Analytics and Systems of Big Data
Reading Assignment 1

M Vinitha (COE16B042)

May, 2020

1 Classifier Algorithms

Classification is a form of data analysis that extracts models describing important data classes. Such models, called classifiers, predict categorical (discrete, unordered) class labels. Classification has numerous applications, including fraud detection, target marketing, performance prediction, manufacturing, and medical diagnosis.

1.1 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. Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

1.1.1 Measurement Parameters

The following measures are used for attribute selection in a decision tree:

1. **Entropy**

   Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Mathematically, Entropy for 1 attribute is represented as:

   $$E(S) = -\sum_{i=1}^{c} p_i \log_2(p_i) \tag{1}$$

   where S $\rightarrow$ Current state, and Pi $\rightarrow$ Probability of an event i of state S or Percentage of class i in a node of state S.

2. **Information Gain**

   Information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values. Mathematically, Information gain is represented as:

   $$InformationGain(T, X) = Entropy(T) - Entropy(T, X) \tag{2}$$

   where T $\rightarrow$ Current state and X $\rightarrow$ Selected attribute.

3. **Gain Ratio**

   Gain ratio is a modification of Information gain that reduces its bias and is usually the best option. It corrects information gain by taking the intrinsic information of a split into account. Mathematically, Gain Ratio is represented as:

$$GainRatio = \frac{InformationGain}{SplitInfo} = \frac{Entropy(before) - \sum_{j=1}^{K} Entropy(j, after)}{\sum_{j=1}^{K} w_j \log_2 w_j} \tag{3}$$

1.1.2 Pseudo Code

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 D_j be the set of data tuples in D satisfying outcome j; // a partition
12. **if** D_j 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**(D_j, attribute list) to node N; **endfor**
15. return N;

1.1.3 Trace of Decision Tree

Let us consider the following dataset for trace of decision tree. Based on the weather we should make a decision on playing football. Decision tree will be constructed using information gain. Let class C_1 correspond to "Yes" and class C_2 correspond to "No".

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

1. Calculating entropy for the decision:
   In the above example, in total there are 9 Yes's and 5 No's. We calculate E(S) using equation (1).

$$\begin{aligned} E(S) &= -p(Yes)\log_2 p(Yes) - p(No)\log_2 p(No) \\ &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

2. Information Gain for each attribute:

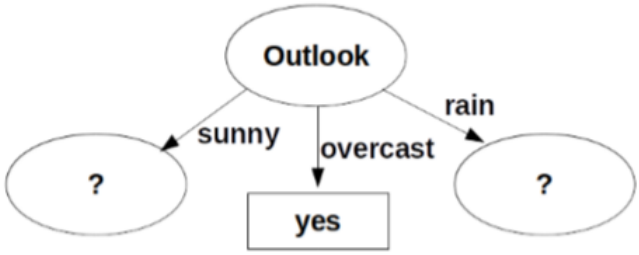
$$\begin{aligned} InformationGain(PlayedFootball, Outlook) &= E(PlayedFootball) - E(PlayedFootball, Outlook) \\ &= 0.247 \\ InformationGain(PlayedFootball, Temperature) &= E(PlayedFootball) - E(PlayedFootball, Temperature) \\ &= 0.029 \\ InformationGain(PlayedFootball, Humidity) &= E(PlayedFootball) - E(PlayedFootball, Humidity) \\ &= 0.152 \\ InformationGain(PlayedFootball, Wind) &= E(PlayedFootball) - E(PlayedFootball, Wind) \\ &= 0.048 \end{aligned}$$

3. From the above values, the attribute with the highest information gain value is chosen to branch. 'Outlook' is placed as the root node of the tree.

