Unit 6
Classification and Prediction

<u>Prepared By</u>

Arjun Singh Saud, Asst. Prof. CDCSIT

Classification and Prediction

- Classification and Prediction are two major categories of prediction problems which are usually dealt with Data mining and machine learning. The terms prediction and regression are used synonymously in in data mining.
- Both of them are supervised learning approaches. Classification is the process of finding or discovering a model or function which helps to predict class label for a given data.
- Prediction is the process of finding a model or function which is used to predict continuous real-valued output for a given data.

Classification and Prediction

- For example, we can build a classification model to categorize bank loan applications as either *safe or risky*. We can also construct a classification model to identify digits.
- On the other hand, we can build a regression model to predict the expenditures of a potential customers on computer equipment given their income and occupation. We can also build a prediction model to predict stock price given historical trading data.

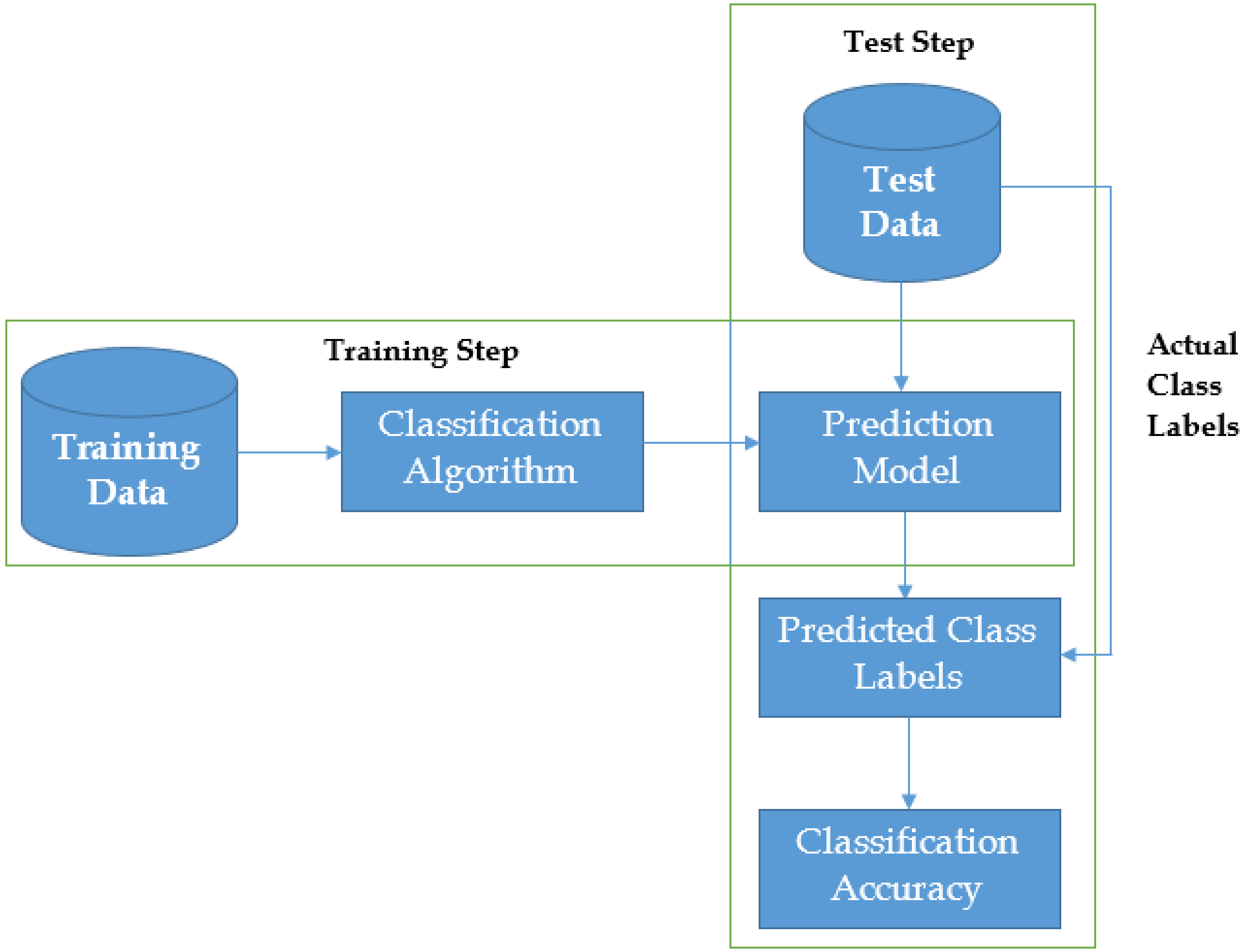
Learning and Testing of Classification

- The Classification process works in following two steps: Learning Step and Testing Step
- **Learning Step:** This step is also called training step or training phase. In this step the learning algorithms build a model on the basis of relationship between input and output in the training dataset. This dataset contains input attributes along with class label for every input tuple.
- Because the class label of each training tuple *is provided,* this step is also known as supervised learning

Learning and Testing of Classification

- **Testing Step:** In this step, the model is used for prediction. Here the test dataset is used to estimate the accuracy of the model. This dataset contains values of input attributes along with class label of the output attribute.
- However, the model only takes values of input attributes and predicts class label of each input tuple.
- Then, accuracy of the model is computed by looking at predicted class labels and actual class labels of test dataset. The model can be applied to the new data tuples if the accuracy is considered acceptable.

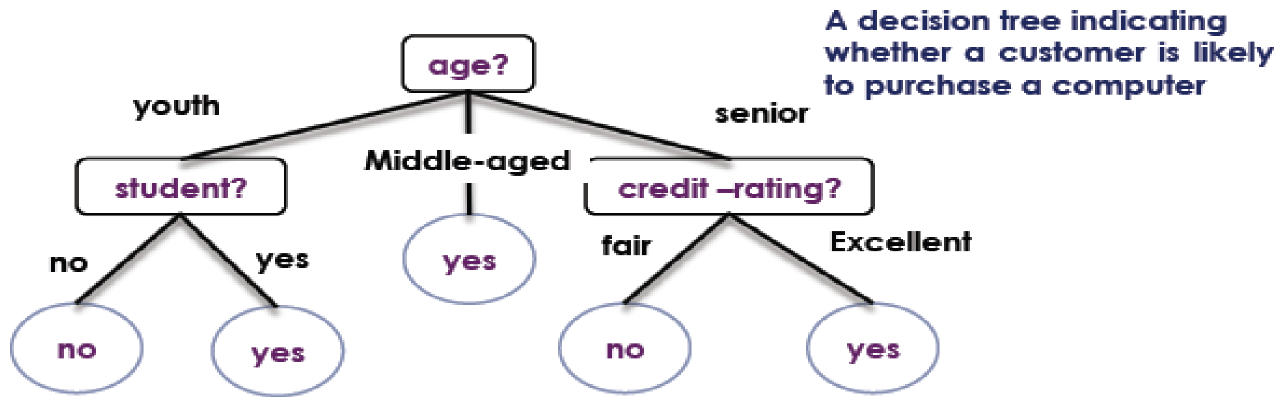
Learning and Testing of Classification



Classification by Decision Tree Induction

- Decision tree induction is the learning of decision trees from class labeled training tuples.
- Decision tree is a flowchart-like tree structure where internal nodes (non leaf node) denotes a test on an attribute, branches represent outcomes of tests, and Leaf nodes (terminal nodes) hold class labels.

Classification by Decision Tree



Class-label Yes: The customer is likely to buy a computer
Class-label no: The customer is unlikely to buy a computer

- In order to make prediction for a tuple, the attributes of a tuple are tested against the decision tree. A path is traced from the root to a leaf node which determines the predicted class for that tuple.

Classification by Decision Tree

Why decision trees classifiers are so popular?

