Chapter VII :

Classification

Outline

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

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

Classification

- A form of data analysis that extracts models describing important data classes
  - Models is called classifiers, predict categorical (discrete, unordered) class labels
- Given a database of records, each with a class label, a classifier generates a concise and meaningful description for each class that can be used to classify subsequent records

Application

- 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
    - Ex. predicts unknown or missing values
- Typical applications
  - Credit/loan approval
  - Weather prediction
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

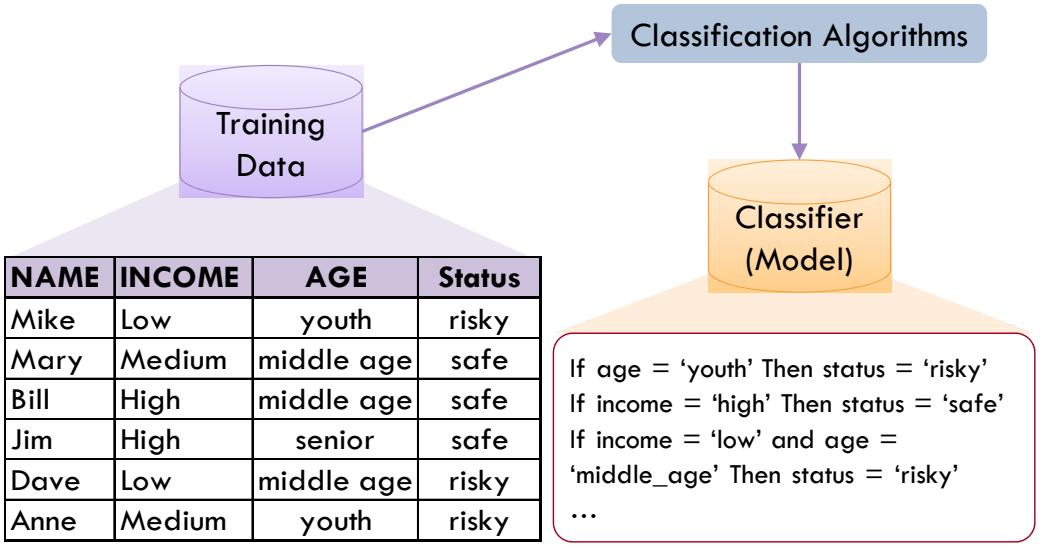
General Approach to Classification

- The general approach to classification as a two-step process
  - The first step : Model construction
    - Build a classification model based on previous data
  - The second step : Model usage
    - Determine the model with acceptable accuracy
    - Use the model to classify new data

Model construction

- 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 individual tuples in a training set are referred to as **training samples**
  - If the class label is provided, this step is known as **supervised learning**, otherwise called **unsupervised learning** (or clustering)
  - The model is represented as classification rules, decision trees, or mathematical formulae