4. There are three possible outcomes for 'Outlook' attribute:

   - Rain
   - Sunny
   - Overcast

When we take Overcast branch for 'Outlook' attribute we directly arrive at the class label C_1. That is, whenever the outlook is Overcast, Played football is always 'Yes'.
At this point, the decision tree looks like:



5. Information gain when we take Sunny branch for 'Outlook' attribute:

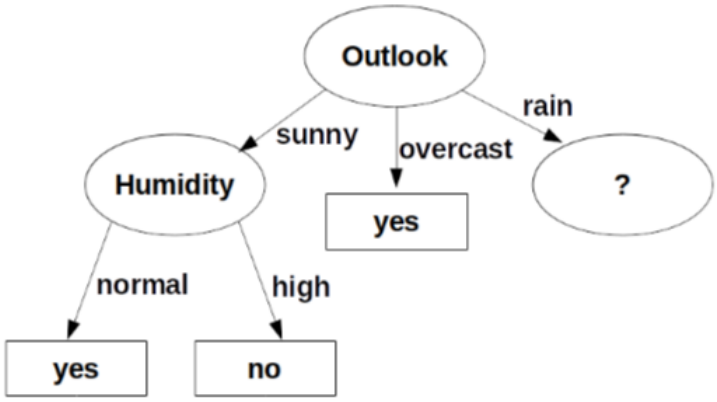
   InformationGain(Outlook=Sunny,Temperature) = 0.571

   InformationGain(Outlook=Sunny,Humidity) = 0.971

   InformationGain(Outlook=Sunny,Wind) = 0.020

   - At the next level of the tree, for the 'Sunny' branch, the splitting attribute is 'Humidity' as it gives the highest information gain.
   - It is observed that the decision is always 'No' if Humidity is High.

   At this point, the decision tree looks like:



6. Information gain when we take Rain branch for 'Outlook' attribute:

   InformationGain(Outlook=Rain,Temperature) = 0.20

   InformationGain(Outlook=Rain,Humidity) = 0.20

   InformationGain(Outlook=Rain,Wind) = 0.97

   - At the next level of the tree, for the 'Rain' branch, the splitting attribute is 'Wind' as it gives the highest information gain.
   - It is observed that the decision is always 'Yes' if the Wind is Strong.

7. A decision (leaf node) has been reached for every possible outcome (branch) starting from the root $\Rightarrow$ Construction of decision tree is complete.

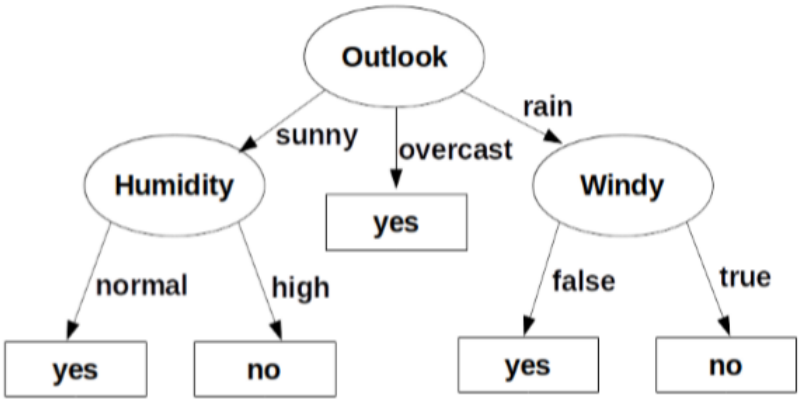


Figure 1: Final Decision Tree

1.1.4 Advantages of Decision Tree

- Easy to use and understand.
- Can handle both categorical and numerical data.
- Resistant to outliers, hence require little data preprocessing.
- New features can be easily added.
- Can be used to build larger classifiers by using ensemble methods.

1.1.5 Disadvantages of Decision Tree

- Data may be overfitted or overclassified.
- For making a decision, only one attribute is tested at an instant thus consuming a lot of time.
- A small change in the data can cause a large change in the structure of the decision tree causing instability.

1.2 Naive Bayesian Classifier (NBC)

- The Naive Bayes Algorithm is a machine learning algorithm for classification problems.
- Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem.
- It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.
- Naive Bayes model is easy to build and particularly useful for very large data sets.

1.2.1 Bayes Theorem

Bayes theorem is a mathematical equation used in probability and statistics to calculate conditional probability. In other words, it is used to calculate the probability of an event based on it's association with another event.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{4}$$

$$P(c|X) = P(x_1|c) * P(x_2|c) * ... * P(x_n|c) * P(c) \tag{5}$$

where,

- $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes)
- P(c) is prior probability of class
- $P(x|c)$ is the likelihood which is the probability of predictor given class
- P(x) is the prior probability of predictor.

