Lecture Slides - 4

Classification & Prediction

Classification and Prediction

- What is classification? What is prediction?
- Issues regarding classification and prediction
- Classification techniques
- Prediction
- Accuracy and error measures
- Ensemble methods
- Model selection
- Summary

2

Classification vs. Prediction

- Classification (System)
  - Models discrete-valued functions to predict categorical class labels (discrete or nominal)
  - Learns to classify data based on the training set with known class labels (model learning) and uses it for determining the class of new data points (prediction)
  - Examples: Naïve Bayes classifier, Decision Tree, etc.
- Prediction (System)
  - Models continuous-valued functions to predict continuous class labels (e.g., predicting unknown or missing values)
  - Example: Mathematical models like $y = x^2 + 1$.
- Typical applications
  - Credit approval, Target marketing, Medical diagnosis, Fraud detection, Email spam filtering

3

Classification: A Mathematical Mapping

- Classification:
  - Predicts discrete/categorical class labels
- Mathematically
  - Given $X = \Re^n$, and $Y = \{+1, -1\}$, we want a (classification) function $f: X \rightarrow Y$
- Example: Homepage classification (i.e., whether a given webpage is homepage or not?)
  - $X_i = (x_1, x_2, \ldots)$, $Y_i = +1$ or -1
    - x_1 : #of times "homepage" appears in a webpage
    - x_2 : #of times "welcome" appears in a webpage
    - …

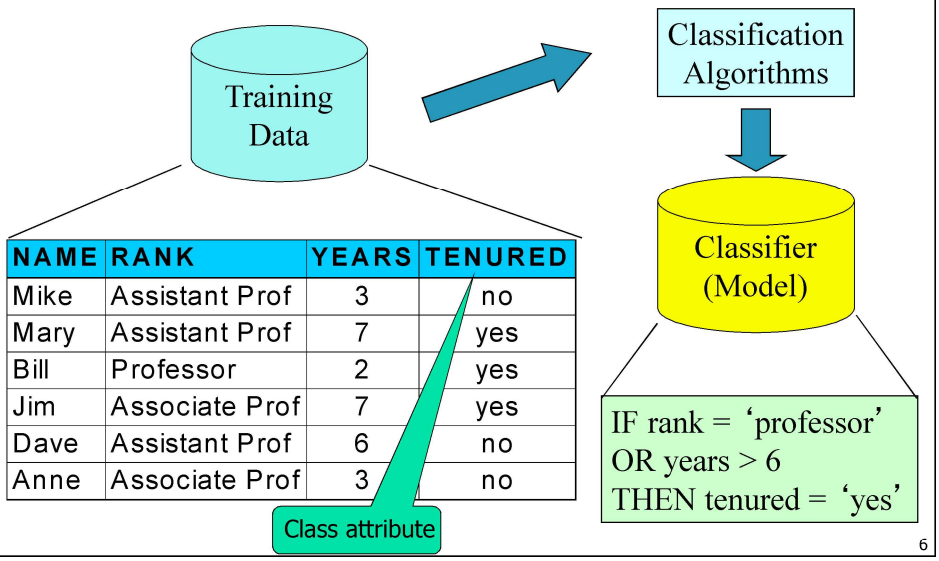
4

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 attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification (if-then) rules, decision trees, or mathematical formulae.
- Model usage: Determining the class labels of future or unknown objects
  - Estimate accuracy of the model using test set
    - The known labels of test set samples are compared with the predicted class labels by the model
    - Accuracy is the percentage of test set samples that are correctly classified by the model
  - If the accuracy is acceptable (beyond a pre-defined threshold), use the model to predict the class labels of the unseen data instances

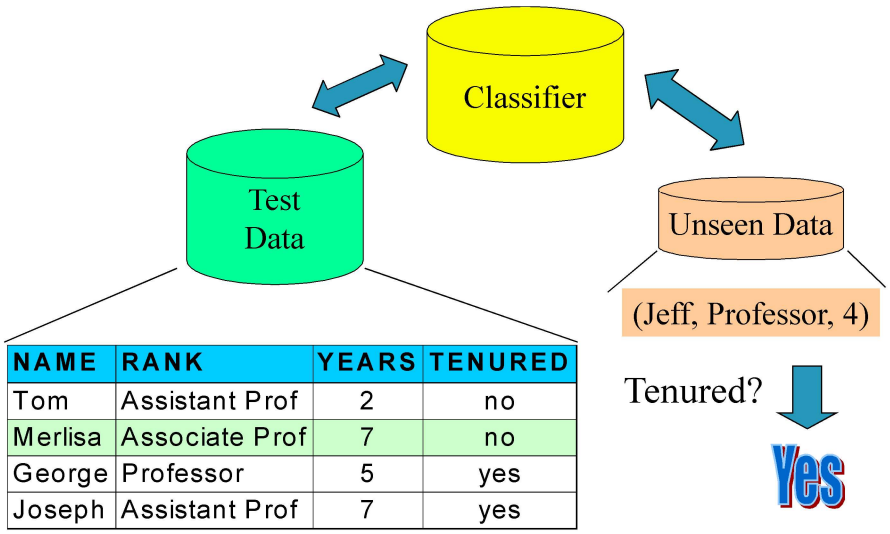
5

Step-1: Model Construction



6

Step-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

7

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by an attribute indicating the class labels 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. the aim is to establish the existence of groups, classes or clusters in the data

