Data Mining

UNIT-V

B.Tech(CSE)-VI SEM

UNIT : IV -Classification

- **Bayes Classification Methods**
  - Bayes' Theorem
  - Naive Bayesian Classification
- **Bayesian Belief Networks**
  - Concepts and Mechanisms
  - Training Bayesian Belief Networks

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

- Total probability Theorem: $P(B) = \sum_{i=1}^{M} P(B|A_i)P(A_i)$
- Bayes' Theorem: $P(H|\mathbf{X}) = \dfrac{P(\mathbf{X}|H)P(H)}{P(\mathbf{X})} = P(\mathbf{X}|H) \times P(H) / P(\mathbf{X})$
  - 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**), (i.e., *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) (likelihood): 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

Prediction Based on 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 viewed as

  posteriori = likelihood x prior/evidence

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

Classification Is to Derive the Maximum Posteriori

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

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

needs to be maximized

Prediction Based on 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 viewed as

  posteriori = likelihood x prior/evidence

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

Na¨ıve Bayesian Classification

The na¨ıve Bayesian classifier, or simple Bayesian classifier, works as follows:

- Let D be a training set of tuples and their associated class labels. As usual, each tuple is represented by an n-dimensional attribute vector, X = (x1, x2,…, xn), depicting n measurements made on the tuple from n attributes, respectively, A1, A2,…, An.
- Suppose that there are m classes, C1, C2,…, Cm. Given a tuple, X, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X. That is, the na¨ıve Bayesian classifier predicts that tuple X belongs to the class Ci if and only if

$$P(C_i | X) > P(C_j | X) \text{ for } 1 \le j \le m, \; j \ne i$$

Na¨ıve Bayesian Classification

- we maximize P(Ci |X). The class Ci for which P(Ci |X) is maximized is called the maximum posteriori hypothesis.

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

- As P(X) is constant for all classes, only P(X|Ci)P(Ci) needs to be maximized. If the class prior probabilities are not known, then it is commonly assumed that the classes are equally likely, that is, $P(C_1) = P(C_2) = \cdots = P(C_m)$, and we would therefore maximize P(X|Ci). Otherwise, we maximize P(X|Ci)P(Ci).

Na¨ıve Bayesian Classification

- Given data sets with many attributes, it would be extremely computationally expensive to compute P(X|Ci).To reduce computation in evaluating P(X|Ci), the na¨ıve assumption of class-conditional independence is made.

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

- We can easily estimate the probabilities P(x1|Ci), P(x2|Ci),…, P(xn|Ci) from the training tuples

Na¨ıve Bayesian Classification

- In, Recall of x_k refers to the value of attribute Ak for tuple X, check whether the attribute is categorical or continuous valued.
- If Ak is categorical, then P(xk |Ci) is the number of tuples of class Ci in D having the value xk for Ak , divided by |Ci,D|, the number of tuples of class Ci in D.
- A continuous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ, defined by

