


Chapter 6. Classification and Prediction

- What is classification? What is prediction? 
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

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

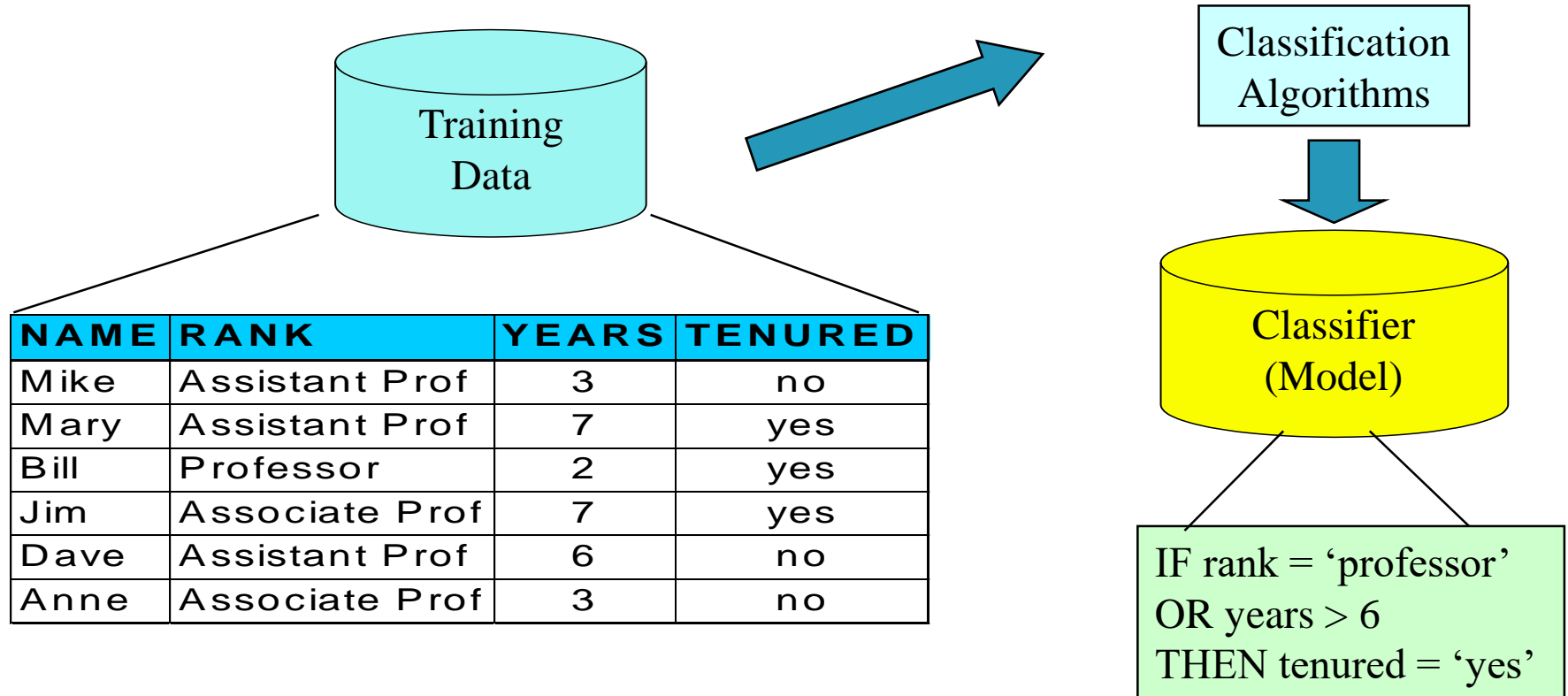
Classification vs. Prediction

- **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
- **Prediction**
 - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
 - Credit approval
 - Target marketing
 - Medical diagnosis
 - Fraud detection

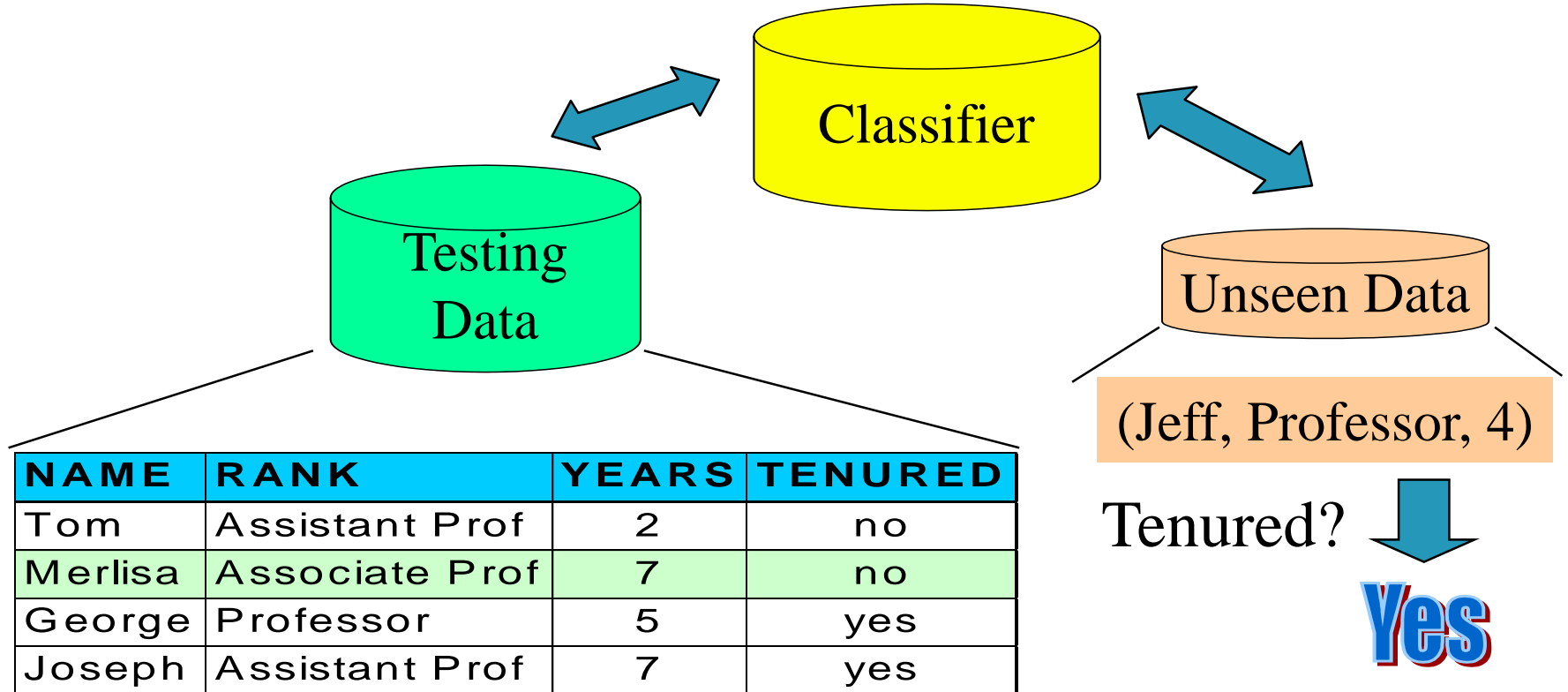
Classification—A Two-Step Process

- **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 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



Process (2): Using the Model in Prediction



Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary


Issues: Data Preparation

- 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 normalize data

Issues: Evaluating Classification Methods

- 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/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 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification 
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- 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

Bayesian Theorem: Basics

- Let \mathbf{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 $P(H|\mathbf{X})$, the probability that the hypothesis holds given the observed data sample \mathbf{X}
- $P(H)$ (*prior probability*), the initial probability
 - E.g., \mathbf{X} will buy computer, regardless of age, income, ...
- $P(\mathbf{X})$: probability that sample data is observed
- $P(\mathbf{X}|H)$ (*posteriori probability of X conditioned on H*), the probability of observing the sample \mathbf{X} , given that the hypothesis holds
 - E.g., Given that \mathbf{X} will buy computer, the prob. that X is 31..40, medium income

Bayesian Theorem

- Given training data \mathbf{X} , *posteriori probability of a hypothesis* H , $P(H|\mathbf{X})$, follows the Bayes theorem

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

- Informally, this can be written as
posteriori = likelihood x prior/evidence
- Predicts \mathbf{X} belongs to C_2 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: require initial knowledge of many probabilities, significant computational cost

Towards Naïve Bayesian Classifier

- Let D be a training set of tuples and their associated class labels, and each tuple is represented by an n -D attribute vector $\mathbf{X} = (x_1, x_2, \dots, x_n)$
- Suppose there are m classes C_1, C_2, \dots, 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

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

where

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

- Since $P(\mathbf{X})$ is constant for all classes, only needs to be maximized $P(C_i|\mathbf{X}) = P(\mathbf{X}|C_i)P(C_i)$

Derivation of Naïve Bayes Classifier

- A simplified assumption: attributes are conditionally independent (i.e., no dependence relation between attributes):

$$\begin{aligned} P(\mathbf{X}|C_i) &= \prod_{k=1}^n P(x_k|C_i) \\ &= P(x_1|C_i) \times P(x_2|C_i) \times \dots \times P(x_n|C_i). \end{aligned}$$

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

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

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

Naïve Bayesian Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data sample

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 Bayesian Classifier: An Example

$P(C_i)$ – Prior probability of each class :

$$P(\text{buys_computer} = \text{"yes"}) = 9/14 = 0.643$$

$$P(\text{buys_computer} = \text{"no"}) = 5/14 = 0.357$$

Compute likelihood $P(X|C_i)$ of each attribute value for each class

$$P(\text{age} = \text{"<=30"} \mid \text{buys_computer} = \text{"yes"}) = 2/9 = 0.222$$

$$P(\text{age} = \text{"<= 30"} \mid \text{buys_computer} = \text{"no"}) = 3/5 = 0.6$$

$$P(\text{income} = \text{"medium"} \mid \text{buys_computer} = \text{"yes"}) = 4/9 = 0.444$$

$$P(\text{income} = \text{"medium"} \mid \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$$

$$P(\text{student} = \text{"yes"} \mid \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{student} = \text{"yes"} \mid \text{buys_computer} = \text{"no"}) = 1/5 = 0.2$$

$$P(\text{credit_rating} = \text{"fair"} \mid \text{buys_computer} = \text{"yes"}) = 6/9 = 0.667$$

$$P(\text{credit_rating} = \text{"fair"} \mid \text{buys_computer} = \text{"no"}) = 2/5 = 0.4$$

age	income	student	credit_rating	com
<=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 Bayesian Classifier: An Example

- For the unseen data, classify using bayes classifier
- $X = (\text{age} \leq 30, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$

$$\begin{aligned} P(X | C_i) : P(X | \text{buys_computer} = \text{"yes"}) &= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044 \\ P(X | \text{buys_computer} = \text{"no"}) &= 0.6 \times 0.4 \times 0.2 \times 0.4 = 0.019 \end{aligned}$$

$$P(C_i | X) = (P(X | C_i) * P(C_i))$$

$$\begin{aligned} P(\text{buys_computer} = \text{yes} | X) &= P(X | \text{buys_computer} = \text{"yes"}) * P(\text{buys_computer} = \text{"yes"}) \\ &= 0.028 \end{aligned}$$

$$\begin{aligned} P(\text{buys_computer} = \text{no} | X) &= P(X | \text{buys_computer} = \text{"no"}) * P(\text{buys_computer} = \text{"no"}) \\ &= 0.007 \end{aligned}$$

$$P(\text{buys_computer} = \text{yes} | X) > P(\text{buys_computer} = \text{no} | X)$$

