MSCIT 5210: Knowledge Discovery and Data Mining

Acknowledgement: Slides modified by Dr. Lei Chen based on the slides provided by Jiawei Han, Micheline Kamber, Jian Pei and Raymond Wong

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- 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

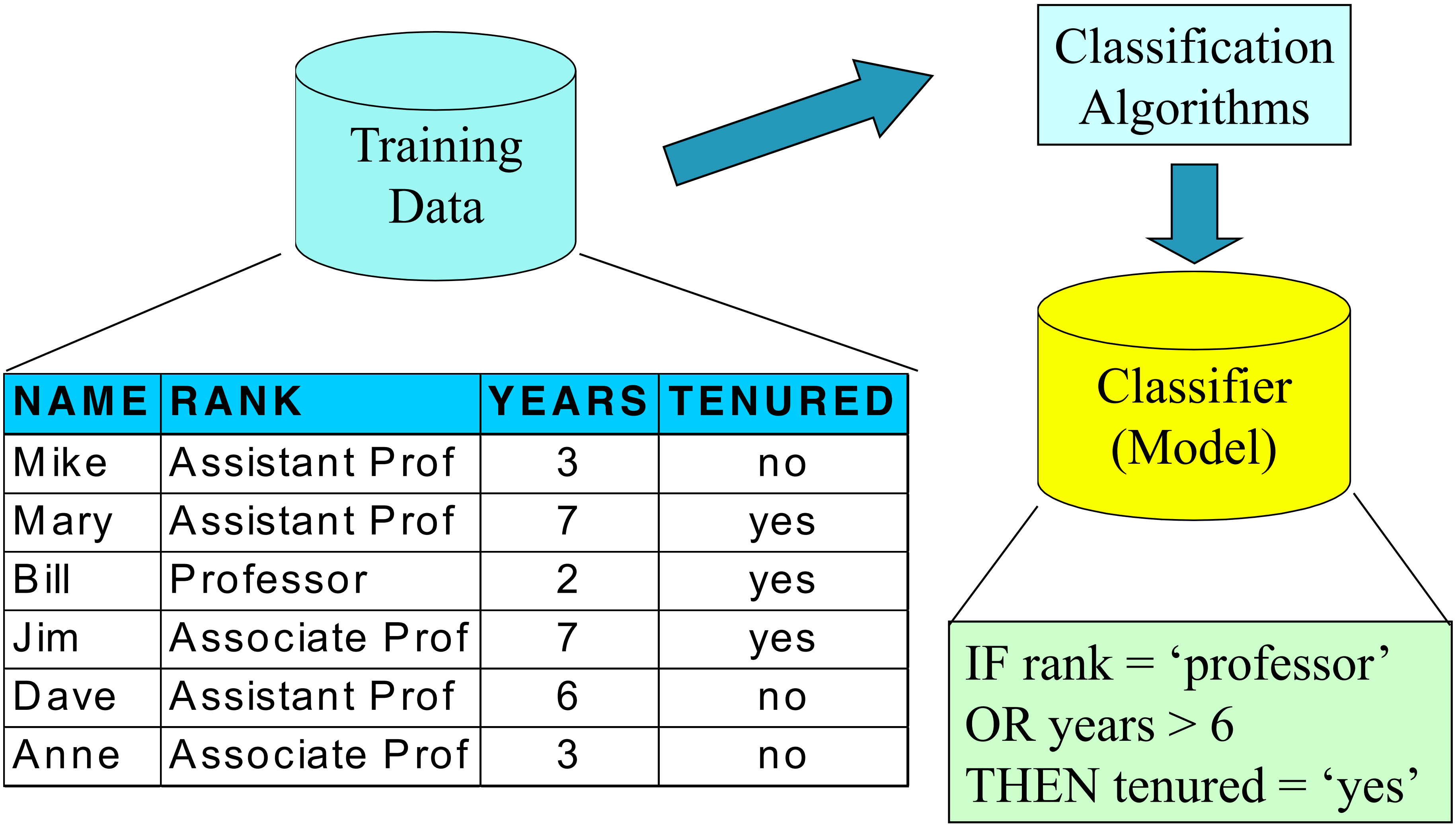
Prediction Problems: Classification vs. Numeric Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Numeric Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values
- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

Classification—A Two-Step Process

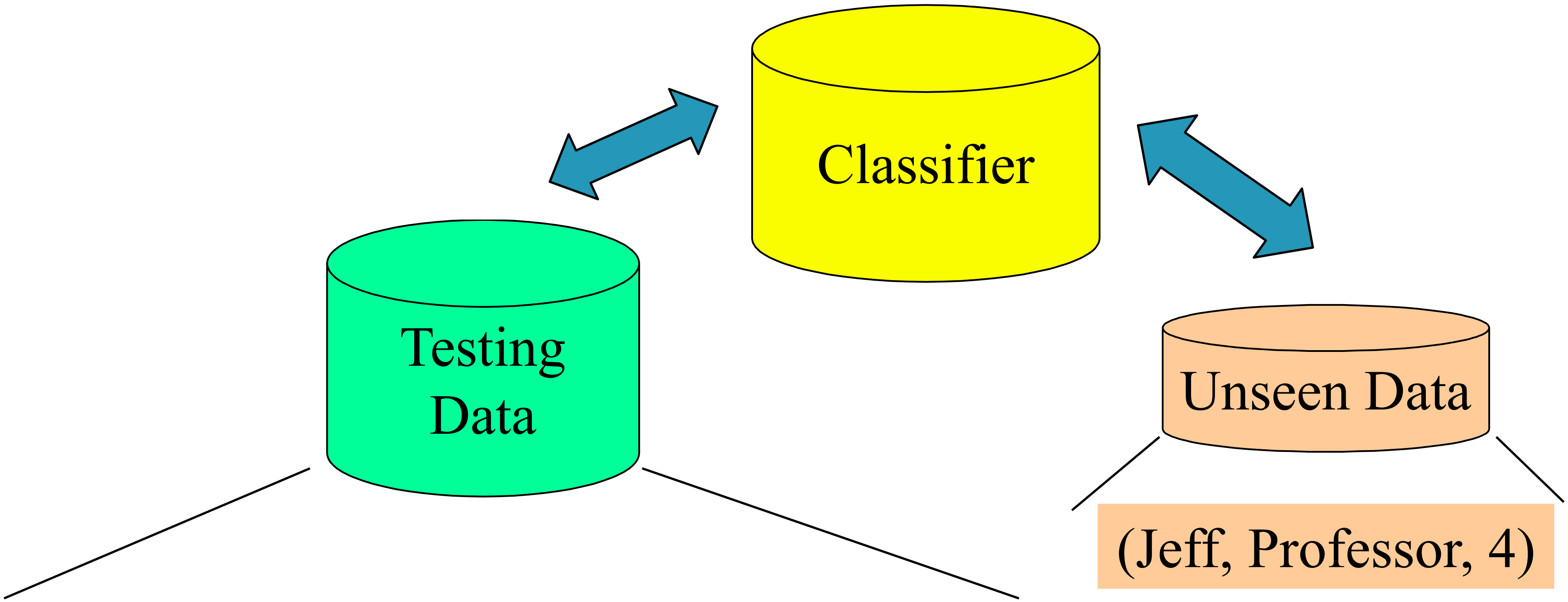
- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data
- Note: If *the test set* is used to select models, it is called validation (test) set

Process (1): Model Construction

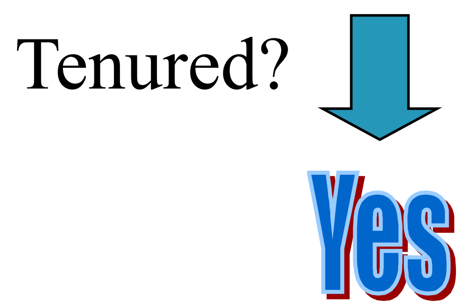


NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Process (2): Using the Model in Prediction



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



Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

The K-Nearest Neighbor Method

- Used for prediction/classification
- Given input x, (e.g., <sunny, normal, ..?>
- #neighbors = K (e.g., k=3)
  - Often a parameter to be determined
    - The form of the distance function
  - K neighbors in training data to the input data x:
    - Break ties arbitrarily
- All k neighbors will vote: majority wins

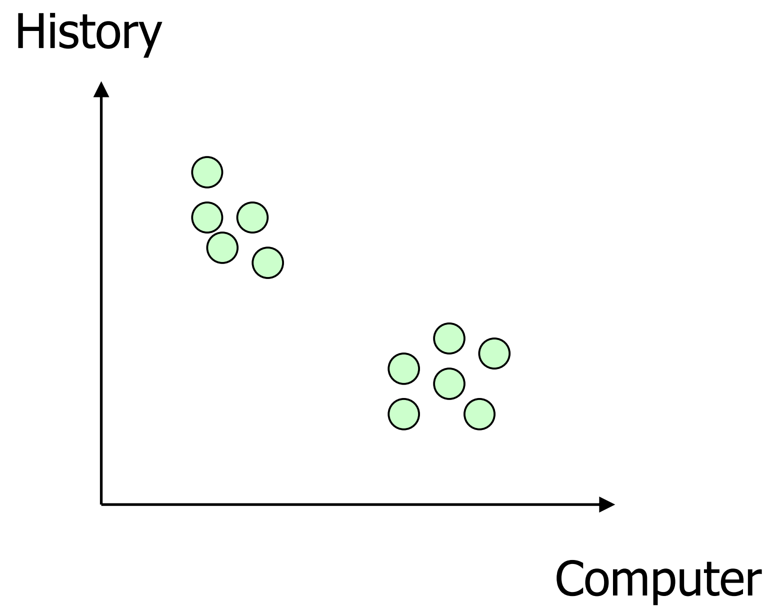
How to decide the distance?

Try 3-NN on this data: assume distance function =‘ # of different attributes.’

	Outlook	Temperature	Humidity	Windy	Play
	sunny	hot	high	FALSE	no
	sunny	hot	high	TRUE	no
	overcast	hot	high	FALSE	yes
	rainy	mild	high	FALSE	yes
	rainy	cool	normal	FALSE	yes
	rainy	cool	normal	TRUE	no
	overcast	cool	normal	TRUE	yes
	sunny	mild	high	FALSE	no
	sunny	cool	normal	FALSE	yes
	rainy	mild	normal	FALSE	yes
	sunny	mild	normal	TRUE	yes
	overcast	mild	high	TRUE	yes
	overcast	hot	normal	FALSE	yes
testing	rainy	mild	high	TRUE	?

