CS 521
Data Mining Techniques
Instructor: Abdullah Mueen

LECTURE 3: CLASSIFICATION: DECISION TREE AND BAYES CLASSIFIER

Chapter 8. Classification: Basic Concepts

Supervised vs. Unsupervised Learning

Supervised learning (classification)

- Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
- New data is classified based on the training set

Unsupervised learning (clustering)

- The class labels of training data is unknown
- Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Prediction Problems: Classification vs. Numeric Prediction

Classification

- predicts categorical class labels (discrete or nominal)
- classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data

Numeric Prediction

- models continuous-valued functions, i.e., predicts unknown or missing values

Typical applications

- Credit/loan approval:
- Medical diagnosis: if a tumor is cancerous or benign
- Fraud detection: if a transaction is fraudulent
- Web page categorization: which category it is

Classification—A Two-Step Process

Model construction: describing a set of predetermined classes

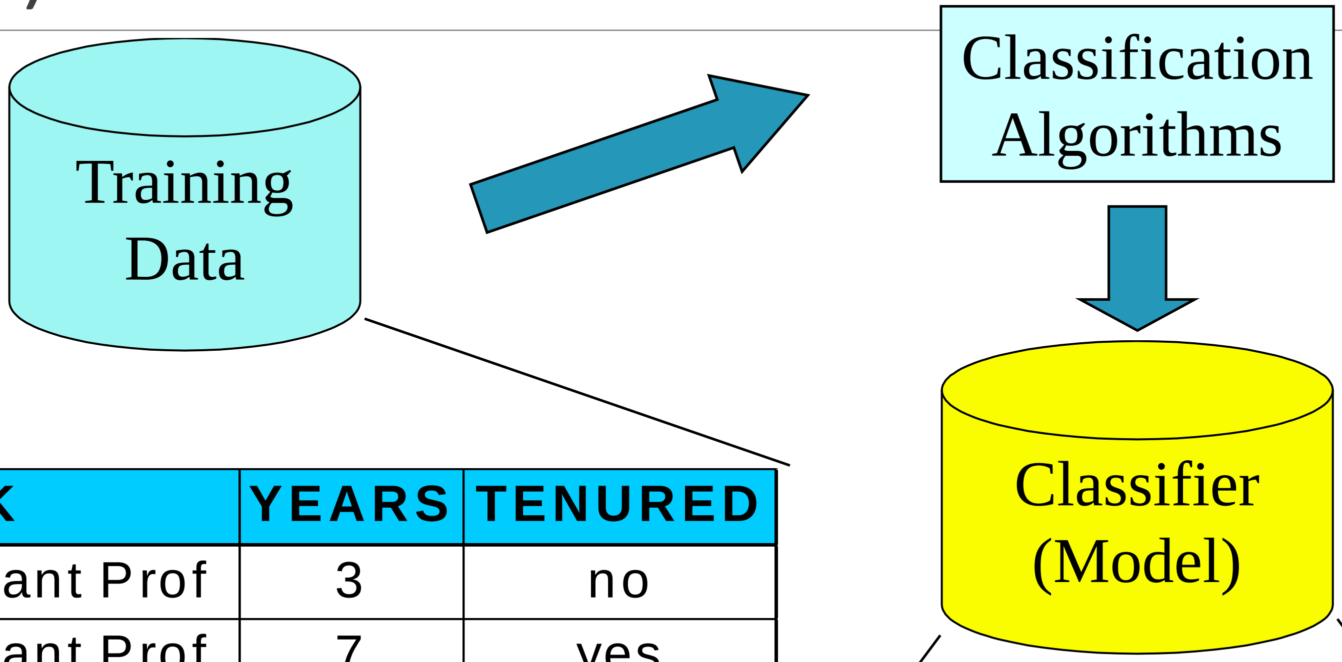
- Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
- The set of tuples used for model construction is training set
- The model is represented as classification rules, decision trees, or mathematical formulae

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 overfitting)
- If the accuracy is acceptable, use the model to classify new data