Model construction

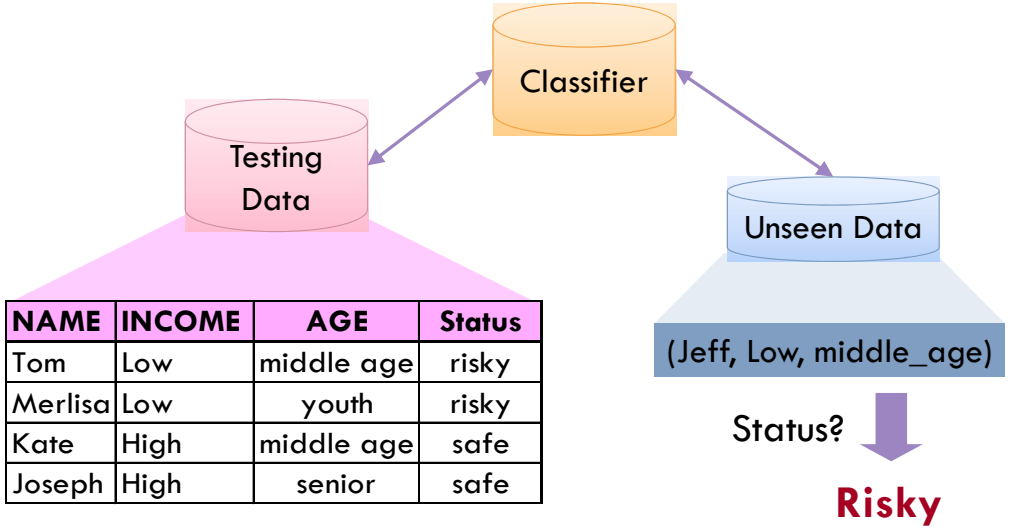


NAME	INCOME	AGE	Status
Mike	Low	youth	risky
Mary	Medium	middle age	safe
Bill	High	middle age	safe
Jim	High	senior	safe
Dave	Low	middle age	risky
Anne	Medium	youth	risky

Model usage

- Model usage: for classifying future or unknown objects
  - First, estimate the predictive accuracy of the model
  - The holdout method is a technique that uses a test set of class-labeled samples which are randomly selected and are independent of the training samples
  - The accuracy of a model on a given test set is the percentage of test set correctly classified by model
  - If the accuracy of the model were estimate based on the training data set -> the model tends to **overfit** the data
  - If the accuracy of the model is considered acceptable, the model can be used to classify future data tuples or objects for which the class label is unknown

Model usage



NAME	INCOME	AGE	Status
Tom	Low	middle age	risky
Merlisa	Low	youth	risky
Kate	High	middle age	safe
Joseph	High	senior	safe

The criteria to evaluate classification methods

- Predictive accuracy
  - An ability of the model to correctly predict the class label of new or unseen data
- Speed
  - The computation costs involved in generating and using the model
- Robustness
  - An ability of the model to make correct predictions given noisy data or data with missing values
- Scalability
  - An ability to construct the model efficiently given large amounts of data
- Interpretability
  - The level of understanding and insight that is provided by the model

Decision tree

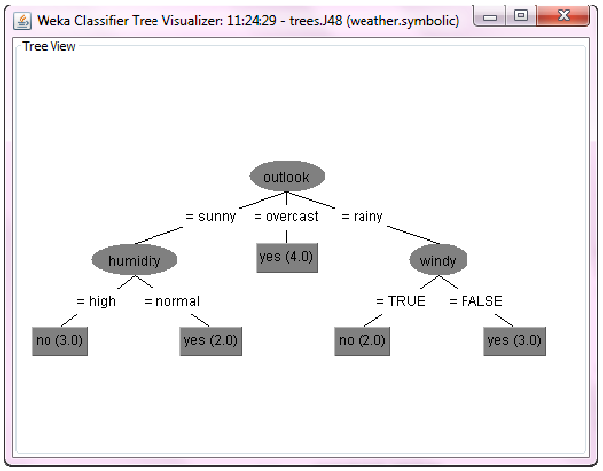
- Decision tree induction is the learning of decision trees from class-labeled training tuples
- A decision tree is a flowchart-like tree structure
  - each internal node (non-leaf node) denotes a test on an attribute
  - each branch represents an outcome of the test
  - each leaf node (or terminal node) holds a class label
  - root node is the topmost node in a tree

Decision tree : Example

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Input

Output



Advantages of decision tree

- Can handle multidimensional data
- Easy to interpret by humans
- The learning and classification steps of decision tree induction are simple and fast
- Have a good accuracy

Decision tree induction

- The basic algorithm for decision tree induction is a greedy algorithm that constructs decision tree in a top-down recursive divide-and-conquer manner
- The basic strategy:
  - The tree starts as a single node representing the training samples
  - If the sample are all of the same class then the node becomes a leaf and is labeled with the class
  - Otherwise, the algorithm uses an entropy-based measure known as *information gain* for selecting the attribute that best separate the samples into individual classes → this attribute becomes the test attribute at the node
  - A branch is created for each known value of the test attribute

Decision tree induction (cont.)

