Introduction to Data Mining

Lecture3 Classification

Jun Huang

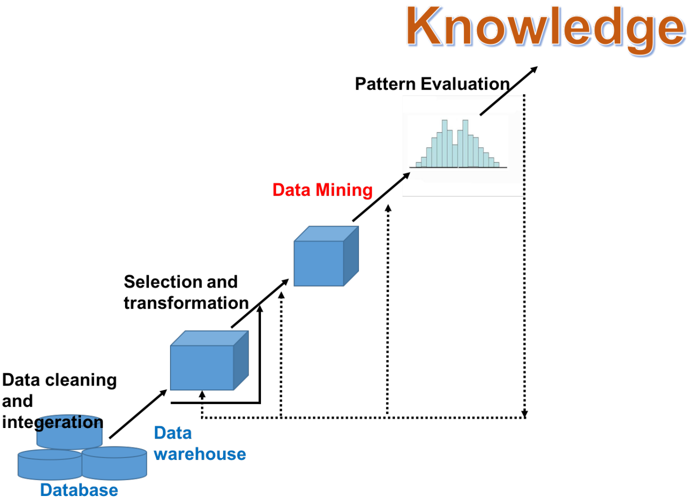
Anhui University of Technology

Spring 2018

huangjun_cs@163.com

KDD Process

Data Mining-Core of Knowledge discovery process



Classification and Prediction

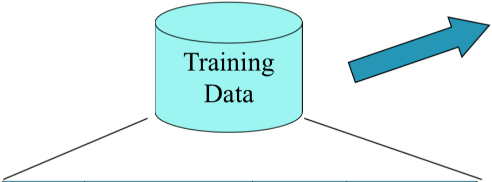
- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Other classification methods
- Prediction
- Accuracy and error measures
- Summary

Classification vs. Prediction

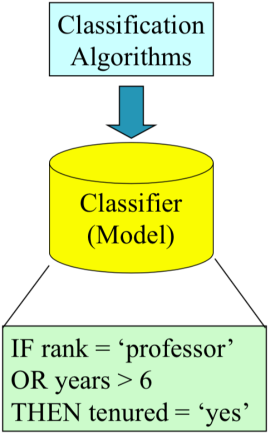
- Classification
  - Predict categorical class labels (discrete or nominal)
  - Classify records (construct a model) based on the training set and the class labels in a classifying attribute and then use the rules to classify new records
- Prediction
  - Model continuous-valued functions, i.e., predict unknown or missing values
- Typical applications
  - Credit approval
  - Target marketing
  - Medical diagnosis
  - Fraud detection
  - Intrusion detection

- **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 traning 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 tranining set, otherwise over-fitting will occur
  - If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known

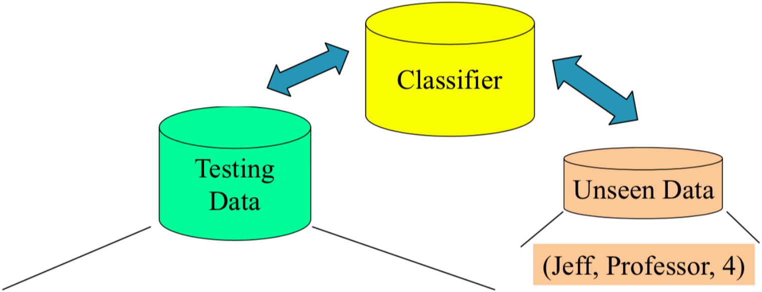
Process(1):Model Construction



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



Process(2): Using the model in Classification



NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

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, establish classes or clusters in the data

- Data Cleaning
  - Preprocess data in order to reduce noise and handle missing values
- Relevance analysis (Feature Selection)
  - Remove the irrelevant or redundant attributes
- Data Transformation
  - Generalize and/or nomalize data

- Accuracy
  - Classifier Accuracy: Predicting Class Label
  - Predictor Accuracy: Guessing value of predicted attributes
- Speed
  - Time to construct the model (training time)
  - Time to use the model (classification or prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
  - Understanding and insigh provided by the model
- Other measures, e.g., goodness of rules, such as decision tree size or compactness of classification rules

Classification by Decision Tree Induction

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

Classification by Decision Tree Induction

Generate_Decision_Tree(D,attribute_list)

create a node N;

if tuples in D are all of the same class C, **then**

  return N as a leaf node labeled with the class C;

if attribute_list is empty, **then**

  return N as a leaf node labeled with the majority class in D;// majority voting

apply Attribute_selection_method(D, attribute_list) to find the highest **information gain**;

label node N with test-attribute;

for each value a_i of test-attribute

  Grow a branch from node N for test-attribute $= a_i$;

