Data Mining

UNIT-III

Classification

Course Outcomes

After Successful completion of the Course,the student will be able to:

CO1: Discuss the basic concepts and techniques of data warehousing and data mining. **(K2)**

CO2: Demonstrate the types of the data to be mined and apply pre-processing methods on raw data. **(K3)**

CO3: Illustrate various Classification Techniques. (K3)

CO4: Discover interesting patterns, analyze supervised and unsupervised models and estimate the accuracy of the algorithms. (K3)

CO5: Use different clustering techniques to cluster data. **(K3)**

UNIT IV: Classification:

- **Classification**
  - Basic Concepts
  - Decision Tree Induction
  - Attribute Selection Measures
  - Tree Pruning
- **Bayes Classification Methods:**
  - Bayes Theorem
  - Naïve Bayesian Classification
- **Bayes Belief Networks:**
  - Concepts and Mechanisms

Classification: Basic Concepts

- 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.

- **Classification is a form** of data analysis that extracts models describing important data classes. These models, called classifiers, predict categorical (discrete, unordered) class labels.

- **For example**, we can build a classification model to categorize bank loan applications as either **safe or risky**.

- Classification has numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis

Classification: Basic Concepts

General Idea behind classification

- The data for a classification task consists of a collection of instances (records).
- Each such instance is characterized by the tuple (x, y), where x is the set of attribute values that describe the instance and y is the class label of the instance.
- The attribute set x can contain attributes of any type, while the class label y must be categorical.
- A classification model is an abstract representation of the relationship between the attribute set and the class label.

Classification: Basic Concepts

Task	Attribute set	Class label
Spam filtering	Features extracted from email message header and content	spam or non-spam
Tumor identification	Features extracted from magnetic resonance imaging (MRI) scans	malignant or benign
Galaxy classification	Galaxy classification Features extracted from telescope images	elliptical, spiral, or irregular-shaped

Basic Concepts

Examples of Classification Tasks

In the below examples, the data analysis task is classification , where a model or a classifier is constructed to predict a class(categorical):

- A **bank loans officer** needs analysis of his / her data to learn which loan applicants are “safe” and which are “risky” for the bank. Here the class label should be safe or risky

- A **marketing manager** at *an Electronics company* needs data analysis to help guess whether a customer with a given profile will buy a new computer. Here the class label should be “yes” or “No”.

- A **medical researcher** wants to analyze breast cancer data to predict which one of three specific treatments a patient should receive. Here the class labels should be “treatment A” , “treatment B” and “treatment C”

Classification: Basic Concepts

- **Supervised learning (Classification)**
  - **Supervision:** The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set.

- **Unsupervised learning (Clustering)**
  - The class labels of training data is unknown.
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Classification: Basic Concepts

- **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

General Approach to Classification

Data classification is a two-step process

1. **Learning step** -where a classification **model is constructed**
   - A classifier is built describing a predetermined set of data classes or concepts.
   - A classification algorithm builds the classifier by analyzing or "learning from" a training set made up of database tuples and their associated class labels
2. **Classification step** -where the **model is used** for classification.
   - First, the predictive accuracy of the classifier is estimated.
   - If the accuracy of the classifier is considered acceptable, the classifier can be used to classify future data tuples for which the class label is not known

General Approach to Classification

Step 1: Learning - Training data are analyzed by a classification algorithm. Here, the class label attribute is loan_decision, and the learned model or classifier is represented in the form of classification rules.

General Approach to Classification

Step 2 : Classification - Test data are used to estimate the accuracy of the classification rules. If the accuracy is considered acceptable, the rules can be applied to the classification of new data tuples.

Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- A **decision tree** is a flowchart-like tree structure, where each **internal node** (nonleaf node) denotes a test on an attribute, each **branch** represents an outcome of the test, and each **leaf node** (or *terminal node*) holds a class label. The topmost node in a tree is the root node.
- **The tree has three types of nodes:**
  - A **root node**, with no incoming links and zero or more outgoing links.
  - **Internal nodes**, each of which has exactly one incoming link and two or more outgoing links.
  - **Leaf or terminal nodes**, each of which has exactly one incoming link and no outgoing links.

Decision Tree Induction

- Every leaf node in the decision tree is associated with a class label. The **non-terminal** nodes, which include the root and internal nodes, contain **attribute test conditions** that are typically defined using a single attribute.
- Each possible outcome of the attribute test condition is associated with exactly one child of this node.