- The basic strategy: (cont.)
  - The algorithm uses the same process recursively to form a decision tree for the samples at each partition
  - The recursive partitioning stop only when any one of the following conditions is true
    - All sample for a given node belong to the same class
    - There are no remaining attribute on which the samples may be further partitioned → majority vote is employed
    - There are no sample for the brach *test-attribute*=a_i → majority vote is employed

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

- The **information gain** measure is used to select the test attribute at each node in the tree
- It is referred to as an attribute selection measure or measure of the **goodness of split**
- The attribute with the **highest information gain is chosen as the test attribute for the current node**

- Let S be a set consisting of s data samples, the class label attribute has m distinct value defining m distinct classes, C_i (for $i=1,...,m$)
- Let s_i be the no of sample of S in class C_i
- The expected information >> $I(s_1,s_2,...,s_m) = -\sum_{i=1}^{m} \frac{s_i}{s} \log_2 \frac{s_i}{s}$

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

- Find an entropy of attribute A
  - Let A have v distinct value $\{a_1,a_2,...,a_v\}$ which can partition S into $\{S_1,S_2,....S_v\}$
  - For each S_j, s_{ij} is the number of samples S_j of class C_i
  - The entropy or expected information based on attribute A is given by

$$E(A)=\sum_{j=1}^{v} \frac{s_{1j}+...+s_{mj}}{s} I(s_{1j},...,s_{mj})$$

- $Gain(A)=I(s_1,s_2,...s_m)-E(A)$
- The algorithm computes the information gain of each attribute
- The attribute with the highest information gain is chosen as the test attribute for the given set S

Once upon a time : Chapter V_Lec

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

- Class y : play = 'yes'
- Class n : play = 'no'
- I(y,n) = 0.94

$-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)$

- E(outlook)=0.69

$5/14*(0.97)+4/14*(0)+5/14*(0.97)$

0.94-0.69

- Gain(outlook)=0.25
- Gain(temp) = 0.03
- Gain(humidity) = 0.15
- Gain(windy) = 0.05

Construct decision tree

- After obtain all gain then sort gain of each attribute order by DESC

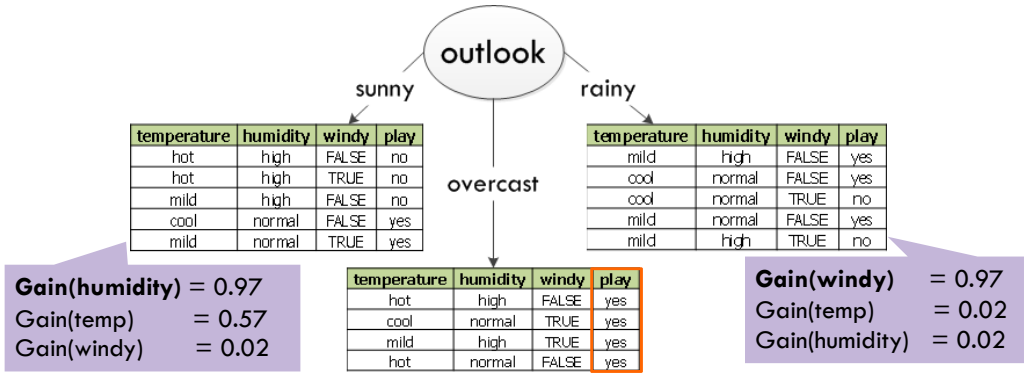
Gain(outlook) = 0.25
Gain(temp) = 0.03
Gain(humidity) = 0.15
Gain(windy) = 0.05