Note: If *the test set* is used to select models, it is called validation (test) set

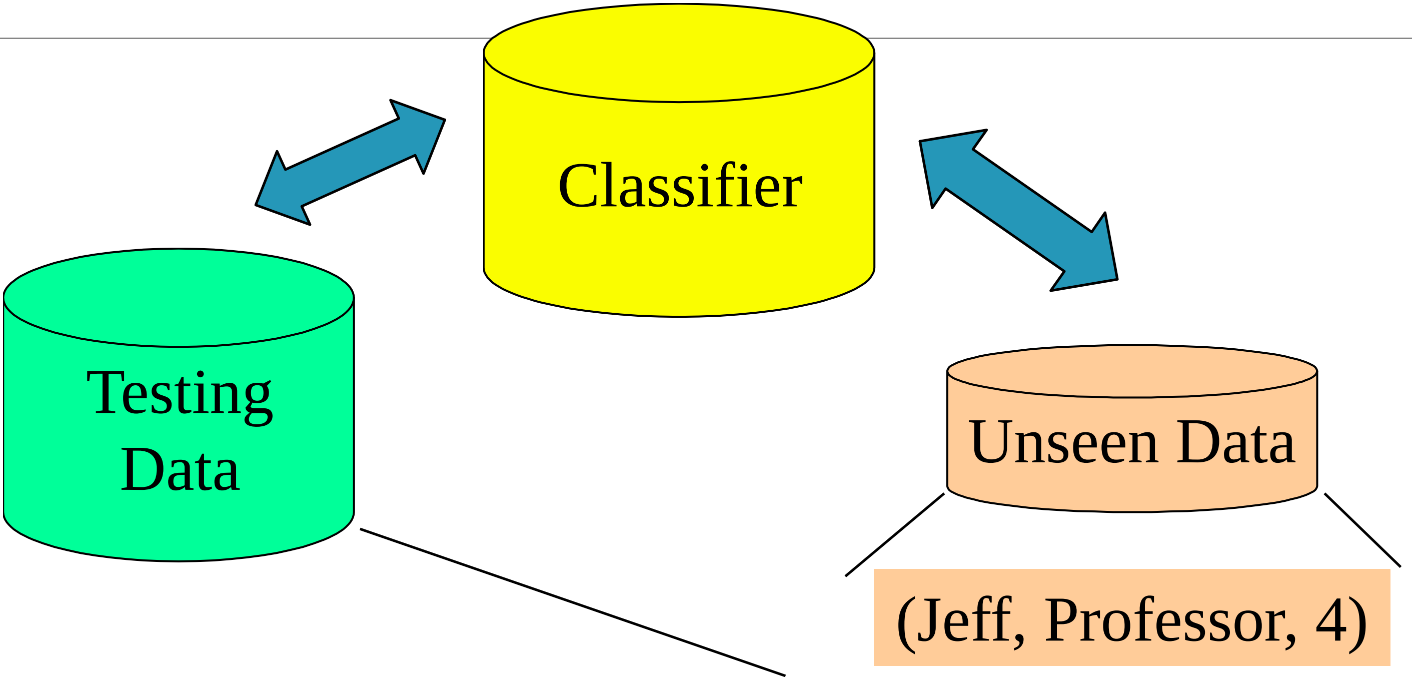
Process (1): Model Construction



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

IF rank = ‘professor’
OR years > 6
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

Tenured?