- The construction of a decision tree does not require any domain knowledge or parameter setting
- They can handle high dimensional data
- Intuitive representation that is easily understood by humans
- Learning and classification are simple and fast
- They have a good accuracy

Classification by Decision Tree

Algorithm for Constructing Decision Tress

- Constructing a Decision tree uses greedy algorithm. Tree is constructed in a top-down divide-and-conquer manner.
  1. At start, all the training tuples are at the root
  2. Tuples are partitioned recursively based on selected attributes
  3. If all samples for a given node belong to the same class
     - Label the class
  4. Else if there are no remaining attributes for further partitioning
     - Majority voting is employed for assigning class label to the leaf
  5. Else
     - Got to step 2

Classification by Decision Tree

- There are many variations of decision-tree algorithms. Some of them are: *ID3 (Iterative Dichotomiser 3), C4.5 (successor of ID3), CART (Classification and Regression Tree) etc*.
- There are different attribute selection measures used by decision tree classifiers. Some of them are: *Information Gain, Gain Ratio, Gini Index* etc.

ID3 Decision Tree

- ID3 stands for Iterative Dichotomiser 3. It uses top-down greedy approach to build decision tree model.
- This algorithm computes information gain for each attribute and then selects the attribute with the highest information gain.
- Information gain measures reduction in entropy after data transformation. It is calculated by comparing entropy of the dataset before and after transformation.
- Entropy is the measure of homogeneity of the sample. Entropy or expected information of dataset D is calculated by using equation 1.

ID3 Decision Tree

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

Where p_i is the probability of a tuple in D belonging to class C_i and is estimated using Equation 2.

$$pi = \frac{|C_{i,D}|}{|D|} \qquad (2)$$

Where $|C_{i,D}|$ is the number of tuples in D belonging to class C_i and $|D|$ is the number of tuples in D.

ID3 Decision Tree

- Suppose we have to partition the tuples in D on some attribute A having v distinct values. The attribute A can be used to split D into v partitions $\{D_1, D_2, \ldots\ldots, D_v\}$. Now, the total entropy of data partitions while partitioning D around attribute A is calculated using Equation 3.

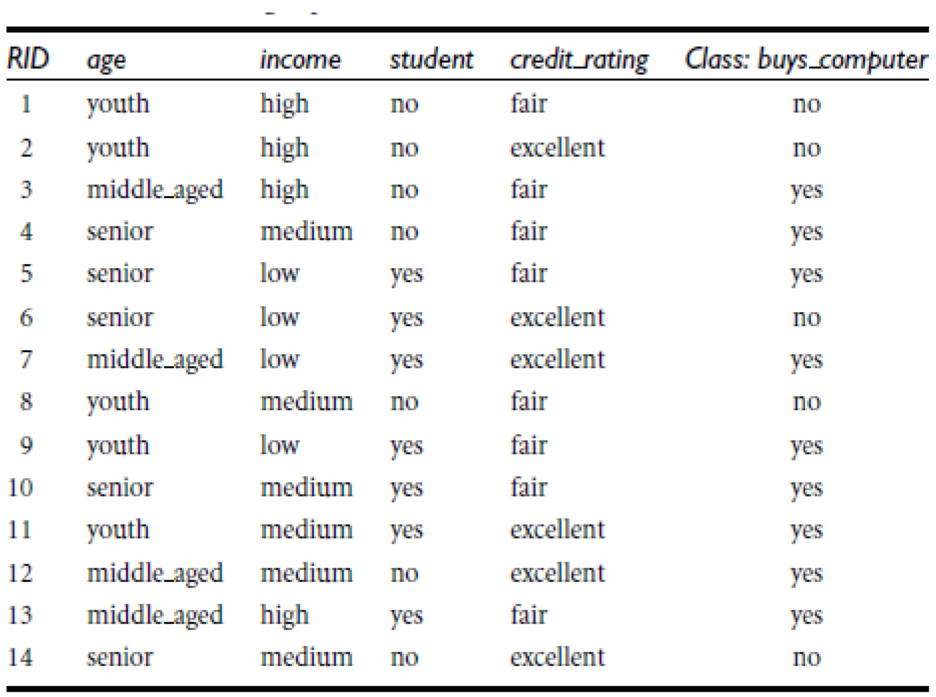
$$E_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times E(D_j) \qquad (3)$$

- Finally, the information gain achieved after partitioning D on attribute A is calculated using Equation 4.

$$IG(A) = E(D) - E_A(D) \qquad (4)$$

ID3 Decision Tree

Example



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

- Construct decision tree from above data using information gain.
- Predict class level of the tuple: ***X*** = (*age* = *youth, income* = *medium, student* = *yes, credit_rating* = *fair*)

ID3 Decision Tree

Solution

Expected information needed to classify a tuple in *D is*:

$$E(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.94$$

Expected information needed to classify a tuple in *D* if the tuples are partitioned according to *age* is:

$$E_{age}(D) = \frac{5}{14}\left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14}\left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) + \frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) = 0.694$$

ID3 Decision Tree

Solution

Hence, the gain in information from such a partitioning would be

$$Gain(Age) = E(D) - E_{Age}(D) = 0.94 - 0.694 = 0.246$$

Similarly,

$$E_{Income}(D) = \frac{4}{14}\left(-\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4}\right) + \frac{6}{14}\left(-\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6}\right) + \frac{4}{14}\left(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4}\right) = ?$$

$$\text{Gain(Income)} = \text{E(D)} - \text{E}_{\text{Income}}\text{(D)} = 0.029$$

ID3 Decision Tree

Solution

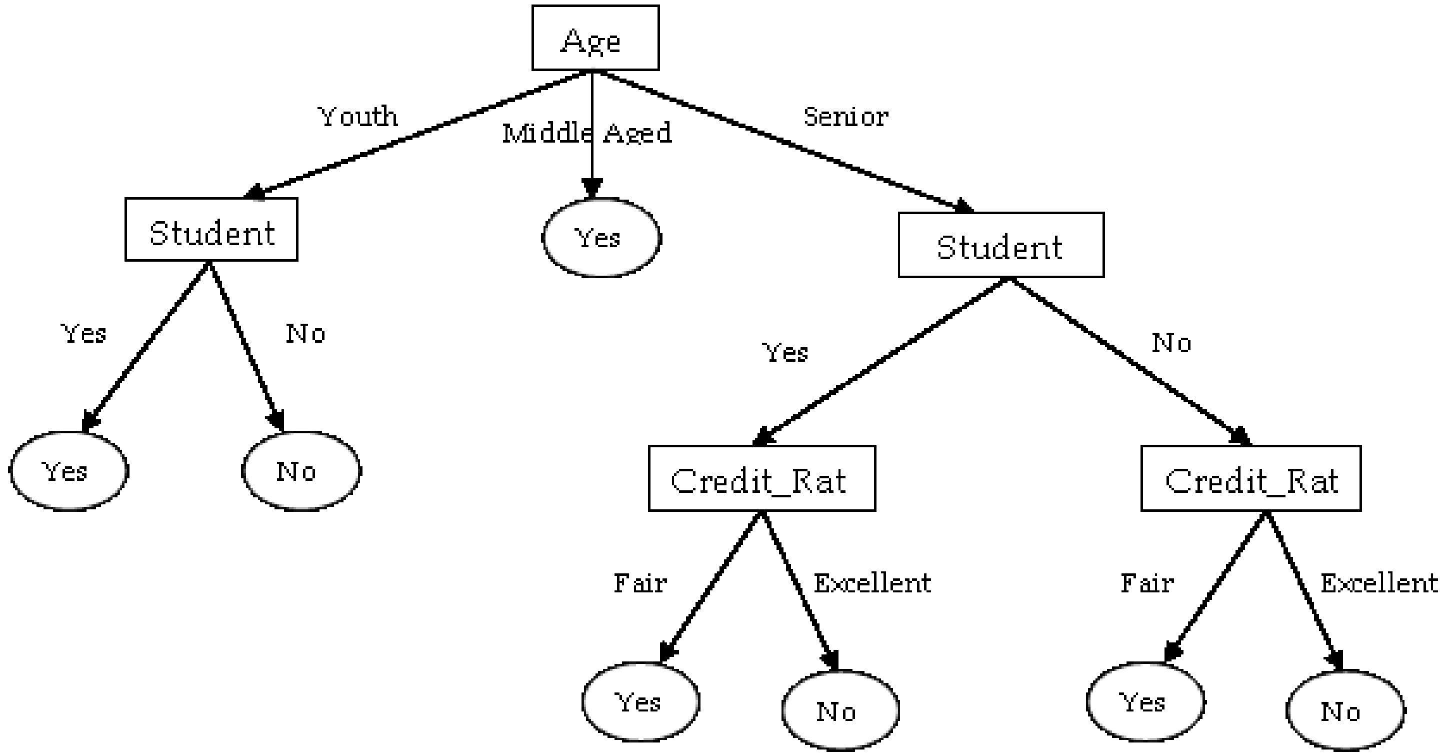
- Similarly, we can get

  Gain (*student*) = 0.151

  Gain (*credit_rating*) = 0.048.