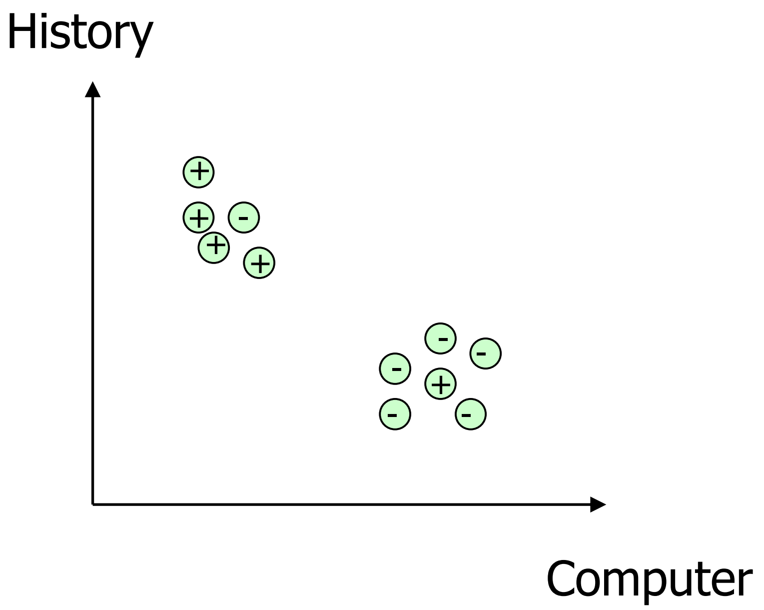
Nearest Neighbor Classifier

Computer	History
100	40
90	45
20	95
…	…



Nearest Neighbor Classifier

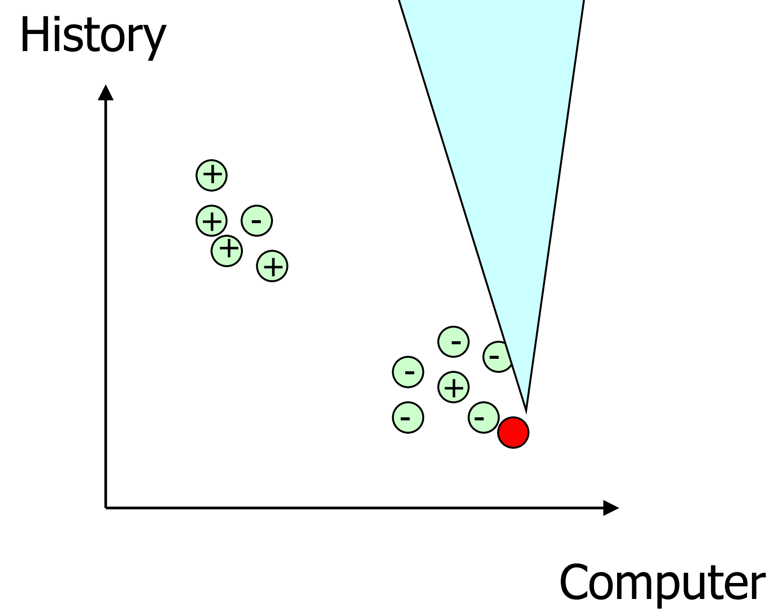
Computer	History	Buy Book?
100	40	No (-)
90	45	Yes (+)
20	95	Yes (+)
…	…	…



Nearest Nei

Nearest Neighbor Classifier:
Step 1: Find the nearest neighbor
Step 2: Use the “label” of this neighbor

Computer	History	Buy Book?
100	40	No (-)
90	45	Yes (+)
20	95	Yes (+)
…	…	…



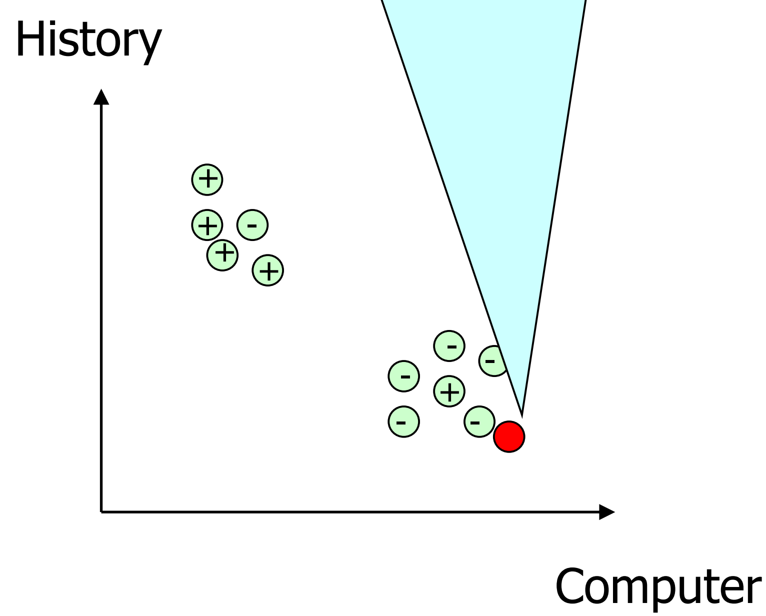
Suppose there is a new person

Computer	History	Buy Book?
95	35	?

Nearest Nei

k-Nearest Neighbor Classifier:
Step 1: Find k nearest neighbors
Step 2: Use the majority of the labels of the neighbors

Computer	History	Buy Book?
100	40	No (-)
90	45	Yes (+)
20	95	Yes (+)
…	…	…



Suppose there is a new person

Computer	History	Buy Book?
95	35	?

Why important?

- Often a baseline
  - Must beat this one to claim innovation
- Forms of KNN
  - Weighted KNN
  - “K” is a variable:
    - Often we experiment with different values of K=1, 3, 5, to find out the optimal one
  - Document similarity
    - Cosine
  - Case based reasoning
    - Edited data base
    - Sometimes, the accuracy (CBR)/accuracy (KNN) can be better than 100%: why?
  - Image understanding
    - Manifold learning
    - Distance metric

K-NN can be misleading

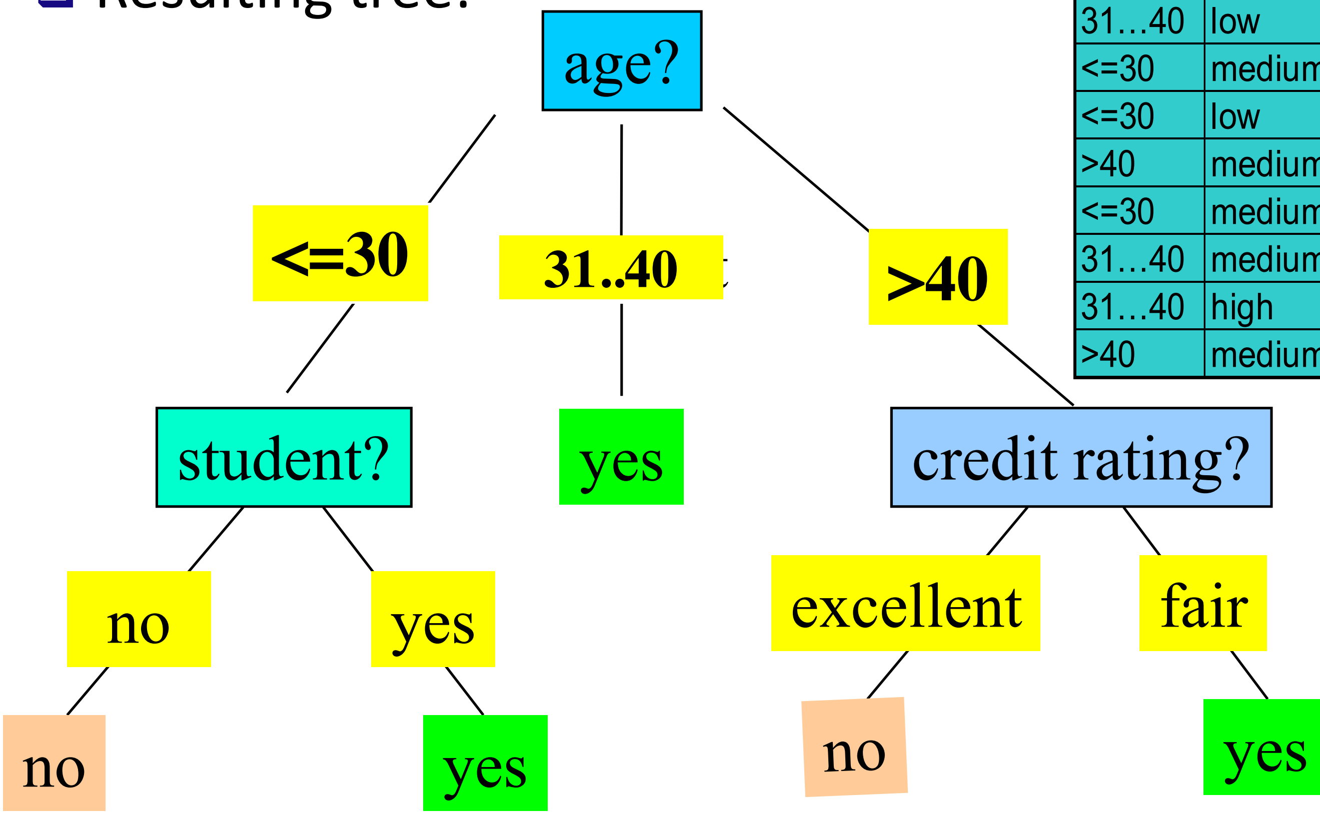
- Consider applying K-NN on the training data
  - What is the accuracy? 100%
  - Why? Distance to self is zero
  - What should we do in testing?
    - Use new data for testing, rather than training data.

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Decision Tree Induction: An Example

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



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

Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

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

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

Computing Information-Gain for Continuous-Valued Attributes

- Let attribute A be a continuous-valued attribute
- Must determine the *best split point* for A
  - Sort the value A in increasing order
  - Typically, the midpoint between each pair of adjacent values is considered as a possible *split point*
    - $(a_i+a_{i+1})/2$ is the midpoint between the values of a_i and a_{i+1}
  - The point with the *minimum expected information requirement* for A is selected as the split-point for A
- Split:
  - D1 is the set of tuples in D satisfying A ≤ split-point, and D2 is the set of tuples in D satisfying A > split-point

Entropy

- **Entropy** is used to measure how informative is a node.
- If we are given a probability distribution $P = (p_1, p_2, \ldots, p_n)$ then the **Information** conveyed by this distribution, also called the **Entropy** of P, is:
  $I(P) = -(p_1 \times \log p_1 + p_2 \times \log p_2 + \ldots + p_n \times \log p_n)$
- All logarithms here are in base 2.

Entropy

- For example,
  - If P is (0.5, 0.5), then I(P) is 1.
  - If P is (0.67, 0.33), then I(P) is 0.92,
  - If P is (1, 0), then I(P) is 0.
- The **entropy** is a way to measure the amount of information.
- The smaller the entropy, the more informative we have.