Yes

Chapter 8. Classification: Basic Concepts

Classification: Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

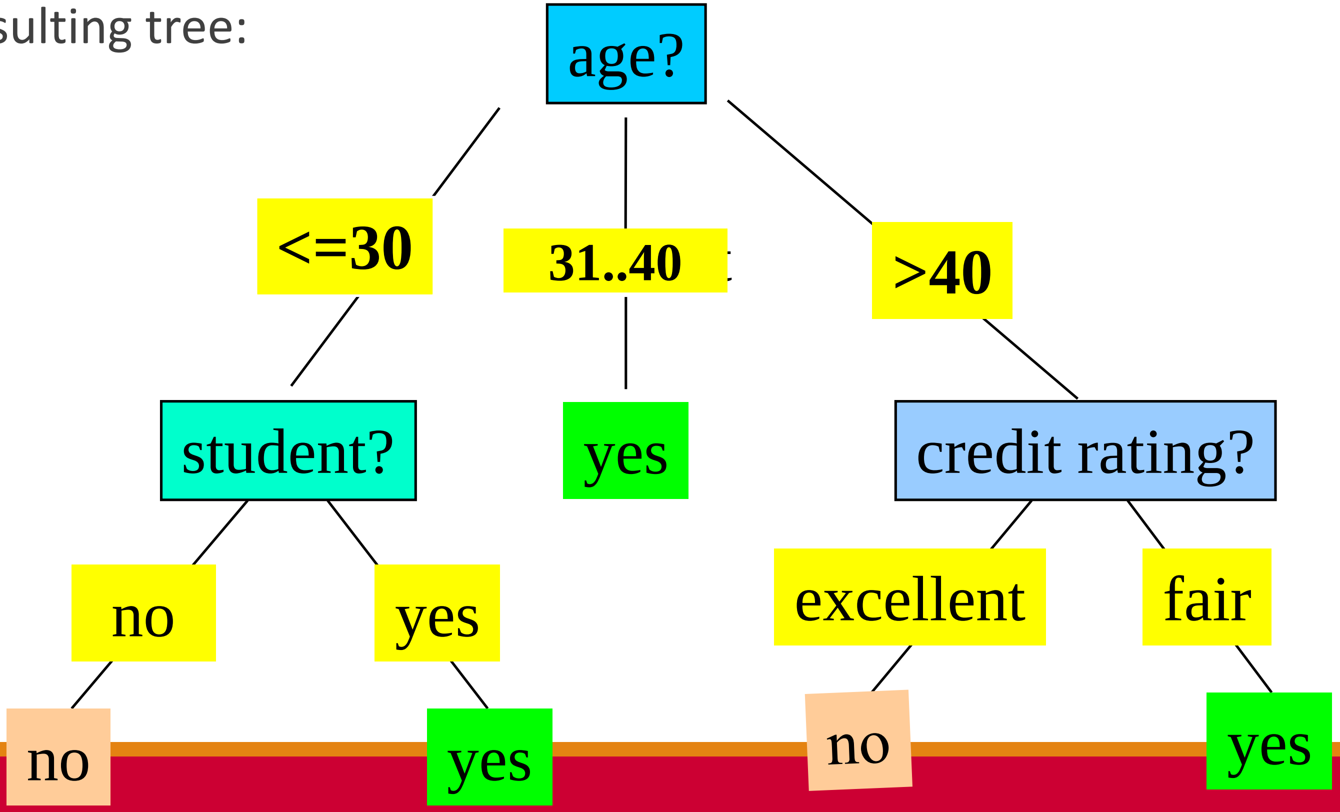
Model Evaluation and Selection

Techniques to Improve Classification Accuracy: Ensemble Methods

Summary

Decision Tree Induction: An Example

- Training data set: Buys_computer
- The data set follows an example of Quinlan's ID3 (Playing Tennis)
- Resulting tree:



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

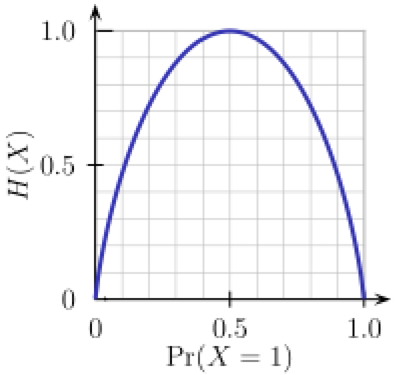
Algorithm for Decision Tree Induction

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 samples for a given node belong to the same class
- There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
- There are no samples left