Therefore, X belongs to class ("buys_computer = yes")

$$\text{Note: } P(X) = 0.028 + 0.007 = 0.035$$

Height Example Data

Name	Gender	Height	Output1	Output2
Kristina	F	1.6m	Short	Medium
Jim	M	2m	Tall	Medium
Maggie	F	1.9m	Medium	Tall
Martha	F	1.88m	Medium	Tall
Stephanie	F	1.7m	Short	Medium
Bob	M	1.85m	Medium	Medium
Kathy	F	1.6m	Short	Medium
Dave	M	1.7m	Short	Medium
Worth	M	2.2m	Tall	Tall
Steven	M	2.1m	Tall	Tall
Debbie	F	1.8m	Medium	Medium
Todd	M	1.95m	Medium	Medium
Kim	F	1.9m	Medium	Tall
Amy	F	1.8m	Medium	Medium
Wynette	F	1.75m	Medium	Medium

$X = \langle \text{Adam}, M, 1.95m \rangle$

$X = \langle \text{Adam, M, 1.95m} \rangle$

Prior Probabilities: $P(\text{short}) = 4/15 = 0.267$; $P(\text{medium}) = 8/15 = 0.533$;

$P(\text{tall}) = 3/15 = 0.2$

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{n-1} \left[\sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2 \right]$$

For continuous data -> assume Gaussian distribution

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

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

So we need to calculate mean and standard deviation of the values for the attribute for class C

From training set, $\mu = 1.9$ and $\sigma = 0.3$ then

1. $P(\text{height} = 1.95 | \text{short})$: Range (1.6m – 1.7m) , $\mu = 1.65$ and $\sigma = 0.05$ then

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi} 0.05} e^{-\frac{(1.95-1.65)^2}{2*0.05^2}}$$

$$g(1.95, \mu_{\text{short}}, \sigma_{\text{short}}) = 1.2154 \times 10^{-7}$$

2. $P(\text{height} = 1.95 | \text{medium})$: Range (1.75m – 1.95m) , $\mu = 1.85$ and $\sigma = 0.1$ then

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi} 0.1} e^{-\frac{(1.95-1.85)^2}{2*0.1^2}}$$

$$g(1.95, \mu_{\text{medium}}, \sigma_{\text{medium}}) = 2.419$$

1.6m	Short
2m	Tall
1.9m	Medium
1.88m	Medium
1.7m	Short
1.85m	Medium
1.6m	Short
1.7m	Short
2.2m	Tall
2.1m	Tall
1.8m	Medium
1.95m	Medium
1.9m	Medium
1.8m	Medium
1.75m	Medium

3. P(height = 1.95|tall): Range (2.0 m – 2.2m) , $\mu = 2.1$ and $\sigma = 0.1$ then

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

$$g(1.95, \mu_{\text{tall}}, \sigma_{\text{tall}}) = 1.29$$

Posteriori probability of M conditioned on C

$$P(M|\text{short}) = 0 = 0$$

$$P(M|\text{medium}) = 2/8$$

$$P(M|\text{tall}) = 3/3$$

Posteriori probability of C conditioned on X

$$P(\text{short}|X) = P(M|\text{short}) \times P(1.95|\text{short}) \times P(\text{short}) = 0$$

$$P(\text{medium}|X) = 2/8 \times 2.419 \times 8/15 = 0.3$$

$$P(\text{tall}|X) = 1 \times 1.29 \times 3/15 = 0.25$$

F	1.6m	Short
M	2m	Tall
F	1.9m	Medium
F	1.88m	Medium
F	1.7m	Short
M	1.85m	Medium
F	1.6m	Short
M	1.7m	Short
M	2.2m	Tall
M	2.1m	Tall
F	1.8m	Medium
M	1.95m	Medium
F	1.9m	Medium
F	1.8m	Medium
F	1.75m	Medium

Since the probability of Medium is highest we classify X as Medium.

Another method for Bayes

$X = \langle \text{Adam}, M, 1.95m \rangle$

Prior Probabilities: $P(\text{short}) = 4/15 = 0.267$; $P(\text{medium}) = 8/15 = 0.533$; $P(\text{tall}) = 3/15 = 0.2$

Since height is continuous data -> split it into ranges

Height Range: $(0, 1.6]$, $(1.6, 1.7]$, $(1.7, 1.8]$, $(1.8, 1.9]$, $(1.9, 2.0]$, $(2.0, \infty)$

Attribute	Value	Count			Probabilities		
		Short	Medium	Tall	Short	Medium	Tall
Gender	M	1	2	3	1/4	2/8	3/3
	F	3	6	0	3/4	6/8	0/3
Height	$(0, 1.6]$	2	0	0	2/4	0	0
	$(1.6, 1.7]$	2	0	0	2/4	0	0
	$(1.7, 1.8]$	0	3	0	0	3/8	0
	$(1.8, 1.9]$	0	4	0	0	4/8	0
	$(1.9, 2]$	0	1	1	0	1/8	1/3
	$(2, \infty)$	0	0	2	0	0	2/3

F	1.6m	Short
M	2m	Tall
F	1.9m	Medium
F	1.88m	Medium
F	1.7m	Short
M	1.85m	Medium
F	1.6m	Short
M	1.7m	Short
M	2.2m	Tall
M	2.1m	Tall
F	1.8m	Medium
M	1.95m	Medium
F	1.9m	Medium
F	1.8m	Medium
F	1.75m	Medium

Posteriori probability of X conditioned on C

$$P(X|\text{short}) = 1/4 \times 0 = 0$$

$$P(X|\text{medium}) = 2/8 \times 1/8 = 0.031$$

$$P(X|\text{tall}) = 3/3 \times 1/3 = 0.333$$

Another method for Bayes

$X = \langle \text{Adam}, M, 1.95\text{m} \rangle$

Prior Probabilities: $P(\text{short}) = 4/15 = 0.267$; $P(\text{medium}) = 8/15 = 0.533$; $P(\text{tall}) = 3/15 = 0.2$

Since height is continuous data -> split it into ranges

Height Range: $(0, 1.6]$, $(1.6, 1.7]$, $(1.7, 1.8]$, $(1.8, 1.9]$, $(1.9, 2.0]$, $(2.0, \infty)$

Posteriori probability of X conditioned on C

$$P(X|\text{short}) = 1/4 \times 0 = 0$$

$$P(X|\text{medium}) = 2/8 \times 1/8 = 0.031$$

$$P(X|\text{tall}) = 3/3 \times 1/3 = 0.333$$

Posteriori probability of C conditioned on X

$$P(\text{short}|X) = 0 \times 0.267 = 0$$

$$P(\text{medium}|X) = 0.031 \times 0.533 = 0.016$$

$$P(\text{tall}|X) = 0.333 \times 0.2 = 0.066$$

Since the probability of Tall is highest we classify X as Tall.

F	1.6m	Short
M	2m	Tall
F	1.9m	Medium
F	1.88m	Medium
F	1.7m	Short
M	1.85m	Medium
F	1.6m	Short
M	1.7m	Short
M	2.2m	Tall
M	2.1m	Tall
F	1.8m	Medium
M	1.95m	Medium
F	1.9m	Medium
F	1.8m	Medium
F	1.75m	Medium

Example for Naïve Bayes Classification

Name	Hair	Height	Weight	Dublin	Result
Katie	Brown	Short	Light	Yes	None
Annie	Blonde	Short	Average	No	Sunburned
Emily	Red	Average	Heavy	No	Sunburned
Sarah	Blonde	Average	Light	No	Sunburned
Pete	Brown	Average	Heavy	No	Sunburned
Sam	Brown	Short	Average	Yes	Sunburned
Alex	Brown	Short	Average	No	None
John	Brown	Average	Heavy	No	None
Dana	Blonde	Tall	Average	No	None
Max	Red	Tall	Light	No	Sunburned

$X = \langle \text{brown, tall, average, no} \rangle$



Avoiding the 0-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 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 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

Chapter 6. Classification and Prediction

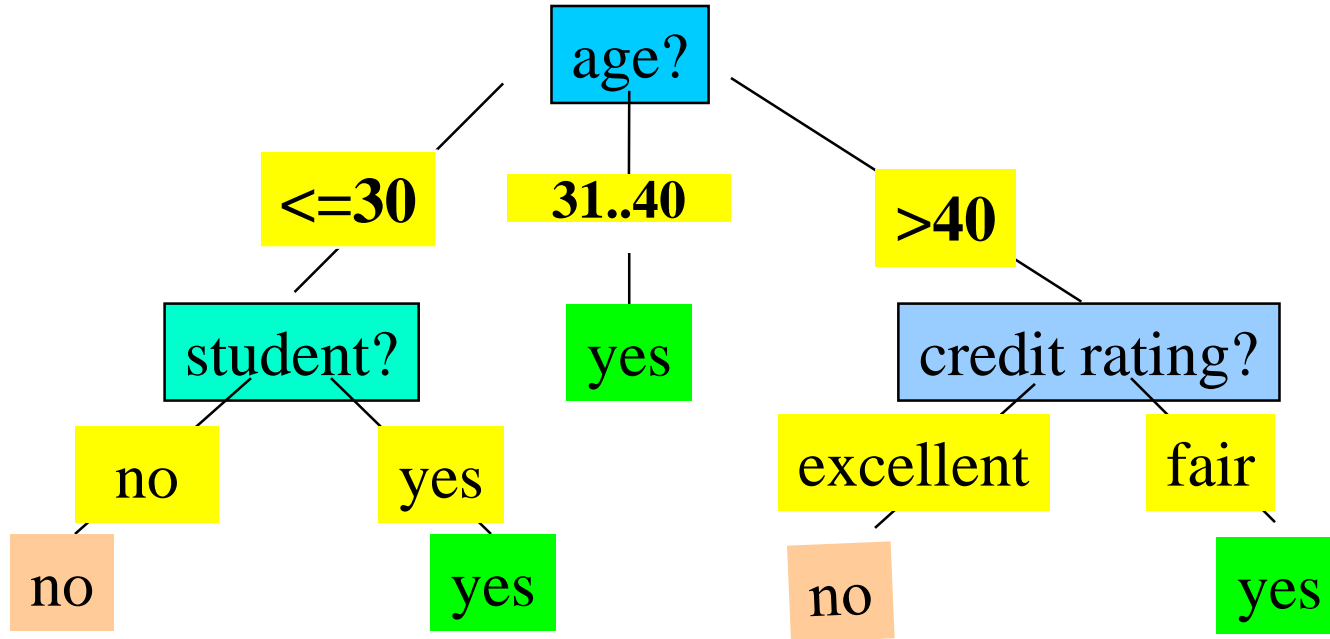
- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction 
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Decision Tree Induction: Training Dataset

This follows an example of Quinlan's ID3 (Playing Tennis)

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

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 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

Partitioning scenarios

Examples

(a)	
(b)	
(c)	

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

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

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

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

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

$$Info(D) = -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

Attribute Selection: Information Gain

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

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

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

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} \right) + \frac{4}{14} \times \left(-\frac{4}{4} \log_2 \frac{4}{4} \right) + \frac{5}{14} \times \left(-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} \right) = 0.694 \text{ bits.}$$

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

Attribute Selection: Information Gain

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

Similarly, find

Gain(income)

Gain(student)

Gain(credit_rating)

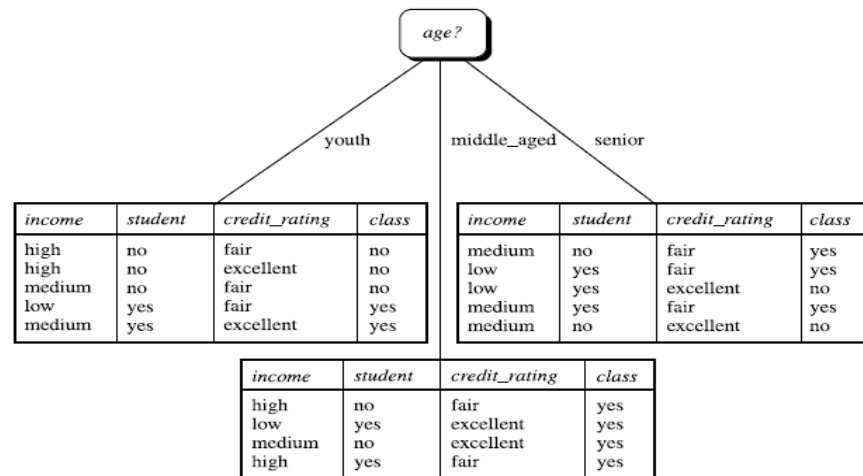
$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit_rating) = 0.048$$

Since age has highest information gain, it is selected as a splitting attribute.