Entrop

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

$\text{Info}(T) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2}$
$= 1$

For attribute Race,

$\text{Info}(T_{black}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.8113$

$\text{Info}(T_{white}) = -\frac{3}{4}\log\frac{3}{4} - \frac{1}{4}\log\frac{1}{4} = 0.8113$

$\text{Info}(\text{Race}, T) = \frac{1}{2} \times \text{Info}(T_{black}) + \frac{1}{2} \times \text{Info}(T_{white}) = 0.8113$

$\text{Gain}(\text{Race}, T) = \text{Info}(T) - \text{Info}(\text{Race}, T) = 1 - 0.8113 = 0.1887$

For attribute Race, $\text{Gain}(\text{Race}, T) = 0.1887$

Entrop

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

$$\text{Info}(T) = - \tfrac{1}{2} \log \tfrac{1}{2} - \tfrac{1}{2} \log \tfrac{1}{2}$$
$$= 1$$

For attribute Income,

$$\text{Info}(T_{high}) = - 1 \log 1 – 0 \log 0 = 0$$

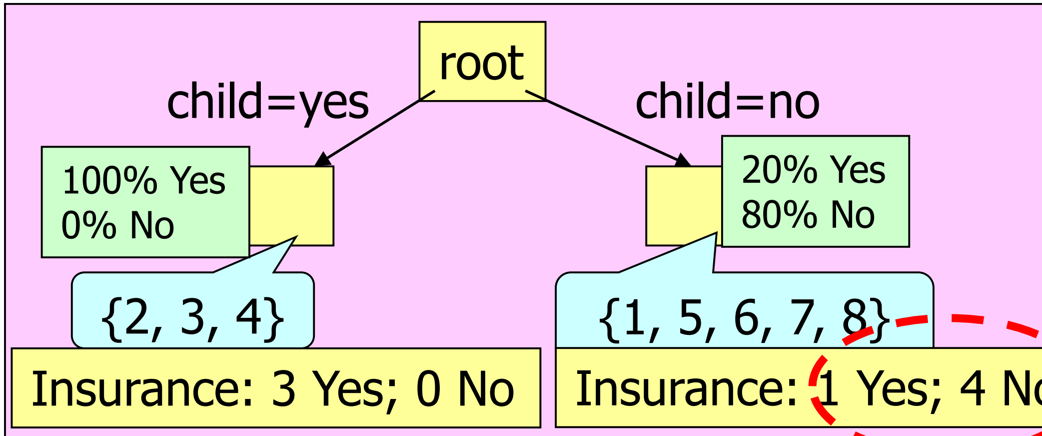
$$\text{Info}(T_{low}) = - 1/3 \log 1/3 – 2/3 \log 2/3 = 0.9183$$

$$\text{Info}(\text{Income}, T) = \tfrac{1}{4} \times \text{Info}(T_{high}) + \tfrac{3}{4} \times \text{Info}(T_{low}) = 0.6887$$

$$\text{Gain}(\text{Income}, T) = \text{Info}(T) – \text{Info}(\text{Income}, T) = 1 – 0.6887 = 0.3113$$

For attribute Race, Gain(Race, T) = 0.1887

For attribute Income, Gain(Income, T) = 0.3113



	Race	Income	Child	Insurance
1	black	high	no	yes
2	white	high	yes	yes
3	white	low	yes	yes
4	white	low	yes	yes
5	black	low	no	no
6	black	low	no	no
7	black	low	no	no
8	white	low	no	no

$\text{Info}(T) = - ½ \log ½ - ½ \log ½$
$= 1$

For attribute Child,

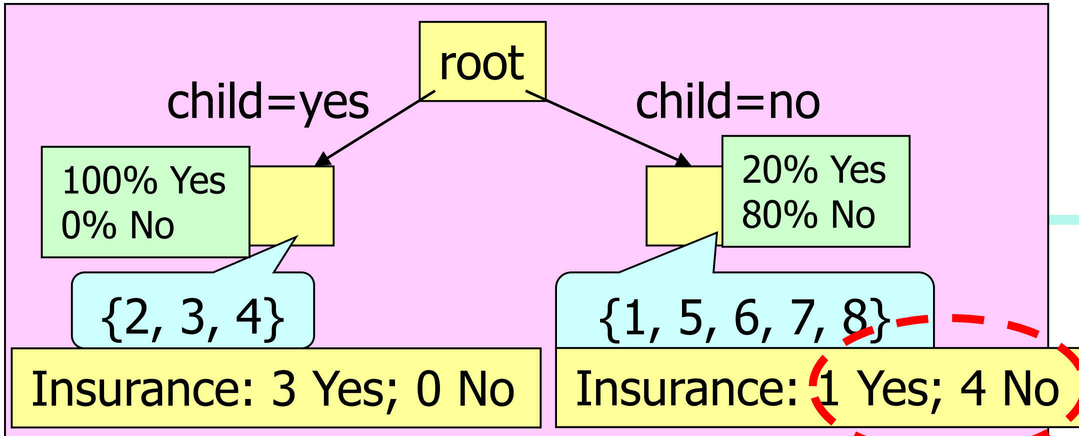
$\text{Info}(T_{yes}) = - 1 \log 1 – 0 \log 0 \quad = 0$

$\text{Info}(T_{no}) = - 1/5 \log 1/5 – 4/5 \log 4/5 \quad = 0.7219$

$\text{Info}(\text{Child}, T) = 3/8 \times \text{Info}(T_{yes}) + 5/8 \times \text{Info}(T_{no}) \quad = 0.4512$

$\text{Gain}(\text{Child}, T) = \text{Info}(T) – \text{Info}(\text{Child}, T) = 1 – 0.4512 \; = 0.5488$

For attribute Race,	Gain(Race, T) = 0.1887
For attribute Income,	Gain(Income, T) = 0.3113
For attribute Child,	Gain(Child, T) = 0.5488



	Race	Income	Child	Insurance
1	black	high	no	yes
2	white	high	yes	yes
3	white	low	yes	yes
4	white	low	yes	yes
5	black	low	no	no
6	black	low	no	no
7	black	low	no	no
8	white	low	no	no

$$\text{Info}(T) = -\ 1/5 \log 1/5 - 4/5 \log 4/5$$
$$= 0.7219$$

For attribute Race,

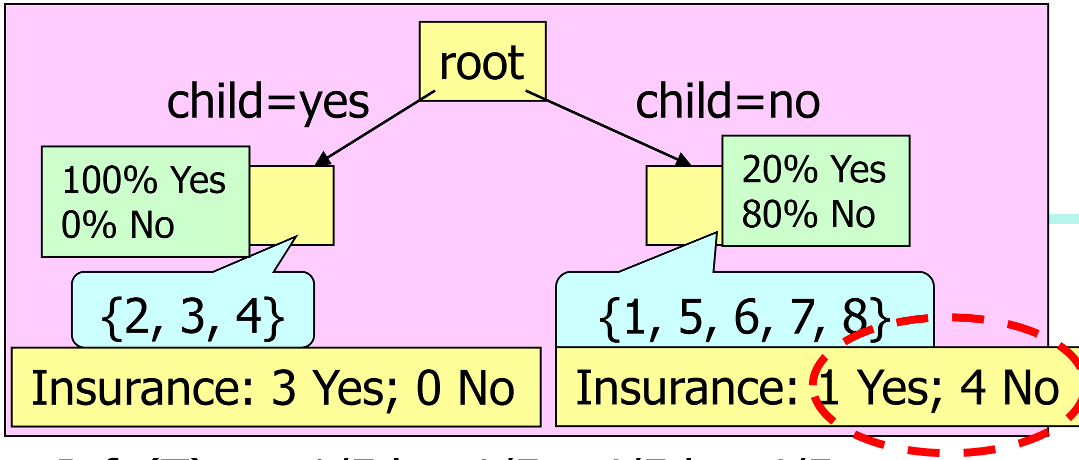
$$\text{Info}(T_{\text{black}}) = -\ ¼ \log ¼ - ¾ \log ¾ \ = 0.8113$$

$$\text{Info}(T_{\text{white}}) = -\ 0 \log 0 - 1 \log 1 \ = 0$$

$$\text{Info}(\text{Race}, T) = 4/5 \times \text{Info}(T_{\text{black}}) + 1/5 \times \text{Info}(T_{\text{white}}) \ = 0.6490$$

$$\text{Gain}(\text{Race}, T) = \text{Info}(T) - \text{Info}(\text{Race}, T) = 0.7219 - 0.6490 \ = 0.0729$$

For attribute Race, Gain(Race, T) = 0.0729



	Race	Income	Child	Insurance
1	black	high	no	yes
2	white	high	yes	yes
3	white	low	yes	yes
4	white	low	yes	yes
5	black	low	no	no
6	black	low	no	no
7	black	low	no	no
8	white	low	no	no

$$\text{Info}(T) = -\ 1/5 \log 1/5 - 4/5 \log 4/5 = 0.7219$$

For attribute Income,

$$\text{Info}(T_{high}) = -\ 1 \log 1 - 0 \log 0 \ = 0$$

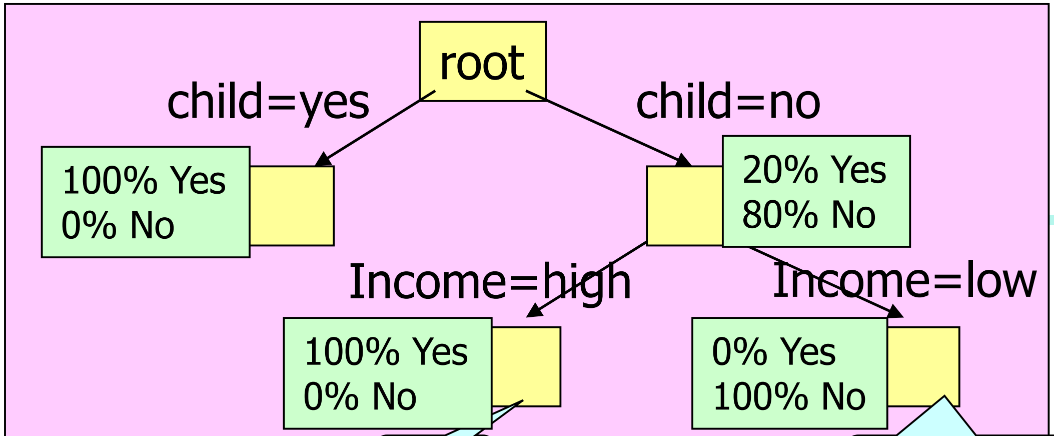
$$\text{Info}(T_{low}) = -\ 0 \log 0 - 1 \log 1 \ = 0$$

$$\text{Info}(\text{Income}, T) = 1/5 \times \text{Info}(T_{high}) + 4/5 \times \text{Info}(T_{low}) = 0$$

$$\text{Gain}(\text{Income}, T) = \text{Info}(T) - \text{Info}(\text{Income}, T) = 0.7219 - 0 = 0.7219$$

For attribute Race, Gain(Race, T) = 0.0729

For attribute Income, Gain(Income, T) = 0.7219



	Race	Income	Child	Insurance
1	black	high	no	yes
2	white	high	yes	yes
3	white	low	yes	yes
4	white	low	yes	yes
[illegible]	black	low	no	no
[illegible]	[illegible]	low	no	no
7	black	low	no	no
8	white	low	no	no