Brief Review of Entropy

- Entropy (Information Theory)
  - A measure of uncertainty associated with a random variable
  - Calculation: For a discrete random variable Y taking m distinct values $\{y_1, \dots, y_m\}$,
    - $H(Y) = -\sum_{i=1}^{m} p_i \log(p_i)$, where $p_i = P(Y = y_i)$
  - Interpretation:
    - Higher entropy => higher uncertainty
    - Lower entropy => lower uncertainty
- Conditional Entropy
  - $H(Y|X) = \sum_x p(x) H(Y|X = x)$



m = 2

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 $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

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

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

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

- Information gained by branching on attribute A $Gain(A) = Info(D) - Info_A(D)$

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

Attribute Selection: Information Gain

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

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

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

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

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

Similarly,

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

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

Computing Information-Gain for Continuous-Valued Attributes

Let attribute A be a continuous-valued attribute

Must determine the *best split point* for A

- Sort the value A in increasing order
- Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
  - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
- The point with the *minimum expected information requirement* for A is selected as the split-point for A

Split:

- D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

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)

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

$$SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$$

- GainRatio(A) = Gain(A)/SplitInfo(A)
- GainRatio(income) = 0.029/1.557 = 0.019

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 of class *j* in *D*

If a data set *D* is split on A into two subsets D_1 and D_2, the *gini* index *gini*(*D*) is defined as

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

Reduction in Impurity: $\Delta gini(A) = gini(D) - gini_A(D)$

The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible splitting points for each attribute*)

Computation of Gini Index

Ex. D has 9 tuples in buys_computer = "yes" and 5 in "no"

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

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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right)Gini(D_1) + \left(\frac{4}{14}\right)Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$
$$= 0.443$$
$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

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

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 is decision tree induction popular?
- 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

RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
- Builds an AVC-list (attribute, value, class label)

Scalability Framework for RainForest

Separates the scalability aspects from the criteria that determine the quality of the tree

Builds an AVC-list**: AVC (Attribute, Value, Class_label)**

AVC-set (of an attribute *X*)

- Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated

AVC-group (of a node *n*)

- Set of AVC-sets of all predictor attributes at the node *n*

Rainforest: 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	2	3
31..40	4	0
>40	3	2

AVC-set on *income*

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

AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

BOAT (Bootstrapped Optimistic Algorithm for Tree Construction)

Use a statistical technique called *bootstrapping* to create several smaller samples (subsets), each fits in memory

Each subset is used to create a tree, resulting in several trees

These trees are examined and used to construct a new tree *T'*

- It turns out that *T'* is very close to the tree that would be generated using the whole data set together

Adv: requires only two scans of DB, an incremental alg.

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

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

Bayes' Theorem: Basics

Total probability Theorem: $P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$

Bayes' Theorem: $$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

- 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(H|**X**), (i.e., *posteriori probability):* the probability that the hypothesis holds given the observed data sample **X**
- P(H) (*prior probability*): the initial probability
  - E.g., **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (likelihood): the probability of observing the sample **X**, given that the hypothesis holds
  - E.g., Given that **X** will buy computer, the prob. that X is 31..40, medium income

Prediction Based on Bayes' Theorem

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

$$P(H|\mathbf{X}) = \frac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$$

Informally, this can be viewed as

posteriori = likelihood x prior/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: It requires initial knowledge of many probabilities, involving significant computational cost

Classification Is to Derive the Maximum Posteriori

Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n-D attribute vector $\mathbf{X} = (x_1, x_2, \ldots, x_n)$

Suppose there are *m* classes $C_1, C_2, \ldots, C_m$.

Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|\mathbf{X})$

This can be derived from Bayes' theorem

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

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

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

needs to be maximized

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 continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = ‘yes’

C2:buys_computer = ‘no’

Data to be classified:

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 Bayes 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")

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

Avoiding the Zero-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)

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 Bayes 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., hospitals: patients: Profile: age, family history, etc.

    Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
  - Dependencies among these cannot be modeled by Naïve Bayes Classifier

How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

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

Model Evaluation and Selection

Evaluation metrics: How can we measure accuracy? Other metrics to consider?