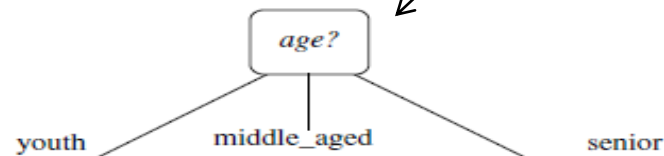
age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

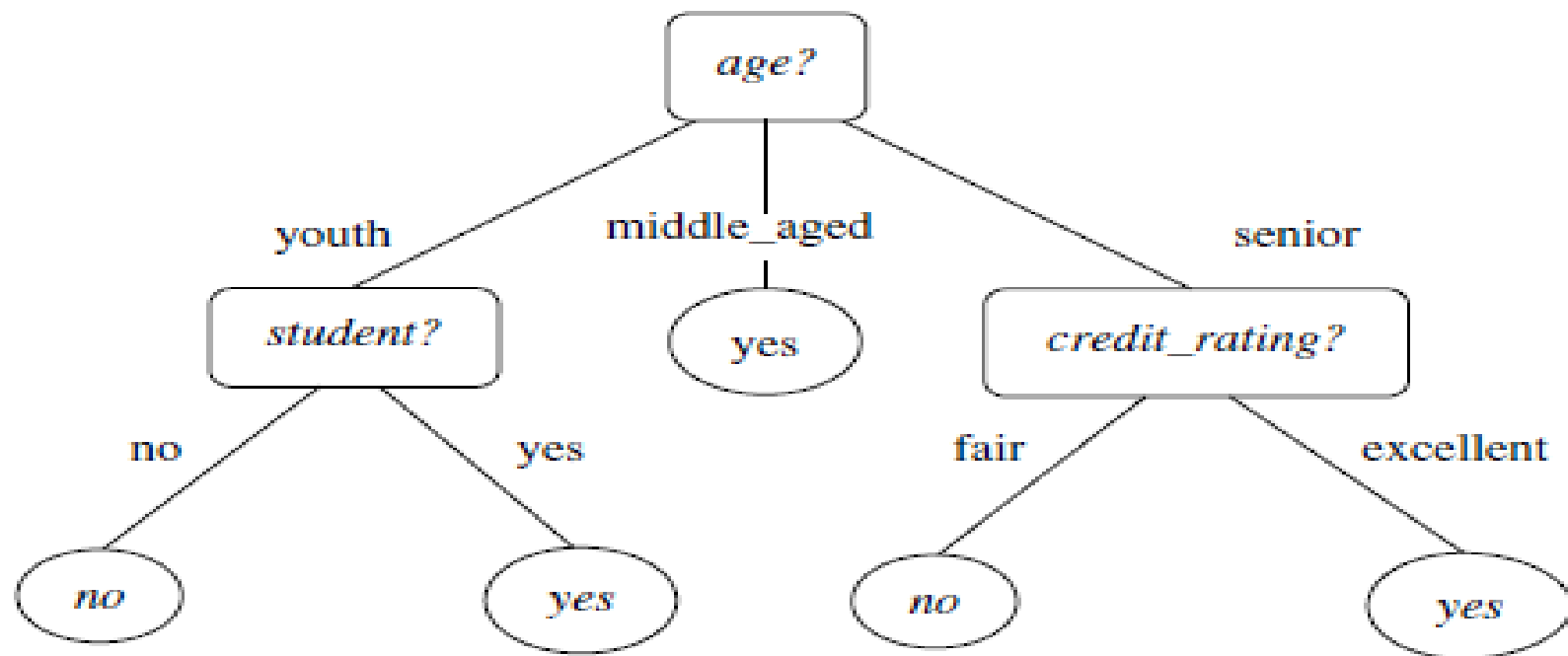


$$Info(D) = -\sum_{i=1}^m p_i \log_2(p_i)$$

$$Info(D_{youth}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5}$$

$$Info(D_{income}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{1}{5} \log_2 \frac{1}{5} + \frac{2}{5} (-\log_2 \frac{1}{5} - \log_2 \frac{1}{5})$$





Computing Information-Gain for Continuous-Value 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 $A \leq \text{split-point}$, and D2 is the set of tuples in D satisfying $A > \text{split-point}$

Computing Information-Gain for Continuous-Value Attributes

- Example: let Age be

- | | | | | | | |
|------|----|----|-----|-----|-----|----|
| Age | 40 | 48 | 60 | 72 | 80 | 90 |
| Buys | No | No | Yes | Yes | Yes | No |

- Determine the threshold where a change in the class is observed : $(48+60)/2 = 54$ and $(80+90)/2 = 85$
- Evaluate the information gain for Age > 54 and Age > 85
- Select threshold on information gain.

Age	40	48	60	72	80	90
Buys	No	No	Yes	Yes	Yes	No

Information gain of Age (>54, <54)

$$D = [3 Y, 3 N] \quad \text{Info}(D) = -3/6 \log_2 3/6 - 3/6 \log_2 3/6 = 1.0$$

$$D(<54) = [0 Y, 2 N] \quad \text{Info}(<54) = 0 - 2/2 \log_2 2/2 = 0$$

$$D(>54) = [3 Y, 1 N] \quad \text{Info}(>54) = -3/4 \log_2 3/4 - 1/4 \log_2 1/4 = 0.81$$

$$\begin{aligned} \text{Gain}(D, \text{Age}>54) &= \text{Info}(D) - \text{Info}_A(D) = \text{Info}(D) - \sum_{v \in \{<54, >54\}} \frac{|D_v|}{|D|} \text{Info}(D_v) = 1 - 2/6 \text{Info}(D<54) - 4/6 \text{Info}(D>54) \\ &= 1 - 0 - 4/6 * 0.81 = 0.45 \end{aligned}$$

Information gain of Age (> 85, <85)

$$D = [3 Y, 3 N] \quad \text{Info}(D) = -3/6 \log_2 3/6 - 3/6 \log_2 3/6 = 1.0$$

$$D(<85) = [3 Y, 2 N] \quad \text{Info}(<85) = -3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.971$$

$$D(>85) = [0 Y, 1 N] \quad \text{Info}(>85) = 0 - 1/1 \log_2 1/1 = 0$$

$$\begin{aligned} \text{Gain}(D, \text{Age}>85) &= \text{Info}(D) - \text{Info}_A(D) = \text{Info}(D) - \sum_{v \in \{<85, >85\}} \frac{|D_v|}{|D|} \text{Info}(D_v) = 1 - 5/6 \text{Info}(D<85) - 1/6 \text{Info}(D>85) \\ &= 1 - 5/6 * 0.971 - 0 = 0.19 \end{aligned}$$

Computing Information-Gain for Continuous-Value Attributes

- Example: let age be

Age	40	48	60	72	80	90
Buys	No	No	Yes	Yes	Yes	No

Gain (D, Age>54) = 0.45

Gain (D, Age>85) = 0.19

- Age >54 has higher gain, hence selected as splitting attribute value
- Now with this discretization we have 2 groups Age < 54 and Age > 54 instead of 6 different values

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)

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

- $GainRatio(A) = Gain(A)/SplitInfo(A)$
- The attribute with the maximum gain ratio is selected as the splitting attribute

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

$$Gain(\text{income}) = 0.029$$

$$\text{gain_ratio}(\text{income}) = 0.029/1.557 = 0.019$$

Gini index (CART, IBM IntelligentMiner)

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

$$gini(D) = 1 - \sum_{i=1}^m p_i^2$$

where p_i is the probability that a tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. The sum is computed over m classes.

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

Gini index (CART, IBM IntelligentMiner)

- 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

$$\begin{aligned} Gini_{income \in \{low, medium\}}(D) &= \frac{10}{14} Gini(D_1) + \frac{4}{14} Gini(D_2) \\ &= \frac{10}{14} \left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2 \right) + \frac{4}{14} \left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 \right) \\ &= 0.443 \\ &= Gini_{income \in \{high\}}(D). \end{aligned}$$

but $gini_{\{medium, high\}}$ is 0.30 and thus the best since it is the lowest

- Similarly, the Gini index values for splits on the remaining subsets are 0.458 (for the subsets {low, high} and {medium}) and 0.450 (for the subsets {medium, high} and {low}).
- Therefore, the best binary split for attribute income is on {low, medium} {high}
- because it minimizes the Gini index.

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

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

Other Attribute Selection Measures

- CHAID: a popular decision tree algorithm, measure based on χ^2 test for independence
- C-SEP: performs better than info. gain and gini index in certain cases
- G-statistics: has a close approximation to χ^2 distribution
- MDL (Minimal Description Length) principle (i.e., the simplest solution is preferred):
 - The best tree as the one that requires the fewest # of bits to both (1) encode the tree, and (2) encode the exceptions to the tree
- Multivariate splits (partition based on multiple variable combinations)
 - CART: finds multivariate splits based on a linear combinations of attributes.
- Which attribute selection measure is the best?
 - Most give good results, none is significantly superior than others

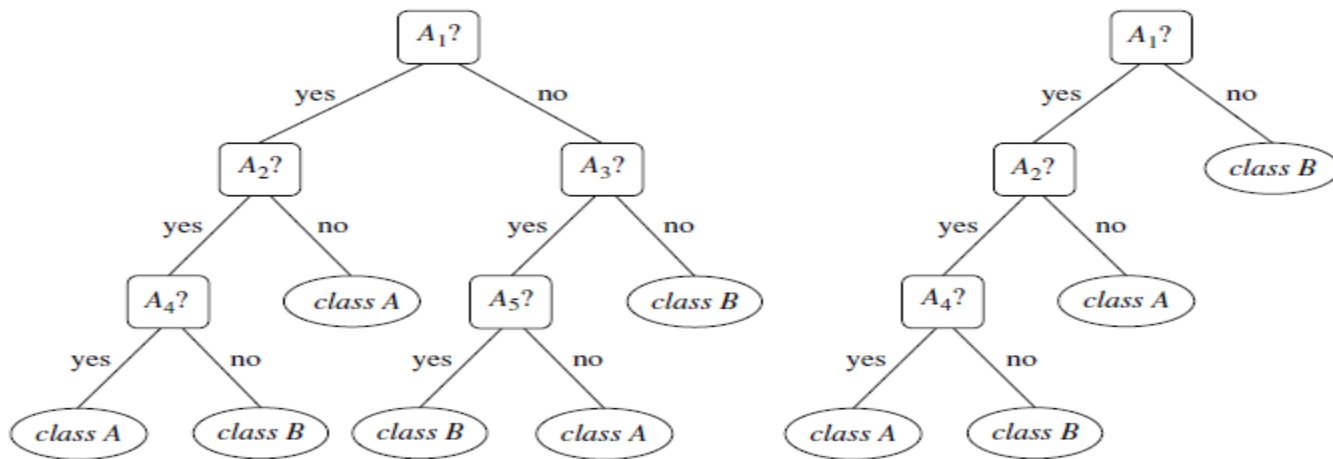
Overfitting and Tree Pruning

- 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
 - Statistical significance, information gain, Gini index to be used for setting threshold
 - Difficult to choose an appropriate threshold
 - High Threshold – over simplified trees
 - Low Threshold – more splits, large trees with very little simplification