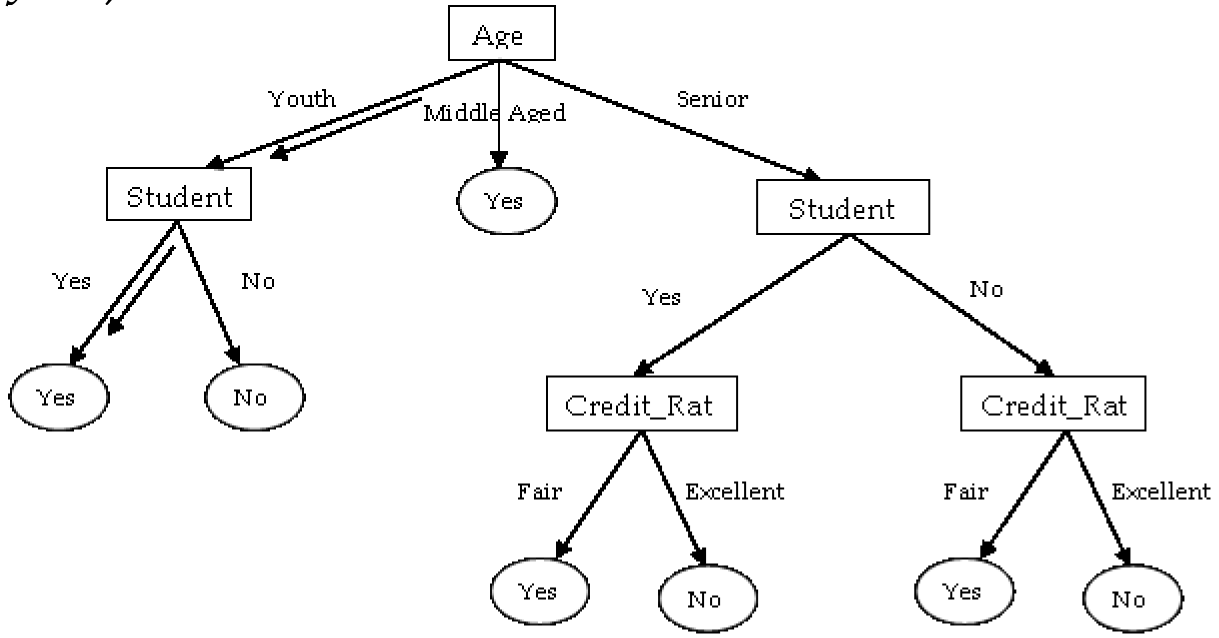
- Thus we partition the tree in the order of attributes: age, student, credit_rating, and income respectively. Thus we get following decision tree.

ID3 Decision Tree



ID3 Decision Tree

Given Test tuple: ***X*** = (*age* = *youth*, *income* = *medium*, *student* = *yes*, *credit_rating* = *fair*)



Thus, The predicted class for the given data is: **Buy_Computer=Yes**

Bayesian Classification

- Bayesian classification is based on Bayes' theorem. It is also called Naïve Bayes Classification or Naïve Bayesian Classification.
- Bayes Theorem is given by:

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

- Bayes' theorem is useful in that it provides a way of calculating the posterior probability, $P(H|\boldsymbol{X})$, from $P(H)$, $P(\boldsymbol{X}|H)$, and $P(\boldsymbol{X})$. Here P(X) and P(H) are prior probability.

Bayesian Classification

- Let D be a database and $C_1, C_2 \ldots\ldots C_m$ are m classes. Now above Bayes rule can be written as:

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}$$

- Given a tuple, ***X***, the classifier will predict that ***X*** belongs to the class having the highest posterior probability, conditioned on ***X***. Thus we need to maximize $P(C_i \mid \boldsymbol{X})$. As $P(\boldsymbol{X})$ is constant for all classes, only $P(\boldsymbol{X} \mid C_i)P(C_i)$ need to be maximized.

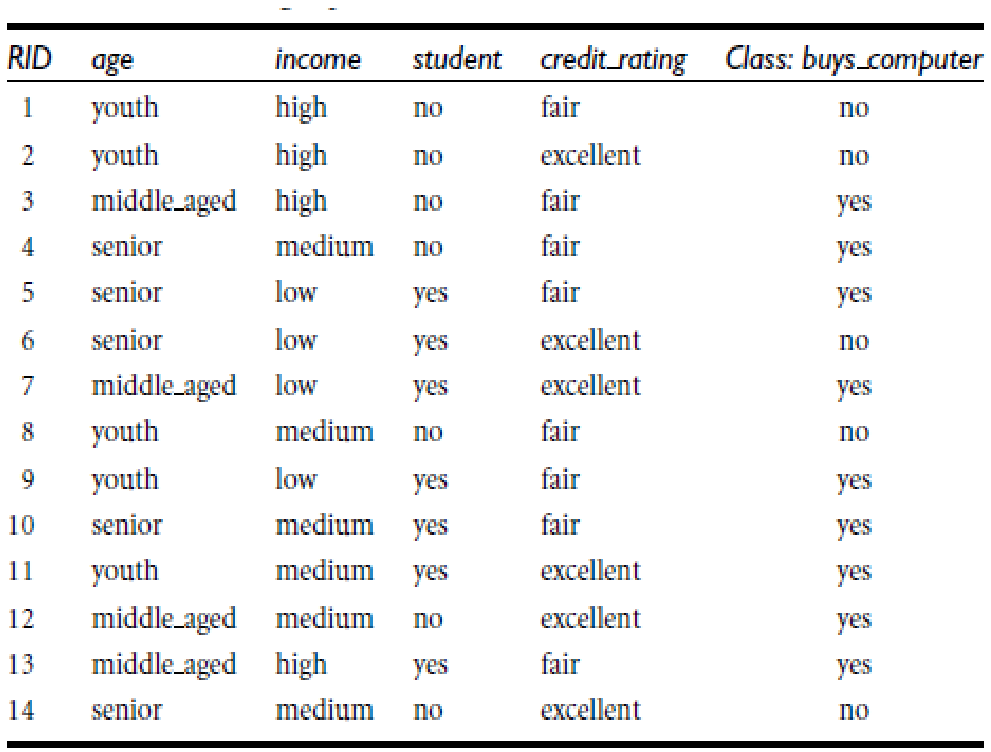
Bayesian Classification

- Let X is the set of attributes {x_1, x_2, x_3…….x_n} where attributes are independent of one another. Now the probability P(X|C$_i$) is given by the equation given below:

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

Bayesian Classification

Example



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

- Predict class level of the tuple: ***X*** = (*age* = *youth, income* = *medium, student* = *yes, credit_rating* = *fair*) using Bayesian classification.