- Internal nodes are denoted by **rectangles,** and leaf nodes are denoted by **ovals**. Some decision tree algorithms produce only *binary* trees (where each internal node branches to exactly two other nodes), whereas others can produce non-binary trees.

Decision Tree Induction

- J. Ross Quinlan, a researcher in machine learning, developed a decision tree algorithm known as ID3 (Iterative Dichotomiser).

- Quinlan later presented C4.5 (a successor of ID3), which became a benchmark to which newer supervised learning algorithms are often compared.

- ID3, C4.5, and **CART(Classification and Regression Trees)** adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.

- Most algorithms for decision tree induction also follow a top-down approach, which starts with a training set of tuples and their associated class labels.

- The training set is recursively partitioned into smaller subsets as the tree is being built.

Decision Tree Induction: An Example

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

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

Class-Labeled Training Tuples from the *AllElectronics* Customer Database

A decision tree for the concept **buys_computer**

Algorithm for Decision Tree Induction

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

Decision Tree Induction Algorithm

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  if tuples in D are all of the same class, C, then
(3)      return N as a leaf node labeled with the class C;
(4)  if attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  if splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

Decision Tree Induction

- Example 2:
- Suppose a new species is discovered by scientists. How can we tell whether it is a mammal or a non-mammal?
- The first question we may ask is whether the species is cold- or warm-blooded.

A decision tree for the mammal classification problem.

Decision Tree Induction

Attribute Selection Measures

- An **attribute selection measure** is a heuristic for selecting the splitting criterion that "best" separates a given data partition, *D*, of class-labeled training tuples into individual classes.
- If we were to split *D* into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure
- Attribute selection measures are also known as **splitting rules** because they determine how the tuples at a given node are to be split.
- The attribute selection measure provides a ranking for each attribute describing the given training tuples. The attribute having the best score for the measure is chosen as the *splitting attribute* for the given tuples.

Attribute Selection Measures

- If the splitting attribute is continuous-valued or if we are restricted to binary trees, then, respectively, either a *split point* or a *splitting subset* must also be determined as part of the splitting criterion.

- The tree node created for partition *D* is labeled with the splitting criterion, branches are grown for each outcome of the criterion, and the tuples are partitioned accordingly.

- Three popular attribute selection measures—
  - *information gain*
  - *gain ratio*
  - *Gini index*.

Attribute Selection Measures

1. Information Gain
   - ID3 uses information gain as its attribute selection measure
   - Let node N represent or hold the tuples of partition D
   - The attribute with the highest information gain is chosen as the splitting attribute for node N

- Select the attribute with the highest information gain
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

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

- Information needed (after using A to split D into v partitions) to classify D:

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

- Information gained by branching on attribute A

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

Attribute Selection Measures

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

Class-Labeled Training Tuples from the *AllElectronics* Customer Database

Attribute Selection Measures (Information Gain)

- In this example, each attribute is discrete-valued
- Continuous-valued attributes have been generalized
- The class label attribute, buys computer, has two distinct values (namely, {yes, no}); therefore, there are two distinct classes (that is, m = 2)
- Let class C_1 correspond to 'yes' and class C_2 correspond to 'no'.
- There are nine tuples of class 'yes' and five tuples of class 'no'.
- A (root) node N is created for the tuples in D

Attribute Selection Measures (Information Gain)

Expected information needed to classify a tuple in *D*

- To find the splitting criterion for these tuples, we must compute the information gain of each attribute
- Let us consider Class: buys computer as decision criteria D
- Calculate information:
- $= -p_y \log_2 (p_y) - p_n \log_2 (p_n)$
- Where p_y is probability of 'yes' and p_n is probability of 'no'

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

Attribute Selection Measures (Information Gain)

Calculation of entropy for ‘ Youth’

- Age can be:
  - youth
  - Middle_aged
  - Senior
- Youth

Youth	Class: buys computer
Yes	2
No	3

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

Attribute Selection Measures (Information Gain)

Calculation of entropy for ‘ Youth’

- Calculate Entropy for youth:
- Entropy youth = $-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}$
- Middle_aged

middle	Class: buys computer
Yes	4
No	0

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

Attribute Selection Measures (Information Gain)