8

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

9

Issues: Evaluating Classification Methods

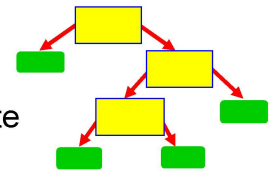
- Accuracy
  - Classifier accuracy: predicting class label
  - Predictor accuracy: guessing value of predicted attributes
- Speed
  - Time to construct the model (training time)
  - Time to use the model (classification/prediction time)
- Robustness:
  - Handling noise and missing values during model learning
- 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

10

Classification Techniques

- Well-known methods:
  - Decision Trees
  - Bayesian Classification
  - Bayesian Belief Networks
- Besides, we can also use
  - Instance-Based Methods
    - k-nearest neighbor (k-NN) approach
  - Artificial Neural Networks
  - Support Vector Machine

Classification by DT Induction



- Decision tree
  - A flow-chart-like tree structure
  - Internal node denotes a test on an attribute
  - Branch represents an outcome of the test
  - Leaf nodes represent class labels or class distribution
- Decision tree generation consists of two phases:
  - Tree construction
  - Tree pruning
    - Identify and remove branches that reflect noise or outliers
- Use of decision tree: Predicting the class labeles of unknown samples

Decision Tree Induction: Training Dataset (buys_computer is class 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

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 samples 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 recursive 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 node
  - There are no samples left

Decision Tree Algorithms (ID3, C4.5, C5.0/See5)

- ID3 (Iterative Dichotomiser 3): A decision tree algorithm invented by Ross Quinlan (RQ)
  - Uses Information Gain for feature selection
- C4.5: An improvement of ID3 by RQ
  - Uses Information Gain Ratio for feature selection
  - Handles both continuous & discrete attributes
  - Handles training data with missing attributes
  - Prunes tree after creation
  - Became a benchmark
- C5.0 (Unix/Linux)/ See5 (Windows): Improvement of C4.5 in terms of speed, memory usage, and smaller decision trees
- CART (Classification and Regression Trees) algorithm
  - The generation of binary decision trees
  - Uses Gini Index for feature selection

Attribute Selection Measure: Information Gain

- Measures how much “information” a feature gives us about the class.
  - Features that perfectly partition should give maximal information.
  - Unrelated features should give no information.
- It measures the reduction in entropy

Information Gain (cont…)

- Measure of the difference in entropy between before-split and after-split of a dataset D based on an attribute A.
- i.e., how much uncertainty in D was reduced after splitting D on attribute A.

$$IG(A,D) = H(D) - \sum_{s \in S} p(s)H(s)$$

  - Where, H(D): Entropy of the dataset D
  - S: The set of subsets created from splitting D by attribute A
  - p(s): The proportion of the number of elements in s to the number of elements in D
  - H(s): Entropy of the subset s
- The attribute with the largest information gain is used to split the dataset D.

Entropy

- Entropy H(D) of a dataset D:
  - Measure of the amount of uncertainty in the dataset D

$$H(D) = -\sum_{c \in C} p(c) \log_2 p(c)$$

  - Where, D: The dataset for which entropy is calculated
  - C: The set of classes in D
  - p(c): The proportion of the number of elements in class c to the number of elements in D, i.e., probability that a randomly selected instance of D will have class label c.
- When H(D) =0, the dataset D is perfectly classified
  - i.e. all elements in D are of the same class

Attribute Selection Measure: Information Gain

- 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, where m is number of classes:

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

- 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

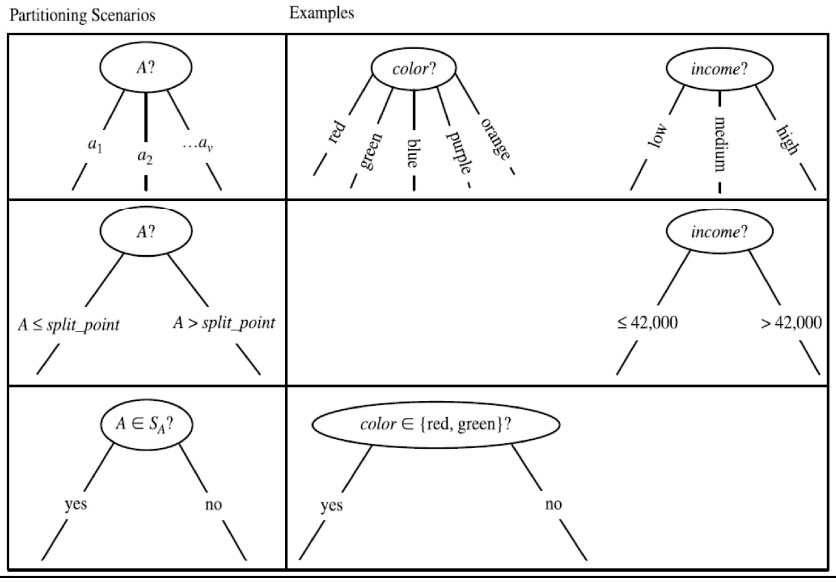
$$Gain(A) = Info(D) - Info_A(D)$$

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

20

Splitting Approaches in DT



21

Attribute Selection: Information Gain

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

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

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

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

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

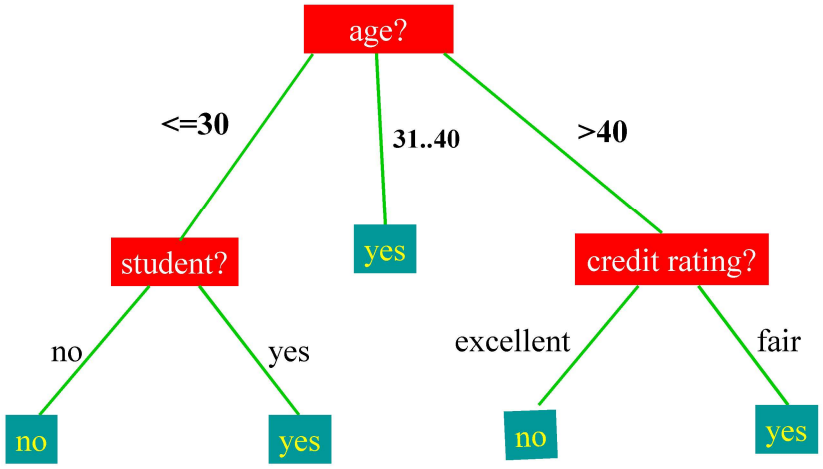
$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes' es and 3 no' s. Hence

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

Similarly,

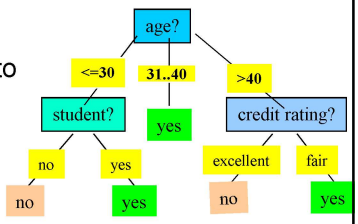
$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Output: A Decision Tree for "buys_computer"



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* <=30 AND *student* = *no* THEN *buys_computer* = *no*

  IF *age* <= 30 AND *student* = *yes* THEN *buys_computer* = *yes*

  IF *age* = 31…40 THEN *buys_computer* = *yes*

  IF *age* > 40 AND *credit_rating* = *excellent* THEN *buys_computer* = *no*

  IF *age* > 40 AND *credit_rating* = *fair* THEN *buys_computer* = *yes*

24

Limitations of Information Gain

- It does not work good for attributes with large number of distinct values (over fitting issue)
- It is biased towards attributes with a large number of values. i.e., it prefers to select attributes having a large number of values.
- Information gain is highest for the attribute which is unique for each data point (i.e., key attribute)
- If the training dataset has key attribute (feature), then the decision tree will terminate in just one iteration, ignoring the importance of other features.

Gain Ratio for Feature Selection

- Gain Ration also called Information Gain Ratio is a ratio of information gain to the intrinsic information (i.e., split information).
- Proposed by Ross Quinlan to reduce a bias towards multi-valued attributes by taking the number and size of branches into account when choosing an attribute.
- Intrinsic information is defined as the information need to determine the branch to which an instance belongs.
- The intrinsic information represents the potential information generated by splitting the dataset into v partitions:

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

26

Gain Ratio for Feature Selection

- High intrinsic info: partitions have more or less the same size
- Low intrinsic info: few partitions hold most of the tuples.
- Gain Ratio if defined as:

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

- The attribute with the maximum gain ratio is selected as the splitting attribute

27

C4.5: Based on Gain Ratio for Feature Selection

- C4.5 (a successor of ID3) uses gain ratio to overcome the problem of over generalization due to IG.
- For example:

$$SplitInfo_A(D) = -\frac{4}{14}\times \log_2(\frac{4}{14}) - \frac{6}{14}\times \log_2(\frac{6}{14}) - \frac{4}{14}\times \log_2(\frac{4}{14}) = 0.926$$

  - gain_ratio(income) = 0.029/0.926 = 0.031

28

Gini Index (CART, IBM IntelligentMiner)

- A measure of impurity of a dataset D.
- Decides how often a randomly chosen element from D would be incorrectly labeled if it were randomly labeled according to the distribution of labels in D.