Info(T) = - 1/5 [illegible] /5 – 4/5 log 4/5

{1}

{5, 6, 7, 8}

Insurance: 1 Yes; 0 No

Insurance: 0 Yes; 4 No

For attribute Income,

$$\text{Info}(T_{high}) = - 1 \log 1 – 0 \log 0 = 0$$

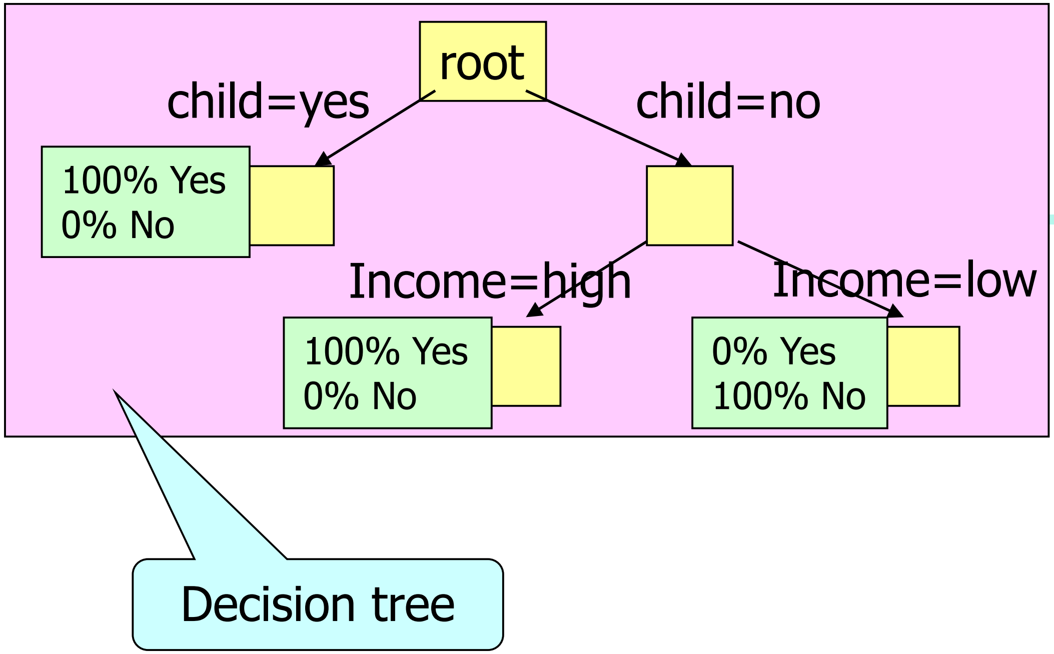
$$\text{Info}(T_{low}) = - 0 \log 0 – 1 \log 1 = 0$$

$$\text{Info(Income, T)} = 1/5 \times \text{Info}(T_{high}) + 4/5 \times \text{Info}(T_{low}) = 0$$

$$\text{Gain(Income, T)} = \text{Info(T)} – \text{Info(Income, T)} = 0.7219 – 0 = 0.7219$$

For attribute Race, Gain(Race, T) = 0.0729

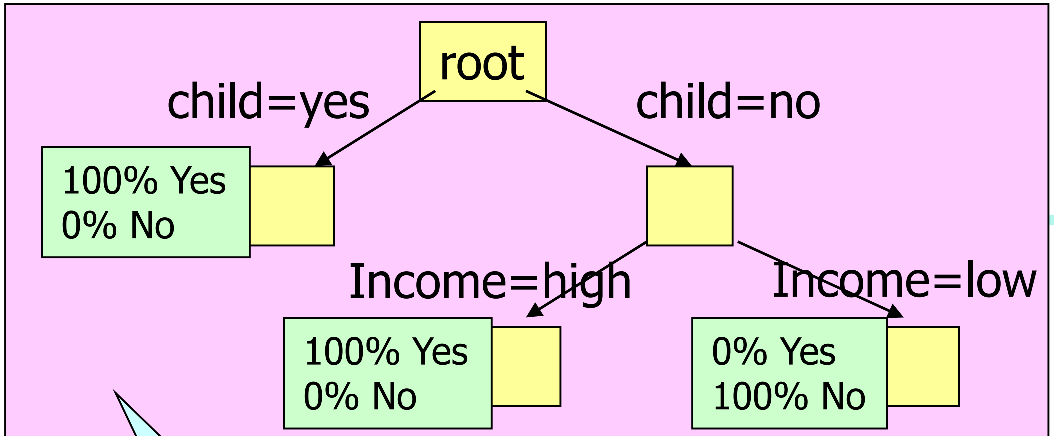
For attribute Income, Gain(Income, T) = 0.7219



	Race	Income	Child	Insurance
1	black	high	no	yes
2	white	high	yes	yes
3	white	low	yes	yes
4	white	low	yes	yes
5	black	low	no	no
6	black	low	no	no
7	black	low	no	no
8	white	low	no	no

Suppose there is a new person.

Race	Income	Child	Insurance
white	high	no	?



	Race	Income	Child	Insurance
1	black	high	no	yes
2	white	high	yes	yes
3	white	low	yes	yes
4	white	low	yes	yes
5	black	low	no	no
6	black	low	no	no
7	black	low	no	no
8	white	low	no	no

Decision tree

Termination Criteria?

e.g., height of the tree
e.g., accuracy of each node

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)
- Ex.

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

  - gain_ratio(income) = 0.029/1.557 = 0.019
- The attribute with the maximum gain ratio is selected as the splitting attribute

C4.5

- ID3
  - Impurity Measurement
    - Gain(A, T)
      = Info(T) – Info(A, T)
- C4.5
  - Impurity Measurement
    - Gain(A, T)
      = (Info(T) – Info(A, T))/SplitInfo(A)
    - where SplitInfo(A) = $-\Sigma_{v \in A}$ p(v) log p(v)

Entrop

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Info(T) = - ½ log ½ - ½ log ½
= 1

For attribute Race,

$\text{Info}(T_{black}) = - \frac{3}{4} \log \frac{3}{4} - \frac{1}{4} \log \frac{1}{4} = 0.8113$

$\text{Info}(T_{white}) = - \frac{3}{4} \log \frac{3}{4} - \frac{1}{4} \log \frac{1}{4} = 0.8113$

$\text{Info}(Race, T) = \frac{1}{2} \times \text{Info}(T_{black}) + \frac{1}{2} \times \text{Info}(T_{white}) = 0.8113$

SplitInfo(Race) = - ½ log ½ - ½ log ½ = 1

Gain(Race, T) = (Info(T) – Info(Race, T))/SplitInfo(Race) = (1 – 0.8113)/1 = 0.1887

For attribute Race, Gain(Race, T) = 0.1887

Entrop

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Info(T) = - ½ log ½ - ½ log ½
= 1

For attribute Income,

$\text{Info}(T_{high}) = - 1 \log 1 – 0 \log 0 = 0$

$\text{Info}(T_{low}) = - 1/3 \log 1/3 – 2/3 \log 2/3 = 0.9183$

$\text{Info}(\text{Income}, T) = ¼ \times \text{Info}(T_{high}) + ¾ \times \text{Info}(T_{low}) = 0.6887$

SplitInfo(Income) = - 2/8 log 2/8 – 6/8 log 6/8 = 0.8113

Gain(Income, T)= (Info(T)–Info(Income, T))/SplitInfo(Income) = (1–0.6887)/0.8113
= 0.3837

For attribute Race, Gain(Race, T) = 0.1887

For attribute Income, Gain(Income, T) = 0.3837

For attribute Child, Gain(Child, T) = ?

Gini Index (CART, IBM IntelligentMiner)

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

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

  where p_j is the relative frequency of class j in D

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

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

- Reduction in Impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

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

CART

- Impurity Measurement
  - Gini

    $I(P) = 1 - \Sigma_j \, p_j^2$

Gini

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

$\text{Info}(T) = 1 - (½)^2 - (½)^2$
$= ½$

For attribute Race,

$\text{Info}(T_{black}) = 1 - (¾)^2 - (¼)^2 = 0.375$

$\text{Info}(T_{white}) = 1 - (¾)^2 - (¼)^2 = 0.375$

$\text{Info}(\text{Race}, T) = ½ \times \text{Info}(T_{black}) + ½ \times \text{Info}(T_{white}) = 0.375$

$\text{Gain}(\text{Race}, T) = \text{Info}(T) - \text{Info}(\text{Race}, T) = ½ - 0.375 = 0.125$

For attribute Race, Gain(Race, T) = 0.125

Gini

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

$\text{Info}(T) = 1 - (½)^2 - (½)^2$
$= ½$

For attribute Income,

$\text{Info}(T_{high}) = 1 - 1^2 - 0^2 = 0$

$\text{Info}(T_{low}) = 1 - (1/3)^2 - (2/3)^2 = 0.444$

$\text{Info}(\text{Income}, T) = 1/4 \times \text{Info}(T_{high}) + 3/4 \times \text{Info}(T_{low}) = 0.333$

$\text{Gain}(\text{Income}, T) = \text{Info}(T) - \text{Info}(\text{Race}, T) = ½ - 0.333 = 0.167$

For attribute Race, Gain(Race, T) = 0.125

For attribute Income, Gain(Race, T) = 0.167

For attribute Child, Gain(Child, T) = ?

Comparing Attribute Selection Measures

- The three measures, in general, return good results but
  - **Information gain**:
    - biased towards multivalued attributes
  - **Gain ratio**:
    - tends to prefer unbalanced splits in which one partition is much smaller than the others
  - **Gini index**:
    - biased to multivalued attributes
    - has difficulty when # of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

Overfitting and Tree Pruning

- Overfitting: An induced tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: *Halt tree construction early*-do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - Postpruning: *Remove branches* from a “fully grown” tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the “best pruned tree”

Enhancements to Basic Decision Tree Induction

- Allow for **continuous-valued attributes**
  - Dynamically define new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals
- Handle **missing attribute values**
  - Assign the most common value of the attribute
  - Assign probability to each of the possible values
- **Attribute construction**
  - Create new attributes based on existing ones that are sparsely represented
  - This reduces fragmentation, repetition, and replication

Classification in Large Databases

- Classification—a classical problem extensively studied by statisticians and machine learning researchers
- Scalability: Classifying data sets with millions of examples and hundreds of attributes with reasonable speed
- Why is decision tree induction popular?
  - relatively faster learning speed (than other classification methods)
  - convertible to simple and easy to understand classification rules
  - can use SQL queries for accessing databases
  - comparable classification accuracy with other methods
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)

Scalability Framework for RainForest

- Separates the scalability aspects from the criteria that determine the quality of the tree
- Builds an AVC-list**: AVC (Attribute, Value, Class_label)**
- **AVC-set** (of an attribute *X*)
  - Projection of training dataset onto the attribute *X* and class label where counts of individual class label are aggregated
- **AVC-group** (of a node *n*)
  - Set of AVC-sets of all predictor attributes at the node *n*

Rainforest: Training Set and Its AVC Sets

Training Examples

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

AVC-set on *Age*

Age	Buy_Computer	
	yes	no
<=30	2	3
31..40	4	0
>40	3	2

AVC-set on *income*

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

AVC-set on *Student*

student	Buy_Computer	
	yes	no
yes	6	1
no	3	4

AVC-set on *credit_rating*

Credit rating	Buy_Computer	
	yes	no
fair	6	2
excellent	3	3

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data
- Standard: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured

Bayes' Theorem: Basics

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

Bayes' Theorem

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

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

- Informally, this can be written as

  posteriori = likelihood x prior/evidence

- Predicts **X** belongs to C_2 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

Towards Naïve Bayes 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 **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 $P(C_i|\mathbf{X})$
- This can be derived from Bayes' theorem

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

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

  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):

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

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k|C_i)$ is the # of tuples in C_i having value x_k for A_k divided by $|C_{i, D}|$ (# of tuples of C_i in D)
- If A_k is continous-valued, $P(x_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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 Bayes Classifier

- Conditional Probability
  - A: a random variable
  - B: a random variable
  - $P(A \mid B) = \dfrac{P(AB)}{P(B)}$

Naïve Bayes Classifier

- Bayes Rule
  - A : a random variable
  - B: a random variable
  -

$$P(A \mid B) = \frac{P(B|A)\, P(A)}{P(B)}$$

Naïve Bayes Cla

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

- Independent Assumption
  - Each attribute are independent
  - e.g.,
    P(X, Y, Z | A) = P(X | A) x P(Y | A) x P(Z | A)

Naïve Bayes Classifier

Suppose there is a new person.