Calculation of entropy for ‘ Middle Age’

- Calculate Entropy for middle_aged
- $= -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}$
- $= 0$

- For Senior

Senior	Class: buys computer
Yes	3
No	2

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for senior

Calculate Entropy for senior

$$= -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}$$

The expected information needed to classify a tuple in D according to age

The expected information needed to classify a tuple in D if the tuples are partitioned according to age is

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14}\times(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}) \\ &+ \frac{4}{14}\times(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}) \\ &+ \frac{5}{14}\times(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}) \\ &= 0.694 \text{ bits.} \end{aligned}$$

Attribute Selection Measures (Information Gain)

Calculation information Gain of Age

- Gain of Age:

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Attribute Selection Measures (Information Gain)

Calculation information Gain of Income

- Calculation of gain for income:
- Income cane be:
  - High
  - Medium
  - Low

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for high

- High :

High	Class: buys computer
Yes	2
No	2

- Calculate Entropy for high:

$= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for 'medium'

- Medium:

Medium	Class: buys computer
Yes	4
No	2

- Calculate Entropy for Medium:

$= -(4/6)\log_2(4/6) - (2/6)\log_2(2/6)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Calculate Entropy for 'low'

- Low :

Low	Class: buys computer
No	1
Yes	3

- Calculate Entropy for Low:

  $= -(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_age	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_age	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_age	medium	no	excellent	yes
13	middle_age	high	yes	fair	yes
14	senior	medium	no	excellent	no

Attribute Selection Measures (Information Gain)

Gain of income

- The expected information needed to classify a tuple in D if the tuples are partitioned according to income is:
- $Info_{income}(D) = (4/14)\,(-(2/4)\log_2(2/4) - (2/4)\log_2(2/4)) +$

  $(6/14)\,(-(4/6)\log_2(4/6) - (2/6)\log_2(2/6)) +$

  $(4/14)\,-(1/4)\log_2(1/4) - (3/4)\log_2(3/4)$

  $= 0.911$

Gain of income : $Info(D) - Info_{income}(D)$

$= 0.94 - 0.911 = 0.029$

Attribute Selection Measures (Information Gain)

Calculation of gain for student

- Calculation of gain for student
- Student can be:
  - Yes
  - No

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for 'Yes'

- Yes :

Yes	Class: buys computer
Yes	6
No	1

- Calculate Entropy for Yes:

  $= -(6/7)\log_2(6/7) - (1/7)\log_2(1/7)$

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

Attribute Selection Measures (Information Gain)

Gain of student

- The expected information needed to classify a tuple in D if the tuples are partitioned according to student is:
- $\text{Info}_{\text{Student}}(D) = (7/14)\,(-(3/7)\log_2(3/7) - (4/7)\log_2(4/7)) +$

  $(7/14)\,(-(6/7)\log_2(6/7) - (1/7)\log_2(1/7))$

  $=0.789$
- Gain(student) :

  $\text{Info}(D) - \text{Info}_{\text{student}}(D)$

  $= 0.94 - 0.789 = 0.151$

Attribute Selection Measures (Information Gain)

Calculation of gain for credit rating

- Calculation of gain for credit rating
- Credit rating can be:
  - Fair
  - Excellent

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for Fair

- Fair :

Fair	Class: buys computer
Yes	6
No	2

- Calculate Entropy for Fair:

$= -(6/8)\log_2(6/8) - (2/8)\log_2(2/8)$

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

Attribute Selection Measures (Information Gain)

Calculate Entropy for Excellent

- Excellent :

Yes	Class: buys computer
Yes	3
No	3

- Calculate Entropy for Excellent:

$= -(3/6)\log_2(3/6) - (3/6)\log_2(3/6)$

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

Attribute Selection Measures (Information Gain)

Gain for credit rating

- The expected information needed to classify a tuple in D if the tuples are partitioned according to Credit rating is:
- $\text{Info}_{\text{Credit rating}}(D) = (8/14)\,(-(6/8)\log_2(6/8) - (2/8)\log_2(2/8)) +$

  $(6/14)\,(-(3/6)\log_2(3/6) - (3/6)\log_2(3/6))$

  $=0.892$
- Gain for credit rating :

  $\text{Info}(D) - \text{Info}_{\text{Credit rating}}(D)$

  $= 0.94 - 0.892 = 0.048$

Attribute Selection Measures (Information Gain)

Independent variable	Information gain
Age	0.246
Income	0.029
Student	0.151
Credit_rating	0.048

Attribute Selection Measures (Information Gain)

Selection of root classifier

- Because age has the highest information gain among the attributes, it is selected as the splitting attribute
- Node N is labelled with age, and branches are grown for each of the attribute's values
- The tuples are then partitioned accordingly
- Notice that the tuples falling into the partition for age = middle aged all belong to the same class
- Because they all belong to class "yes," a leaf should therefore be created at the end of this branch and labelled with "yes."