  Let s_i be the set of samples in D for which test-attribute $= a_i$;

  if s_i is empty **then**

    attach a leaf labeled with the majority class in D to node N;

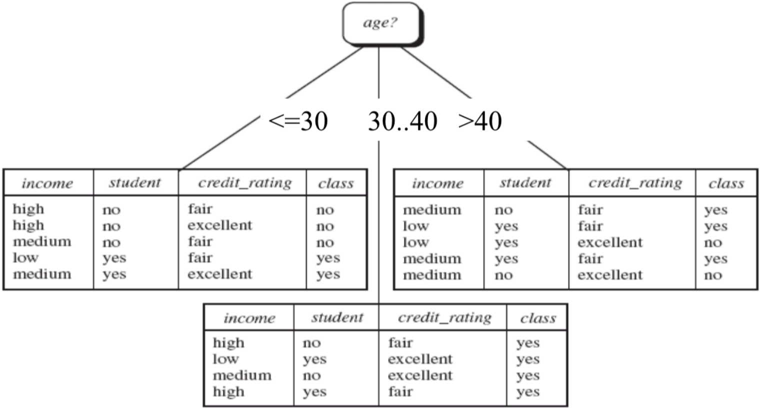
  else attach the node returned by **Generate_Decision_Tree**(s_i,**attribute_list**) to node N;

end for

Decision Tree Induction: Training Dataset

age	income	student	credit_rating	buy_computer
<= 30	high	no	fair	no
<= 30	high	no	excellent	no
31 - 34	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

Decision Tree

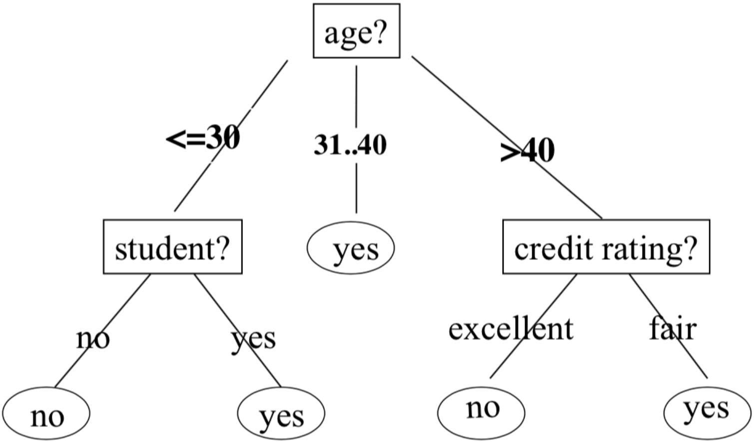


income	student	credit_rating	class
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	class
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

income	student	credit_rating	class
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

Output: A Decision Tree for "buys_computer"



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 partioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain, Gini index)
- 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

Information Gain (ID3/C4.5)

- Select the attribute with the **highest information gain**

$$
\begin{aligned}
Info(\mathcal{D}) &= -\sum_{i=1}^{m} p_i \log_2(p_i) \\
Info_A(\mathcal{D}) &= \sum_{j=1}^{v} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Info(\mathcal{D}_j)
\end{aligned}
$$

- where the dataset has m class labels, and the attribute A has v different values
- Assume there two classes, P and N
  - Let the set of examples $\mathcal{D}$ contain p elements of class P and n elements of class N
  - The amount of information, needed to classify sample

$$Info(\mathcal{D}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that attribute A have v distinct values $\{a_1, a_2, ..., a_v\}$
- Training set $\mathcal{D}$ will be partitioned into sets $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_v\}$
  - If $\mathcal{D}_i$ contains p_i examples of P and n_i examples of N, the entropy, or the expected information based on partitioning into subsets by attribute A is

$$Info_A(\mathcal{D}) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} Info(\mathcal{D}_i)$$

- Information gain of A

$$Gain(A) = Info(\mathcal{D}) - Info_A(\mathcal{D})$$

Attribute Selection by Information Gain Computation

- class P: buys_computer = "yes"
- class N: buys_computer = "no"
- $Info(\mathcal{D}) = 0.940$
- Compute the entropy for age:

age	p_i	n_i	$Info_{\text{age}}(\mathcal{D}_i)$
<= 30	2	3	0.971
30...40	4	0	0
>40	3	2	0.971

- $Info_{\text{age}}(\mathcal{D}) = \frac{5}{14} Info_{\text{age}}(\mathcal{D}_1) + \frac{4}{14} Info_{\text{age}}(\mathcal{D}_2) + \frac{5}{14} Info_{\text{age}}(\mathcal{D}_3)$
- Hence $Gain(age) = Info(\mathcal{D}) - Info_{\text{age}}(\mathcal{D}) = 0.246$

Exercise

1. Please calculate the information gain of *income, student, and credit_rating*, respectively.

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

Problem of Infomation Gain

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

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

$$Gain(A) = Info(\mathcal{D}) - Info_A(\mathcal{D})$$

What is disadvantage(s) of Information Gain?