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

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

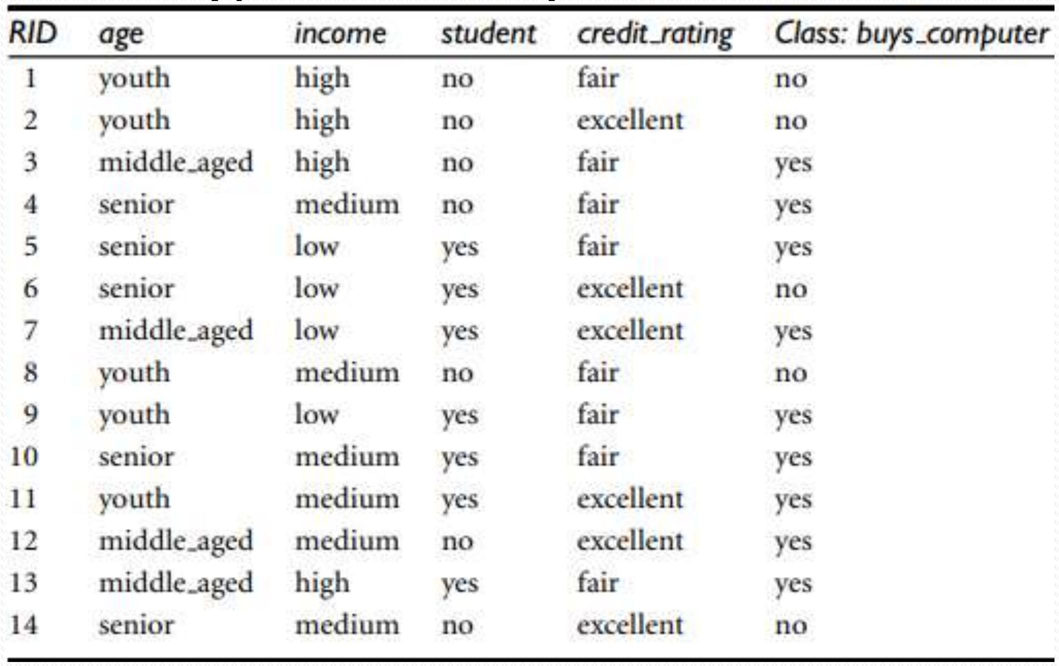
Na¨ıve Bayesian Classification

- To predict the class label of X, P(X|Ci)P(Ci) is evaluated for each class Ci . The classifier predicts that the class label of tuple X is the class Ci if and only if
- $P(X|C_i)P(C_i) > P(X|C_j)P(C_j)$ for $1 \leq j \leq m, j \neq i$.
- i.e., the predicted class label is the class Ci for which P(X|Ci)P(Ci) is the maximum.

Example Na¨ıve Bayesian Classification

- **Predicting a class label using na¨ıve Bayesian classification.**
- predict the class label

of a tuple using na¨ıve Bayesian classification, given the training data. The training data was shown in below table.



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

The data tuples are described by the attributes age, income, student, and credit rating. The class label attribute, buys computer, has two distinct values (namely, {yes, no}). Let C1 correspond to the class buys computer = yes and C2 correspond to buys computer = no.

Example Na¨ıve Bayesian Classification

- The tuple we wish to classify is
- X = (age = youth, income = medium, student = yes, credit rating = fair)

- We need to maximize P(X|Ci)P(Ci), for i = 1, 2. P(Ci)
- The prior probability of each class, can be computed based on the training tuples

  P(buys computer = yes) = 9/14 = 0.643

  P(buys computer = no) = 5/14 = 0.357

Example Na¨ıve Bayesian Classification

- To compute P(X|Ci), fori = 1, 2, we compute the following conditional probabilities:

  P(age = youth | buys computer = yes) = 2/9 = 0.222

  P(age = youth | buys computer = no) = 3/5 = 0.600

  P(income = medium | buys computer = yes) = 4/9 = 0.444

  P(income = medium | buys computer = no) = 2/5 = 0.400

  P(student = yes | buys computer = yes) = 6/9 = 0.667

P(student = yes | buys computer = no) = 1/5 = 0.200

P(credit rating = fair | buys computer = yes) = 6/9 = 0.667

P(credit rating = fair | buys computer = no) = 2/5 = 0.400

Example Na¨ıve Bayesian Classification

- Using these probabilities, we obtain
- **P(X|buys computer = yes)** =

  P(age = youth | buys computer = yes) ×

  P(income = medium | buys computer = yes) ×

  P(student = yes | buys computer = yes) ×

  P(credit rating = fair | buys computer = yes)

  = 0.222 × 0.444 × 0.667 × 0.667 = 0.044.

- Similarly, P(X|buys computer = no) =

  0.600 × 0.400 × 0.200 × 0.400 = 0.019.

Example Na¨ıve Bayesian Classification

- To find the class, Ci , that maximizes P(X|Ci)P(Ci), we compute
- P(X|buys computer = yes)P(buys computer = yes)

  = 0.044 × 0.643 = 0.028
- P(X|buys computer = no)P(buys computer = no)

  = 0.019 × 0.357 = 0.007

- Therefore, the na¨ıve Bayesian classifier predicts buys computer = yes for tuple X.

Example Na¨ıve Bayesian Classification

Using the Laplacian correction to avoid computing probability values of zero

- Assume for the class buys_computer = yes in some training database, D, containing 1000 tuples
- Among all those, 0 tuples with income = low, 990 tuples with income = medium, and 10 tuples with income = high.
- The probabilities of these events, without the Laplacian correction, are

  0, 0.990 (from 990/1000), and 0.010 (from 10/1000), respectively.

Example Na¨ıve Bayesian Classification

- Using the Laplacian correction for the three quantities, we will add 1 more tuple for each income-value pair.
- By this way, the following probabilities (rounded up to three decimal places) were obtained

  1/ 1003 = 0.001,

  991/ 1003 = 0.988

  11/ 1003 = 0.011
- The "corrected" probability estimates are close to their "uncorrected" counterparts, yet the zero probability value is avoided.