Attribute Selection Measures (Information Gain)

Decision tree

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

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

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

Attribute Selection Measures (Information Gain)

Decision tree

- The final decision tree returned by the algorithm is shown in Figure

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

Attribute Selection Measures (Gain Ratio)

- **Disadvantages of Inforamtion Gain**
  - Need to calculate entroy for every attribute.
  - If any attribute have unique values, then the entropy value will be 0, In this case, the partition will be useless.

- The information gain measure is biased toward tests with many outcomes.
- It prefers to select attributes having a large number of values
- C4.5, a successor of ID3, uses an extension to information gain known as *gain ratio*, which attempts to overcome this bias.

Attribute Selection Measures (Gain Ratio)

- It applies a kind of normalization to information gain using a "split information" value defined with *Info(D)* as

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

- This value represents the potential information generated by splitting the training data set, *D*, into *v* partitions, corresponding to the *v* outcomes of a test on attribute *A*.

Attribute Selection Measures (Gain Ratio)

- The gain ratio is defined as

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)}.$$

- The attribute with the maximum gain ratio is selected as the splitting attribute.
- If the split information approaches 0, the ratio becomes unstable.

Attribute Selection Measures (Gain Ratio)

- Example:
- A test on *income* splits the data of data set into three partitions, namely *low*, *medium*, and *high*, containing four, six, and four tuples, respectively.
- To compute the gain ratio of *income*

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

$$= 1.557.$$

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

Attribute Selection Measures (Gini Index)

- 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*)

Computation of 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

$$gini_{income \in \{low,medium\}}(D) = \left(\frac{10}{14}\right) Gini(D_1) + \left(\frac{4}{14}\right) Gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right)$$

$$= 0.443$$

$$= Gini_{income \in \{high\}}(D).$$

$Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450. Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index

- 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

Comparing Attribute Selection Measures

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

Other Attribute Selection Measures

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

Tree Pruning

- When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers.

- Tree pruning methods address this problem of ***overfitting*** the data. Such methods typically use statistical measures to remove the **least-reliable branches**. An unpruned tree and a pruned version of it are shown below:

- Pruned trees tend to be smaller and less complex and thus, easier to comprehend.

- They are usually faster and better at correctly classifying independent test data (i.e., of previously unseen tuples) than unpruned trees.

How does tree pruning work

- There are two common approaches to tree pruning:
  1) Prepruning
  2) Postpruning

1) Prepruning

•Here, a tree is "pruned" by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node) by using measures such as statistical significance, **information gain, Gini index**, and so on, can be used to assess the **goodness of a split** based on prespecified **threshold.**

•High thresholds could result in oversimplified trees, whereas low thresholds could result in very little simplification.

•Upon halting, the node becomes a leaf.

•The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.

Tree Pruning

2) Postpruning

•It removes subtrees from a "fully grown" tree, by removing its branches and replacing it with a ***leaf***.

•The **leaf** is labeled with the **most frequent class** among the subtree being replaced.

•The **cost complexity** pruning algorithm used in CART is an example of the postpruning approach.

•This approach considers the cost complexity of a tree to be a function of the number of leaves in the tree and the error rate of the tree (where the error rate is the percentage of tuples misclassified by the tree).

•It starts from the bottom of the tree.

Tree Pruning

2) Post pruning:

•For each internal node, N, it computes the cost complexity of the subtree at N, and the cost complexity of the subtree at N if it were to be pruned (i.e., replaced by a leaf node).

•The two values are compared. If pruning the subtree at node N would result in a smaller cost complexity, then the subtree is pruned. Otherwise, it is kept.