Bayesian Classification

Solution

Prior probability of each class can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$

$$P(buys\_computer = no) = 5/14 = 0.357$$

Compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$

$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.6$$

Bayesian Classification

Solution

$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$

$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.4$$

$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$

$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.2$$

$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$

$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.4$$

Bayesian Classification

Solution

Using the above probabilities, we obtain

$P(\mathbf{X}|buys\_computer{=}yes){=}P(age{=}youth|buys\_computer{=}yes)$
$\times P(income{=}medium|buys\_computer{=}yes)\times P(student{=}yes|$
$buys\_computer{=}yes)\times P(credit\_rating{=}fair|buys\_computer{=}yes){=}$
$0.222\times 0.444\times 0.667\times 0.667 = 0.044$

$P(\mathbf{X}|buys\ \ computer{=}no) = P(age{=}youth|buys\_computer{=}no)$
$\times P(income = medium|buys\_computer{=}no) \times P(student{=}yes|$
$buys\_computer{=}no) \times P(credit\ \ rating{=}\ fair|buys\_computer{=}no){=}$
$0.6*0.4*0.2*0.4 = 0.019$

Bayesian Classification

Solution

Now,

$$P(buys\_computer=yes \mid \text{X})=P(\mathbf{X} \mid buys\_computer=yes) \times P(buys\_computer=yes)= 0.044*0.643 = 0.028$$

$$P(buys\_computer=no \mid \text{X})=P(\mathbf{X} \mid buys\_computer=no) \times P(buys\_computer=no)= 0.019*0.357 = 0.007$$

Since

$P(buys\_computer=yes \mid \text{X}) > P(buys\_computer=no \mid \text{X})$,

Prediction for X is: ***buys_computer = yes.***

Laplace Smoothing

- if probability value is zero for some $P(x_k|C_i)$, it results to zero probability for $P(X|C_i)$.
- Zero probability cancels the effects of all the other (posteriori) probabilities (on C_i) involved in the product.
- Laplace Smoothing is the technique to solve this problem. It adds 1 to each count. The main philosophy behind the concept is that the database is normally large and adding 1 to the count makes negligible difference in probability.

Classification By Backpropagation

- A popular method for the training of multilayer perceptrons is the backpropagation algorithm. The training proceeds in two phases:
  - **Forward Phase**: In this phase, the synaptic weights of the network are fixed and the input signal is propagated through the network, layer by layer, until it reaches the output. Thus, in this phase, changes are confined to the activation potentials and outputs of the neurons in the network.
  - **Backward Phase**: In this phase, an error signal is produced by comparing the output of the network with a desired response. The resulting error signal is propagated in backward direction through the network, again layer by layer. In this second phase, successive adjustments are made to the synaptic weights of the network.

Classification By Backpropagation

Back-Propagation Algorithm

1. Initialize all weights and biases in *network*
2. While terminating condition is not satisfied
3. for each training tuple ***X*** in *D*
4. for each input layer unit *j: Oj* = *Ij*; //output of an input unit is its actual input value
5. for each hidden or output layer unit *j*

   compute the net input of unit j with respect to the previous layer, $I_j = \sum_i w_{ij} O_i + \Theta j$

   compute the output of each unit j $O_j = \frac{1}{1+e^{-I_j}}$

Classification By Backpropagation

<u>**Back-Propagation Algorithm**</u>

```
6.  for each unit j in the output layer
        $Err_j = O_j(1-O_j)(T_j-O_j)$; // compute the error
7.  for each unit j in the hidden layers, from the last to the first hidden layer
        $Err_j = O_j(1-O_j) \sum_k Err_k w_{jk}$
        //compute the error with respect to the next higher layer, k
8.  for each weight $w_{ij}$ in network
        $\Delta wij = (l) Err_j O_i$;
        //weight increment, where l is learning rate
        $wij = wij + \Delta wij$; // weight update
```

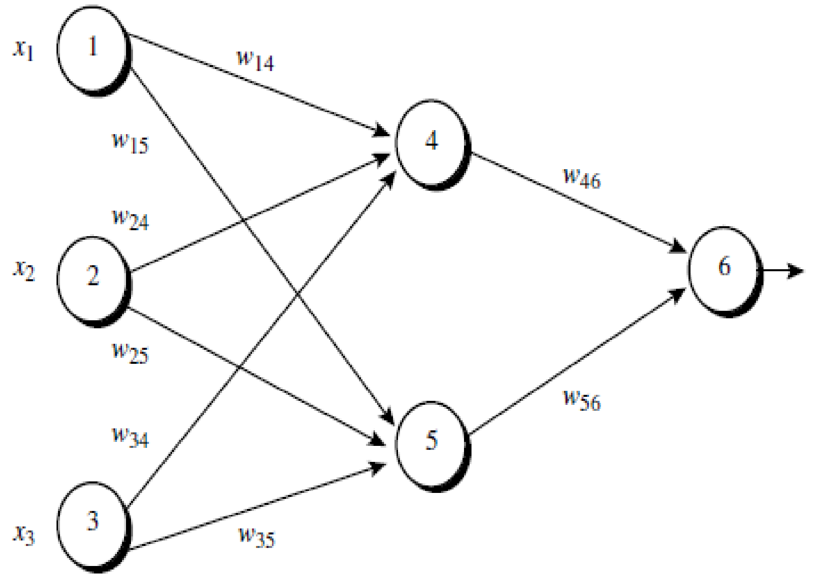
Classification By Backpropagation

Back-Propagation Algorithm

```
9.  for each bias $\theta j$ in network
        $\Delta\theta j = (l)Err_j$; // bias increment
        $\theta j = \theta j + \Delta\theta j$;// bias update
```

Classification By Backpropagation

Example: Consider a multilayer feed-forward neural network given below. Let the learning rate be 0.9. The initial weight and bias values of the network are given in table, along with the first training tuple, ***X*** = (1, 0, 1), whose target output is 1. Show weight updates by using back-propagation algorithm.



x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	−0.3	0.4	0.1	−0.5	0.2	−0.3	−0.2	−0.4	0.2	0.1

Classification By Backpropagation

Solution

Net input and output calculation

Unit j	Net input, I_j	Output, O_j
4	$0.2+0-0.5-0.4=-0.7$	$1/(1+e^{0.7})=0.332$
5	$-0.3+0+0.2+0.2=0.1$	$1/(1+e^{-0.1})=0.525$
6	$(-0.3)(0.332)-(0.2)(0.525)+0.1=-0.105$	$1/(1+e^{0.105})=0.474$

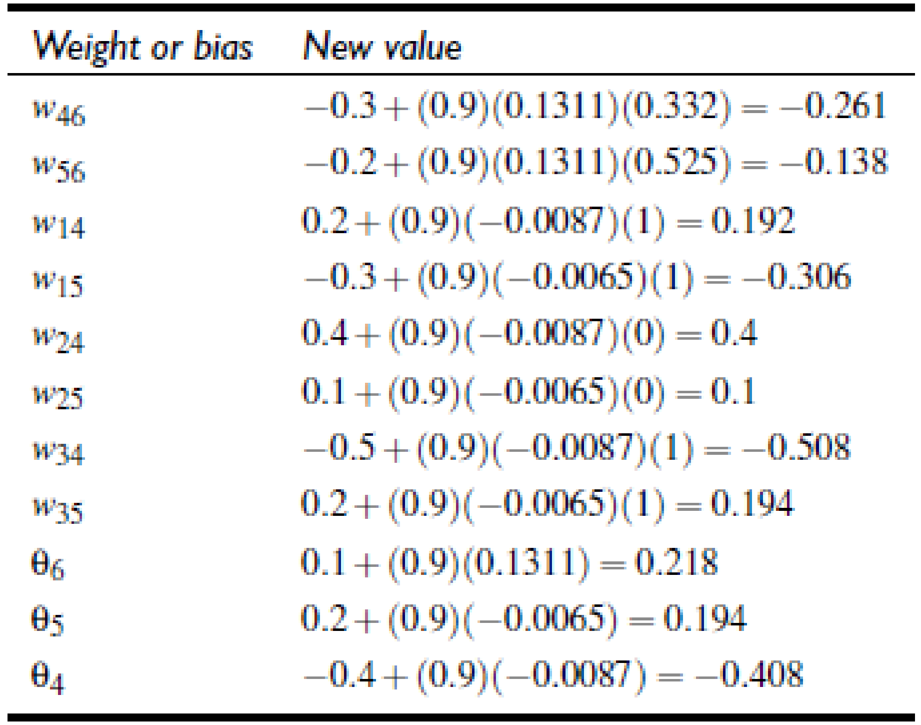
Errors at each node

Unit j	Err_j
6	$(0.474)(1-0.474)(1-0.474)=0.1311$
5	$(0.525)(1-0.525)(0.1311)(-0.2)=-0.0065$
4	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

Classification By Backpropagation

Solution

Calculate weight and bias update



Weight or bias	New value
w_{46}	$-0.3+(0.9)(0.1311)(0.332)=-0.261$
w_{56}	$-0.2+(0.9)(0.1311)(0.525)=-0.138$
w_{14}	$0.2+(0.9)(-0.0087)(1)=0.192$
w_{15}	$-0.3+(0.9)(-0.0065)(1)=-0.306$
w_{24}	$0.4+(0.9)(-0.0087)(0)=0.4$
w_{25}	$0.1+(0.9)(-0.0065)(0)=0.1$
w_{34}	$-0.5+(0.9)(-0.0087)(1)=-0.508$
w_{35}	$0.2+(0.9)(-0.0065)(1)=0.194$
θ_6	$0.1+(0.9)(0.1311)=0.218$
θ_5	$0.2+(0.9)(-0.0065)=0.194$
θ_4	$-0.4+(0.9)(-0.0087)=-0.408$

Rule Based Classification

- A **rule-based classifier** uses a set of If-Then rules for classification. An **If-Then** rule is an expression of the form

  IF *condition* THEN *conclusion*.