Race	Income	Child	Insurance
white	high	no	?

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Insurance = Yes

For attribute Race,

P(Race = black | Yes) = ¼
P(Race = white | Yes) = ¾
P(Race = black | No) = ¾
P(Race = white | No) = ¼

P(Yes) = ½

P(No) = ½

For attribute Income,

P(Income = high | Yes) = ½
P(Income = low | Yes) = ½
P(Income = high | No) = 0
P(Income = low | No) = 1

For attribute Child,

P(Child = yes | Yes) = ¾
P(Child = no | Yes) = ¼
P(Child = yes | No) = 0
P(Child = no | No) = 1

Naïve Bayes Classifier

P(Race = white, Income = high, Child = no| Yes)
= P(Race = white | Yes) x P(Income = high | Yes)
x P(Child = no | Yes)
= ¾ x ½ x ¼
= 0.09375

P(Race = white, Income = high, Child = no| No)
= P(Race = white | No) x P(Income = high | No)
x P(Child = no | No)
= ¼ x 0 x 1
= 0

Suppose there is a new person.

Race	Income	Child	Insurance
white	high	no	?

Naïve Bayes C

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Insurance = Yes

For attribute Race,

P(Race = black | Yes) = ¼

P(Race = white | Yes) = ¾

P(Race = black | No) = ¾

P(Race = white | No) = ¼

P(Yes) = ½

P(No) = ½

For attribute Income,

P(Income = high | Yes) = ½

P(Income = low | Yes) = ½

P(Income = high | No) = 0

P(Income = low | No) = 1

For attribute Child,

P(Child = yes | Yes) = ¾

P(Child = no | Yes) = ¼

P(Child = yes | No) = 0

P(Child = no | No) = 1

Naïve Bayes Classifier

P(Race = white, Income = high, Child = no| Yes)

= 0.09375

P(Race = white, Income = high, Child = no| No)

= P(Race = white | No) x P(Income = high | No)

x P(Child = no | No)

= ¼ x 0 x 1

= 0

Suppose there is a new person.

Race	Income	Child	Insurance
white	high	no	?

Naïve Bayes Classifier

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Insurance = Yes

For attribute Race,

P(Race = black | Yes) = ¼
P(Race = white | Yes) = ¾
P(Race = black | No) = ¾
P(Race = white | No) = ¼

P(Yes) = ½

P(No) = ½

For attribute Income,

P(Income = high | Yes) = ½
P(Income = low | Yes) = ½
P(Income = high | No) = 0
P(Income = low | No) = 1

For attribute Child,

P(Child = yes | Yes) = ¾
P(Child = no | Yes) = ¼
P(Child = yes | No) = 0
P(Child = no | No) = 1

Naïve Bayes Classifier

P(Race = white, Income = high, Child = no| Yes)
= 0.09375

P(Race = white, Income = high, Child = no| No)
= P(Race = white | No) x P(Income = high | No)
x P(Child = no | No)
= ¼ x 0 x 1
= 0

Suppose there is a new person.

Race	Income	Child	Insurance
white	high	no	?

Naïve Bayes

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Insurance = Yes

For attribute Race,

P(Race = black | Yes) = ¼

P(Race = white | Yes) = ¾

P(Race = black | No) = ¾

P(Race = white | No) = ¼

P(Yes) = ½

P(No) = ½

For attribute Income,

P(Income = high | Yes) = ½

P(Income = low | Yes) = ½

P(Income = high | No) = 0

P(Income = low | No) = 1

For attribute Child,

P(Child = yes | Yes) = ¾

P(Child = no | Yes) = ¼

P(Child = yes | No) = 0

P(Child = no | No) = 1

Naïve Bayes Classifier

P(Race = white, Income = high, Child = no| Yes)

= 0.09375

P(Race = white, Income = high, Child = no| No)

= 0

Suppose there is a new person.

Race	Income	Child	Insurance
white	high	no	?

Naïve Bayes C

Race	Income	Child	Insurance
black	high	no	yes
white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Insurance = Yes

For attribute Race,

P(Race = black | Yes) = ¼
P(Race = white | Yes) = ¾
P(Race = black | No) = ¾
P(Race = white | No) = ¼

P(Yes) = ½

P(No) = ½

For attribute Income,

P(Income = high | Yes) = ½
P(Income = low | Yes) = ½
P(Income = high | No) = 0
P(Income = low | No) = 1

For attribute Child,

P(Child = yes | Yes) = ¾
P(Child = no | Yes) = ¼
P(Child = yes | No) = 0
P(Child = no | No) = 1

Naïve Bayes Classifier

P(Race = white, Income = high, Child = no| Yes)
= 0.09375

P(Race = white, Income = high, Child = no| No)
= 0

Suppose there is a new person.

Race	Income	Child	Ins
white	high	no	?

$$P(Yes \mid Race = white, Income = high, Child = no) = \frac{0.046875}{P(Race = white, Income = high, Child = no)}$$

Naïve Bayes C

white	high	yes	yes
white	low	yes	yes
white	low	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Insurance = Yes

For attribute Race,

$P(Race = black \mid Yes) = 1/4$

$P(Race = white \mid Yes) = 3/4$

$P(Race = black \mid No) = 3/4$

$P(Race = white \mid No) = 1/4$

$P(Yes) = 1/2$

$P(No) = 1/2$

For attribute Income,

$P(Income = high \mid Yes) = 1/2$

$P(Income = low \mid Yes) = 1/2$

$P(Income = high \mid No) = 0$

$P(Income = low \mid No) = 1$

For attribute Child,

$P(Child = yes \mid Yes) = 3/4$

$P(Child = no \mid Yes) = 1/4$

$P(Child = yes \mid No) = 0$

$P(Child = no \mid No) = 1$

Naïve Bayes Classifier

$P(Race = white, Income = high, Child = no \mid Yes) = 0.09375$

$P(Race = white, Income = high, Child = no \mid No) = 0$

$$P(Yes \mid Race = white, Income = high, Child = no) = \frac{P(Race = white, Income = high, Child = no \mid Yes)\, P(Yes)}{P(Race = white, Income = high, Child = no)}$$

$$= \frac{0.09375 \times 0.5}{P(Race = white, Income = high, Child = no)}$$

$$= \frac{0.046875}{P(Race = white, Income = high, Child = no)}$$

Suppose there is a new person.

Race	Income	Child	Ins
white	high	no	

Insuran

$$P(Yes \mid Race = white, Income = high, Child = no) = \frac{0.046875}{P(Race = white, Income = high, Child = no)}$$

$$P(No \mid Race = white, Income = high, Child = no) = \frac{0}{P(Race = white, Income = high, Child = no)}$$

white	high	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

For attribute Race,

$P(Race = black \mid Yes) = ¼$

$P(Race = white \mid Yes) = ¾$

$P(Race = black \mid No) = ¾$

$P(Race = white \mid No) = ¼$

$P(Yes) = ½$

$P(No) = ½$

For attribute Income,

$P(Income = high \mid Yes) = ½$

$P(Income = low \mid Yes) = ½$

$P(Income = high \mid No) = 0$

$P(Income = low \mid No) = 1$

For attribute Child,

$P(Child = yes \mid Yes) = ¾$

$P(Child = no \mid Yes) = ¼$

$P(Child = yes \mid No) = 0$

$P(Child = no \mid No) = 1$

Naïve Bayes Classifier

$P(Race = white, Income = high, Child = no \mid Yes)$
$= 0.09375$

$P(Race = white, Income = high, Child = no \mid No)$
$= 0$

$P(No \mid Race = white, Income = high, Child = no)$

$$= \frac{P(Race = white, Income = high, Child = no \mid No)\, P(No)}{P(Race = white, Income = high, Child = no)}$$

$$= \frac{0 \times 0.5}{P(Race = white, Income = high, Child = no)}$$

$$= \frac{0}{P(Race = white, Income = high, Child = no)}$$

Suppose there is a new person.

Race	Income	Child	Ins
white	high	no	

$$P(\text{Yes} \mid \text{Race} = \text{white}, \text{Income} = \text{high}, \text{Child} = \text{no}) = \frac{0.046875}{P(\text{Race} = \text{white}, \text{Income} = \text{high}, \text{Child} = \text{no})}$$

$$P(\text{No} \mid \text{Race} = \text{white}, \text{Income} = \text{high}, \text{Child} = \text{no}) = \frac{0}{P(\text{Race} = \text{white}, \text{Income} = \text{high}, \text{Child} = \text{no})}$$

Insurar

For attribute Race,

P(Race = black | Yes) = ¼

P(Race = white | Yes) = ¾

P(Race = black | No) = ¾

P(Race = white | No) = ¼

P(Yes) = ½

P(No) = ½

white	high	yes	yes
black	low	no	no
black	low	no	no
black	low	no	no
white	low	no	no

Naïve Bayes Classifier

For attribute Income,

P(Income = high | Y

P(Income = low | Y

P(Income = high | N

P(Income = low | N

For attribute Child,

P(Child = yes | Yes)

P(Child = no | Yes)

P(Child = yes | No)

P(Child = no | No) = 1

Since P(Yes | Race = white, Income = high, Child = no) > P(No | Race = white, Income = high, Child = no).

we predict the following new person will buy an insurance.

Race	Income	Child	Insurance
white	high	no	?

Naïve Bayes Classifier: Training Dataset

Class:

C1:buys_computer = 'yes'

C2:buys_computer = 'no'

Data to be classified:

X = (age <=30,

Income = medium,

Student = yes

Credit_rating = Fair)

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

Naïve Bayes Classifier: An Example

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

- $P(C_i)$: P(buys_computer = “yes”) = 9/14 = 0.643
  P(buys_computer = “no”) = 5/14= 0.357
- Compute $P(X|C_i)$ for each class
  P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222
  P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6
  P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
  P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
  P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
  P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
  P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
  P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4
- **X = (age <= 30 , income = medium, student = yes, credit_rating = fair)**

$P(X|C_i)$: P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019

$P(X|C_i)*P(C_i)$: P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028
P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Therefore, X belongs to class (“buys_computer = yes”)

Avoiding the Zero-Probability Problem

- Naïve Bayesian prediction requires each conditional prob. be **non-zero**. Otherwise, the predicted prob. will be zero

$$P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i)$$

- Ex. Suppose a dataset with 1000 tuples, income=low (0), income= medium (990), and income = high (10)
- Use **Laplacian correction** (or Laplacian estimator)
  - *Adding 1 to each case*

    Prob(income = low) = 1/1003

    Prob(income = medium) = 991/1003

    Prob(income = high) = 11/1003
  - The "corrected" prob. estimates are close to their "uncorrected" counterparts

Naïve Bayes Classifier: Comments

- Advantages
  - Easy to implement
  - Good results obtained in most of the cases
- Disadvantages
  - Assumption: class conditional independence, therefore loss of accuracy
  - Practically, dependencies exist among variables
    - E.g., hospitals: patients: Profile: age, family history, etc.