•A **pruning set** of class-labeled tuples is used to estimate **cost complexity.**

•C4.5 uses a method called **pessimistic pruning**, which is similar to the cost complexity method in that it also uses error rate estimates to make decisions regarding subtree pruning.

•Pessimistic pruning, however, does not require the use of a prune set. Instead, it uses the training set to estimate error rates.

Tree Pruning

- Alternatively, Prepruning and postpruning may be interleaved for a combined approach.
- Postpruning requires more computation than prepruning, yet generally leads to a more reliable tree.
- No single pruning method has been found to be superior over all others.
- Although pruned trees tend to be more compact than their unpruned counterparts, they may still be rather large and complex.
- Decision trees can suffer from repetition and replication, making them overwhelming to interpret.

Tree Pruning

An example of: (a) subtree **repetition**, where an attribute is repeatedly tested along a given branch of the tree (e.g., *age*) and (b) subtree **replication**, where duplicate subtrees exist within a tree (e.g., the subtree headed by the node "*credit_rating?*").

Tree Pruning

- These situations can impede the accuracy and comprehensibility of a decision tree.
- The use of multivariate splits (splits based on a combination of attributes) can prevent these problems.
- Another approach is to use a different form of knowledge representation, such as rules, instead of decision trees.

Bayes' Classification Methods

- Bayesian classifiers are statistical classifiers.
- They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.
- Bayesian classification is based on Bayes' theorem
- Bayesian classifiers exhibit high accuracy and speed when applied to large databases
- Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence.
- It is made to simplify the computations involved and, in this sense, is considered —naïve.

Bayes' Theorem

- Bayes' theorem is named after Thomas Bayes
- Bayes' theorem is useful as it provides a way of calculating the posterior probability, P(H|X), from P(H),P(X|H) and P(X)
- **Bayes' Theorem is** $$P(H \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid H)P(H)}{\mathrm{P(X)}}$$

  - Let X be a data tuple. In Bayesian terms, X is considered ―evidence‖
  - As usual, it is described by measurements made on a set of n attributes.
  - Let H be some hypothesis such as that the data tuple X belongs to a specified class C.
  - For classification problems, we want to determine P(H|X), the probability that the hypothesis H holds given the ―evidence‖ or observed data tuple X
  - P(H|X) is the posterior probability, or a posteriori probability, of H conditioned on X
  - In contrast, P(H) is the prior probability, or a priori probability, of H
  - P(X|H) is the posterior probability of X conditioned on H
  - P(X) is the prior probability of X

Bayes' Theorem

- For example, suppose our world of data tuples is confined to customers described by the attributes age and income, respectively, and that X is a 35- year-old customer with an income of $40,000.

- Suppose that H is the hypothesis that our customer will buy a computer. Then P(H|X) reflects the probability that customer X will buy a computer given that we know the customer's age and income.

- As P(H) is the prior probability, for our example, this is the probability that any given customer will buy a computer, regardless of age, income, or any other information, for that matter.

- The posterior probability, P(H|X), is based on more information (e.g., customer information) than the prior probability, P(H), which is independent of X.

- Similarly, P(X|H) is the posterior probability of X conditioned on H. That is, it is the probability that a customer, X, is 35 years old and earns $40,000, given that we know the customer will buy a computer

Naïve Bayesian Classification

- Naïve Bayes algorithm is a supervised learning algorithm, which is **based on Bayes theorem** and used for solving classification problems.

- It is mainly used in **text classification** that includes a high-dimensional training dataset.

- Naïve Bayes provides **probabilistic approach (**it predicts on the basis of the probability of an object) to solve classification problem.

- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast **machine learning models** that can make quick predictions.

- This algorithm is quite popular to be used in NLP.

- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Naïve Bayesian Classification

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

Naïve: It is called Naïve because it assumes that the occurrence of a certain feature is **independent of the occurrence of other features**. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.

Bayes: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' theorem is also known as Bayes' Rule or Bayes' law, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

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

Naïve Bayesian Classification

- With many attributes it is computationally expensive to compute $P(X|C_i)$
- Naïve assumption of **"class conditional independence"**

$$P(X|C_i) = P(x_1, x_2, \ldots x_n|C_i)$$

$$= P(x_1|C_i) * P(x_2|C_i) * \ldots P(x_n|C_i)$$

$$= \prod_{K=1}^{n} P(x_k|C_i)$$

To compute $P(X|C_i)$

- **A_k is categorical,**

$$P(x_k|C_i) = \frac{\text{the number of tuples of class } C_i \text{ in D having value } x_k \text{ for } A_k}{\text{the number of tuples of class } C_i \text{ in D}}$$

.