Use **validation 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, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*

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

A\P	C	¬C	
C	**TP**	**FN**	**P**
¬C	**FP**	**TN**	**N**
	P'	**N'**	**All**

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

Accuracy = (TP + TN)/All

Error rate: *1 – accuracy*, or

Error rate = (FP + FN)/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: exactness – what % of tuples that the classifier labeled as positive are actually positive

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

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

Perfect score is 1.0

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

Inverse relationship between precision & recall

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

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

$F_ß$: weighted measure of precision and recall

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

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

Classifier Evaluation Metrics: Example

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

- *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 boostrap**

- 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:

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

Estimating Confidence Intervals: Classifier Models M_1 vs. M_2

Suppose we have 2 classifiers, M_1 and M_2, which one is better?

Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$

These mean error rates are just *estimates* of error on the true population of *future* data cases

What if the difference between the 2 error rates is just attributed to *chance*?

- Use a **test of statistical significance**
- Obtain **confidence limits** for our error estimates

Estimating Confidence Intervals: Null Hypothesis

Perform 10-fold cross-validation

Assume samples follow a **t distribution** with *k–1* **degrees of freedom** (here, *k=10*)

Use **t-test** (or **Student's t-test**)

Null Hypothesis: M_1 & M_2 are the same

If we can **reject** null hypothesis, then

- we conclude that the difference between M_1 & M_2 is **statistically significant**
- Chose model with lower error rate

Estimating Confidence Intervals: t-test

If only 1 test set available: **pairwise comparison**

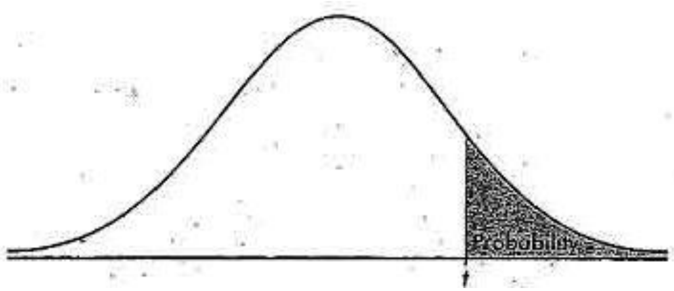
- For i^{th} round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M_1)_i$ and $err(M_2)_i$
- Average over 10 rounds to get $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- **t-test** computes **t-statistic** with *k-1* **degrees of freedom:**

$$t = \frac{\overline{err}(M_1) - \overline{err}(M_2)}{\sqrt{var(M_1 - M_2)/k}}$$

where

$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))\right]^2$$

Estimating Confidence Intervals: Table for t-distribution



Symmetric

Significance level, e.g., *sig = 0.05* or *5%* means M_1 & M_2 are *significantly different* for 95% of population