Problem of Infomation Gain

$$
\begin{aligned}
Info(\mathcal{D}) &= -\sum_{i=1}^{m} p_i \log_2(p_i) \\
Info_A(\mathcal{D}) &= \sum_{j=1}^{v} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} Info(\mathcal{D}_j) \\
Gain(A) &= Info(\mathcal{D}) - Info_A(\mathcal{D})
\end{aligned}
$$

What is disadvantage(s) of Information Gain?

- Attribute is selected with the highest information gain
- Information gain measure is biased towards attributes with a large number of values

Gain Ratio for Attribute Selection(C4.5)

- C4.5(a successor of ID3), uses gain ratio to overcome the problem (normalization to information gain)

$$SplitInfo_A(\mathcal{D}) = -\sum_{j=1}^{v} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \log_2(\frac{|\mathcal{D}_j|}{|\mathcal{D}|})$$

- $SplitInfo_A(\mathcal{D}) = -\frac{4}{14} \times \log_2(\frac{4}{14}) - \frac{6}{14} \times \log_2(\frac{6}{14}) - \frac{4}{14} \times \log_2(\frac{4}{14}) = 0.926$
- GainRatio(A) = Gain(A)/SplitInfo(A), e.g., gain_ratio (income) = 0.029/0.926 = 0.031
- The attribute with the **maximum gain ratio** is selected as the splitting attribute

Problem of Gain Ratio

$$SplitInfo_A(\mathcal{D}) = -\sum_{j=1}^{v} \frac{|\mathcal{D}_j|}{|\mathcal{D}|} \times \log_2(\frac{|\mathcal{D}_j|}{|\mathcal{D}|})$$

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

What is disadvantage(s) of Gain Ratio?

Problem of Gain Ratio

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

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

What is disadvantage(s) of Gain Ratio?

- Attribute is selected with the highest gain ratio
- Gain ratio tends to prefer unbalanced splits in which one partition is much smaller than the other

Gini Index (CART, IBM Intelligent Miner)

- If a data set $\mathcal{D}$ contains examples from n classes, gini index, $gini(\mathcal{D})$ is defined as

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

- where p_i is the relative frequency of class j in $\mathcal{D}$.
- If a data set $\mathcal{D}$ is split into two subsets $\mathcal{D}_1$ and $\mathcal{D}_2$ with sizes N_1 and N_2 respectively, the $gini$ index of the split data contains examples from n classes, the $gini$ index of the spit is defined as

$$gini_{split}(\mathcal{D}) = \frac{N_1}{N} gini(\mathcal{D}_1) + \frac{N_2}{N} gini(\mathcal{D}_2)$$

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

Gini index (CART, IBM IntelligentMiner)

- The lowest is the best
- All attributes are assumed continuous-valued
- Can be modified for categorical attributes
- Ex. $\mathcal{D}$ has 9 tuples in bus_computer = "yes" and 5 in "no", $gini(\mathcal{D}) = 1 - (\frac{9}{14})^2 - (\frac{5}{14})^2 = 0.459$
- Suppose the attribute income partitions $\mathcal{D}$ into 10 in $\mathcal{D}_1$:{medium, high} and 4 in $\mathcal{D}_2$

$$\begin{aligned} gini_{income \in \{medium, high\}}(\mathcal{D}) &= \frac{10}{14} gini(\mathcal{D}_1) + \frac{4}{14} gini(\mathcal{D}_2) \\ &= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2) \\ &= 0.450 \end{aligned}$$

Problem of Gini Index

$$
\begin{aligned}
gini(\mathcal{D}) &= 1 - \sum_{j=1}^{n} p_j^2 \\
gini_{split}(\mathcal{D}) &= \frac{N_1}{N} gini(\mathcal{D}_1)
\end{aligned}
$$

What is disadvantage(s) of Gini Index?

Problem of Gini Index

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

$$gini_A(\mathcal{D}) = \sum_{i=1}^{v} \frac{|\mathcal{D}_i|}{|\mathcal{D}|} gini(\mathcal{D}_i)$$

What is disadvantage(s) of Gini Index?