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

Where p_i is the relative frequency of class i in D

- If a data set D is split on A into two subsets D_1 and D_2, then *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 that provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node
- The Gini index considers a binary split for each attribute.

29

Example: Gini index

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

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

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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_1)$$
$$= \frac{10}{14}(1 - (\frac{6}{10})^2 - (\frac{4}{10})^2) + \frac{4}{14}(1 - (\frac{1}{4})^2 - (\frac{3}{4})^2)$$
$$= 0.450$$
$$= Gini_{income \in \{high\}}(D)$$

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

- All attributes are assumed continuous-valued
- May need other tools, e.g., clustering, to get the possible split values
- Can be modified for categorical attributes

30

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 number of classes is large
    - tends to favor tests that result in equal-sized partitions and purity in both partitions

31

Overfitting and Tree Pruning

- Overfitting: Occurs when a model describes random error or noise instead of the underlying relationship.
- 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"

32

Pros & Cons of Using DT

- Pros:
  - Simple to understand and interpret
  - Requires little data preparation
  - Able to handle both numerical and categorical data
  - Robust
  - Perform well with large data in short time
- Cons:
  - Learning an optimal decision tree is NP-complete
  - Decision-tree learners create over-complex trees that do not generalize the data well.

33

Bayesian Classification

- A statistical classifier: predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: 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
- Assumes class conditional independence to simplify the computation.

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

Bayesian Theorem

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

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

- Informally, this can be written as

  posteriori = likelihood × prior / evidence

- Predicts X belongs to C_i iff the probability $P(C_i \mid \mathbf{X})$ is the highest among all the $P(C_k \mid X)$ for all the *k* classes
- Practical difficulty: require initial knowledge of many probabilities, significant computational cost

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 **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 $P(C_i \mid \mathbf{X})$
- This can be derived from Bayes' theorem $P(C_i|\mathbf{X}) = \frac{P(\mathbf{X}|C_i)P(C_i)}{P(\mathbf{X})}$
- Since P(X) is constant for all classes, only

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

needs to be maximized

Derivation of Naïve Bayes Classifier

- Assumption: attributes are conditionally independent

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

- This greatly reduces the computation cost: Only counts the class distribution
- If A_k is categorical, $P(x_k \mid C_i)$ 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, $P(x_k \mid C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

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

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

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	s_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(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|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(buys_computer="yes"|X) = P(X|buys_computer = "yes") × P(buys_computer = "yes") = 0.028
P(X|buys_computer= "no"|X) = P(X|buys_computer = "no") × P(buys_computer = "no") = 0.007

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

Avoiding the 0-Probability Problem

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

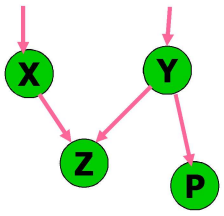
    Prob(income = high) = 11/1003

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., 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

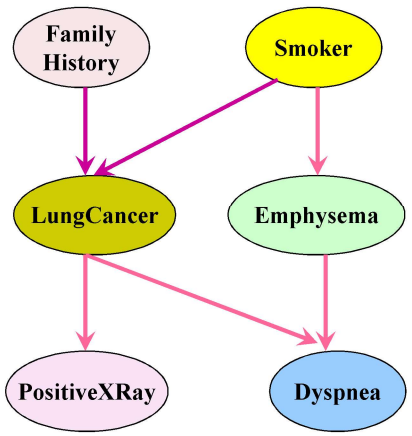
Bayesian Belief Networks

- Bayesian belief network allows a *subset* of the variables conditionally independent
- A graphical model of causal 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 or cycles

Bayesian Belief Network: An Example



The **conditional probability table** (**CPT**) for variable LungCancer:

	(FH, S)	(FH, ~S)	(~FH, S)	(~FH, ~S)
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

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:

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

Bayesian Belief Networks

Lazy vs. Eager Learning

- Lazy learning: Simply stores training data (or only minor processing) and waits until it is given a test tuple. (e.g., Naïve Bayes Classifier)
- Eager learning: Given a set of training set, constructs a classification model before receiving new (e.g., test) data to classify (e.g., Decision Tree classifier)
- Pros & Cons:
  - Lazy requires less time in training but more time in predicting
  - Lazy methods are more accurate as they use many local linear functions to form its implicit global approximation to the target function. Whereas, eager methods must commit to a single hypothesis that covers the entire instance space.

45