Gain(outlook) = 0.25
Gain(humidity) = 0.15
Gain(windy) = 0.05
Gain(temp) = 0.03

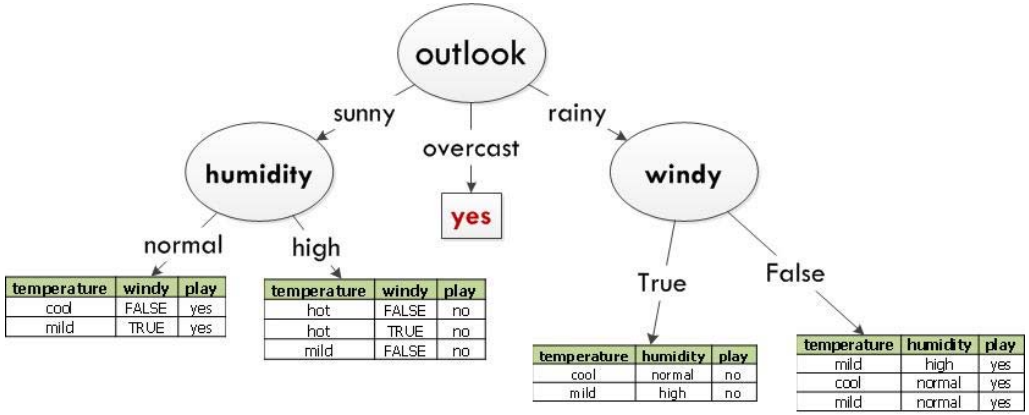
- Construct root of decision tree followed by the highest information gain >> **outlook**

Construct decision tree

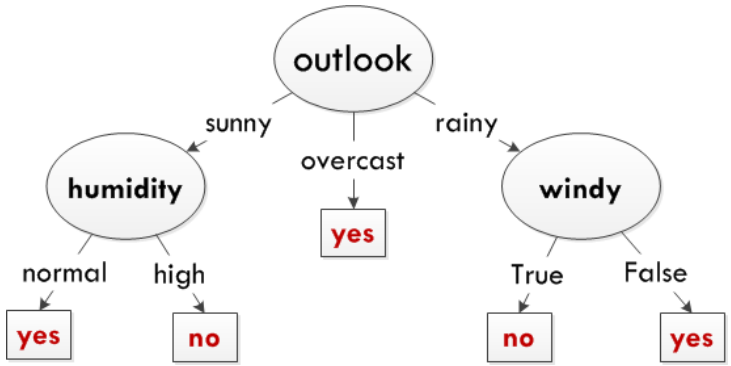
Find information gain for each attribute, then select the attribute with the highest information gain as a test node



Construct decision tree (cont.)



Construct decision tree (cont.)



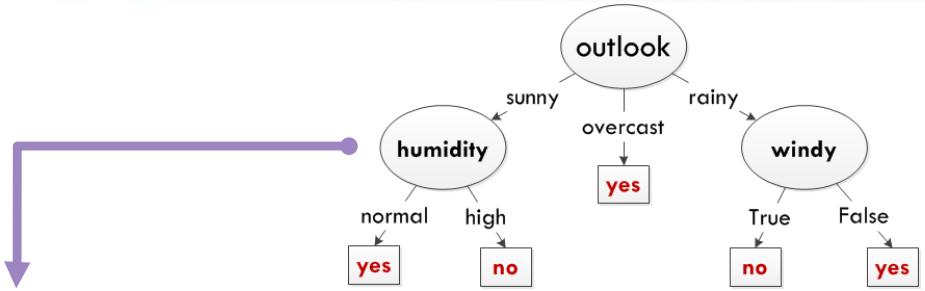
Overfitting and Tree Pruning

- 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"

Rule-Based Classification

- The knowledge represented in decision tree can be extracted and represented in the form of classification **IF-THEN** rules
- One rule is created for each path from the root to a leaf node
- **IF-part**: a conjunction of attribute value along a given path
- **THEN-part**: the class prediction the leaf node holds
- **IF-THEN rules** may be easier for humans to understand, particularly if the given tree is very large

Example…Rule extraction from decision-tree



- IF outlook=overcast THEN play=yes
- IF outlook=sunny and humidity=high THEN play=no
- IF outlook=sunny and humidity=normal THEN play=yes
- IF outlook=rainy and windy=false THEN play=yes
- IF outlook=rainy and windy= true THEN play=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

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)

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

  - GainRatio(A) = Gain(A)/SplitInfo(A)

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

- Ex. $SplitInfo_{income}(D) = -\frac{4}{14} \times \log_2\left(\frac{4}{14}\right) - \frac{6}{14} \times \log_2\left(\frac{6}{14}\right) - \frac{4}{14} \times \log_2\left(\frac{4}{14}\right) = 1.557$
  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

Gini index : (CART)

- All attributes are assumed continuous-valued
- Assume there exist several possible split values for each attribute
- 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 provides the smallest ginisplit(T) is chosen to split the node (need to enumerate all possible splitting points for each attribute)