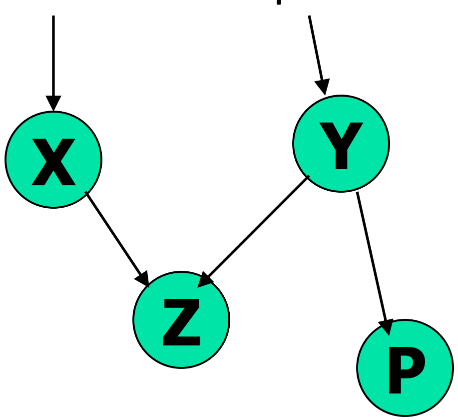
      Symptoms: fever, cough etc., Disease: lung cancer, diabetes, etc.
    - Dependencies among these cannot be modeled by Naïve Bayes Classifier
- How to deal with these dependencies? Bayesian Belief Networks (Chapter 9)

Bayesian Belief Network

- Naïve Bayes Classifier
  - Independent Assumption
- Bayesian Belief Network
  - Do not have independent assumption

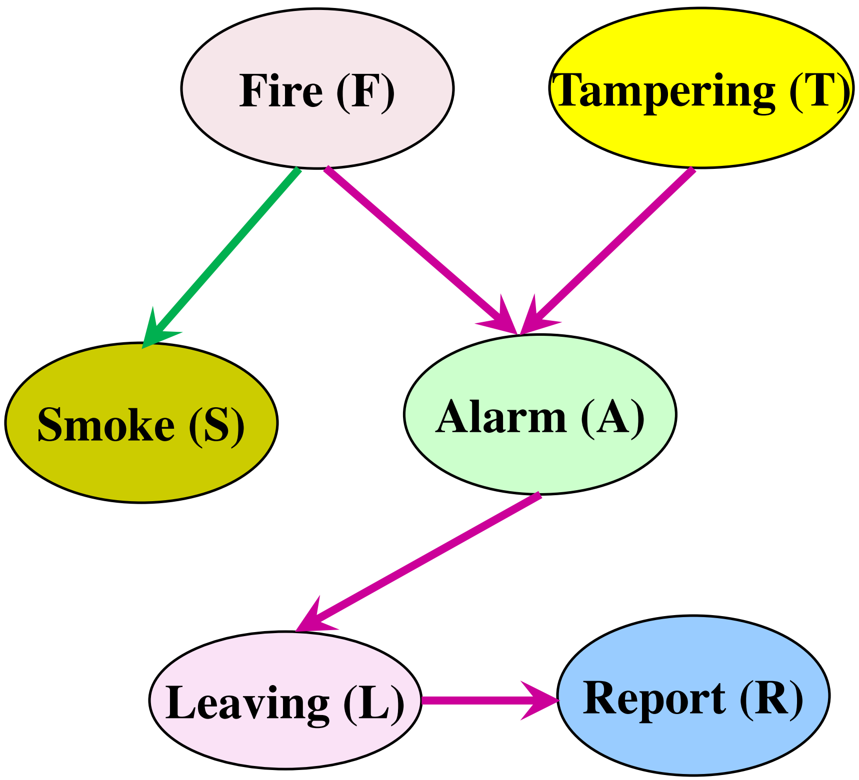
Bayesian Belief Networks

- **Bayesian belief network** (also known as **Bayesian network**, **probabilistic network**): allows *class conditional independencies* between *subsets* of variables
- Two components: (1) A *directed acyclic graph* (called a structure) and (2) a set of *conditional probability tables* (CPTs)
- A (*directed acyclic*) graphical model of *causal influence* relationships
  - Represents dependency among the variables
  - Gives a specification of joint probability distribution



- Nodes: random variables
- Links: dependency
- X and Y are the parents of Z, and Y is the parent of P
- No dependency between Z and P
- Has no loops/cycles

A Bayesian Network and Some of Its CPTs



CPT: Conditional Probability Tables

Fire	Smoke	$\Theta_{s\|f}$
True	True	.90
False	True	.01

Fire	Tampering	Alarm	$\Theta_{a\|f,t}$
True	True	True	.5
True	False	True	.99
False	True	True	.85
False	False	True	.0001

CPT shows the conditional probability for each possible combination of its parents

Derivation of the probability of a particular combination of values of **X**, from CPT:

$$P(x_1,\ldots,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(x_i))$$

Yes/No

Healthy/ Unhealthy

Yes/No

High/Low

Yes/No

Yes/No

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
Yes	Healthy	No	High	Yes	No
No	Unhealthy	Yes	Low	Yes	No
No	Healthy	Yes	High	No	Yes
…	…	…	…	…	…

Some attributes are dependent on other attributes.

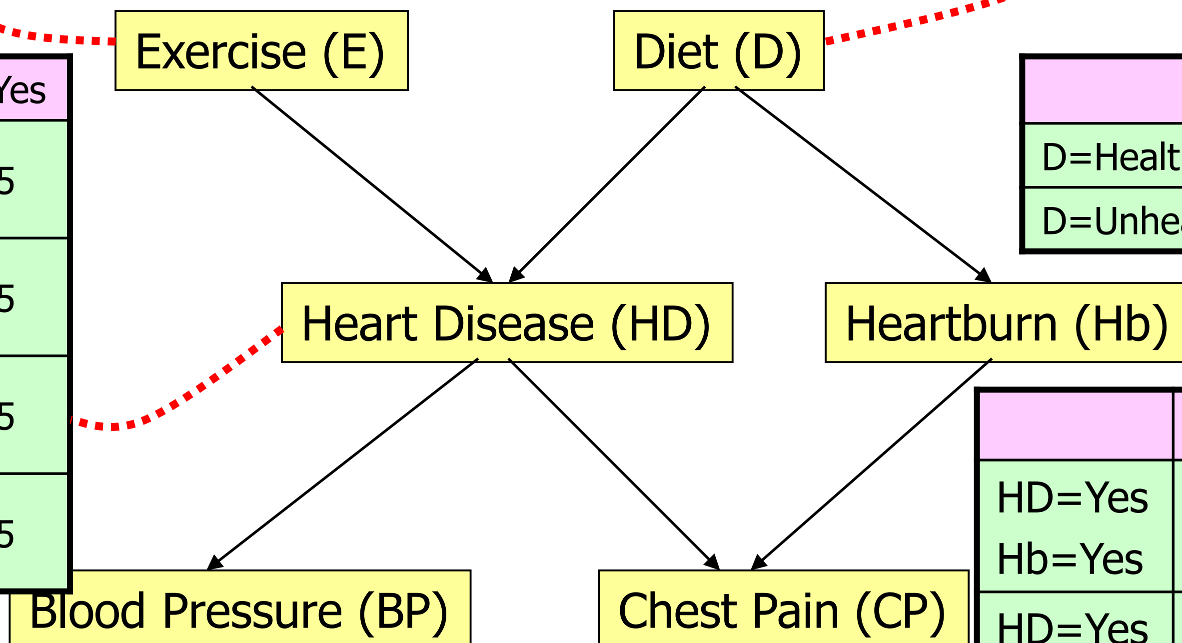
e.g., doing exercises may reduce the probability of suffering from Heart Disease

$$\begin{tikzcd} \text{Exercise (E)} \arrow[d] \\ \text{Heart Disease} \end{tikzcd}$$

Bayesian Belief N k

E = Yes
0.7

D = Healthy
0.25



	HD=Yes
E=Yes D=Healthy	0.25
E=Yes D=Unhealthy	0.45
E=No D=Healthy	0.55
E=No D=Unhealthy	0.75

	Hb=Yes
D=Healthy	0.85
D=Unhealthy	0.2

	BP=High
HD=Yes	0.85
HD=No	0.2

	CP=Yes
HD=Yes Hb=Yes	0.8
HD=Yes Hb=No	0.6
HD=No Hb=Yes	0.4
HD=No Hb=No	0.1

Let X, Y, Z be three random variables.
X is said to be **conditionally independent** of Y given Z if the following holds.

$$P(X \mid Y, Z) = P(X \mid Z)$$

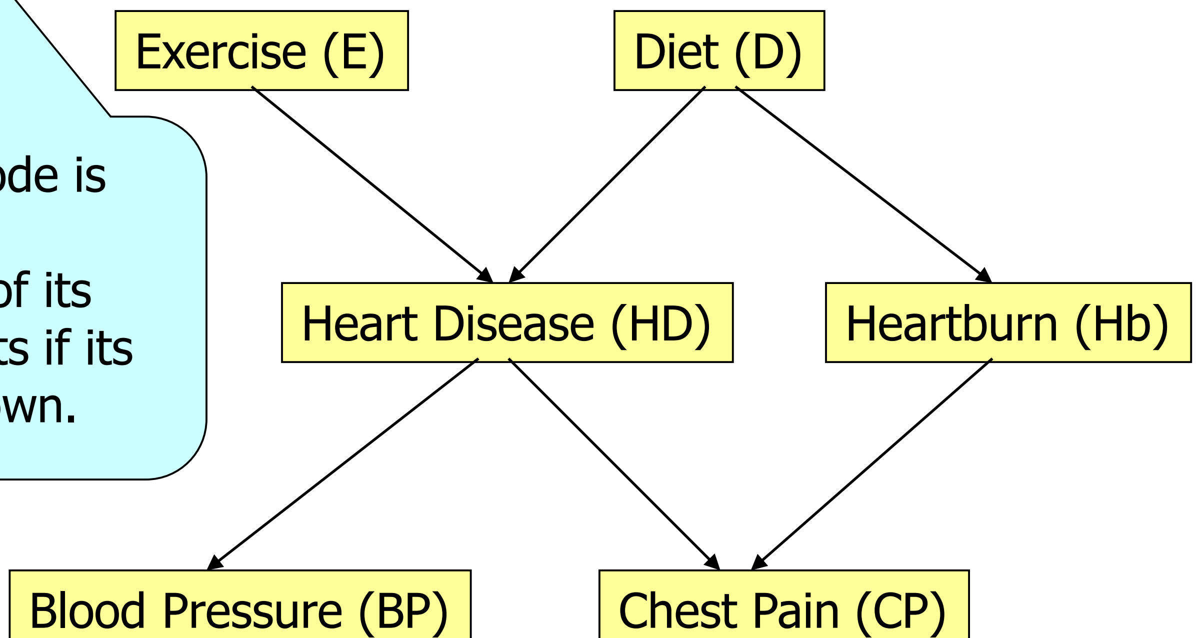
Lemma:
If X is conditionally independent of Y given Z,

$$P(X, Y \mid Z) = P(X \mid Z) \times P(Y \mid Z) \ ?$$

Let X, Y, Z be three random variables.
X is said to be **conditionally independent** of Y given Z if the following holds.

$$P(X \mid Y, Z) = P(X \mid Z)$$

Property: A node is **conditionally independent** of its non-descendants if its parents are known.



e.g., P(BP = High | HD = Yes, D = Healthy) = P(BP = High | HD = Yes)

“BP = High” is **conditionally independent** of “D = Healthy” given “HD = Yes”

e.g., P(BP = High | HD = Yes, CP=Yes) = P(BP = High | HD = Yes)

“BP = High” is **conditionally independent** of “CP = Yes” given “HD = Yes”

Yes/No

Healthy/
Unhealthy

Yes/No

High/Low

Yes/No

Yes/No

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
Yes	Healthy	No	High	Yes	No
No	Unhealthy	Yes	Low	Yes	No
No	Healthy	Yes	High	No	Yes
…	…	…	…	…	…

Suppose there is a new person and I want to know whether he is likely to have Heart Disease.