Overfitting and Tree Pruning

- Pruning technique

- Postpruning: Remove branches from a “fully grown”
 - Replace branches by a leaf node labeled with most frequent class among the subtree
 -



Overfitting and Tree Pruning

- PostPruning technique

1. Cost complexity pruning: used by CART

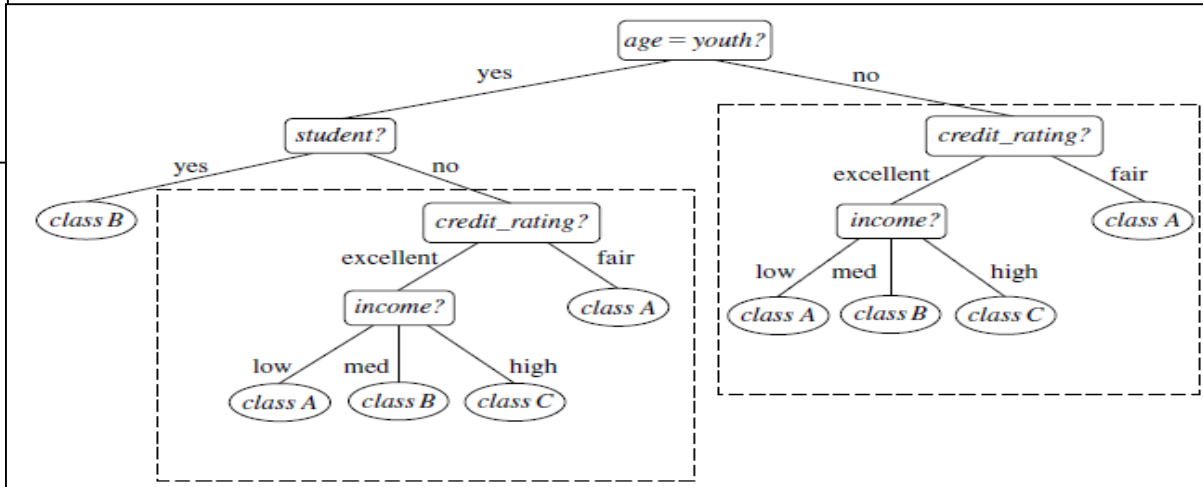
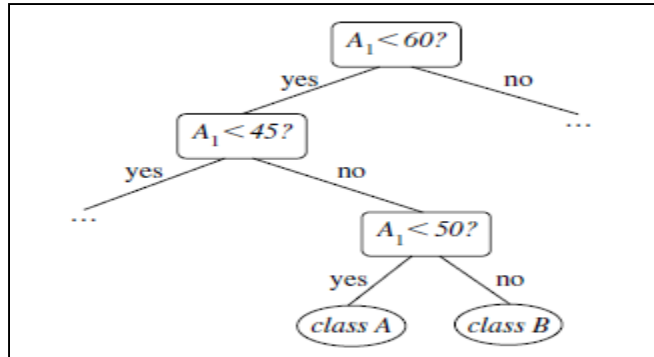
- Cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree).
- It starts from the bottom of the tree.
- For each internal node, N , it computes the cost complexity of the subtree at N , and the cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node).
- The two values are compared.
- If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned.
- Otherwise, it is kept.
- A separate pruning set is used to compute the cost complexity.

Overfitting and Tree Pruning

- PostPruning technique
 2. Pessimistic pruning: used by C4.5 similar to cost complexity
 - It also uses error rate estimates to make decisions regarding subtree pruning.
 - However, does not require the use of a prune set.
 - Instead, it uses the training set to estimate error rates.
 - Adjusts the error rates obtained from the training set by adding a penalty, so as to counter the bias incurred by using the training set as the prune set.
- Prune trees based on the number of bits required to encode them.
 - The “best” pruned tree is the one that minimizes the number of encoding bits.
 - This method adopts the MDL principle
- Which pruning is most suitable???
 - Postpruning requires more computation than prepruning
 - Interleaved pre and post pruning can be done
 - No one technique is best

Problems with Decision Trees


- Repetition and replication



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

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification 
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

Using IF-THEN Rules for Classification

- Represent the knowledge in the form of **IF-THEN** rules
- R: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes
 - Rule antecedent/precondition vs. rule consequent
- If the condition (i.e., all the attribute tests) in a rule antecedent holds true for a given tuple, we say that the rule antecedent is **satisfied** (or simply, that the rule is satisfied) and that the rule **covers** the tuple.
- Assessment of a rule: *coverage* and *accuracy*
 - n_{covers} = # of tuples covered by R
 - n_{correct} = # of tuples correctly classified by R

$$\text{coverage}(R) = \frac{n_{\text{covers}}}{|D|}$$
$$\text{accuracy}(R) = \frac{n_{\text{correct}}}{n_{\text{covers}}}$$

■

<i>RID</i>	<i>age</i>	<i>income</i>	<i>student</i>	<i>credit_rating</i>	<i>Class: buys_computer</i>
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$coverage(R) = \frac{n_{covers}}{|D|}$$

$$accuracy(R) = \frac{n_{correct}}{n_{covers}}$$

Example

- R1: (age = youth) \wedge (student = yes) \rightarrow (buys_computer = yes)
- R1 covers 2/14 tuples and correctly classify both the tuples.
 - Coverage(R1) = 2/14 = 14.28%
 - Accuracy(R1) = 2/2 = 100%

Using IF-THEN Rules for Classification

- **Predicting class label** using If-Then Rules:
 - If a rule is satisfied by X, then the rule is said to be triggered.
 - E.g. X = (age=youth, income=medium, student = yes, credit_rating = fair)
 - X **satisfies** R1, which **triggers** the rule
 - If R1 is the only rule, then the rule is **fired and class label is returned as prediction**.
 - If more than one rule is triggered, need **conflict resolution**
- **Conflict resolution – Size ordering and rule ordering**
- **Size ordering:**
 - assign the highest priority to the triggering rules that has the “toughest” requirement
 - Toughness is measured by rule antecedent size. i.e., triggering rule with the *most attribute test*

Using IF-THEN Rules for Classification

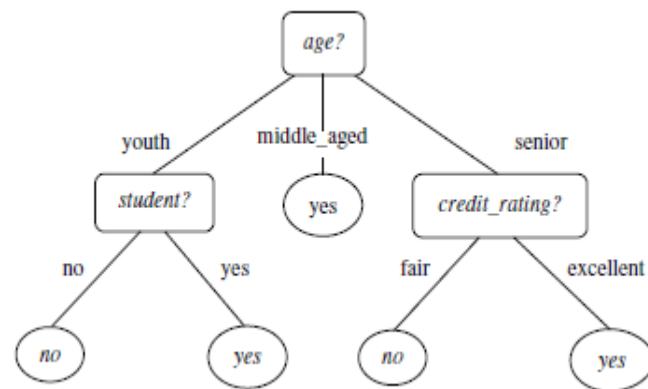
- **Rule ordering:**
 - Prioritizes the rules beforehand
 - **Class-based ordering:**
 - Classes are sorted in decreasing order of importance i.e. *order of prevalence or misclassification cost per class*
 - Rules with more prevalent class come first followed by next prevalent and so on.
 - Or rules with least misclassification cost come first and so on
 - Within each class rules are not ordered (as they predict the same class)
 - **Rule-based ordering (decision list):**
 - rules are organized into one long priority list, according to some measure of rule quality such as quality, accuracy or coverage or based on advice by experts
 - With rule ordering, the triggering rule that appears earliest in the list has the highest priority, and so it gets to fire its class prediction. Any other rule that satisfies X is ignored.
 - In class-based, rules are unordered. They can be applied in any order. Each rule represents a standalone nugget or piece of knowledge
 - In rule-based, rules are applied in strict prescribed order to avoid conflicts. Each rule negates, the one that comes before it. E.g. decision tree rules

Rule Extraction from a Decision Tree

- Rules are easier to understand than large trees
- 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 holds the class prediction
- Rules are mutually exclusive and exhaustive
- Example: Rule extraction from our *buys_computer* decision-tree

IF *age* = young AND *student* = no THEN *buys_computer* = no
IF *age* = young AND *student* = yes THEN *buys_computer* = yes
IF *age* = mid-age THEN *buys_computer* = yes
IF *age* = old AND *credit_rating* = excellent THEN *buys_computer* = yes
IF *age* = young AND *credit_rating* = fair THEN *buys_computer* = no

- Rule pruning to be used
 - C4.5 uses pessimistic pruning i.e. training set with their class labels for rule pruning
 - For resolving conflicts - class-based ordering scheme is used.