- **A_k is continuous,**

A continous-valued attribute is typically assumed to have a Gaussian distribution with a mean μ and standard deviation σ

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

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

Naïve Bayes Classifier: Training Dataset

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

Class: C1:buys_computer = ‘yes’ ,C2:buys_computer = ‘no’

Data to be classified:

X = (age = “youth”, income= “medium”, student =“yes”, credit_rating = “fair”)

Naïve Bayesian Classifier: Example

age	income	student	credit_rating	buys computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_age	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_age	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_age	medium	no	excellent	yes
middle_age	high	yes	fair	yes
senior	medium	no	excellent	no

- **P(Ci):**

P(buys_computer = "yes") = 9/14 = 0.643

P(buys_computer = "no") = 5/14= 0.357

- **Compute P(X|C) for each class**

•P(age = "youth" | buys_computer = "yes") = 2/9 = 0.222

•P(age = "youth" | 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

Naïve Bayesian Classifier: Example

- **X = (age="youth", income = "medium", student = "yes", credit_rating = "fair")**

- 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 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

- **Find class C_i , that maximizes $P(X| C_i)*P(C_i)$:**
  - P(X|buys_computer = "yes") * P(buys_computer = "yes") = 0.028
  - P(X|buys_computer = "no") * P(buys_computer = "no") = 0.007

- **Prediction: Therefore, X belongs to class ("buys_computer = yes")**

Text classification – An Application of Naïve Bayes Classifier:

- Classify news stories as world, business,Technical, Sports, health etc.
- Classify email as spam / not spam
- Classify business names by industry
- Classify movie reviews as favorable, unfavorable and neutral
- Classify joke as funny, not funny
- Classify opinion as like, hate, neutral

Naïve Bayes Classification

- **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

Bayesian Belief Network(BBN): Concepts and Mechanisms

- Bayesian Belief Networks are probabilistic graphical models.
- They allow the representation of dependencies among subsets of attributes.
- Bayesian Belief Networks can be used for classification
- The Naïve Bayesian classifier makes the assumption of class conditional independence, that is, given the class label of a tuple, the values of the attributes are assumed to be conditionally independent of one another.
- This simplifies computation. When the assumption holds true, then the naïve Bayesian classifier is the most accurate in comparison with all other classifiers.
- In practice, however, dependencies can exist between variables

Bayesian Belief Network(BBN): Concepts and Mechanisms

- Bayesian Belief Networks specify joint conditional probability distributions.
- They allow class conditional independencies to be defined between subsets of variables.
- They provide a graphical model of causal relationships, on which learning can be performed.
- Trained Bayesian belief networks can be used for classification
- Bayesian Belief Networks are also known as belief networks, Bayesian networks, and probabilistic networks.
- A belief network is defined by two components—a directed acyclic graph and a set of conditional probability tables

Bayesian Belief Network(BBN): Concepts and Mechanisms

(a)

Simple Bayesian belief network.

(a)A proposed causal model, represented by a directed acyclic graph.

(b)The conditional probability table

	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)

Bayesian Belief Network(BBN) Concepts and Mechanisms

- 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 Network(BBN) Concepts and Mechanisms

- 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.

- The conditional probability for each known value of LungCancer is given for each possible combination of the values of its parents.

- **For Example:**

  P(LungCancer=yes | FamilyHistory = yes, Smoker = yes)= 0.8

  P(LungCancer=no | FamilyHistory = no, Smoker = no)= 0.9

Bayesian Belief Network(BBN) Concepts and Mechanisms

- Let $X = (x_1, \ldots, x_n)$ be a data tuple described by the variables or attributes $Y_1, \ldots, Y_n$, respectively
- Joint probability distribution is computed as :

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

- where $P(x_1, \ldots, x_n)$ is the probability of a particular combination of values of X, and the values for $P(x_i \mid Parents(Y_i))$ correspond to the entries in the CPT for Y_i

Example of Bayesian Belief Network(BBN)

Exercise=Yes	0.7
Exercise=No	0.3

Diet=Healthy	0.25
Diet=Unhealthy	0.75

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

	HD=Yes	HD=No
CP=Yes	0.8	0.01
CP=No	0.2	0.99

	HD=Yes	HD=No
BP=High	0.85	0.2
BP=Low	0.15	0.8

Example of Bayesian Belief Network(BBN)

Given: X = (E=No, D=Healthy, CP=Yes, BP=High) Compute P(HD|E,D,CP,BP)?