Equation (4) is for a single variable x while equation (5) is for multivariable X = $(x_1, x_2, ..., x_n)$.

1.2.2 Conditional Class Independence

Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.

1.2.3 Prior Probability

Prior probability, often simply called the prior, is the probability of an event before new data is collected.It is used for assessment of the probability of an outcome based on the current knowledge before an experiment is performed.

1.2.4 Posterior Probability

Posterior Probability is the conditional probability of a given event, computed after observing a second event whose conditional and unconditional probabilities were known in advance. It is computed by revising the prior probability, that is, the probability assigned to the first event before observing the second event.

1.2.5 Laplacian estimator for zero probability scenario

If categorical variable has a category (in test data set), which was not observed in training data set, then the model will assign a zero probability and will be unable to make a prediction. This is often known as Zero Frequency. To solve this, we can use smoothing techniques. One of the simplest smoothing techniques is called Laplace estimation.

We can assume that our training database, D, is so large that adding one to each count that we need would only make a negligible difference in the estimated probability value, yet would conveniently avoid the case of probability values of zero. This technique for probability estimation is known as the Laplacian correction or Laplace estimator. If we have, say, q counts to which we each add one, then we must remember to add q to the corresponding denominator used in the probability calculation.

1.2.6 Pseudo Code

1. Calculate the prior probability for given class labels.
2. Calculate conditional probability with each attribute for each class.
3. Multiply same class conditional probability to get likelihood.
4. Multiply prior probability with output of step 3 to get posterior probability.
5. Class with higher probability belongs to given input set.

1.2.7 Trace

Let us consider the following dataset for trace of Naive Bayes' Classifier. Based on the weather we should make a decision on playing football. Let class C_1 correspond to "Yes" and class C_2 correspond to "No".

Outlook	Temperature	Humidity	Wind	Played football(yes/no)
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Learning Phase

P(Outlook=o|Class Play=Yes|No)	Frequency		Probability in Class	
Outlook =	Play=Yes	Play=No	Play=Yes	Play=No
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rain	3	2	3/9	2/5
	total= 9	total=5		

P(Temperature=t|Class Play=Yes|No)	Frequency		Probability in Class	
Temperature =	Play=Yes	Play=No	Play=Yes	Play=No
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
	total= 9	total=5		

P(Humidity=h|Class Play=Yes|No)	Frequency		Probability in Class	
Humidity =	Play=Yes	Play=No	Play=Yes	Play=No
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
	total= 9	total=5		

P(Wind=w|Class Play=Yes|No)	Frequency		Probability in Class	
Wind =	Play=Yes	Play=No	Play=Yes	Play=No
strong	3	3	3/9	3/5
weak	6	2	6/9	2/5
	total= 9	total=5		

Figure 2: Likelihood Table

Test Phase

Input: Test entry where the Weather is Sunny, Temperature is Cool, the Humidity is High and the Wind is Strong. Using Naive Bayes classifier we'll decide whether to play football or not.

- Look up from the tables
  - $P(Outlook = Sunny|Play = Yes) = 2/9$
  - $P(Outlook = Sunny|Play = No) = 3/5$
  - $P(Temperature = Cool|Play = Yes) = 3/9$
  - $P(Temperature = Cool|Play = No) = 1/5$
  - $P(Humidity = High|Play = Yes) = 3/9$
  - $P(Humidity = High|Play = No) = 4/5$
  - $P(Wind = Strong|Play = Yes) = 3/9$
  - $P(Wind = Strong|Play = No) = 3/5$
  - $P(Play = Yes) = 9/14$
  - $P(Play = No) = 5/14$
- $P(Yes|X) \approx P(X|Yes)P(Yes) \approx [P(Sunny|Yes)P(Cool|Yes)P(High|Yes)P(Strong|Yes)]P(Yes) = 0.0053$
- $P(No|X) \approx P(X|Yes)P(Yes) \approx [P(Sunny|No)P(Cool|No)P(High|No)P(Strong|No)]P(No) = 0.0171$
- Given the fact $P(Yes|X) < P(No|X)$, we label X to be C_2. Hence, the given entry is classified as No.