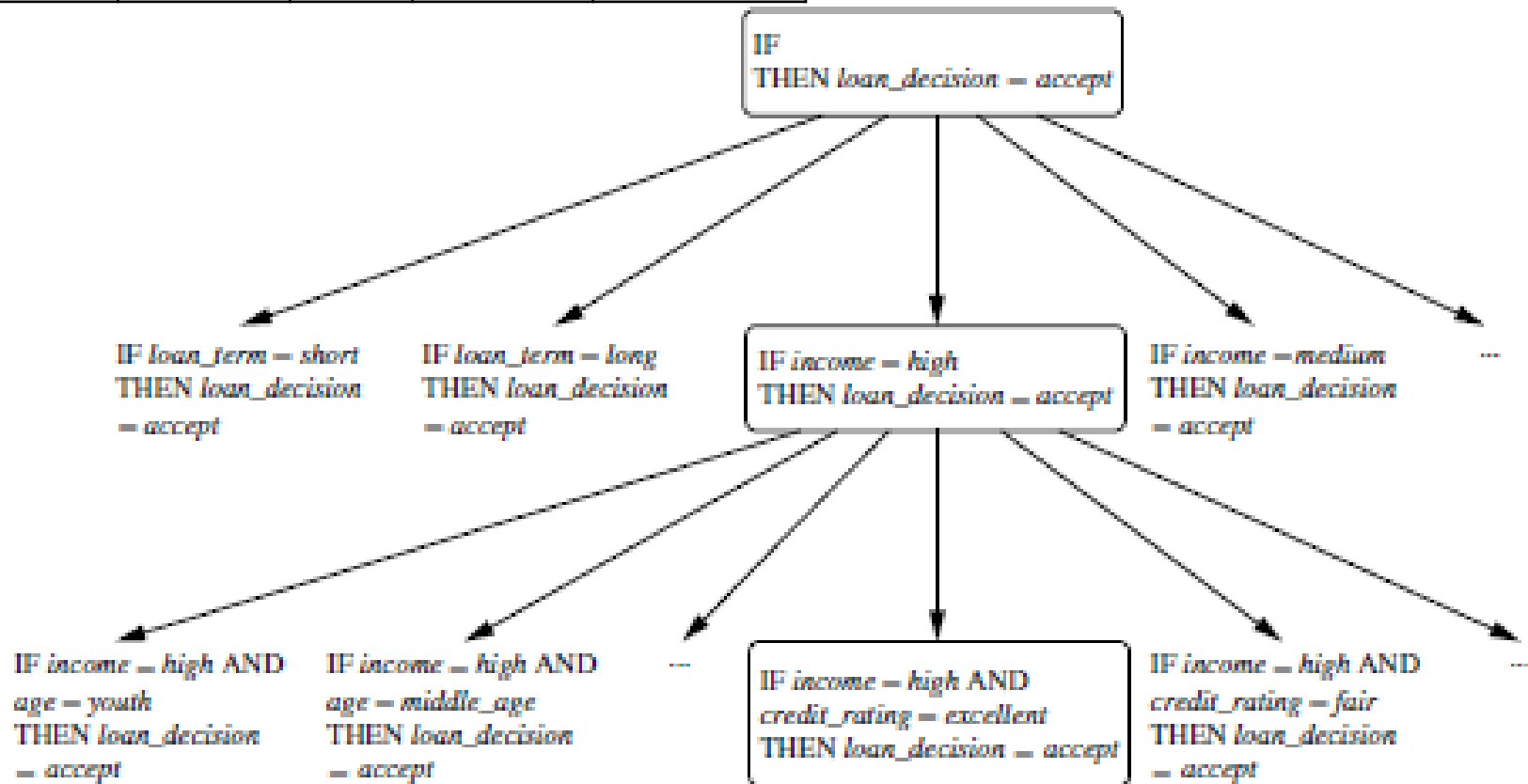


Rule Induction using Sequential Covering Algorithm

- Extracts rules directly from training data
- Typical sequential covering algorithms: FOIL, AQ, CN2, RIPPER
- Rules are learned *sequentially*, each for a given class C_i will cover many tuples of C_i but none (or few) of the tuples of other classes
- Steps:
 - Rules are learned one at a time
 - Each time a rule is learned, the tuples covered by the rules are removed
 - The process repeats on the remaining tuples unless *termination condition*, e.g., when no more training examples or when the quality of a rule returned is below a user-specified threshold
- Comp. w. decision-tree induction: learning a set of rules *simultaneously*

How rules are learnt

Loan_term	Income	Age	Credit_rating	Loan_decision
—				



How to Learn-One-Rule?

- Start with the most general rule possible: condition = empty
- Adding new attributes by adopting a greedy depth-first strategy
 - Picks the one that most improves the rule quality
- Rule-Quality measures: consider both coverage and accuracy
 - Foil-gain (in FOIL & RIPPER): assesses info_gain by extending condition

$$FOIL_Gain = pos' \times (\log_2 \frac{pos'}{pos' + neg'}) - \log_2 \frac{pos}{pos + neg}$$

It favors rules that have high accuracy and cover many positive tuples


- Rule pruning based on an independent set of test tuples

$$FOIL_Prune(R) = \frac{pos - neg}{pos + neg}$$

Pos/neg are # of positive/negative tuples covered by R.

If *FOIL_Prune* is higher for the pruned version of R, prune R

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
 - Issues regarding classification and prediction
 - Classification by decision tree induction
 - Bayesian classification
 - Rule-based classification
 - Classification by back propagation
 - Support Vector Machines (SVM)
 - Associative classification
 - Lazy learners (or learning from your neighbors)
 - Other classification methods
 - Prediction
 - Accuracy and error measures
 - Ensemble methods
 - Model selection
 - Summary
- 

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

Linear Regression

- Linear regression: involves a response variable y and a single predictor variable x

$$y = w_0 + w_1 x$$

where w_0 (y-intercept) and w_1 (slope) are regression coefficients

- Method of least squares: estimates the best-fitting straight line

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2} \quad w_0 = \bar{y} - w_1 \bar{x}$$

- Multiple linear regression: involves more than one predictor variable
 - Training data is of the form $(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \dots, (\mathbf{X}_{|D|}, y_{|D|})$
 - Ex. For 2-D data, we may have: $y = w_0 + w_1 x_1 + w_2 x_2$
 - Solvable by extension of least square method or using SAS, S-Plus
 - Many nonlinear functions can be transformed into the above

Nonlinear Regression

- Some nonlinear models can be modeled by a polynomial function
- A polynomial regression model can be transformed into linear regression model. For example,

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

convertible to linear with new variables: $x_2 = x^2$, $x_3 = x^3$

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

- Other functions, such as power function, can also be transformed to linear model
- Some models are intractable nonlinear (e.g., sum of exponential terms)
 - possible to obtain least square estimates through extensive calculation on more complex formulae

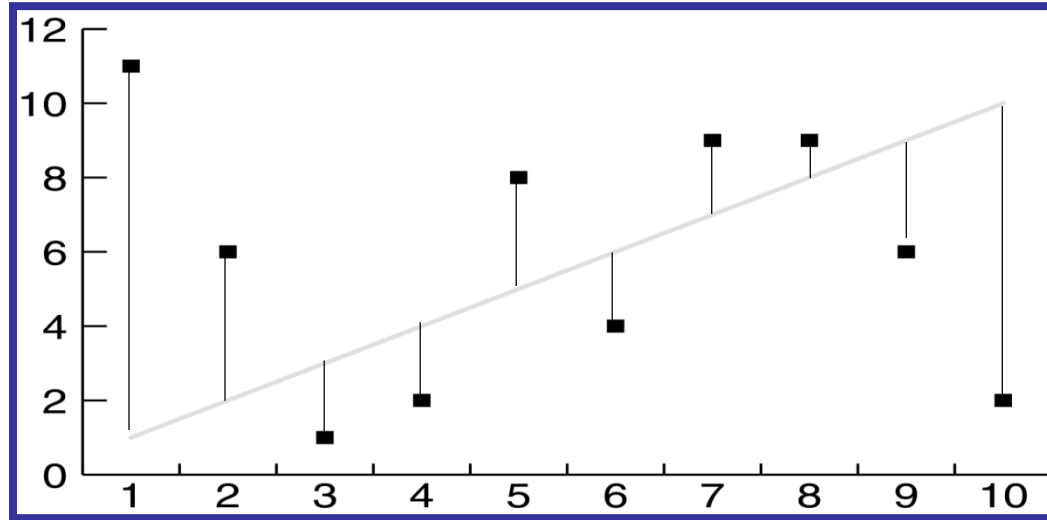
Other Regression-Based Models

- Generalized linear model:
 - Foundation on which linear regression can be applied to modeling categorical response variables
 - Variance of y is a function of the mean value of y , not a constant
 - Logistic regression: models the prob. of some event occurring as a linear function of a set of predictor variables
 - Poisson regression: models the data that exhibit a Poisson distribution
- Log-linear models: (for categorical data)
 - Approximate discrete multidimensional prob. distributions
 - Also useful for data compression and smoothing
- Regression trees and model trees
 - Trees to predict continuous values rather than class labels

Regression Trees and Model Trees

- Regression tree: proposed in CART system (Breiman et al. 1984)
 - CART: Classification And Regression Trees
 - Each leaf stores a *continuous-valued prediction*
 - It is the *average value of the predicted attribute* for the training tuples that reach the leaf
- Model tree: proposed by Quinlan (1992)
 - Each leaf holds a regression model—a multivariate linear equation for the predicted attribute
 - A more general case than regression tree
- Regression and model trees tend to be more accurate than linear regression when the data are not represented well by a simple linear model

Linear Regression Poor Fit



Regression

- Assume data fits a predefined function
- Determine best values for *regression coefficients* c_0, c_1, \dots, c_n .
- Assume an error: $y = c_0 + c_1x_1 + \dots + c_nx_n + \epsilon$
- To minimize the error, least mean squared error is used for training set:

$$y_i = c_0 + c_1x_{1i} + \epsilon_i, i = 1, \dots, k$$

$$L = \sum_{i=1}^k \epsilon_i^2 = \sum_{i=1}^k (y_i - c_0 - c_1x_{1i})^2$$

- Taking derivatives w.r.t c_0 and c_1 , and setting it equal to zero, we can obtain least squares estimates for the coefficients

Classification Using Regression

- ***Division:*** Use regression function to divide area into regions.
- ***Prediction:*** Use regression function to predict a class membership function. Input includes desired class.

Division

Classifying a person as short and medium based on his/her height

Height: {1.6, 1.9, 1.88, 1.7, 1.85, 1.6, 1.7, 1.8, 1.95, 1.9, 1.8, 1.75}

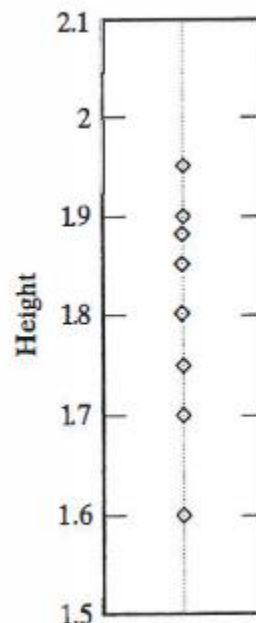
$$y = c_0 + \epsilon$$

We wish to minimize
$$L = \sum_{i=1}^{12} \epsilon_i^2 = \sum_{i=1}^{12} (y_i - c_0)^2$$

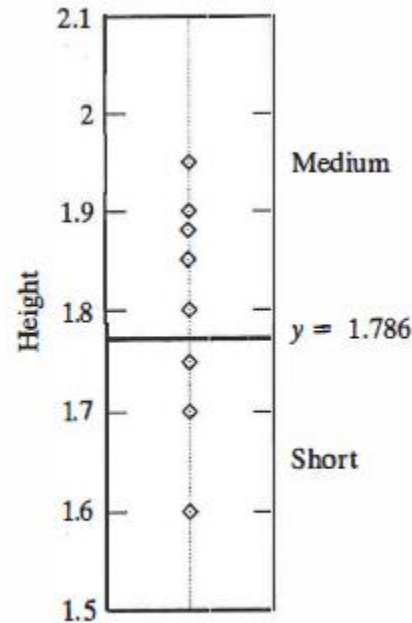
Taking the derivative with respect to c_0 and setting equal to zero we get

$$-2 \sum_{i=1}^{12} y_i + \sum_{i=1}^{12} 2c_0 = 0$$

$$c_0 = \frac{\sum_{i=1}^{12} y_i}{12} = 1.786$$



(a) Short and medium heights



(b) Division

Prediction

$\{ (1.6, 0), (1.9, 1), (1.88, 1), (1.7, 0), (1.85, 1), (1.6, 0), (1.7, 0), (1.8, 1), (1.95, 1), (1.9, 1), (1.8, 1), (1.75, 1) \}$.

$$y = c_0 + c_1 x_1 + \epsilon$$

We thus wish to minimize

$$L = \sum_{i=1}^{12} \epsilon_i^2 = \sum_{i=1}^{12} (y_i - c_0 - c_1 x_{1i})^2$$

Taking the partial derivative with respect to c_0 and setting equal to zero we get

$$\frac{\partial L}{\partial c_0} = -2 \sum_{i=1}^{12} y_i + \sum_{i=1}^{12} 2c_0 + \sum_{i=1}^{12} 2c_1 x_{1i} = 0$$