  An example is rule *R*1,

  *R*1: If *age*=*youth* AND *student*=*'yes'* Then *buys_computer*=*'yes'*

- The "If" part of a rule is known as the **antecedent** or **precondition**. The "Then" part is the **consequent**. In the antecedent, the condition consists of one or more *attribute tests* that are logically ANDed. The rule's consequent contains a class prediction.

Rule Based Classification

- If the condition in an antecedent holds true for a given tuple, we say that the antecedent is satisfied (or simply, that the rule is satisfied) and that the rule covers the tuple.
- A rule R can be assessed by its coverage and accuracy. Given a tuple, X, from a class labeled data set, D, let n_{covers} be the number of tuples that satisfy rules antecedent; $n_{correct}$ be the number of tuples correctly classified by R; and $|D|$ be the number of tuples in D. We can define the coverage and accuracy of R as below.

Rule Based Classification

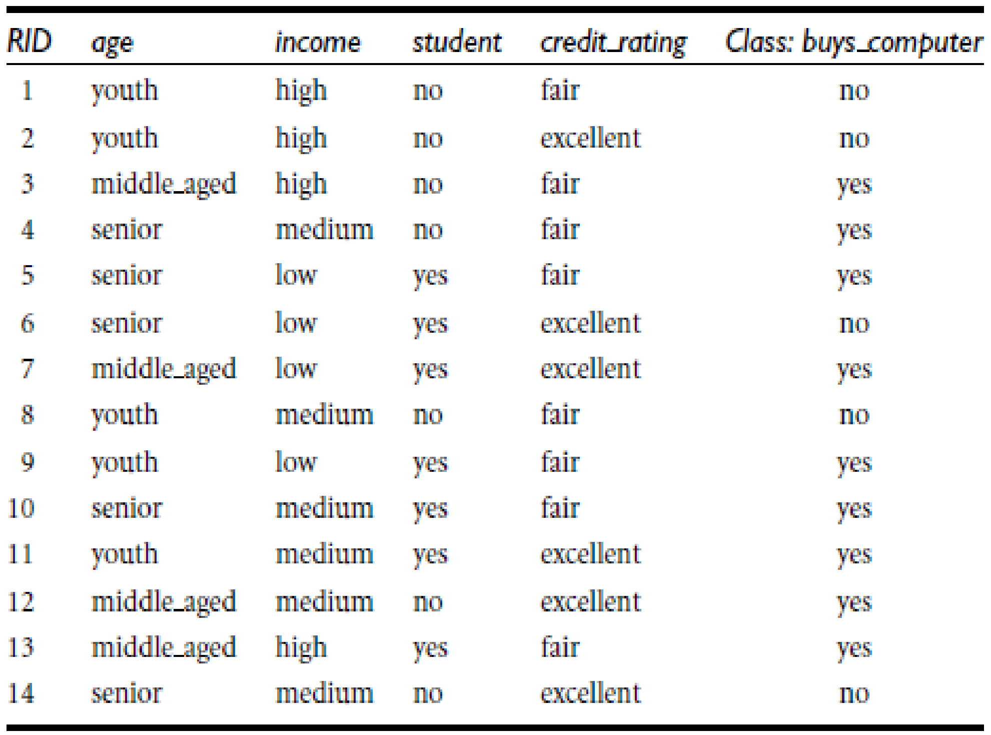
- **Rule Coverage**: A rule's coverage is the percentage of tuples that that satisfy rule's antecedent.
- **Rule Accuracy**: A rule's accuracy is the percentage of tuples that are correctly classified by the rule.

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

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

Rule Based Classification

Example



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

- Find coverage and accuracy of the rule *R*1: If *age=youth* AND *student='yes'* Then *buys_computer='yes'*

Rule Based Classification

Solution

The rule covers 2 of the 14 tuples (i.e. n_{cover}=2)

Therefore, *coverage* (*R*1)=2/14= 14.28%

It can correctly classify both tuples (i.e $n_{correct}$=2)

Therefore, *accuracy*(*R*1)=2/2=100%.

Rule Based Classification

Predicting Class Using Rule Based Classification

- If a rule is satisfied by ***X***, the rule is said to be **triggered**. For example, suppose we have ***X***= (*age*=*youth*, *income*=*medium*, *student*=*yes*, *credit_rating*=*fair*).
- We would like to classify ***X*** according to *buys_computer*. ***X*** satisfies *R*1, which triggers the rule.
- If *R*1 is the only rule satisfied, then the rule **fires** by returning the class prediction for ***X***. If more than one rule is triggered, we may have a potential problem.
- What if they each specify a different class? Or what if no rule is satisfied by ***X***?

Rule Based Classification

- If more than one rule is triggered, we need a **conflict resolution strategy** to figure out which rule gets to fire. Two such strategies are: *size ordering* and *rule ordering*.
- The **size ordering** scheme fires rule with the most attribute tests in its antecedent.
- The **rule ordering** scheme prioritizes the rules beforehand. The ordering may be *class-based* or *rule-based*.
- With **class-based ordering**, the classes are sorted in order of decreasing importance. That is, all the rules for the most prevalent (or most frequent) class come first, the rules for the next prevalent class come next, and so on.