Bayesian Belief Networks

- They are also known as Belief Networks, Bayesian Networks, or Probabilistic Networks.
- The na¨ıve Bayesian classifier makes the assumption of class conditional independence. This simplifies computation
- When the assumption holds true, then the naïve Bayesian classifier is the most accurate in comparison with all other classifiers.
- Bayesian Belief Networks specify joint conditional probability distributions.
- They allow class conditional independencies to be defined between subsets of variables

Bayesian Belief Networks

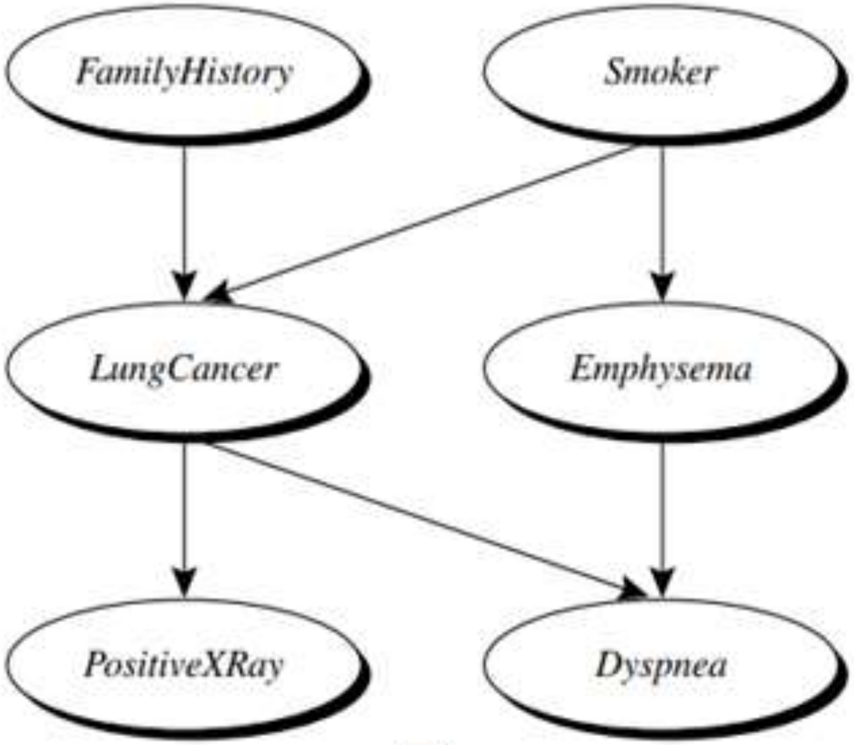
- They provide a graphical model of causal relationships, on which learning can be performed.
- Trained Bayesian belief networks can be used for classification.
- A belief network is defined by two components
  - A direct acyclic graph
  - A set of conditional probability tables

Bayesian Belief Networks

Direct Acyclic Graph

- Each node in the directed acyclic graph represents a random variable. The variables may be discrete- or continuous-valued.
- They may correspond to actual attributes given in the data or to “hidden variables” believed to form a relationship
- Each arc represents a probabilistic dependence.
- If an arc is drawn from a node Y to a node Z, then Y is a parent or immediate predecessor of Z, and Z is a descendant of Y.
- Each variable is conditionally independent of its nondescendants in the graph, given its parents.

Bayesian Belief Networks



(a)

	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

(b)

Simple Bayesian belief network. (a) A proposed causal model, represented by a directed acyclic graph. (b) The conditional probability table for the values of the variable *LungCancer* (*LC*) showing each possible combination of the values of its parent nodes, *FamilyHistory* (*FH*) and *Smoker* (*S*).

Bayesian Belief Networks

Conditional Probability Table

- A belief network has one conditional probability table (CPT) for each variable.
- The CPT for a variable Y specifies the conditional distribution P(Y|Parents(Y)), where Parents(Y) are the parents of Y.
- In the figure, the conditional probability for each known value of LungCancer is given for each possible combination of the values of its parents.

- P(LungCancer = yes|FamilyHistory = yes, Smoker = yes) = 0.8
- P(LungCancer = no |FamilyHistory = no, Smoker = no) = 0.9.