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
?	?	?	?	?	?

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
?	?	?	High	?	?

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
Yes	Healthy	?	High	?	?

Bayesian Belief Network

Suppose there is a new person and I want to know whether he is likely to have Heart Disease.

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
?	?	?	?	?	?

$$P(HD = Yes) = \sum_{x \in \{Yes,\ No\}} \sum_{y \in \{Healthy,\ Unhealthy\}} P(HD=Yes|E=x, D=y) \times P(E=x, D=y)$$

$$= \sum_{x \in \{Yes,\ No\}} \sum_{y \in \{Healthy,\ Unhealthy\}} P(HD=Yes|E=x, D=y) \times P(E=x) \times P(D=y)$$

$$= 0.25 \times 0.7 \times 0.25 + 0.45 \times 0.7 \times 0.75 + 0.55 \times 0.3 \times 0.25 + 0.75 \times 0.3 \times 0.75$$

$$= 0.49$$

$$P(HD = No) = 1- P(HD = Yes)$$

$$= 1- 0.49$$

$$= 0.51$$

Bayesian Belief Network

Suppose there is a new person and I want to know whether he is likely to have Heart Disease.

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
?	?	?	High	?	?

$$P(BP = High) = \sum_{x \in \{Yes,\ No\}} P(BP = High|HD=x) \times P(HD = x)$$
$$= 0.85 \times 0.49 + 0.2 \times 0.51$$
$$= 0.5185$$

$$P(HD = Yes|BP = High) = \frac{P(BP = High|HD=Yes) \times P(HD = Yes)}{P(BP = High)}$$
$$= \frac{0.85 \times 0.49}{0.5185}$$
$$= 0.8033$$

$$P(HD = No|BP = High) = 1 - P(HD = Yes|BP = High)$$
$$= 1 - 0.8033$$
$$= 0.1967$$

Bayesian Belief Network

Suppose there is a new person and I want to know whether he is likely to have Heart Disease.

Exercise	Diet	Heartburn	Blood Pressure	Chest Pain	Heart Disease
Yes	Healthy	?	High	?	?

$$P(HD = Yes \mid BP = High, D = Healthy, E = Yes)$$

$$= \frac{P(BP = High \mid HD = Yes, D = Healthy, E = Yes)}{P(BP = High \mid D = Healthy, E = Yes)} x\ P(HD = Yes|D = Healthy, E = Yes)$$

$$= \frac{P(BP = High|HD = Yes)\ P(HD = Yes|D = Healthy, E = Yes)}{\sum_{x \in \{Yes, No\}} P(BP=High|HD=x)\ P(HD=x|D=Healthy, E=Yes)}$$

$$= \frac{0.85x0.25}{0.85x0.25 + 0.2x0.75}$$

$$= 0.5862$$

$$P(HD = No \mid BP = High, D = Healthy, E = Yes)$$

$$= 1- P(HD = Yes \mid BP = High, D = Healthy, E = Yes)$$

$$= 1-0.5862$$

$$= 0.4138$$

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap
- Comparing classifiers:
  - Confidence intervals
  - Cost-benefit analysis and ROC Curves

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

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

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

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

  Accuracy = (TP + TN)/All

- **Error rate:** *1 – accuracy*, or

  Error rate = (FP + FN)/All

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class
  - **Sensitivity**: True Positive recognition rate
    - **Sensitivity = TP/P**
  - **Specificity**: True Negative recognition rate
    - **Specificity = TN/N**

Classifier Evaluation Metrics: Precision and Recall, and F-measures

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive

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

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

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

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

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

- ***$F_ß$*:** weighted measure of precision and recall
  - assigns ß times as much weight to recall as to precision

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

Classifier Evaluation Metrics: Example

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

- *Precision* = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

Evaluating Classifier Accuracy: Holdout & Cross-Validation Methods

- **Holdout method**
  - Given data is randomly partitioned into two independent sets
    - Training set (e.g., 2/3) for model construction
    - Test set (e.g., 1/3) for accuracy estimation
  - Random sampling: a variation of holdout
    - Repeat holdout k times, accuracy = avg. of the accuracies obtained
- **Cross-validation** (*k*-fold, where k = 10 is most popular)
  - Randomly partition the data into *k mutually exclusive* subsets, each approximately equal size
  - At *i*-th iteration, use D_i as test set and others as training set
  - Leave-one-out: *k* folds where *k* = # of tuples, for small sized data
  - ***Stratified cross-validation***: folds are stratified so that class dist. in each fold is approx. the same as that in the initial data

Evaluating Classifier Accuracy: Bootstrap

- **Bootstrap**
  - Works well with small data sets
  - Samples the given training tuples uniformly *with replacement*
    - i.e., each time a tuple is selected, it is equally likely to be selected again and re-added to the training set
- Several bootstrap methods, and a common one is **.632 boostrap**
  - A data set with *d* tuples is sampled *d* times, with replacement, resulting in a training set of *d* samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the bootstrap, and the remaining 36.8% form the test set (since $(1 – 1/d)^d \approx e^{-1} = 0.368$)
  - Repeat the sampling procedure *k* times, overall accuracy of the model:

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

Estimating Confidence Intervals: Classifier Models M_1 vs. M_2

- Suppose we have 2 classifiers, M_1 and M_2, which one is better?
- Use 10-fold cross-validation to obtain $\overline{err}(M_1)$ and $\overline{err}(M_2)$
- These mean error rates are just *estimates* of error on the true population of *future* data cases
- What if the difference between the 2 error rates is just attributed to *chance*?
  - Use a **test of statistical significance**
  - Obtain **confidence limits** for our error estimates

Estimating Confidence Intervals: Null Hypothesis

- Perform 10-fold cross-validation
- Assume samples follow a **t distribution** with *k–1* **degrees of freedom** (here, *k=10*)
- Use **t-test** (or **Student's t-test**)
- **Null Hypothesis**: M_1 & M_2 are the same
- If we can **reject** null hypothesis, then
  - we conclude that the difference between M_1 & M_2 is **statistically significant**
  - Chose model with lower error rate

Estimating Confidence Intervals: t-test

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

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

where

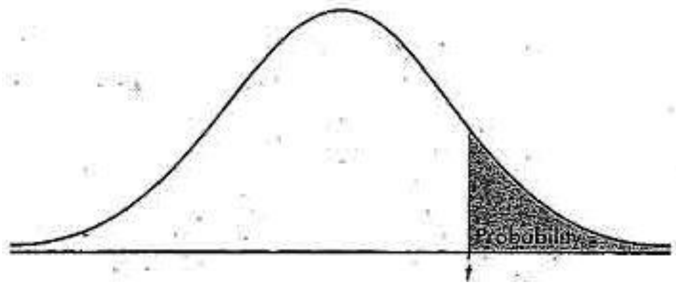
$$var(M_1 - M_2) = \frac{1}{k}\sum_{i=1}^{k}\left[err(M_1)_i - err(M_2)_i - (\overline{err}(M_1) - \overline{err}(M_2))\right]^2$$

- If two test sets available: use **non-paired t-test**

where $$var(M_1 - M_2) = \sqrt{\frac{var(M_1)}{k_1} + \frac{var(M_2)}{k_2}},$$

where k_1 & k_2 are # of cross-validation samples used for M_1 & M_2, resp.

Estimating Confidence Intervals: Table for t-distribution



- Symmetric
- **Significance level**, e.g., *sig = 0.05* or *5%* means M_1 & M_2 are *significantly different* for 95% of population
- **Confidence limit**, *z = sig/2*

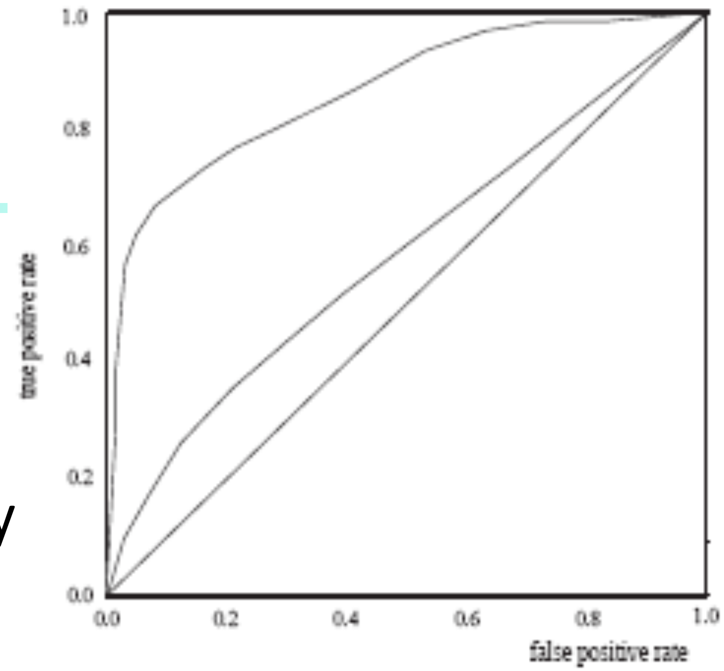
TABLE B: *t*-DISTRIBUTION CRITICAL VALUES

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

Estimating Confidence Intervals: Statistical Significance

- Are M_1 & M_2 **significantly different**?
  - Compute *t.* Select *significance level* (e.g. *sig = 5%)*
  - Consult table for t-distribution: Find *t value* corresponding to *k-1 degrees of freedom* (here, 9)
  - t-distribution is symmetric: typically upper % points of distribution shown → look up value for **confidence limit** *z=sig/2* (here, 0.025)
  - **If t > z or t < -z**, then t value lies in rejection region:
    - **Reject null hypothesis** that mean error rates of M_1 & M_2 are same
    - Conclude: statistically significant difference between M_1 & M_2
  - **Otherwise**, conclude that any difference is **chance**