Rule Based Classification

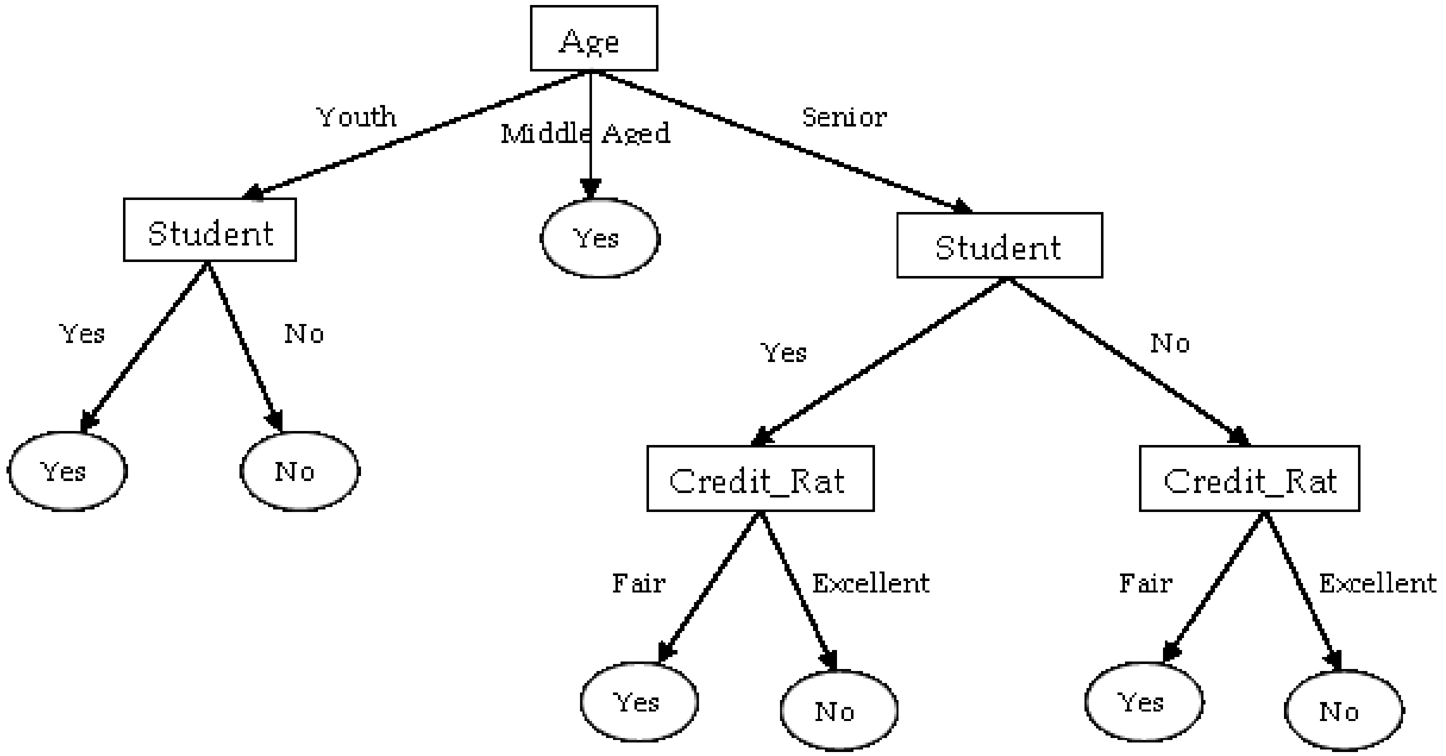
- With **rule-based ordering**, the rules are organized into one long priority list, according to some measure of rule quality, such as accuracy, coverage, or size, or based on advice from domain experts.
- With rule ordering, the triggering rule that appears earliest in the list has the highest priority, and so it gets to fire its class prediction. Any other rule that satisfies ***X*** is ignored.

Rule Based Classification

Rule Extraction from Decision Tree

- To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent. The leaf node holds the class prediction, forming the rule consequent.
- Example: Consider the following decision tree and extract the rules.

Rule Based Classification



R1: If Age=Youth AND Student=Yes
Then Buys_Computer=Yes

R2: If Age=Youth AND Student=NO
Then Buys_Computer=NO

R3: If Age=Middle_Aged
Then Buys_Computer=Yes

R4: If Age=Senior AND Student=Yes
AND Credit_Rating=Fair
Then Buys_Computer=Yes
…….

Rule Based Classification

Rule Extraction from Decision Tree

- Rules extracted from decision trees are **mutually exclusive** and **exhaustive**.
- Rules are said to be mutually exclusive if no two rules are triggered by the same tuple.
- Rules said to be exhaustive if there is one rule for each possible attribute-value combination.
- Rules extracted from decision tree can be simplified. Rules can be simplified by pruning the conditions in the rule antecedent that do not improve the estimated accuracy of the rule.

Rule Based Classification

Rule Simplification: Example

Initial Rule:

- R: If Age=Senior AND Student=Yes AND Credit_Rating=Fair

  Then Buys_Computer=Yes

Simplified Rule:

- R: If Age=Senior AND Credit_Rating=Fair

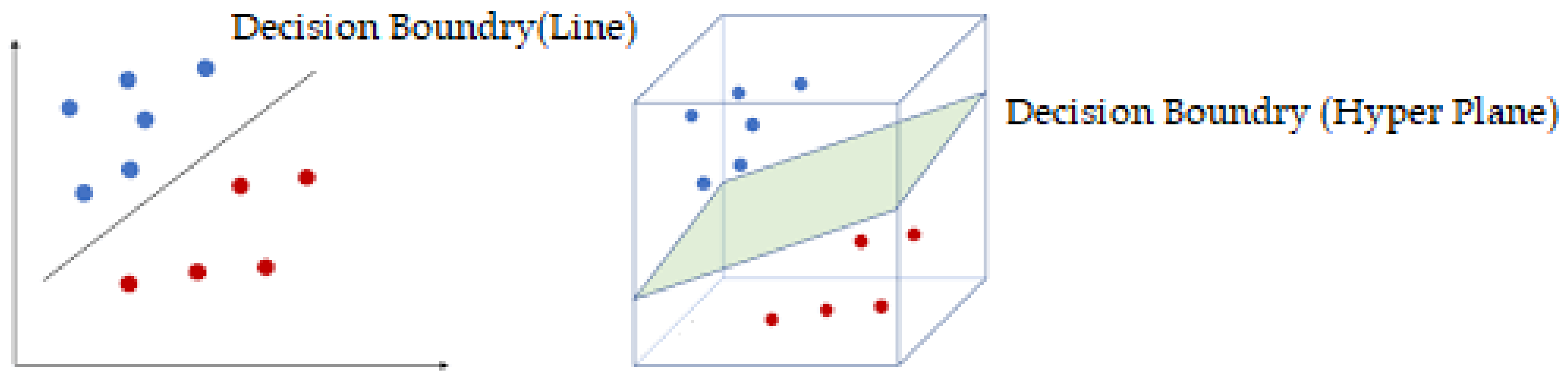
  Then Buys_Computer=Yes

- Effect of rule simplification is that rules are no longer mutually exclusive and exhaustive.

Support Vector Machine

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- A support vector machine takes input data points and outputs the hyperplane (which in two dimensions it’s simply a line) that best separates the data points into two classes.
- This line or hyperplane is the **decision boundary**: any data points that falls to one side of it is classified in one class and, and the data points that falls to the other of it is classified in another class.

Support Vector Machine



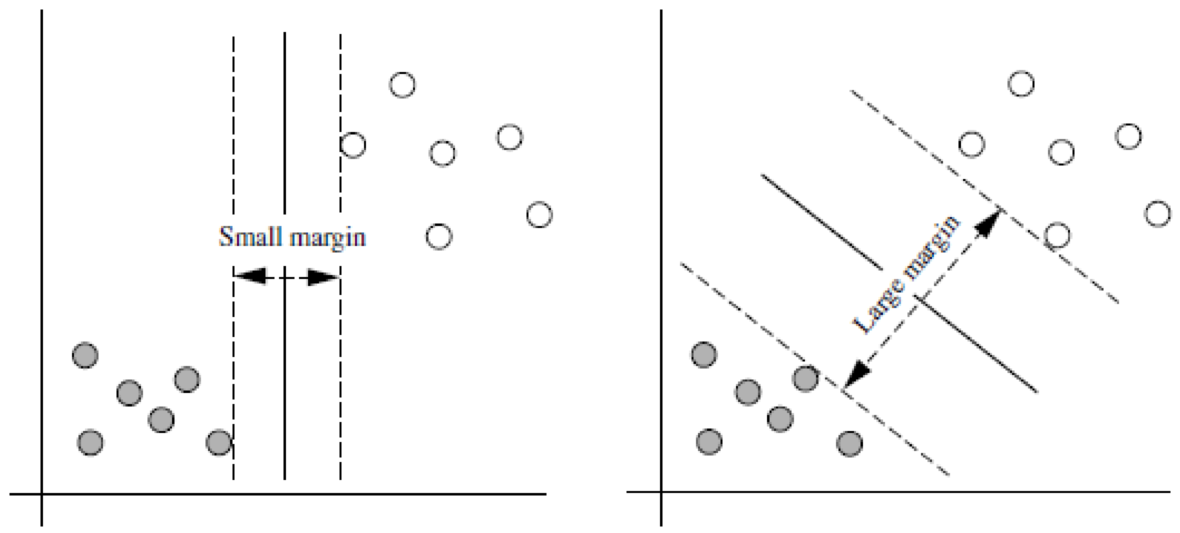
- Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousands).