Solving for c_0 , we find that:

$$c_0 = \frac{\sum y_i - \sum c_1 x_{1i}}{12}$$

Now taking the partial of L with respect to c_1 , substituting the value for c_0 , and setting equal to zero we obtain

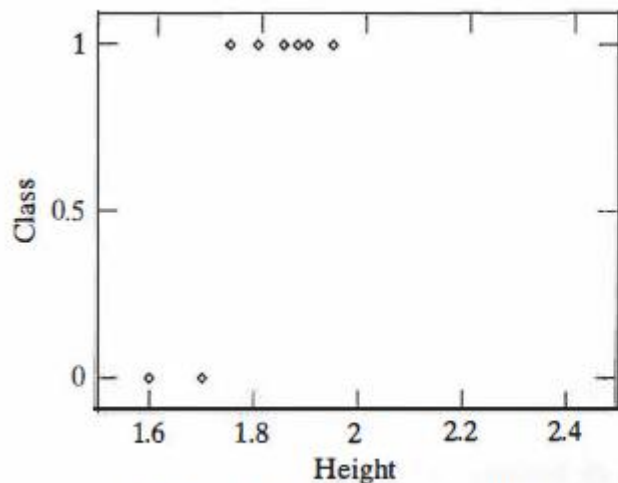
$$\frac{\partial L}{\partial c_1} = 2 \sum (y_i - c_0 - c_1 x_{1i})(-x_{1i}) = 0$$

Solving for c_1 , we finally have

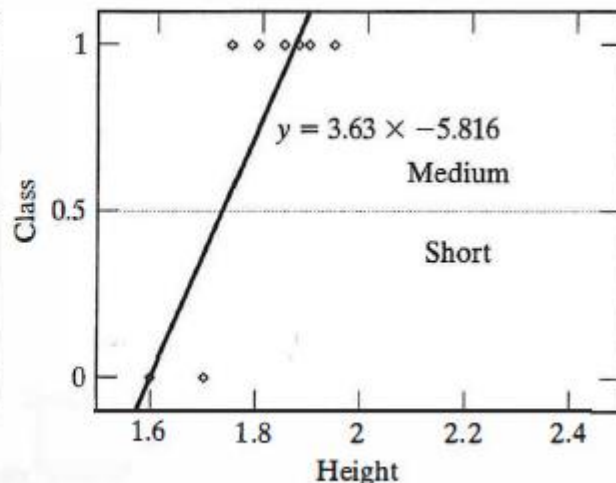
$$c_1 = \frac{\sum (x_{1i} y_i) - \frac{\sum x_{1i} \sum y_i}{12}}{\sum (x_{1i}^2) - \frac{(\sum x_{1i})^2}{12}}$$

$$\sum x_{1i} = 21.43 \quad \sum y_i = 8 \quad \sum (x_{1i} y_i) = 14.83 \quad \sum (x_{1i}^2) = 38.42 \quad c_1 = 3.63 \text{ and } c_0 = -5.816$$

$$y = -5.816 + 3.63x_1$$



(a) Short and medium heights with classes



(b) Prediction

Non-Linear Regression

- If the predictors in the linear regression function are modified by a function (square, square root etc) then the model looks like

$$y = c_0 + f_1(x_1) + \dots + f_n(x_n)$$

- Where f_i is the function used to transform the predictor and is called non-linear regression
- Drawbacks of linear regression:
 - Not applicable to complex data mining application
 - Do not work well with non-numeric data
 - Assume that the input-output variables have a linear relationship
 - The data may not always in a straight line
 - Cannot be used as the probability of occurrence of a class
- Hence, logistic regression is used

Non-Linear Regression

- Univariate logistic regression curve:

$$p = \frac{e^{(c_0 + c_1 x_1)}}{1 + e^{(c_0 + c_1 x_1)}}$$

- The logistic curve gives values between 0 and 1
- So it can be interpreted as probability of class membership
- Like linear regression, it can be used for 2 class classification
- To perform linear regression, apply logarithmic function to obtain the logistic:

$$\log_e \left(\frac{p}{1-p} \right) = c_0 + c_1 x_1$$

- Here, p is probability of being in the class and (1- p) is not being in the class

Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function:** measures the error between. y_i and the predicted value y_i'
 - Absolute error: $|y_i - y_i'|$
 - Squared error: $(y_i - y_i')^2$
- Test error (generalization error): the average loss over the test set

■ Mean absolute error:	$\frac{\sum_{i=1}^d y_i - y_i' }{d}$	Mean squared error:	$\frac{\sum_{i=1}^d (y_i - y_i')^2}{d}$
■ Relative absolute error:	$\frac{\sum_{i=1}^d y_i - y_i' }{\sum_{i=1}^d y_i - \bar{y} }$	Relative squared error:	$\frac{\sum_{i=1}^d (y_i - y_i')^2}{\sum_{i=1}^d (y_i - \bar{y})^2}$

The mean squared-error exaggerates the presence of outliers

Popularly use (square) root mean-square error, similarly, root relative squared error

Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary



Metrics for evaluating classifier

- Training data – overoptimistic
- Testing data - $\sqrt{}$
- Positive tuples – tuples of main class of interest
- Negative tuples – other tuples
- True Positives (TP)
- True Negatives (TN)
- False Positives (FP)
- False Negatives (FN)

Confusion matrix

- Tool for analysing how well your classifier can recognize tuples of different classes
- TP & TN – indicates when classifier is getting things right
- FP & FN – indicates when classifier is getting things wrong

		Predicted class		
		Yes	No	Total
Actual class	Yes	True positive (TP)	False negative (FN)	P
	No	False positive (FP)	True negative (TN)	N
Total		P'	N'	P+N

- Given m classes ($m \geq 2$), confusion matrix is $m \times m$ table
- Entry $CM_{i,j}$ no. of tuples of class i labeled as class j
- Ideal confusion matrix – diagonal entries have more values and others are zero or close to zero

Confusion matrix

		Predicted class		
		Yes	No	Total
Actual class	Yes	True positive (TP)	False negative (FN)	P
	No	False positive (FP)	True negative (TN)	N
Total		P'	N'	P+N

P = positive class tuples (TP + FN)

N = negative class tuples (FP + TN)

P' = tuples classified as positive (TP + FP)

N' = tuples classified as negative (FN + TN)

Total = TP+TN+FP+FN = P+N = P' + N'

Alternative Evaluation Measures

- Sensitivity = TP/P
 - true positive (recognition) rate
 - Proportion of positive tuples that are correctly identified
- Specificity = TN/N
 - true negative rate
 - Proportion of negative tuples that are correctly identified
- accuracy = sensitivity * $P/(P + N)$ + specificity * $N/(P + N)$
- Precision = $TP/(TP + FP)$
 - Measure of exactness
 - What percentage of tuples labeled as positive are actually positive
- Recall = $TP / (TP + FN)$
 - Measure of completeness
 - What percentage of positive tuples are labeled as positive

Evaluation Measures

- Accuracy of a classifier M, $\text{acc}(M)$: percentage of test set tuples that are correctly classified by the model M

$$\text{accuracy} = \frac{TP+TN}{P+N}$$

- Error rate (misclassification rate) of M = $1 - \text{acc}(M)$
- $\text{error rate} = 1 - \text{accuracy} = \frac{FP+FN}{P+N}$
- Accuracy measure works well when the distribution of data is balanced
- For imbalanced data other measures are used e.g. (cancer diagnosis)

classes	buy_computer = yes	buy_computer = no	total	recognition(%)
buy_computer = yes	6954	46	7000	99.34
buy_computer = no	412	2588	3000	86.27
total	7366	2634	10000	95.42

Accuracy = $6954 + 2588 / 10000 = 95.42$

Sensitivity = $6954 / 7000 = 99.34$

Specificity = $2588 / 3000 = 86.27$

Precision = $6954 / 7366 = 94.40$

Recall = $6954 / 7000 = 99.34$

Confusion Matrix for information retrieval	
Ret , Rel	Rel, Not Ret
Ret, Not Rel	Not Rel, Not Ret

Alternative Evaluation Measures

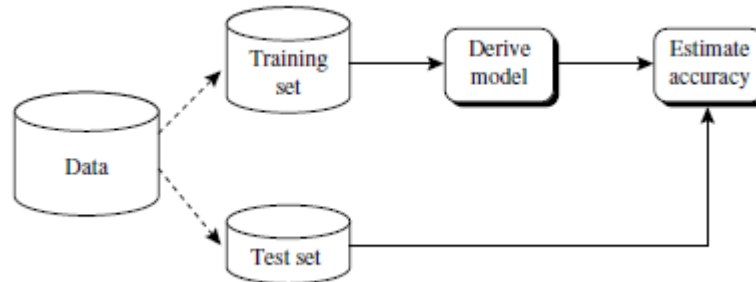
- ■ F-measure:
 - Harmonic mean of precision and recall
 - Gives equal weight to precision and recall
 - $$F = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
- F_β : weighted measure of precision and recall
 - $$F_\beta = \frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$$
 - F_2 – which weights recall twice as much as precision
 - $F_{0.5}$ – which weights precision twice as much as recall

Other aspects to compare classifiers

- Speed – computational costs involved in generating and using the given classifier
- Robustness – ability of classifier to make correct predictions give noisy data or data with missing values
- Scalability – ability to construct the classifier efficiently given large amount of data
- Interpretability – level of understanding and insight provided by classifier
 - E.g. decision tree and classification rules are easy to interpret

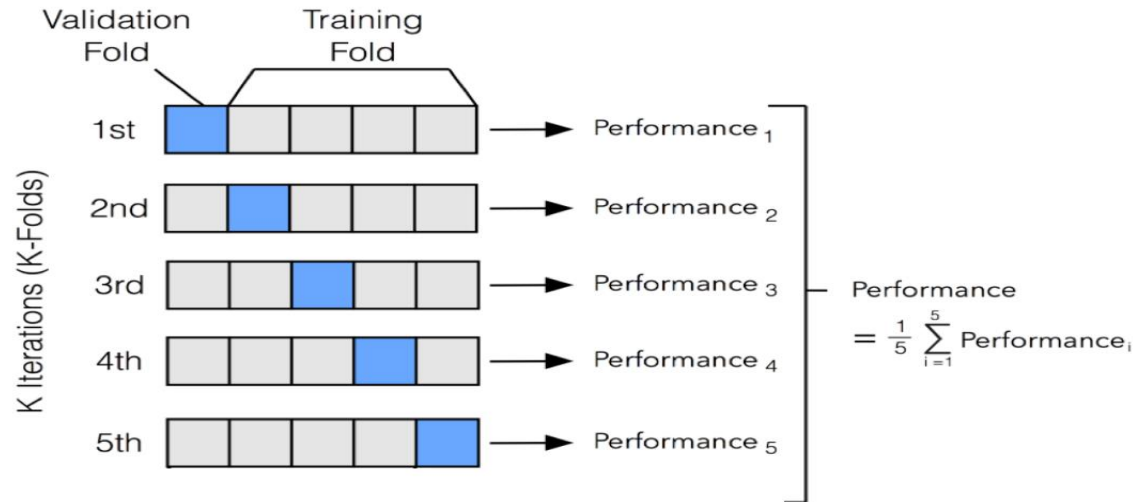
Evaluating the Accuracy of a Classifier or Predictor (I)

- 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 subsampling: a variation of holdout
 - Repeat holdout k times,
 - accuracy = avg. of the accuracies obtained from each iteration



Evaluating the Accuracy of a Classifier or Predictor (I)