Model Selection: ROC Curves



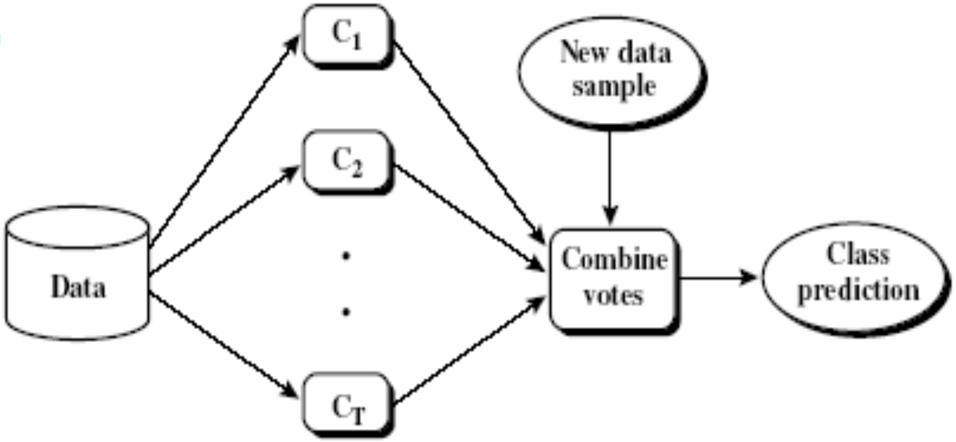
- **ROC** (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate and the false positive rate
- The area under the ROC curve is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model

- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Ensemble Methods: Increasing the Accuracy



- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Bagging: Boostrap Aggregation

- Analogy: Diagnosis based on multiple doctors' majority vote
- Training
  - Given a set D of *d* tuples, at each iteration *i*, a training set D_i of *d* tuples is sampled with replacement from D (i.e., bootstrap)
  - A classifier model M_i is learned for each training set D_i
- Classification: classify an unknown sample **X**
  - Each classifier M_i returns its class prediction
  - The bagged classifier M* counts the votes and assigns the class with the most votes to **X**
- Prediction: can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple
- Accuracy
  - Often significantly better than a single classifier derived from D
  - For noise data: not considerably worse, more robust
  - Proved improved accuracy in prediction

Boosting

- Analogy: Consult several doctors, based on a combination of weighted diagnoses—weight assigned based on the previous diagnosis accuracy
- How boosting works?
  - **Weights** are assigned to each training tuple
  - A series of k classifiers is iteratively learned
  - After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, M_{i+1}, to **pay more attention to the training tuples that were misclassified** by M_i
  - The final **M* combines the votes** of each individual classifier, where the weight of each classifier's vote is a function of its accuracy
- Boosting algorithm can be extended for numeric prediction
- Comparing with bagging: Boosting tends to have greater accuracy, but it also risks overfitting the model to misclassified data

Adaboost (Freund and Schapire, 1997)

- Given a set of *d* class-labeled tuples, ($\mathbf{X_1}$, y_1), …, ($\mathbf{X_d}$, y_d)
- Initially, all the weights of tuples are set the same (1/d)
- Generate k classifiers in k rounds. At round i,
  - Tuples from D are sampled (with replacement) to form a training set D_i of the same size
  - Each tuple's chance of being selected is based on its weight
  - A classification model M_i is derived from D_i
  - Its error rate is calculated using D_i as a test set
  - If a tuple is misclassified, its weight is increased, o.w. it is decreased
- Error rate: err($\mathbf{X_j}$) is the misclassification error of tuple $\mathbf{X_j}$. Classifier M_i error rate is the sum of the weights of the misclassified tuples:

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

- The weight of classifier M_i's vote is

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

Random Forest (Breiman 2001)

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Classification of Class-Imbalanced Data Sets

- Class-imbalance problem: Rare positive example but numerous negative ones, e.g., medical diagnosis, fraud, oil-spill, fault, etc.
- Traditional methods assume a balanced distribution of classes and equal error costs: not suitable for class-imbalanced data
- Typical methods for imbalance data in 2-class classification:
  - **Oversampling**: re-sampling of data from positive class
  - **Under-sampling**: randomly eliminate tuples from negative class
  - **Threshold-moving**: moves the decision threshold, t, so that the rare class tuples are easier to classify, and hence, less chance of costly false negative errors
  - Ensemble techniques: Ensemble multiple classifiers introduced above
- Still difficult for class imbalance problem on multiclass tasks

Chapter 8. Classification: Basic Concepts

- Classification: Basic Concepts
- K Nearest Neighbor Classification Methods
- Decision Tree Induction
- Bayes Classification Methods
- Rule-Based Classification
- Model Evaluation and Selection
- Techniques to Improve Classification Accuracy: Ensemble Methods
- Summary

Summary (I)

- Classification is a form of data analysis that extracts models describing important data classes.
- Effective and scalable methods have been developed for decision tree induction, Naive Bayesian classification, rule-based classification, and many other classification methods.
- Evaluation metrics include: accuracy, sensitivity, specificity, precision, recall, F measure, and F_{β} measure.
- Stratified k-fold cross-validation is recommended for accuracy estimation. Bagging and boosting can be used to increase overall accuracy by learning and combining a series of individual models.

Summary (II)

- Significance tests and ROC curves are useful for model selection.
- There have been numerous comparisons of the different classification methods; the matter remains a research topic
- No single method has been found to be superior over all others for all data sets
- Issues such as accuracy, training time, robustness, scalability, and interpretability must be considered and can involve trade-offs, further complicating the quest for an overall superior method

References (1)

- C. Apte and S. Weiss. **Data mining with decision trees and decision rules**. Future Generation Computer Systems, 13, 1997
- C. M. Bishop, **Neural Networks for Pattern Recognition**. Oxford University Press, 1995
- L. Breiman, J. Friedman, R. Olshen, and C. Stone. **Classification and Regression Trees**. Wadsworth International Group, 1984
- C. J. C. Burges. **A Tutorial on Support Vector Machines for Pattern Recognition**. *Data Mining and Knowledge Discovery*, 2(2): 121-168, 1998
- P. K. Chan and S. J. Stolfo. **Learning arbiter and combiner trees from partitioned data for scaling machine learning**. KDD'95
- H. Cheng, X. Yan, J. Han, and C.-W. Hsu, Discriminative Frequent Pattern Analysis for Effective Classification, ICDE'07
- H. Cheng, X. Yan, J. Han, and P. S. Yu, Direct Discriminative Pattern Mining for Effective Classification, ICDE'08
- W. Cohen. **Fast effective rule induction**. ICML'95
- G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. **Mining top-k covering rule groups for gene expression data**. SIGMOD'05

References (2)

- A. J. Dobson. **An Introduction to Generalized Linear Models**. Chapman & Hall, 1990.
- G. Dong and J. Li. **Efficient mining of emerging patterns: Discovering trends and differences**. KDD'99.
- R. O. Duda, P. E. Hart, and D. G. Stork. **Pattern Classification**, 2ed. John Wiley, 2001
- U. M. Fayyad. **Branching on attribute values in decision tree generation**. AAAI'94.
- Y. Freund and R. E. Schapire. **A decision-theoretic generalization of on-line learning and an application to boosting**. J. Computer and System Sciences, 1997.
- J. Gehrke, R. Ramakrishnan, and V. Ganti. **Rainforest: A framework for fast decision tree construction of large datasets**. VLDB'98.
- J. Gehrke, V. Gant, R. Ramakrishnan, and W.-Y. Loh, **BOAT -- Optimistic Decision Tree Construction**. SIGMOD'99.
- T. Hastie, R. Tibshirani, and J. Friedman. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction.** Springer-Verlag, 2001.
- D. Heckerman, D. Geiger, and D. M. Chickering. **Learning Bayesian networks: The combination of knowledge and statistical data**. Machine Learning, 1995.
- W. Li, J. Han, and J. Pei, **CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules**, ICDM'01.

References (3)

- T.-S. Lim, W.-Y. Loh, and Y.-S. Shih. **A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms.** Machine Learning, 2000.
- J. Magidson. **The Chaid approach to segmentation modeling: Chi-squared automatic interaction detection**. In R. P. Bagozzi, editor, Advanced Methods of Marketing Research, Blackwell Business, 1994.
- M. Mehta, R. Agrawal, and J. Rissanen. **SLIQ : A fast scalable classifier for data mining**. EDBT'96.
- T. M. Mitchell. **Machine Learning**. McGraw Hill, 1997.
- S. K. Murthy, **Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey**, Data Mining and Knowledge Discovery 2(4): 345-389, 1998
- J. R. Quinlan. **Induction of decision trees**. *Machine Learning*, 1:81-106, 1986.
- J. R. Quinlan and R. M. Cameron-Jones. **FOIL: A midterm report**. ECML'93.
- J. R. Quinlan. **C4.5: Programs for Machine Learning**. Morgan Kaufmann, 1993.
- J. R. Quinlan. **Bagging, boosting, and c4.5**. AAAI'96.

References (4)

- R. Rastogi and K. Shim. **Public: A decision tree classifier that integrates building and pruning**. VLDB’98.
- J. Shafer, R. Agrawal, and M. Mehta. **SPRINT : A scalable parallel classifier for data mining**. VLDB’96.
- J. W. Shavlik and T. G. Dietterich. **Readings in Machine Learning**. Morgan Kaufmann, 1990.
- P. Tan, M. Steinbach, and V. Kumar. **Introduction to Data Mining**. Addison Wesley, 2005.
- S. M. Weiss and C. A. Kulikowski. **Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems**. Morgan Kaufman, 1991.
- S. M. Weiss and N. Indurkhya. **Predictive Data Mining**. Morgan Kaufmann, 1997.
- I. H. Witten and E. Frank. **Data Mining: Practical Machine Learning Tools and Techniques**, 2ed. Morgan Kaufmann, 2005.
- X. Yin and J. Han. **CPAR: Classification based on predictive association rules**. SDM'03
- H. Yu, J. Yang, and J. Han. **Classifying large data sets using SVM with hierarchical clusters**. KDD'03.

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

Predictor Error Measures

- Measure predictor accuracy: measure how far off the predicted value is from the actual known value
- **Loss function**: measures the error betw. 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

Scalable Decision Tree Induction Methods

- SLIQ (EDBT'96 — Mehta et al.)
  - Builds an index for each attribute and only class list and the current attribute list reside in memory
- SPRINT (VLDB'96 — J. Shafer et al.)
  - Constructs an attribute list data structure
- PUBLIC (VLDB'98 — Rastogi & Shim)
  - Integrates tree splitting and tree pruning: stop growing the tree earlier
- RainForest (VLDB'98 — Gehrke, Ramakrishnan & Ganti)
  - Builds an AVC-list (attribute, value, class label)
- BOAT (PODS'99 — Gehrke, Ganti, Ramakrishnan & Loh)
  - Uses bootstrapping to create several small samples

Data Cube-Based Decision-Tree Induction

- Integration of generalization with decision-tree induction (Kamber et al.'97)
- Classification at primitive concept levels
  - E.g., precise temperature, humidity, outlook, etc.
  - Low-level concepts, scattered classes, bushy classification-trees
  - Semantic interpretation problems
- Cube-based multi-level classification
  - Relevance analysis at multi-levels
  - Information-gain analysis with dimension + level