- Attribute is selected with the lowest Gini index
- Gini index is biased towards multivalued attributes
- Gini index has difficulty when # of classes is large
- Gini index tends to favor tests that result in equal-sized partitions and purity in both partitions

Extracting Classification Rules from Trees

- Represent the knowledge in the form of If-Then rules
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The leaf node holds the class distribution
- Rules are easier for humans to understand
- Example
  - If age = "<=30" AND student = "no" THEN buys_computer = "no"
  - If age = "<=30" AND student = "yes" THEN buys_computer = "yes"
  - IF age =">40" AND credit_rating = "excellent" THEN buys_computer = "no"
  - IF age = "<=30" AND credit_rating = "fair" THEN buys_computer = "yes"

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"

Summary of Decision Tree

- ID3
  - Select the attribute with the **highest information gain**
  - Information Gain is biased towards attributes with a large number of values
- C4.5
  - Select the attribute with the **highest gain ratio**
  - Gain ratio tends to prefer unbalanced splits in which one partition is much smaller than the other
- CART
  - Select the attribute with the **lowest gini index**
  - Gini index is biased towards multivalued attributes
  - Gini index has difficulty when # of classes is large
  - Gini index tends to favor tests that result in equal-sized partitions and purity in both partitions

Summary of Decision Tree

- The maximum number of leaf nodes in tree is N, where N is the number of examples in the training dataset
- The maximum length of the tree is a, where a is the number of attributes in the training dataset
- The maximum number of nodes in the tree is $N + a$

Evaluating Classifier Accuracy

- Holdout
  - Train on $2/3$
  - Test on $1/3$
- Cross validation: k-fold cross validation
  - Partition data set into k parts
  - Train on random $(k-1)$ parts, test on 1 part
  - Repeat k times, or more
  - Average accuracy

Comment on Decision Tree Induction

- Relatively faster learning speed(than other classification methods)
- Convertible to simple and easy to understand classification rules
- Comparable classification accuracy with other methods
- Comparably scalable to large database

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
- Attribute construction
  - Create new attributes based on existing ones

Bayesian Classification

- A statistical classifier
  - Perform probabilistic prediction, i.e., predict class membership probabilities
- Foundation
  - Based on Bayes' Theorem
- Assumption
  - The effect of an attribute on a given class is independent of other attributes
- Performance
  - A simple Bayesian classifier, naïve Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers

Bayesian Theorem: Basics

- Let X be a data sample, class label is unknown
- Let H be a hypothesis, e.g., X belongs to class C
- Classification is to determine $P(H|X)$, the probability that the hypothesis holds given the observed data sample X
- $P(H)$: the initial probability
  - E.g., X will buy computer, regardless of age, income,...
- $P(X)$: probability that sample data is observed
- $P(X|H)$: the probability of observing the sample X, given that the hypothesis holds
  - E.g., Given that X will buy computer, what is the prob. that X is 31..40?

Bayesian Theorem

- Given training data X, probability of a hypothesis H, $P(H|X)$ follows the Bayesian Theorem

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

- Predict X belongs to C_i iff the probability $P(C_i|X)$ is the highest among all the $P(C_k|X)$ for all the k classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

Naïve Bayesian Classifier

- Let $\mathcal{D}$ be a training set of tuples and their associated class labels, and each tuple is represented by an $n-$ dimensional attribute vector $X = (x_1, x_2, ..., x_n)$
- Suppose there are m classes $C_1, C_2, ..., C_m$
- Classification is to derive the maximum posteriori, i.e., the maximal $P(C_i|X)$
- This can be derived from Bayes Theorem

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

- Since $P(X)$ is constant for all classes, only $P(C_i|X) = P(X|C_i)P(C_i)$ needs to be maximized
- $P(C_i)$ can be obtained from training data set s_i/s

Derivation of Naïve Bayes Classifier

- Assumption: attribute are conditionally independent (i.e., no dependence relation between attributes), $X = (x_1, x_2, ..., x_n)$

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

- This greatly reduces the computation cost: Only counts the class distribution
- If attribute A_k is **categorical**, $P(x_k|C_i) = \frac{s_{ik}}{s_i}$, count the distribution

Derivation of Naïve Bayes Classifier

- If attribute A_k is **continuous-valued**, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard derivation σ,

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

- Then, $P(x_k|C_i)$ is calculated by

$$P(x_k|C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$$

- The mean μ and standard derivation σ can be easily estimated according the training data

Exercise

age	income	student	credit_rating	buy_computer
$<= 30$	high	no	fair	no
$<= 30$	high	no	excellent	no
31 - 34	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

- Predict what class does the data sample $X = (age <= 30, Income = medium, Student = yes, Credit\_rating = Fair)$ belong to?
- Class: C_1-buys_computer ="yes", C_2-buys_computer = "no"

Solution

- Compute $P(C_i)$:
- $P(buys\_computer = \text{"}yes\text{"}) = 9/14 = 0.643$
- $P(buys\_computer = \text{"}no\text{"}) = 5/14 = 0.357$