Example…Gini index

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

- Gini index of the entire set
  $= 1-(9/14)^2-(5/14)^2$
  $= 0.459$
- Gini (split) = $5/14(gini_{sunny}) + 4/14(gini_{overcast})+5/14(gini_{rainy})$
- $Gini_{(sunny)} = 1 - (2/5)^2 - (3/5)^2 = 0.48$
- $Gini_{(overcast)} = 1 - (4/4)^2 - 0 = 0$
- $Gini_{(rainy)} = 1 - (3/5)^2 - (2/5)^2 = 0.48$

∴ split = $5/14(0.48) + 4/14(0) + 5/14(0.48)$
$= 0.34$

Comparing Attribute Selection Measures

- **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

Bayes Classification Methods

- Bayesian classifier are statistical classifier that used for predict class membership probabilities (probability that a given sample belong to a particular class)
- Bayesian classifier is based on Bayes' theorem
- The simple Bayesian classifier known as naïve Bayesian classifier
- Many studies found naïve Bayesian classifier to be comparable in performance with decision tree and neural network classifier

Bayesian Theorem

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

- Classification is to determine P(H|**X**), (*posteriori probability*), the probability that the hypothesis holds given the observed data sample **X**
- X : data tuple
- H : hypothesis
  - ex. data tuple X belongs to a specified class C
- P(H) (*prior probability*), the initial probability
  - ex. **X** will buy computer, regardless of age, income, …
- P(**X**): probability that sample data is observed
- P(**X**|H) (likelyhood): the probability of observing the sample **X**, given that the hypothesis holds
  - ex. Given that **X** will buy computer, the prob. that X is 31..40, medium income

Naïve Bayesian Classification

Step 1.

- Each data sample is represented by an *n*-dimensional feature
- $X=(x_1,x_2,\ldots,x_n)$ from n attributes, repectively, $A_1,A_2,\ldots,A_n$

$A_1,A_2,\ldots,A_n$

Outlook	Temperature	Humidity	Windy	Play
Rainy	Mild	Normal	False	Y
Overcast	Cool	Normal	True	Y
Sunny	Hot	High	True	N
Overcast	Hot	High	False	Y
X → Sunny	Hot	High	False	

X=(sunny, hot, high, false) unknown class

Naïve Bayesian Classification (cont.)

Step 2.

- Suppose that there are *m* classes, $C_1,C_2,\ldots C_m$
  - Given an unknown data sample *X*
  - The classifier will predict that *X* belongs to the class having the highest posterior probability, condition on *X*
  - The naïve Bayesian will assigns an unknown *X* to class C_i if and only if

$$P(C_i \mid X) > P(C_j \mid X) \text{ for } 1 \le j \le m, j \ne I$$

    - That is, it will find the maximum posterior probability among $P(C_1 \mid X)$, $P(C_2 \mid X)$, ….,$P(C_m \mid X)$
  - The class C_i for which $P(C_i \mid X)$ is maximized is called the maximum posteriori hypothesis

Naïve Bayesian Classification (cont.)

m =2

C_1: Play="Y" and C_2: Play="N"

$A_1,A_2,\ldots,A_n$

Outlook	Temperature	Humidity	Windy	Play
Training Samples: Rainy	Mild	Normal	False	Y
Overcast	Cool	Normal	True	Y
Sunny	Hot	High	True	N
Overcast	Hot	High	False	Y
X → Sunny	Hot	High	False	

X=(sunny,hot,high,false) unknown class

If (Play="Y" | X) > (Play='N'| X)

Naïve Bayesian Classification (cont.)

Step 3.