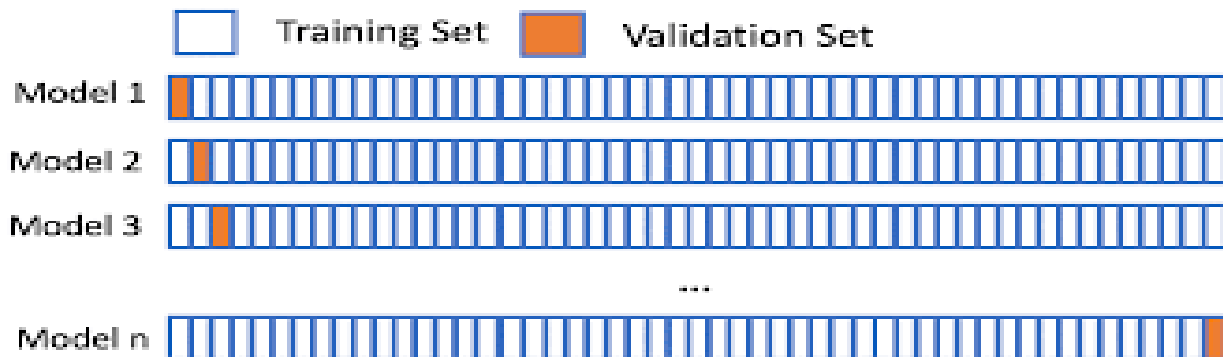
- Cross-validation (k -fold, where $k = 10$ is most popular)
 - Randomly partition the data into k *mutually exclusive* subsets D_1, D_2, \dots, D_k , each approximately equal size
 - At i -th iteration, use D_i as test set and others as training set
 - Each sample is used the same number of times for training and once for testing



Evaluating the Accuracy of a Classifier or Predictor (I)

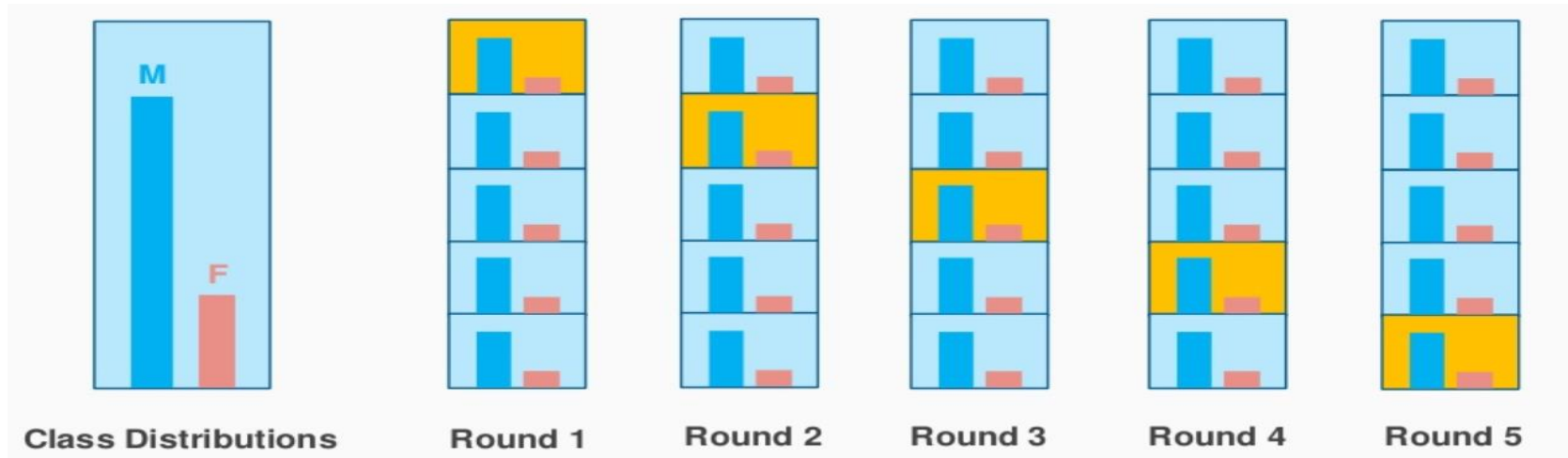
Leave-one-out:

- k folds where $k = \#$ of initial tuples, for small sized data
- One sample is left out at a time for test set



Evaluating the Accuracy of a Classifier or Predictor (I)

- Stratified cross-validation:
 - folds are stratified so that class distribution in each fold is approx. the same as that in the initial data
 - Stratified 10-fold cross-validation is recommended



Different cross validations

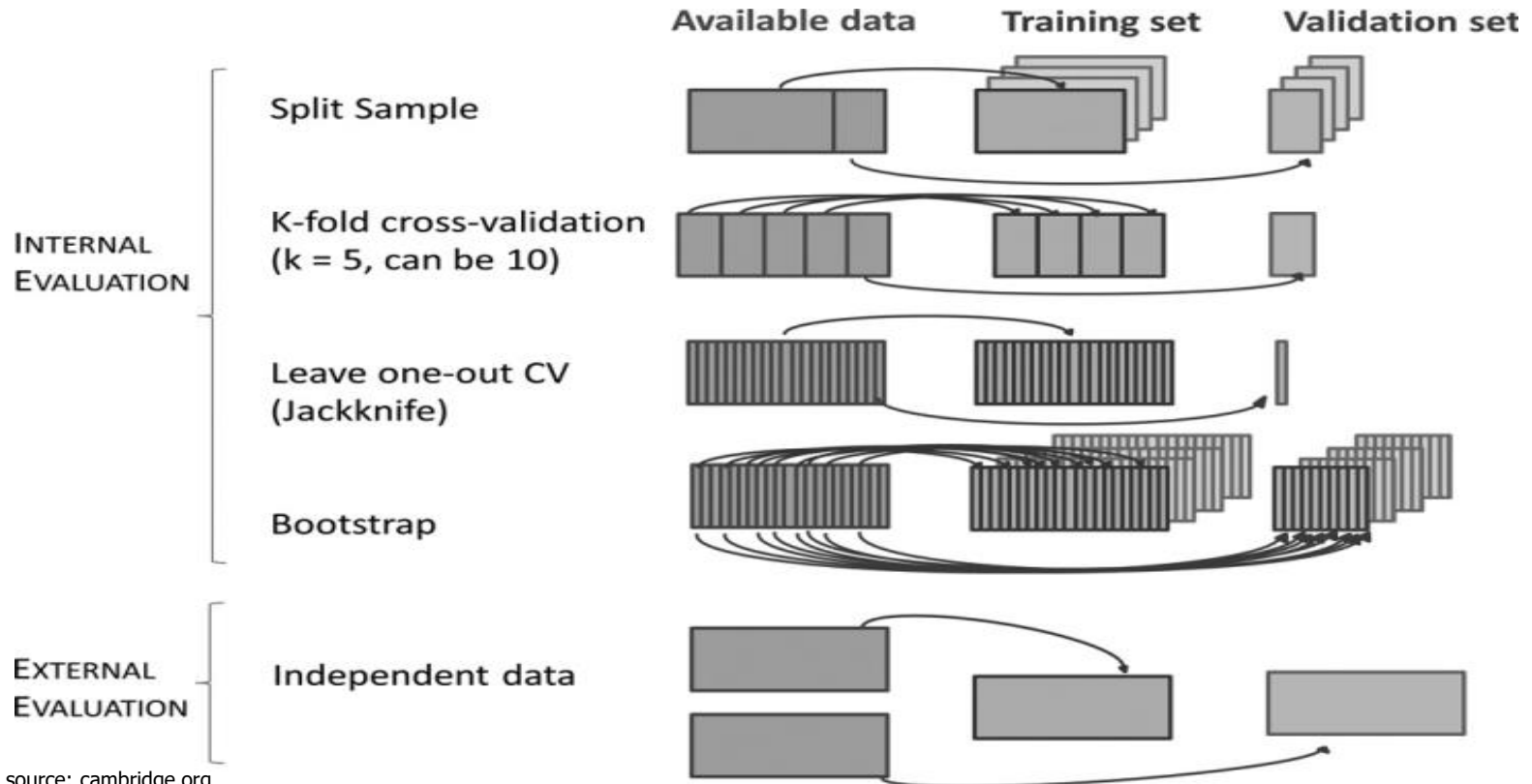


Image source: cambridge.org

October 13, 2022

Kiran Bhowmick

91

Evaluating the Accuracy of a Classifier or Predictor (II)

- 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




Evaluating the Accuracy of a Classifier or Predictor (II)

- Several bootstrap methods, and a common one is **.632 bootstrap**
 - Suppose we are given a data set of d tuples.
 - The data set 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 will end up in the bootstrap, and the remaining 36.8% will 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:

$$acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times acc(M_i)_{test_set} + 0.368 \times acc(M_i)_{train_set})$$

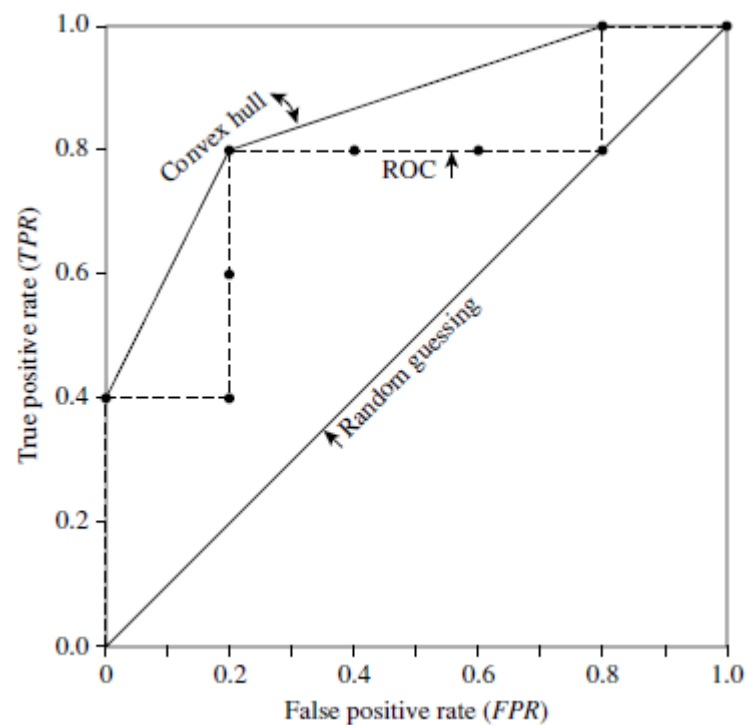
Chapter 6. Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection 
- Summary