- Compute $P(X|C_i)$ for each class:
- $P(age = \text{"} <= 30\text{"}|buys\_computer = \text{"}yes\text{"}) = 2/9 = 0.222$
- $P(age = \text{"} <= 30\text{"}|buys\_computer = \text{"}no\text{"}) = 3/5 = 0.6$
- $P(income = \text{"}medium\text{"}|buys\_computer = \text{"}yes\text{"}) = 4/9 = 0.444$
- $P(income = \text{"}medium\text{"}|buys\_computer = \text{"}no\text{"}) = 2/5 = 0.4$
- $P(student = \text{"}yes\text{"}|buys\_computer = \text{"}yes) = 6/9 = 0.667$
- $P(student = \text{"}yes\text{"}|buys\_computer = \text{"}no\text{"}) = 1/5 = 0.2$
- $P(credit\_rating = \text{"}fair\text{"}|buys\_computer = \text{"}yes\text{"}) = 6/9 = 0.667$
- $P(credit\_rating = \text{"}fair\text{"}|buys\_computer = \text{"}no\text{"}) = 2/5 = 0.4$

Solution

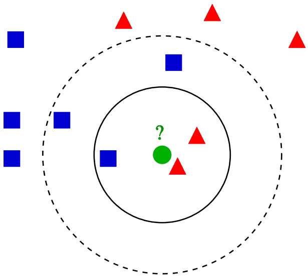
- Test example: $X = (age <= 30, Income = medium, Student = yes, Credit\_rating = Fair)$
- Compute $P(X|C_i)$:
- $P(X|buys\_computer = \text{“}yes\text{”}) = 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044$
- $P(X|buys\_computer = \text{“}no\text{”}) = 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019$
- Compute $P(C_i|X) = P(X|C_i) * P(C_i)$:
- $P(X|buys\_computer = \text{“}yes\text{”}) \times P(buys\_computer = \text{“}yes\text{”}) = 0.028$
- $P(X|buys\_computer = \text{“}no\text{”}) \times P(buys\_computer = \text{“}no\text{”}) = 0.007$
- **Therefore, X belongs to class “buys_computer = yes” ”**

Naïve Bayesian 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 do 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 Bayesian Classifier
- How to deal with these dependencies?
  - Bayesian Belief Networks

k Nearest Neighbors Algorithm

- All instances correspond to points in the $\mathbb{R}^D$ space
- The nearest neighbor is defined in terms of Euclidean distance, dist(X_1, X_2), or other distance measures
- Target function could be discrete-valued or real-valued
- For discrete-valued, k-NN returns the most common value among the k training examples nearest to X_q



Exercise

- Consider the one-dimensional data set. Please classify the data point $x = 5.0$ according to its 1-, 3-, and 5-nearest neighbors (using majority vote).

x	y
0.5	-
3.0	-
4.5	+
4.6	+
4.9	+
5.2	-
5.3	-
5.5	+
7.0	-
9.5	-

Exercise

- Consider the one-dimensional data set. Please classify the data point $x = 5.0$ according to its 1-, 3-, and 5-nearest neighbors (using majority vote).

x	y	dis $(\lvert x_1 - x_2 \rvert)$
0.5	-	4.5
3.0	-	2
4.5	$+$	0.5
4.6	$+$	0.4
4.9	$+$	0.1
5.2	-	0.2
5.3	-	0.3
5.5	$+$	0.5
7.0	-	2
9.5	-	4.5

How about $k = 4$?

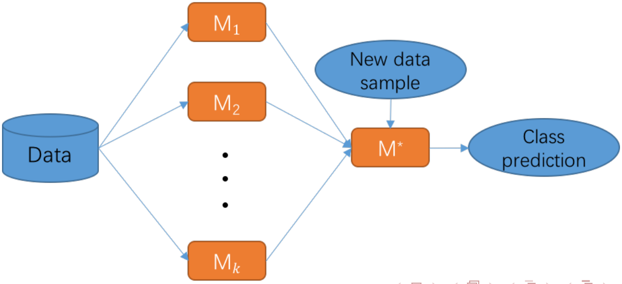
Discussion on the k-NN Algorithm

- k-NN for real-value prediction for a given unknown tuple
  - Returns the mean value of the k nearest neighbors
- Robust to noisy data by averaging k-nearest neighbors
- Distance between neighbors could be dominated by the irrelevant attributes
  - To overcome it, eliminate irrelevant attributes
- Lazy-learner
  - Not build a classifier
  - Store all the training samples
  - High computational cost for each new tuple

Issues to kNN Algorithm

- The choice of k
  - If k is too small, then the resut can be sensitive to noise points
  - If k is too large, then the neighborhood may include too many points from other classes
- Combing the nearest neighbors class labels
  - Majority vote
  - The nearest neighbors may vary widely in their distance, and the closer neighbors more reliably indicate the class of the object
  - Weights each object's vote by its distance
- The choice of distance measure
  - Euclidean distance, cosine similarity, Manhattan distance, Metric Learning, .etc
- High computation
  - Find the k nearest neighbors for each test example
  - Make use of structure of data, e.g., nearest neighbor graphs, minimum spanning trees, relative neighborhood graphs, Delaunay triangulations, and Gabriel graphs,...