Confidence limit, *z = sig/2*

TABLE B: t-DISTRIBUTION CRITICAL VALUES

	Tail probability *p*											
df	.25	.20	.15	.10	.05	.025	.02	.01	.005	.0025	.001	.0005
1	1.000	1.376	1.963	3.078	6.314	12.71	15.89	31.82	63.66	127.3	318.3	636.6
2	.816	1.061	1.386	1.886	2.920	4.303	4.849	6.965	9.925	14.09	22.33	31.60
3	.765	.978	1.250	1.638	2.353	3.182	3.482	4.541	5.841	7.453	10.21	12.92
4	.741	.941	1.190	1.533	2.132	2.776	2.999	3.747	4.604	5.598	7.173	8.610
5	.727	.920	1.156	1.476	2.015	2.571	2.757	3.365	4.032	4.773	5.893	6.869
6	.718	.906	1.134	1.440	1.943	2.447	2.612	3.143	3.707	4.317	5.208	5.959
7	.711	.896	1.119	1.415	1.895	2.365	2.517	2.998	3.499	4.029	4.785	5.408
8	.706	.889	1.108	1.397	1.860	2.306	2.449	2.896	3.355	3.833	4.501	5.041
9	.703	.883	1.100	1.383	1.833	2.262	2.398	2.821	3.250	3.690	4.297	4.781
10	.700	.879	1.093	1.372	1.812	2.228	2.359	2.764	3.169	3.581	4.144	4.587
11	.697	.876	1.088	1.363	1.796	2.201	2.328	2.718	3.106	3.497	4.025	4.437
12	.695	.873	1.083	1.356	1.782	2.179	2.303	2.681	3.055	3.428	3.930	4.318
13	.694	.870	1.079	1.350	1.771	2.160	2.282	2.650	3.012	3.372	3.852	4.221
14	.692	.868	1.076	1.345	1.761	2.145	2.264	2.624	2.977	3.326	3.787	4.140
15	.691	.866	1.074	1.341	1.753	2.131	2.249	2.602	2.947	3.286	3.733	4.073
16	.690	.865	1.071	1.337	1.746	2.120	2.235	2.583	2.921	3.252	3.686	4.015
17	.689	.863	1.069	1.333	1.740	2.110	2.224	2.567	2.898	3.222	3.646	3.965
18	.688	.862	1.067	1.330	1.734	2.101	2.214	2.552	2.878	3.197	3.611	3.922
19	.688	.861	1.066	1.328	1.729	2.093	2.205	2.539	2.861	3.174	3.579	3.883
20	.687	.860	1.064	1.325	1.725	2.086	2.197	2.528	2.845	3.153	3.552	3.850
21	.686	.859	1.063	1.323	1.721	2.080	2.189	2.518	2.831	3.135	3.527	3.819
22	.686	.858	1.061	1.321	1.717	2.074	2.183	2.508	2.819	3.119	3.505	3.792
23	.685	.858	1.060	1.319	1.714	2.069	2.177	2.500	2.807	3.104	3.485	3.768
24	.685	.857	1.059	1.318	1.711	2.064	2.172	2.492	2.797	3.091	3.467	3.745
25	.684	.856	1.058	1.316	1.708	2.060	2.167	2.485	2.787	3.078	3.450	3.725
26	.684	.856	1.058	1.315	1.706	2.056	2.162	2.479	2.779	3.067	3.435	3.707
27	.684	.855	1.057	1.314	1.703	2.052	2.158	2.473	2.771	3.057	3.421	3.690
28	.683	.855	1.056	1.313	1.701	2.048	2.154	2.467	2.763	3.047	3.408	3.674
29	.683	.854	1.055	1.311	1.699	2.045	2.150	2.462	2.756	3.038	3.396	3.659
30	.683	.854	1.055	1.310	1.697	2.042	2.147	2.457	2.750	3.030	3.385	3.646
40	.681	.851	1.050	1.303	1.684	2.021	2.123	2.423	2.704	2.971	3.307	3.551
50	.679	.849	1.047	1.299	1.676	2.009	2.109	2.403	2.678	2.937	3.261	3.496
60	.679	.848	1.045	1.296	1.671	2.000	2.099	2.390	2.660	2.915	3.232	3.460
80	.678	.846	1.043	1.292	1.664	1.990	2.088	2.374	2.639	2.887	3.195	3.416
100	.677	.845	1.042	1.290	1.660	1.984	2.081	2.364	2.626	2.871	3.174	3.390
1000	.675	.842	1.037	1.282	1.646	1.962	2.056	2.330	2.581	2.813	3.098	3.300
∞	.674	.841	1.036	1.282	1.645	1.960	2.054	2.326	2.576	2.807	3.091	3.291
	50%	60%	70%	80%	90%	95%	96%	98%	99%	99.5%	99.8%	99.9%
	Confidence level *C*											

Estimating Confidence Intervals: Statistical Significance

Are M_1 & M_2 **significantly different**?