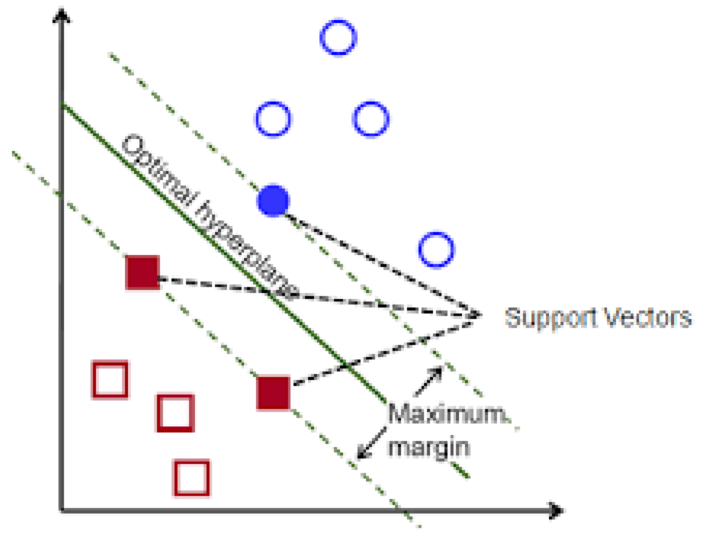
Support Vector Machine

- Support vectors are data points that are closest to the hyperplane and influence the position and orientation of the hyperplane.
- To separate the two classes of data points, there are many possible hyperplanes that could be chosen.
- SVM algorithm selects optimal hyperplane by choosing hyperplane with largest margin. Such hyperplane is called maximum marginal hyperplane (MMH).
- Let H1 and H2 are planes that passes through support vectors and parallel to the hyperplane of decision boundary.

Support Vector Machine

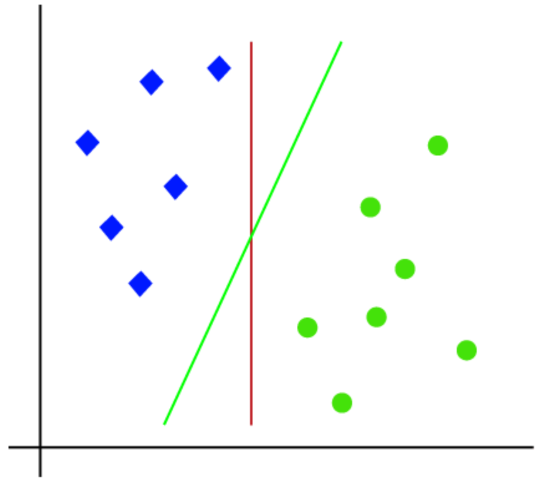
- Distance between plane H1 and the hyperplane should be equal to distance between plane H2 and the hyperplane. Margin is the distance between planes H1 and H2.



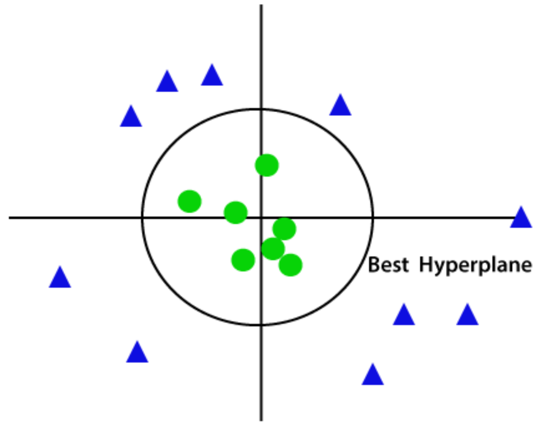


Support Vector Machine

- Support Vector Machine(SVM) can be of two types: Linear SVM and Non-linear SVM.
- SVM that is used to classify linearly separable data points is called linear SVM whereas the SVM that is used to classify non-linearly separable data points is called non-linear SVM.



Linearly Separable Data Points



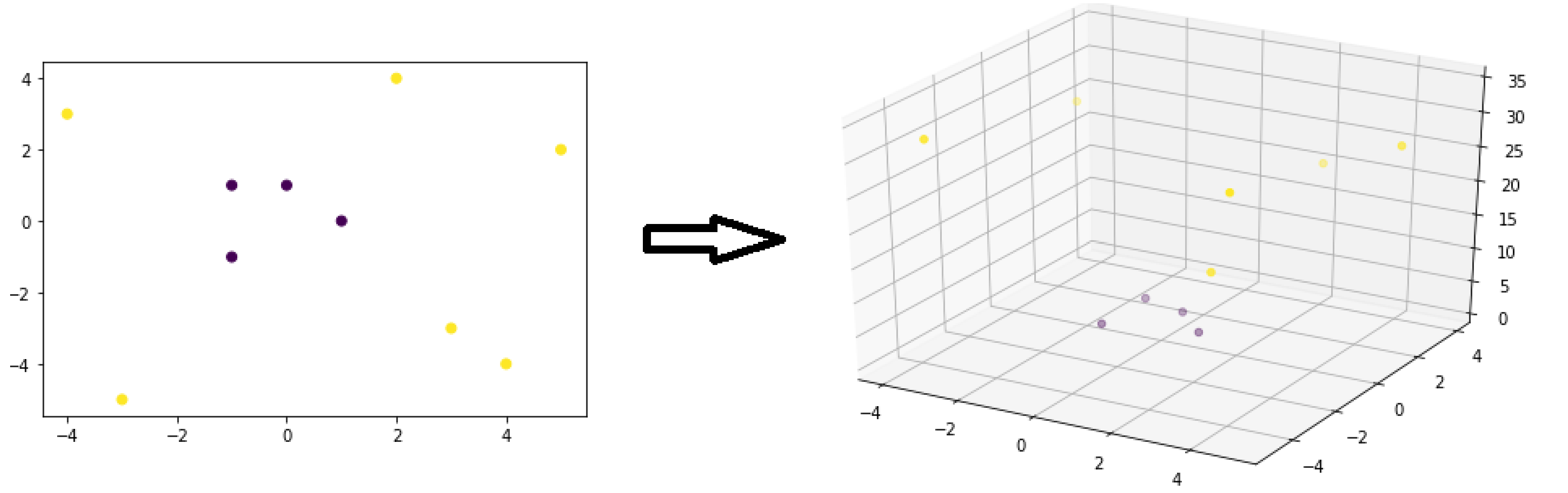
Non-linearly Separable Data Points

Support Vector Machine

- Non-linear SVM works in following two steps:
  - It transforms low–dimensional data points into high-dimensional data points, that are linearly separable, by using kernel-trick.
  - Then, it classifies data points using linear-hyperplane.
- SVM algorithms use a set of mathematical functions that are defined as the kernel.
- These functions are used to transform non-linearly separable low-dimensional data points into linearly separable high-dimensional data points.
- Popular kernels are: Linear Kernel, Polynomial Kernel, Gaussian Kernel, Radial Basis Function (RBF), Sigmoid Kernel, etc.

Support Vector Machine

- Consider the following 2-D data points which are linearly inseparable. We can transform the data points into linearly separable data points by adding third dimension $z=x^2+y^2$.



Support Vector Machine

Example

- Consider following data points:
  - Positively Labelled Data Points:(3,1),(3,-1),(6,1),(6,-1)
  - Negatively Labelled Data Points:(1,0),(0,1),(0,-1),(-1,0)
- Determine the equation of hyperplane that divides the above data points into two classes:

Support Vector Machine

Solution

Support vectors are

s1=(1,0),   s2=(3,1), s3=(3,-1)   why???