- By Bayes theorem >> $P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$
  - As P(X) is constant for all classes
    - only $P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$ need to be maximized
  - If $P(C_i)$ are not known,
    - it is commonly assume that $P(C_1) = P(C_2) = \ldots = P(C_m)$, therefore only $P(X|C_i)$ need to be maximized
  - Otherwise, we maximize $P(X|C_i)\,P(C_i)$,
    - where $P(C_i) = \frac{S_i}{S}$ (S_i: # of training sample of Class Ci; S: Total # of training sample)

Naïve Bayesian Classification (cont.)

m =2

C_1: Play="Y" and C_2: Play="N"

$A_1, A_2, \ldots, A_n$

Outlook	Temperature	Humidity	Windy	Play
Rainy	Mild	Normal	False	Y
Overcast	Cool	Normal	True	Y
Sunny	Hot	High	True	N
Overcast	Hot	High	False	Y
Sunny	Hot	High	False	

Training Samples

X →

X=(sunny, hot, high, false) unknown class

(Play="Y"|X) = P(X|Play="Y") P(Play="Y")
= P(X|Play="Y") (3/4)

(Play="N"|X) = P(X|Play="N") P(Play="N")
= P(X|Play="N") (1/4)

Naïve Bayesian Classification (cont.)

Step 4.

- Given a data sets with many attribute → it is expensive to compute $P(X|C_i)$
  - To reduce computation, naïve made an assumption of class conditional independence (there are no dependence relationship among the attribute)

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

If A_k is **categorical** values → $P(x_k|C_i) = \frac{S_{ik}}{S}$ (S_{ik}: # of training sample of Class C_i having the value x_k for A_k; S: Total # of training sample belong to class Ci)

If A_k is **continuous** values → perform Gaussian distribution (not focus in this class) with a mean μ and standard deviation σ

$$P(\mathbf{X}|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i}) \;;\; g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Naïve Bayesian Classification (cont.)

Step 5.

- In order to classify an unknown X, $P(X|C_i)\,P(C_i)$ is evaluated for each class C_i
  - Sample X is assign to the class C_i for which $P(X|C_i)\,P(C_i)$ is the maximum

Example : Predicting a class label using naïve Bayesian Classifier

RID	age	income	student	Credit_rating	Class:buys_computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	31…40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31…40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	no
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31…40	Medium	No	Excellent	Yes
13	31…40	High	Yes	Fair	Yes
14	>40	medium	no	Excellent	No
15	<=30	medium	yes	fair	

Example : Predicting a class label using naïve Bayesian Classifier (cont.)

- C_1: buys_computer ="Yes" , C_2: buys_computer ="No"
- The unknown sample we wish to classify is
  - X=(age="<=30", income="medium", student="yes", credit_rating="fair")
- We need to maximize $P(X|C_i)\,P(C_i)$, for i=1,2

i=1

P(X | buys_computer="yes") P(buys_computer="yes")

P(buys_computer="yes") = 9/14 = 0.64
P(X | buys_computer="yes") = P(*age*="<=30" | buys_computer="yes") *
P(*income*="medium" | buys_computer="yes") *
P(*student*="yes" | buys_computer="yes") *
P(credit_rating="fair" | buys_computer="yes")
= 2/9 * 4/9 * 6/9 * 6/9
= 0.044

P(X | buys_computer="yes") P(buys_computer="yes") = 0.64*0.044 = 0.028

Example : Predicting a class label using naïve Bayesian Classifier (cont.)

i=2

P(X | buys_computer="no") P(buys_computer="no")

P(buys_computer="no") = 5/14 = 0.36
P(X | buys_computer="no") = P(*age*="<=30" | buys_computer="no") *
P(*income*="medium" | buys_computer="no") *
P(*student*="yes" | buys_computer="no") *
P(credit_rating="fair" | buys_computer="no")
= 3/5 * 2/5 * 1/5 * 2/5
= 0.019

P(X | buys_computer="yes") P(buys_computer="yes") = 0.36*0.019 = 0.007

Therefore,

X=(age="<=30", income="medium", student="yes", credit_rating="fair")
should be in class buys_computer= **"yes"**

Naïve Bayesian Classifier: Advantages vs. Disadvantages

- **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
    - Ex. - Hospitals: patients: Profile: age, family history, etc.
      - Symptoms: fever, cough etc.
      - Disease: lung cancer, diabetes, etc.
  - Dependencies among these cannot be modeled by Naïve Bayesian Classifier