P(HD=Yes| E=No,D=Healthy) = 0.45
P(CP=Yes| HD=Yes) = 0.8
P(BP=High| HD=Yes) = 0.85
P(HD=Yes|E=No,D=Healthy,CP=Yes,BP=High)
$\approx 0.45 \times 0.8 \times 0.85 = 0.306$

P(HD=No| E=No,D=Healthy) = 0.55

P(CP=Yes| HD=No) = 0.01

P(BP=High| HD=No) = 0.2
P(HD=No|E=No,D=Healthy,CP=Yes,BP=High)
$\approx 0.55 \times 0.01 \times 0.2 = 0.0011$

Classify X as Yes

Training Bayesian Belief Networks (BBN)

- 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
- The hidden data case is also referred to as missing values or incomplete data
- If the network topology is known and the variables are observable, then training the network is straightforward.
- It consists of computing the CPT entries, as is similarly done when computing the probabilities involved in Naïve Bayesian classification
- When the network topology is given and some of the variables are hidden, there are various methods to choose from for training the belief network.

Training Bayesian Belief Networks (BBN)

Gradient Descent method

- 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 $w_{ijk} \equiv (Y_i = y_{ij} \mid U_i = u_{ik})$
- for eg. Y_i= {LungCancer=¯yes‖} and U_i={FamilyHistory=¯yes‖, Smoker=¯yes‖}
- 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

Training Bayesian Belief Networks (BBN)

Gradient Descent method

- A gradient descent strategy performs greedy hill-climbing
- At each iteration, the weights are updated and will eventually converge to a local optimum solution
- A gradient descent strategy is used to search for the w_{ijk} values that best model the data, based on the assumption that each possible setting of w_{ijk} is equally likely. Such a strategy is iterative.
- It searches for a solution along the negative of the gradient (i.e., steepest descent) of a criterion function
- We want to find the set of weights, W, that maximize this function. To start with, the weights are initialized to random probability values.

Training Bayesian Belief Networks (BBN)

Gradient Descent method

- The gradient descent method performs greedy hill-climbing in that, at each iteration or step along the way, the algorithm moves toward what appears to be the best solution at the moment, without backtracking. The weights are updated at each iteration
- Eventually, they converge to a local optimum solution. For our problem, we maximize: $P_w(D) = \prod_{d=1}^{|D|} P_w(\boldsymbol{X}_d)$
- This can be done by following the gradient of $\ln P_w(S)$, which makes the problem simpler.

Training Bayesian Belief Networks (BBN)

Gradient Descent method

- Given the network topology and initialized w_{ijk} , the algorithm proceeds as follows:

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

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

  - The probability on the right side of equation is to be calculated for each training tuple, X_d, in D(referred to as p).
  - When the variables represented by Y_i and U_i are hidden for some X_d, then the corresponding probability p can be computed from the observed variables of the tuple using standard algorithms for Bayesian network inference

Training Bayesian Belief Networks (BBN)

Gradient Descent method

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

$$w_{ijk} \leftarrow w_{ijk} + (l)\frac{\partial \ln P_w(D)}{\partial w_{ijk}}, \qquad (2)$$

◦where l is the learning rate representing the step size and is con $\frac{\partial \ln P_w(D)}{\partial w_{ijk}}$ m Eq. (1)

◦The learning rate is set to a small constant and helps with convergence

3. Renormalize the weights: Because the weights w_{ijk} are probability values, they must be between 0.0 and 1.0, and must equal 1 for all i, k.

◦These criteria are achieved by renormali $\sum_j w_{ijk}$ veights after they have been updated by Eq. (2).

▪Algorithms that follow this learning form are called adaptive probabilistic networks

Bayesian Belief Networks (BBN)

- 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.

Reference:

- Data Mining Concepts and Techniques, Jiawei Han, Micheline Kamber, Jian Pei,3rd Edition, Morgan Kaufmann Publishers