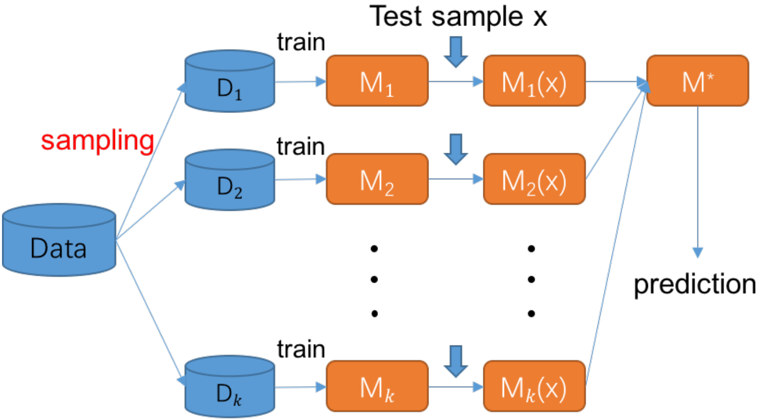
- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, ...,M_k, with the aim of creating an improved model M^*
- Popular ensemble methods
  - Bagging
  - Boosting



Bagging: Bootstrap Aggregation

- Analogy:Diagnosis based on multiple doctors' majority vote
- Training
  - Give a data set $\mathcal{D}$ of N samples, at each iteration i, a training set $\mathcal{D}_i$ is sampled with replacement from $\mathcal{D}$
  - A classifier model M_i is learned for each training set $\mathcal{D}_i$
- Classification: classify an unknown data 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

Bagging: Bootstrap Aggregation



- $M^*(x) = \text{maxcount}_t M_t(x)$

Bagging: Bootstrap Aggregation

- 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 significant better than a single classifier derived from $\mathcal{D}$
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Exercise

Following is a data set to construct a bagging classifier

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
y	1	1	1	-1	-1	-1	-1	1	1	1

Examples chosen for training in each round are shown below:

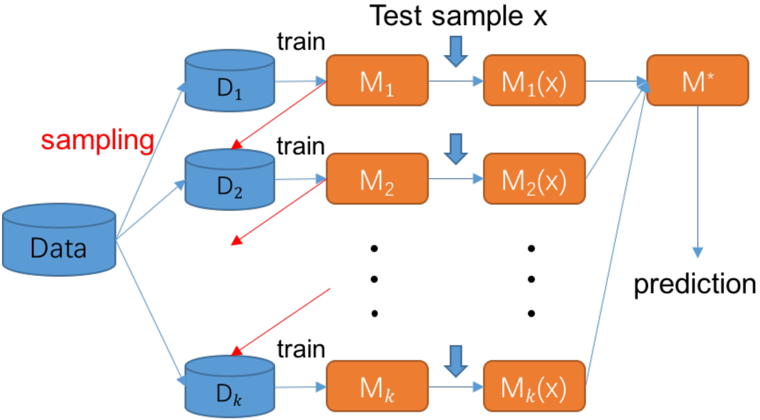
x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	-1	- 1
Classifier: ① x <= 0.35 -> y=1, ② x>0.35 -> y = -1										
x	0.1	0.2	0.3	0.5	0.5	0.8	0.9	1	1	1
y	1	1	1	-1	-1	1	1	1	1	1
Classifier: ① 0.4<= x <=0.55 -> y=-1, ② x>0.55 -> y=1, ③ x<0.4 -> y=1										
x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1
Classifier: ① x<=0.35 -> y=1, ② 0.35<=x<=0.75 -> y=-1, ③ x>0.75 -> y=1										

Please predict the class label for x = 0.38 ?