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

Metrics for Evaluating Classifier Performance

- Accuracy
- Error Rate
- Sensitivity
- Specificity
- Precision
- Recall
- F-measures

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	play=yes	play=no	Total
play = yes	6954	46	7000
play = no	412	2588	3000
Total	7366	2634	10000

- Given *m* classes
- An entry, $CM_{i,j}$ 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

Confusion matrix : 4 terms

- True positives (TP): These refer to the positive tuples that were correctly labeled by the classifier
  - Let TP be the number of true positives
- True negatives (TN): These are the negative tuples that were correctly labeled by the classifier
  - Let TN be the number of true negatives
- False positives (FP): These are the negative tuples that were incorrectly labeled as positive (e.g., tuples of class buys_computer = no for which the classifier predicted buys_computer = yes)
  - Let FP be the number of false positives
- False negatives (FN): These are the positive tuples that were mislabeled as negative (e.g., tuples of class buys_computer = yes for which the classifier predicted buys_computer = no)
  - Let FN be the number of false negatives

Classifier Evaluation Metrics: (cont.)

- Classifier Accuracy(recognition rate)
  - Percentage of test set tuples that are correctly classified
  - Accuracy = (TP + TN)/All
- Error rate
  - Misclassification rate of a classifier
  - Error rate = 1 – accuracy, or Error rate = (FP + FN)/All
- Sensitivity
  - True Positive recognition rate
  - Sensitivity = TP/P
- Specificity
  - True Negative recognition rate
  - Specificity = TN/N

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

Classifier Evaluation Metrics: (cont.)

- **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?

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

- Perfect score is 1.0
- Inverse relationship between precision & recall
- ***F* measure (F_1 or *F*-score):** harmonic mean of precision and recall

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

- F_β: weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

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

Estimating Classifier Accuracy

- Holdout method
  - The given data are randomly partitioned into 2 independent sets, *a training data* and *a test set*
  - Typically, 2/3 are training set, 1/3 are test set
  - Training set: used to derive the classifier
  - Test set: used to estimate the derived classifier
- Random subsampling
  - The variation of the hold out method
  - Repeat hold out method *k* times
  - The overall accuracy estimate is the average of the accuracies obtained from each iteration

Estimating Classifier Accuracy (cont.)

- *k*-fold cross validation
  - The initial data are randomly partitioned into *k* equal sized subsets ("folds") $S_1, S_2, \ldots, S_k$
  - Training and testing are performed *k* times
  - In iteration *i*, the subset S_i is the test sets, and the remaining subset are collectively used to train the classifier
  - Accuracy = $\frac{\text{overall no. of correct classifiers from the k iterations}}{\text{total no. of samples in the initial data}}$

Estimating Classifier Accuracy (cont.)

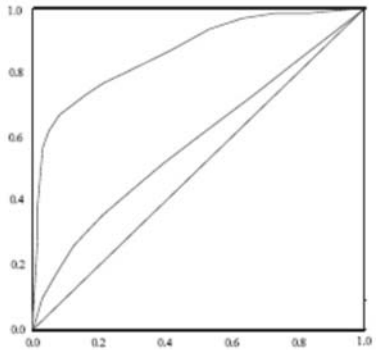
- Bootstrap
  - Bootstrap method samples the given training tuples uniformly with replacement
  - Each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
    - For instance, imagine a machine that randomly selects tuples for our training set
    - In sampling with replacement, the machine is allowed to select the same tuple more than once

Comparing classifiers : Model Selection by ROC Curves

- ROC : Receiver Operating Characteristics
- Visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate of one or more classifiers
- 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

ROC Curve

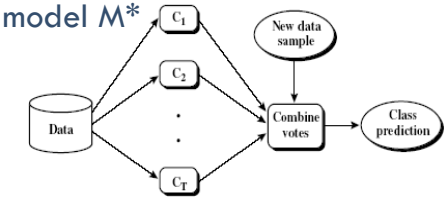
- Vertical axis represents the true positive rate
- Horizontal axis represent the false positive rate
- The plot also shows a diagonal line



- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model
- A model with perfect accuracy will have an area of 1.0

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

- 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
  - 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

Random Forest

- 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 one
  - ex. 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