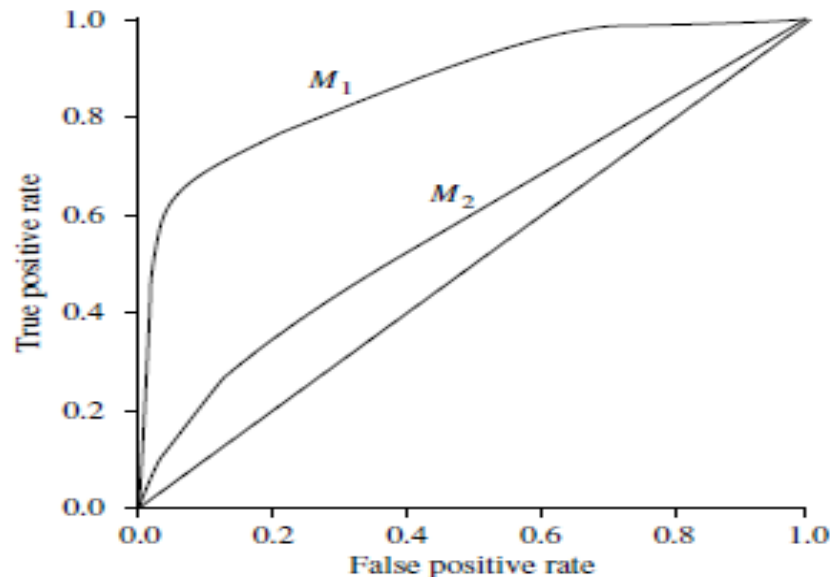
Model Selection: ROC Curves

- 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

<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>TPR</i>	<i>FPR</i>
1	<i>P</i>	0.90	1	0	5	4	0.2	0
2	<i>P</i>	0.80	2	0	5	3	0.4	0
3	<i>N</i>	0.70	2	1	4	3	0.4	0.2
4	<i>P</i>	0.60	3	1	4	2	0.6	0.2
5	<i>P</i>	0.55	4	1	4	1	0.8	0.2
6	<i>N</i>	0.54	4	2	3	1	0.8	0.4
7	<i>N</i>	0.53	4	3	2	1	0.8	0.6
8	<i>N</i>	0.51	4	4	1	1	0.8	0.8
9	<i>P</i>	0.50	5	4	0	1	1.0	0.8
10	<i>N</i>	0.40	5	5	0	0	1.0	1.0




Model Selection: ROC Curves



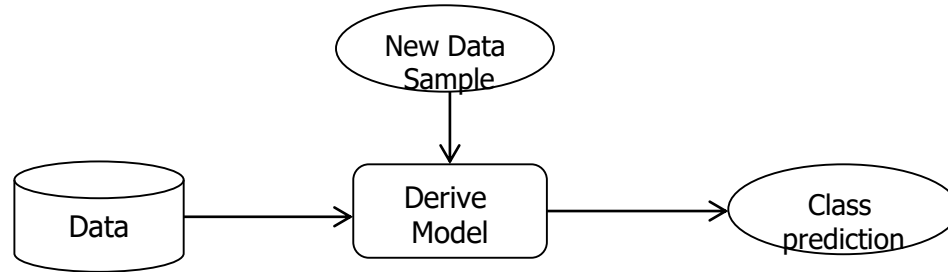
- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line – random guessing
- Measure the area under the curve to assess the accuracy of a model
- The closer the area is to 0.5, the less accurate the corresponding model is.
- A model with perfect accuracy will have an area of 1.0

Chapter 6. Classification and Prediction

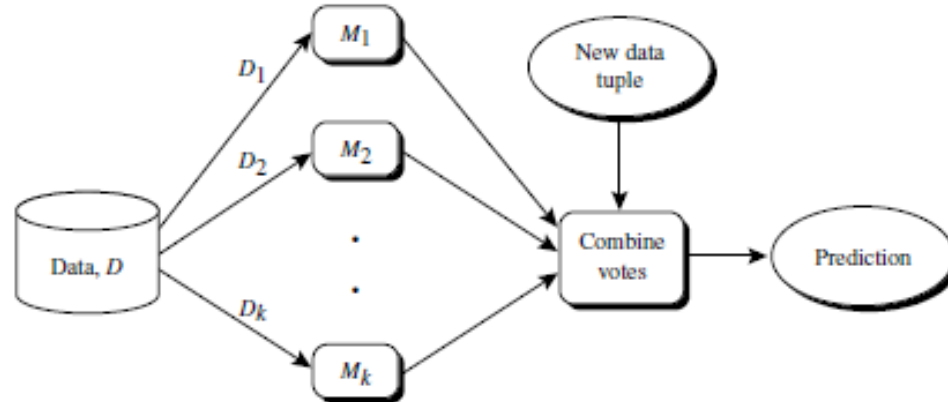
- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification by decision tree induction
- Bayesian classification
- Rule-based classification
- Classification by back propagation
- Support Vector Machines (SVM)
- Associative classification
- Lazy learners (or learning from your neighbors)
- Other classification methods
- Prediction
- Accuracy and error measures
- Ensemble methods 
- Model selection
- Summary

Techniques to improve classifier accuracy

Ensemble Methods



Ensemble
methods



Ensemble Methods

- Ensemble methods
 - Use a combination of models to increase accuracy
 - Combine a series of k learned models, M_1, M_2, \dots, 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: Bootstrap 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 \mathbf{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 \mathbf{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 significant better than a single classifier derived from D
 - For noise data: not considerably worse, more robust
 - Proved improved accuracy in prediction

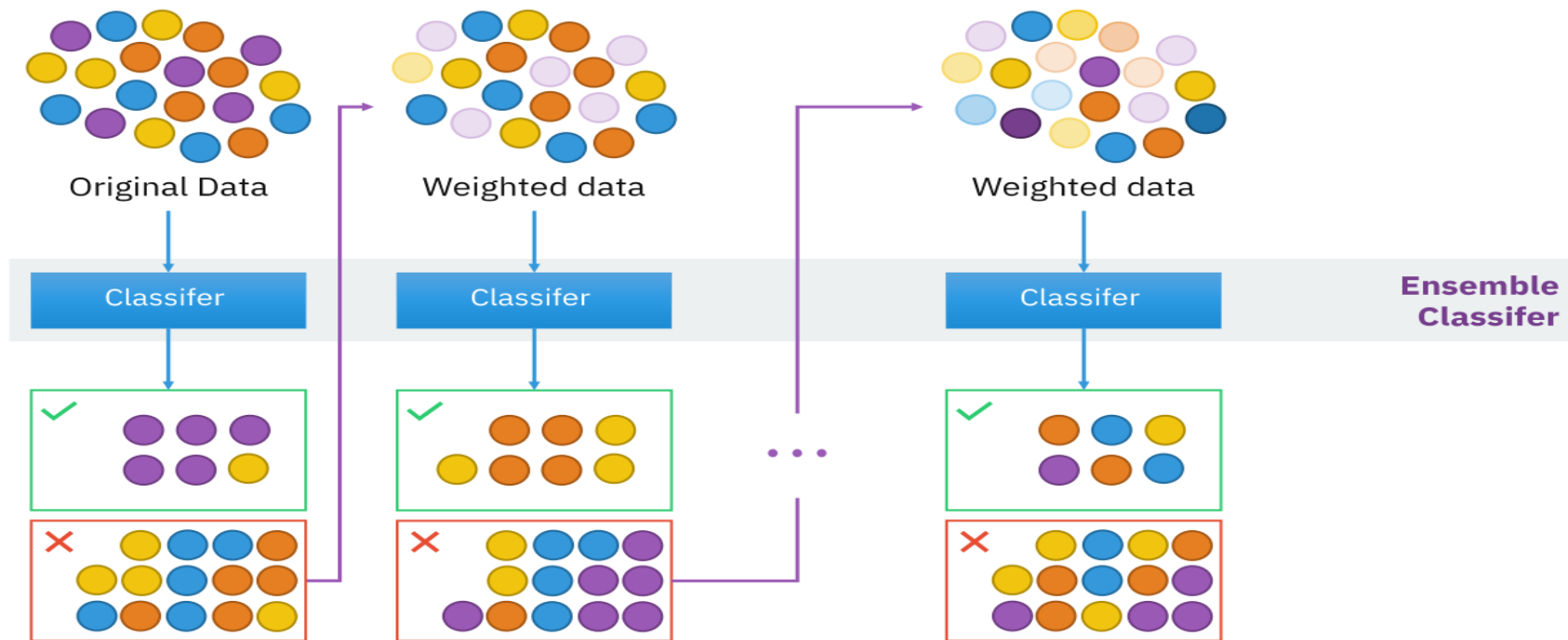


Bagging Classifier Process Flow

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
- 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 misclassified data

Boosting



Adaboost (Freund and Schapire, 1997)

- Given a set of d class-labeled tuples, $(\mathbf{X}_1, y_1), \dots, (\mathbf{X}_d, y_d)$
- 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, else 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:

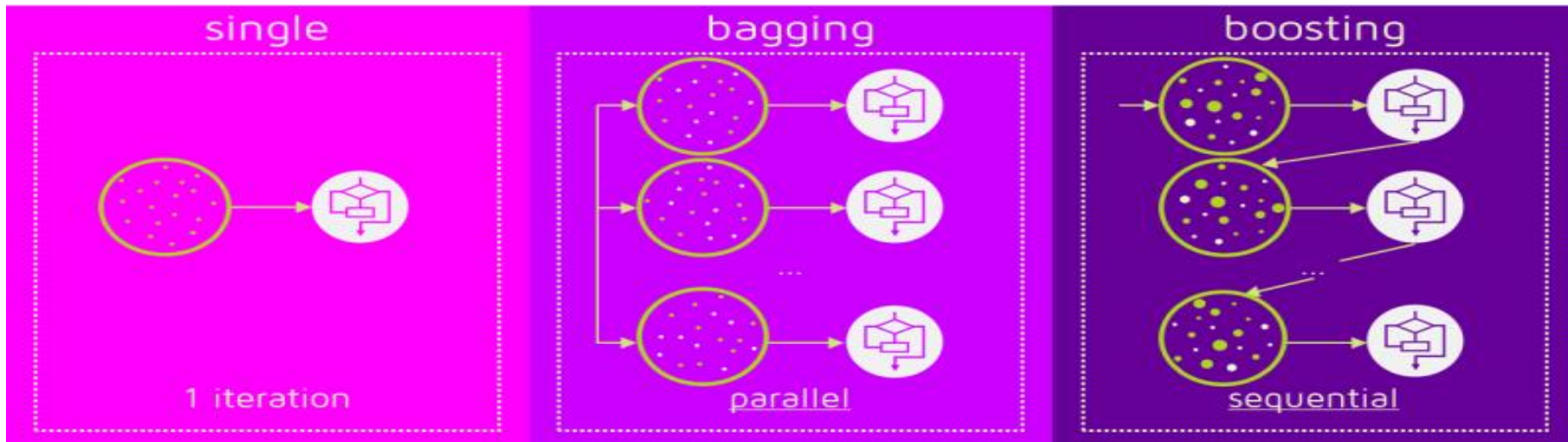
$$error(M_i) = \sum_j^d w_j \times err(\mathbf{X}_j)$$

Adaboost (Freund and Schapire, 1997)

- The weights of the tuples are updated as follows:
- Tuples that are correctly classified, the weight is multiplied by $\text{error}(M_i)/(1-\text{error}(M_i))$
- Once the weights of correctly classified tuples are updated, the weights for all tuples are normalized as
 - $\text{Weight} \times \text{sum (old weights)} / \text{sum (updated weights)}$
- This will increase weights of misclassified tuple and decrease those of correctly classified tuples
- Classifying unseen tuples
 - Weight of each classifier's vote

$$\log \frac{1 - \text{error}(M_i)}{\text{error}(M_i)}$$

- Image source: [Internet](#)



Handling Imbalanced data

- Problem of imbalance – very few samples of minority class
- Applications
 - Fraud detection
 - Spam detection
 - Medical diagnosis
- Majority class ----- Minority class
 - Legal transaction Fraudulent transaction
 - Ham email Spam email
 - Negative cases Positive cases

Handling Imbalanced data

- Methods of handling
 - Data-based methods
 - Oversampling
 - Undersampling
 - Hybrid sampling
 - Algorithmic-based methods
 - Ensemble learning
 - Cost-sensitive learning
 - Hybrid methods