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses - weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier M_{i+1} pay more attention to the training tuples that were misclassified by M_i
  - A series of k classifiers are iteratively learned
  - The final M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy

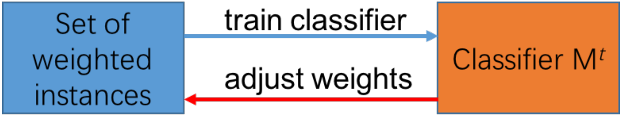
Boosting



- $\mathrm{M}^*(\mathrm{x}) = \mathrm{argmax}_{M_c} \sum_t^k w_t M_t(x)$

Boosting

- The boosting algorithm can be extended for the prediction of continuous values
- Comparing with bagging: boosting tends to achieve greater accuracy, but it also risks overfitting the model to the misclassified data



Bagging vs. Boosting

- Model training:
  - Bagging: random sampling, independent classifiers
  - Boosting: subsequent classifier M_{i+1} pay more attention to the training tuples that were misclassified by M_i
- Model usage:
  - Bagging: equal weight
  - Boosting: different weights assigned

Random Forest

Tree bagging

- Given a training set $\mathcal{D} = \{x_i, y_i\}_{i=1}^{n}$, $x_i \in \mathbb{R}^d$, and y_i is the corresponding class label.
- The procedures of Tree bagging is summarized as following:
  1. For $b = 1$ *to* B
  2. Sample, with replacement, n training examples from $\mathcal{D}$, call $\mathcal{D}_b = \{x_i, y_i\}_{i=1}^{n}$;
  3. Train a classification or regression tree f_b on $\mathcal{D}_b$;
  4. End
- Predictions for unseen samples x' can be made by taking the majority vote in the case of classification trees.
- or by averaging the predictions from all the individual regression trees on x'

$$\hat{f} = \frac{1}{B} f_b(x')$$

Random Forest

- Random forests differ in only one way from Tree Bagging
  - They use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features.

1. For $b = 1$ *to* B
2. Sample, with replacement, n training examples **with p features** from $\mathcal{D}$, call $\mathcal{D}_b = \{x_i, y_i\}_{i=1}^{n}$, $x_i \in \mathbb{R}^p$;
3. Train a classification or regression tree f_b on $\mathcal{D}_b$;
4. End

- Typically, for a classification problem with d features, $\sqrt{d}$ (rounded down) features are used in each split.
- For regression problems the inventors recommend $d/3$ (rounded down) with a minimum node size of 5 as the default. (The Elements of Statistical Learning, 2nd ed.)

Random Forest

- Decision trees are a popular method for various machine learning tasks. Tree learning comes closest to meeting the requirements for serving as an off-the-shelf procedure for data mining
- It is invariant under scaling and various other transformations of feature values, is robust to inclusion of irrelevant features
- Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks
- Random decision forests correct for decision trees' habit of overfitting to their training set

What is Prediction?

- Numerical prediction is similar to classification
  - Construct a model
  - Use model to predict continuous or ordered value for a given input
- Prediction is different from classification
  - Classification refers to predict categorical class label
  - Prediction models continuous valued functions
- Major method for prediction: **regression**
  - Model the relationship between one or more **independent or predictor** variables and a **dependent or response** variable
- Regression analysis
  - Linear and multiple regression
  - Non-linear regression
  - Other regression methods: generalized linear model, Poisson regression, log-linear models, regression trees, logistic regression

Linear Regression

- Linear regression: a response variable y and a single predictor variable x, $y = w_0 + w_1 x$, where w_0 and w_1 are regression
- Method of least squares: estimate the best-fitting straight line, $w_1 = \frac{\sum_{i=1}^{|D|}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|}(x_i - \bar{x})^2}$, and $w_0 = \bar{y} - w_1 \bar{x}$

- Multiple linear regression: more than one predictor variable
  - Training data is of the form $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_{|D|}, y_{|D|})$
  - Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{d \times n}$, $\mathbf{y} = [y_1, y_2, ..., y_n]^T$
  - Linear regression: $y = \mathbf{x}^T\mathbf{w} + w_0$
  - where $\mathbf{w} = [w_1, w_2, ..., w_d]^T \in \mathbb{R}^d$ is the regression coefficients, the bias w_0 can be absorbed into $\mathbf{w}$ when the constant value 1 is added as an additional dimension for each data $\mathbf{x}_i (1 \leq i \leq n)$, so we can obtain $y = \mathbf{x}^T\mathbf{w}$
- Apply the least square loss:

$$
\begin{aligned}
J(\mathbf{w}) &= \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^T\mathbf{w} - y_i)^2 \\
&= \frac{1}{2}\|\mathbf{X}^T\mathbf{w} - \mathbf{y}\|_2^2 \\
&= \frac{1}{2}(\mathbf{X}^T\mathbf{w} - \mathbf{y})^T(\mathbf{X}^T\mathbf{w} - \mathbf{y})
\end{aligned}
$$

$$
\begin{aligned}
J(\mathbf{w}) &= \frac{1}{2}(\mathbf{X}^T\mathbf{w}-\mathbf{y})^T(\mathbf{X}^T\mathbf{w}-\mathbf{y}) \\
&= \frac{1}{2}(\mathbf{w}^T\mathbf{X}\mathbf{X}^T\mathbf{w}-\mathbf{w}^T\mathbf{X}\mathbf{y}-\mathbf{y}^T\mathbf{X}^T\mathbf{w}+\mathbf{y}^T\mathbf{y})
\end{aligned}
$$