- Compute *t.* Select *significance level* (e.g. *sig = 5%)*
- Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
- t-distribution is symmetric: typically upper % points of distribution shown → look up value z for **confidence limit** *sig/2* (here, 0.025)
- **If t > z or t < -z**, then t value lies in rejection region:
  - **Reject null hypothesis** that mean error rates of M_1 & M_2 are not the same
  - Conclude: statistically significant difference between M_1 & M_2
- **Otherwise**, conclude that any difference is **chance**

Model Selection: ROC Curves

ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models

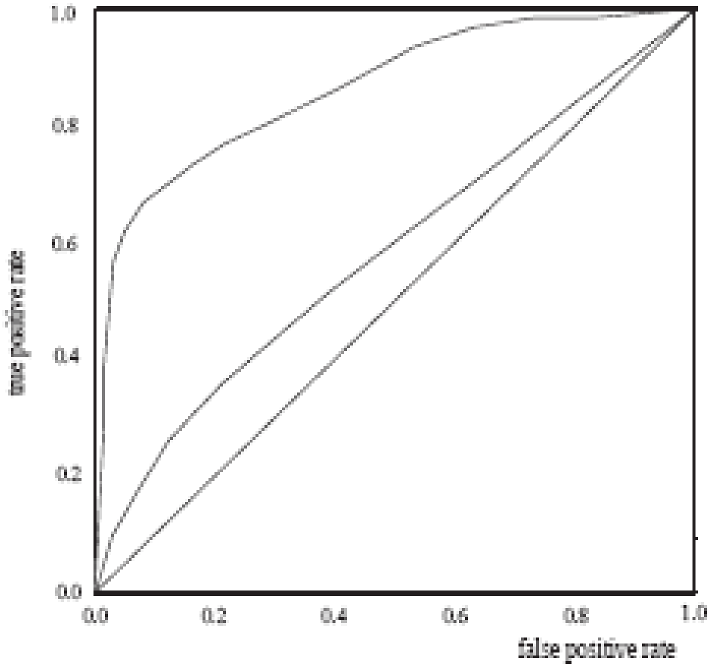
Originated from signal detection theory

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 tuples 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

Chapter 8. Classification: Basic Concepts

Classification: Basic Concepts

Decision Tree Induction

Bayes Classification Methods

Rule-Based Classification

Model Evaluation and Selection

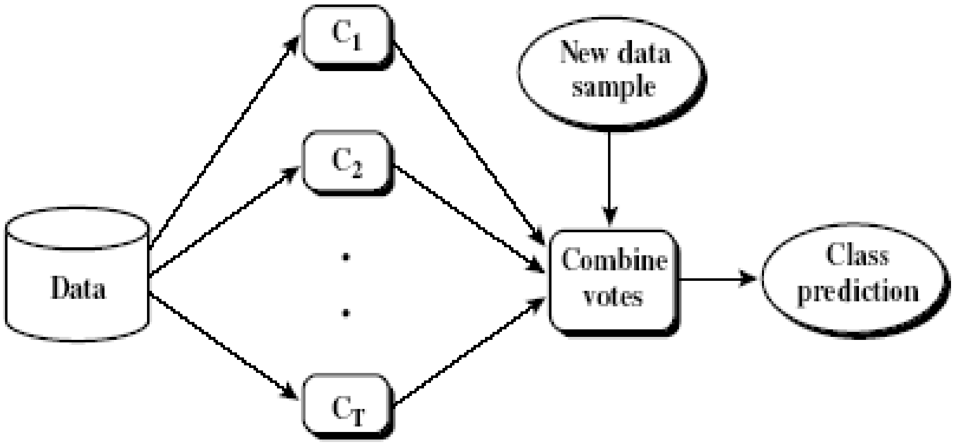
Techniques to Improve Classification Accuracy: Ensemble Methods

Summary

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
- 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., bootstrap)
- 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 significantly better than a single classifier derived from D
- For noise data: not considerably worse, more robust
- Proven 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, $(\mathbf{X_1}, y_1), \ldots, (\mathbf{X_d}, y_d)$

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:

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

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 combining a series of individual models.

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