Bayesian Belief Networks

- Let X = (x1,…, xn) be a data tuple described by the variables or attributes Y1,…, Yn, respectively.
- We know that, each variable is conditionally independent of its nondescendants in the network graph, given its parents.
- This allows the network to provide a complete representation of the existing joint probability distribution with the following equation:

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

- where P(x1,…, xn) is the probability of a particular combination of values of X, and
- the values for P(xi |Parents(Yi)) correspond to the entries in the CPT for Yi .

Bayesian Belief Networks

- A node within the network can be selected as an "output" node, representing a class label attribute. There may be more than one output node.
- Various algorithms for inference and learning can be applied to the network.
- Rather than returning a single class label, the classification process can return a probability distribution that gives the probability of each class.
- Belief networks are used to find probability of evidence and most probable explanation queries.

Bayesian Belief Networks

Examples for BBN

- **genetic linkage analysis** (e.g., the mapping of genes onto a chromosome)

By casting the gene linkage problem in terms of inference on Bayesian networks, and using state-of-the art algorithms, the scalability of such analysis has advanced considerably.

- computer vision (e.g., image restoration and stereo vision)
- document and text analysis
- Decision support systems
- sensitivity analysis.

Training Bayesian Belief Networks

How does a Bayesian belief network learn?

- In the learning or training of a belief network, a number of scenarios are possible.
- The network topology (or “layout” of nodes and arcs) may be constructed by human experts or inferred from the data.
- The network variables may be observable or hidden in all or some of the training tuples.
- Several algorithms exist for learning the network topology from the training data given observable variables.
- The problem is one of discrete optimization

Training Bayesian Belief Networks

Human experts usually have a good grasp of the direct conditional dependencies that hold in the domain under analysis, which helps in network design.

Experts must specify conditional probabilities for the nodes that participate in direct dependencies. These probabilities can then be used to compute the remaining probability values.

Training Bayesian Networks: Several Scenarios

- Scenario 1: Given both the network structure and all variables observable: *compute only the CPT entries*
- Scenario 2: Network structure known, some variables hidden: *gradient descent* (greedy hill-climbing) method, i.e., search for a solution along the steepest descent of a criterion function
  - Weights are initialized to random probability values
  - At each iteration, it moves towards what appears to be the best solution at the moment, w.o. backtracking
  - Weights are updated at each iteration & converge to local optimum
- Scenario 3: Network structure unknown, all variables observable: search through the model space to *reconstruct network topology*
- Scenario 4: Unknown structure, all hidden variables: No good algorithms known for this purpose

Training Bayesian Belief Networks

Let D be a training set of data tuples, $X_1, X_2, \ldots, X_{|D|}$

Training the belief network means that we must learn the values of the CPT entries.

Let w_{ijk} be a CPT entry for the variable $Y_i = y_{ij}$ having the parents $U_i = u_{ik}$, where $w_{ijk} \equiv P(Y_i = y_{ij} | U_i = u_{ik})$.

The w_{ijk} are viewed as weights, analogous to the weights in hidden units of neural networks

The set of weights is collectively referred to as W. The weights are initialized to random probability values.

A gradient descent strategy is performed for updating the weights.

Training Bayesian Belief Networks

At each iteration, the weights are updated and converge to a local optimal solution.

For our problem, we maximize

$$P_w(D) = \prod_{d=1}^{|D|} P_w(X_d).$$

Given the network topology and initialized wijk, the algorithm proceeds as follows:

1. Compute the gradients: For each i, j, k, compute

$$\frac{\partial \ln P_w(D)}{\partial w_{ijk}} = \sum_{d=1}^{|D|} \frac{P(Y_i = y_{ij}, U_i = u_{ik} | \mathbf{X}_d)}{w_{ijk}}.$$

Training Bayesian Belief Networks

- The probability is to be calculated for each training tuple, Xd, in D.

2 Take a small step in the direction of the gradient: The weights are updated by

$$w_{ijk} \leftarrow w_{ijk} + (l)\,\frac{\partial \ln P_w(D)}{\partial w_{ijk}}$$

3. Renormalize the weights: Because the weights wijk are probability values, they must be between 0.0 and 1.0 and $\sum_j$ wijk must equal 1 for all i, k.

Training Bayesian Belief Networks

Improving the learning rate:

- Algorithms that follow this learning form are called adaptive probabilistic networks.
- Belief networks are computationally intensive.
- Because belief networks provide explicit representations of causal structure, a human expert can provide prior knowledge to the training process in the form of network topology and/or conditional probability values.
- This can significantly improve the learning rate.