- Solving $\mathbf{w}$: Setting the derivative of $J(\mathbf{w})$ with respect to $\mathbf{w}$ to zero

$$
\begin{aligned}
\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= \mathbf{X}\mathbf{X}^T\mathbf{w}-\mathbf{X}\mathbf{y}=0 \\
\mathbf{w} &= (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{y}
\end{aligned}
$$

- **Note**: $\mathbf{X}\in\mathbb{R}^{(d+1)\times n}$, $\mathbf{w}\in\mathbb{R}^{d+1}$

Nonlinear Regression

- A polynomial regression model can be transformed into linear regression model. For example

$$y = w_0 + w_1 x + w_2 x^2 + w_3 x^3$$

- It can be convert to linear with new variables: $x_1 = x, x_2 = x^2, x_3 = x^3$

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$$

- There are many nonlinear regression models, e.g., Exponential, power, and log functions
- $y = \beta_0 e^{\beta_1 x}$: let $y' = \ln y, \beta_0' = \ln \beta_0, x' = x$, than it can be convert to a linear model $y' = \beta_0' + \beta_1 x'$

Classifier Accuracy Measures

- Accuracy of a classifier M: percentage of test samples that are correctly classified by the model M
  - Given m classes, $CM_{i,j}$ is an entry in a **confusion matrix** and it indicates # of samples in class i that are labeled by the classifier as class j
  - Accuracy $=$ (t-pos $+$ t-neg)/(pos $+$ neg)
  - Error rate (misclassification rate) of $M = 1$ - Accuracy

		Predicted class		
		C_1	C_2	Total
Actual class	C_1	True positive	False negative	pos
	C_2	Flase positive	True negative	neg
	Total	t-pos $+$ f-pos	t-neg $+$ f-neg	pos+neg

Classifier Accuracy Measures

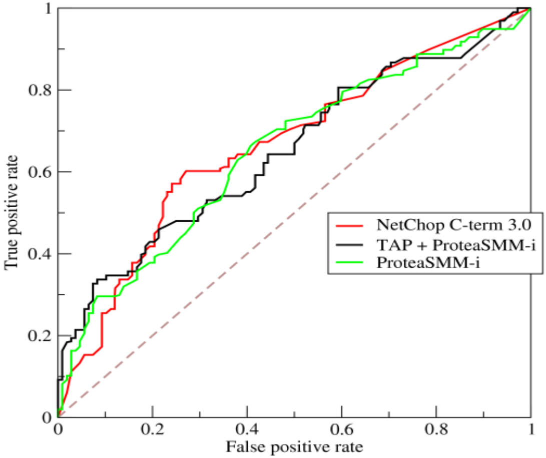
- Alternative accuracy measures
- sensitivity = t-pos/pos, true positive recognition rate, also called "recall"
- specificity = t-neg/neg, true negative recognition rate
- precision = t-pos/(t-pos + f-pos)
- recall = t-pos/(t-pos + f-neg)
- accuracy = (t-pos + t-neg)/(pos + neg)
- $f_1 = (1+\alpha^2)\times$precision$\times$recall / (α^2precision + recall) = 2$\times$t-pos/(2$\times$t-pos+f-neg + f-pos), α is usually set to be 1

ROC and AUC

- The **ROC** (Receiver Operating Characteristic) curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.
- **true positive rate (TPR)**: t-pos/pos, **recall, sensitivity**
- **false positive rate (FPR)**: f-pos/neg, **1-specificity**

ROC and AUC

Example



ROC and AUC

Key points

- (0,0): f-pos=0, t-pos=0. It means that all the tuples are classified as negative.
- (0,1): f-pos=0, t-pos=pos. It indicates that all the tuples are correctly classified.
- (0,1): f-pos=neg, t-pos=0. It indicates that all the tuples are incorrectly classified.
- (1,1): f-pos=neg, t-pos=pos. It indicates that all the tuples are classified as positive.

ROC and AUC

- **AUC**: Area under ROC curve, the AUC value is equivalent to the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative example

Summary

- Bagging and Boosting can be used to increase overall accuracy by learning and combing a series of individual models
- No single methods has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, interpretability, and scalability must be considered
- k-fold cross validation is a recommended method for accuracy estimation