Augment support vectors with b=1

s1=(1,0,1),   s2=(3,1,1), s3=(3,-1,1)

Since there are three support vectors, we need to calculate three variables

Support Vector Machine

Solution

Since there are three support vectors, we need to calculate three variables

Thus, three linear equations can be written as:

$$\alpha s_1.s_1 + \beta s_2.s_1 + \lambda s_3.s_1 = -1$$

$$\alpha s_1.s_2 + \beta s_2.s_2 + \lambda s_3.s_2 = 1$$

$$\alpha s_1.s_3 + \beta s_2.s_3 + \lambda s_3.s_3 = 1$$

After simplifying above equations, we get

Support Vector Machine

Solution

After simplifying above equations, we get

$$2\alpha + 4\beta + 4\lambda = -1$$

$$4\alpha + 11\beta + 9\lambda = 1$$

$$4\alpha + 9\beta + 11\lambda = 1$$

Solving these equations, we get

$$\alpha = -3.5 \quad \beta = 0.75 \quad \lambda = 0.75$$

Now, we can compute weight vector of hyperplane as below

$$w = \alpha s_1 + \beta s_2 + \lambda s_3 = (1,0,-2)$$

Support Vector Machine

Solution

Thus, $w = (1,0)$ and b = -2

=> line is vertical line parallel to y-axis

Hence, equation of hyper plane that divides data points is x-2=0

Performance Measures of Classification

- Before selecting a classification algorithm to solve a problem, we need to evaluating its performance. Confusion matrix is widely used for computing performance measures of classification algorithm.
- A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes.
- The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

Performance Measures of Classification

- For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:

		Actual Classes	
		Positive	Negative
Predicted Classes	Positive	TP	FP
	Negative	FN	TN

- **True Positive(TP)**: It represents correctly classified positive classes. Both actual and predicted class are positive here.
- **False Positive (FP)**: It represents incorrectly classified positive classes. These are the positive classes predicted by the model that were actually negative. This is called Type I error.

Performance Measures of Classification

- **True Negative(TN)**: It represents correctly classified Negative classes. Both actual and predicted class are negative here.
- **False Negative (FN)**: It represents incorrectly classified negative classes. These are the negative classes predicted by the model that were actually positive. This is called Type II error.
- Four widely used performance measures used for evaluating classification models are: *Accuracy, Recall, Precision, F1-score*.

Performance Measures of Classification

- **Accuracy:** It is the percentage of correct predictions made by the model and is given as below:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{TP} + \text{TN}}{\text{Total Number of Instances}}$$

- **Precision:** It is percentage of predicted positives that are actually positive and is given by:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{Total Number of Predicted Positives}}$$

Performance Measures of Classification

- **Recall:** It is the percentage of actual positives that are correctly classified by the model and is given as below:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{Total Number of actual positives}}$$

- **F1-score:** It is the harmonic mean of recall and precision. It becomes high only when both precision and recall are high. This score is given by:

$$F1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

Performance Measures of Classification

Example

- Suppose that we have to classify 100 people as pregnant or not pregnant. This includes 40 pregnant women and the remaining 60 are not pregnant. Out of 40 pregnant women 30 pregnant women are classified correctly and the remaining 10 pregnant women are classified as not pregnant by the machine learning algorithm. On the other hand, out of 60 people in the not pregnant category, 55 are classified as not pregnant and the remaining 5 are classified as pregnant.
- Compute accuracy, precision, recall, and F1-score for the above example.

Performance Measures of Classification

Solution

Cross-Validation

- Cross-validation is a re-sampling procedure used to evaluate machine learning models on a limited data sample.
- k-Fold Cross-Validation (Muti-Fold Cross-Validation) is widely used for cross validation. If we choose k=10, it becomes 10-fold cross validation.
- In this approach, the original sample is randomly partitioned into k equal sized subsamples. Out of the k subsamples, a single subsample is used as the testing set, and the remaining $k - 1$ subsamples are used as training set.

Cross-Validation

- The cross-validation process is then repeated *k* times, with each of the *k* subsamples used exactly once as the testing data. The *k* results can then be averaged to produce a single estimation.
- The advantage of this method is that all observations are used for both training and testing, and each observation is used for testing exactly once. 10-fold cross-validation is commonly used, but in general *k* remains an unfixed parameter.
- For example, setting $k = 2$ results in 2-fold cross-validation. In 2-fold cross-validation, we randomly shuffle the dataset into two sets d_0 and d_1, so that both sets are equal size. We then train on d_0 and test on d_1, followed by training on d_1 and testing on d_0.

Cross-Validation

- To make the cross-validation procedure concrete, let's look at a worked example. Imagine we have a data sample with 6 observations: {(1,3),(2,5),(3,7),(4,9),(5,11),(6,13)}
- Let k=3. That means we will shuffle the data and then split the data into 3 groups.

  Fold 1= {(1,3),(5,11)}

  Fold 2= {(6,13),(3,7)}

  Fold 3= {(2,5),(4,9)}

Cross-Validation

- Three models are trained and evaluated with each fold given a chance to be the held out test set. For example:
  - **Model 1**: Trained on Fold 1 + Fold 2, Tested on Fold 3
  - **Model 2**: Trained on Fold 2 + Fold 3, Tested on Fold 1
  - **Model3**: Trained on Fold 1 + Fold 3, Tested on Fold 2

Overfitting and Underfitting

- Overfitting is the result of using an excessively complicated model.
- It happens when the model learns the details and noise of the training data to the extent that it negatively impacts the performance of the model on new data.
- This means that the model learns noise or random fluctuations in the training data as concepts.
- This negatively impacts the model's ability to generalize because these concepts do not apply to new data.

Overfitting and Underfitting

- On the other hand, underfitting is the result of using an excessively simple model or using very few training samples.
- in such situations, a machine learning algorithm cannot capture the underlying trend of the data.
- Thus, it refers to a model that can neither model the training data nor generalize to new data.
- Obviously an underfitted machine learning model is not a suitable model for making predictions because it has poor performance on the training data too.

Overfitting and Underfitting

- On the other hand, underfitting is the result of using an excessively simple model or using very few training samples.
- in such situations, a machine learning algorithm cannot capture the underlying trend of the data.
- Thus, it refers to a model that can neither model the training data nor generalize to new data.
- Obviously an underfitted machine learning model is not a suitable model for making predictions because it has poor performance on the training data too.

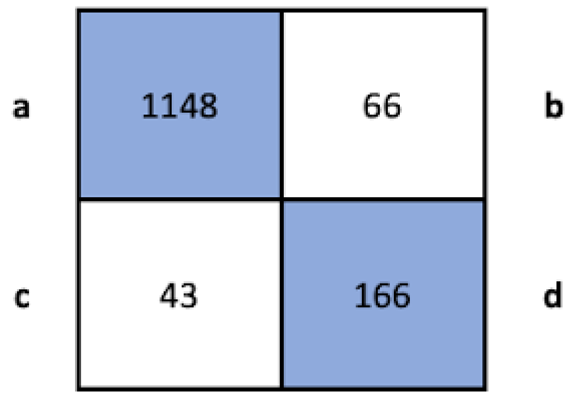
Overfitting and Underfitting

- On the other hand, underfitting is the result of using an excessively simple model or using very few training samples.
- in such situations, a machine learning algorithm cannot capture the underlying trend of the data.
- Thus, it refers to a model that can neither model the training data nor generalize to new data.
- Obviously an underfitted machine learning model is not a suitable model for making predictions because it has poor performance on the training data too.

McNemar’s Test

- In terms of comparing two binary classification algorithms, McNemar’s test is the test commenting on whether the two models disagree in the same way (or not).
- It is not commenting on whether one model is more or less accurate or error prone than another.
- In McNemar's test, we begin by creating a contingency table like the one below to compare the performance of model 1 and model 2.

McNemar’s Test



a	1148	66	b
c	43	166	d

- In this table, the regions are:
- a - both models are correct
- b - model 1 is correct, and model 2 is wrong
- c - model 2 is correct, and model 1 is wrong
- d - both models are wrong

McNemar’s Test

- We can then compute the chi-square statistic as

$$\chi^2 = \frac{(b-c)^2}{(b+c)}$$

- Additively, we need to obtain a p-value from the Chi-Square test. p-value is the area under the Chi-Square density.
- If the p-value is greater than 0.05, it can be concluded that there isn’t a significant difference between false negatives and false positives of two classification algorithms.
- And, if the p-value is less than or equal to 0.05, it means there is a significant difference.

Classification Issues

- The major issue is preparing the data for Classification and Prediction. Preparing the data involves the following activities:
- **Data Cleaning −** Data cleaning involves removing the *noise and treatment of missing values*.
- **Relevance Analysis −** Database may also have the irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.
- **Data Normalization:** Data is transformed using normalization. Normalization involves scaling all values for given attribute in order to make them fall within a small specified range.