$$

Maximize a concave function

Training a Logistic Regression Classifier (2)

The Optimization Problem

$$\max_{\overline{\Theta}} \mathcal{LL}(\overline{\Theta})$$

- Gradient ascent

$$\nabla \mathcal{LL}(\overline{\Theta}) = \left(\frac{\partial \mathcal{LL}(\overline{\Theta})}{\partial \theta_0} \cdots \frac{\partial \mathcal{LL}(\overline{\Theta})}{\partial \theta_d} \right)$$

$$\begin{aligned}
\frac{\partial \mathcal{LL}(\overline{\Theta})}{\partial \theta_i} &= \sum_{\overline{X_k} \in \mathcal{D}_+} \frac{x_k^i}{1 + e^{(\theta_0 + \sum_{i=1}^d \theta_i x_i)}} - \sum_{\overline{X_k} \in \mathcal{D}_-} \frac{x_k^i}{1 + e^{-(\theta_0 + \sum_{i=1}^d \theta_i x_i)}} \\
&= \sum_{\overline{X_k} \in \mathcal{D}_+} P(\overline{X_k} \in \mathcal{D}_-) x_k^i - \sum_{\overline{X_k} \in \mathcal{D}_-} P(\overline{X_k} \in \mathcal{D}_+) x_k^i \\
&= \sum_{\overline{X_k} \in \mathcal{D}_+} P(\text{Mistake on } \overline{X_k}) x_k^i - \sum_{\overline{X_k} \in \mathcal{D}_-} P(\text{Mistake on } \overline{X_k}) x_k^i
\end{aligned}$$

Training a Logistic Regression Classifier (2)

The Optimization Problem

$$\max_{\overline{\Theta}} \mathcal{LL}(\overline{\Theta})$$

- Gradient ascent
  - ✓ The updating rule

$$\theta_i \leftarrow \theta_i + \alpha \left(\sum_{\overline{X_k} \in \mathcal{D}_+} P(\overline{X_k} \in \mathcal{D}_-) x_k^i - \sum_{\overline{X_k} \in \mathcal{D}_-} P(\overline{X_k} \in \mathcal{D}_+) x_k^i \right)$$

  - ✓ α is a step size
  - ✓ Converge to the global optimum
  - ✓ Stochastic version: use a subset of data

Training a Logistic Regression Classifier (3)

Regularized Problem

$$\max_{\overline{\Theta}} \mathcal{LL}(\overline{\Theta}) - \frac{\lambda}{2}\sum_{i=1}^{d}\theta_i^2$$

- Gradient ascent
  - ✓ The updating rule

$$\theta_i \leftarrow \theta_i + \alpha\left(\sum_{\overline{X_k}\in\mathcal{D}_+} P(\overline{X_k}\in\mathcal{D}_-)x_k^i - \sum_{\overline{X_k}\in\mathcal{D}_-} P(\overline{X_k}\in\mathcal{D}_+)x_k^i\right) + \alpha\lambda\theta_i$$

  - ✓ α is a step size
  - ✓ Converge to the global optimum
  - ✓ Stochastic version: use a subset of data

Linear Models

- Logistic Regression
  - Logistic function

$$\mathcal{LL}(\overline{\Theta}) = \log(\mathcal{L}(\overline{\Theta})) = -\sum_{\overline{X_k}\in\mathcal{D}_+} \log(1+e^{-(\theta_0+\sum_{i=1}^{d}\theta_i x_k^i)}) - \sum_{\overline{X_k}\in\mathcal{D}_-} \log(1+e^{(\theta_0+\sum_{i=1}^{d}\theta_i x_k^i)})$$

- Neural Networks
  - Squared error

$$\sum_{\overline{X_k}\in\mathcal{D}}(y_k - \mathrm{sign}(\theta_0+\sum_{i=1}^{d}\theta_i x_i^k))^2$$

- Fisher's linear discriminant
  - Squared error
- Support Vector Machine
  - Hinge loss

Outline

- Introduction
- Feature Selection for Classification
- Decision Trees
- Rule-Based Classifiers
- Probabilistic Classifiers
- **Summary**

Summary

- Feature Selection for Classification
  - Filter Models (LDA), Wrapper, Embedded
- Decision Trees
  - Split, Stopping and Pruning Criteria
- Rule-Based Classifiers
  - Rule Generation from Decision Trees
  - Sequential Covering Algorithms
  - Associative Classifiers
  - Rule Pruning
- Probabilistic Classifiers
  - Naive Bayes, Logistic regression