1.2.8 Advantages

- It is easy and fast to predict class of test data set. It also perform well in multi class prediction.
- When assumption of independence holds, a Naive Bayes classifier performs better compare to other models like logistic regression and you need less training data.
- It requires less training data.

1.2.9 Disadvantages

- Limitation of Naive Bayes is the assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.

1.3 Neural Network Classifier (Back Propagation Network(BPN))

Artificial neural networks are crude electronic networks of neurons based on the neural structure of the brain. They process records one at a time and learn by comparing their classification of the record (i.e., largely arbitrary) with the known actual classification of the record. The errors from the initial classification of the first record is fed back into the network, and used to modify the networks algorithm for further iterations. Neurons are organized into layers: input, hidden and output.

1.3.1 Terminologies

- **Topology of a Network:** The way in which the neurons (basic units of a neural network) are connected together determines the topology of the neural network. In supervised learning, the most common topology is the fully connected, three-layer, feed-forward network. In unsupervised learning it is a direct mapping of inputs to a collection of units that represents categories.
- **Input Layer:** The input layer is the first layer of the workflow for the artificial neural network. The input layer of a neural network is composed of artificial input neurons and brings the initial data into the system for further processing by subsequent layers of artificial neurons.
- **Hidden Layer:** The layer/layers that lie between a network's input and output layers. It is called hidden, because its neuron values are not visible outside the network. The neurons in this layer apply weights to the inputs and direct them through an activation function as the output for the subsequent layer. The hidden layers extend a neural network's abilities to learn logical operations.
- **Output Layer:** The last layer of a neural network, that produces the output value (class label) of the network.
- **Activation Function:** A mathematical function that a neuron uses to produce an output. Usually this input value has to exceed a specified threshold value that determines if an output to other neurons should be generated. Functions such as Sigmoid are often used. Sometimes called the transfer function.
- **Bias:** Bias is used to delay the triggering of the activation function. A bias value allows a shift in the activation function to the left or right and it helps getting a better fit for the data.
- **Weight:** In a neural network, a weight is associated with each input. Weight increases the steepness of activation function. This means weight decides how fast the activation function will trigger.

1.3.2 Strengths

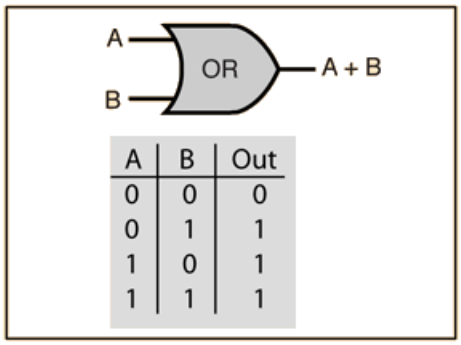
- Ability to handle very large datasets.
- Can be used in a variety of fields including speech, image recognition, image captioning, natural language processing, handwriting recognition, etc.
- Ability to encode features useful across problem domains (e.g. you can train the lower levels of an image recognizer on one dataset, and on the next dataset you don't have to start from scratch).
- Storing information on the entire network : Information such as in traditional programming is stored on the entire network, not on a database. The disappearance of a few pieces of information in one place does not prevent the network from functioning.
- Ability to work with incomplete knowledge: After ANN training, the data may produce output even with incomplete information. The loss of performance here depends on the importance of the missing information.
- Having fault tolerance: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault tolerant.
- Ability to make the machine learning: Artificial neural networks learn events and make decisions by commenting on similar events.
- Parallel processing capability: Artificial neural networks have numerical strength that can perform more than one job at the same time.

1.3.3 Limitations

- Hardware dependence: Artificial neural networks require processors with parallel processing power, in accordance with their structure.
- Unexplained behavior of the network: This is the most important problem of ANN. When ANN produces a probing solution, it does not give a clue as to why and how. This reduces trust in the network.
- Determination of proper network structure: There is no specific rule for determining the structure of artificial neural networks. Appropriate network structure is achieved through experience and trial and error.
- Difficulty of showing the problem to the network: ANNs can work with numerical information. Problems have to be translated into numerical values before being introduced to ANN. The display mechanism to be determined here will directly influence the performance of the network . This depends on the user's ability.
- The duration of the network is unknown: The network is reduced to a certain value of the error on the sample means that the training has been completed. This value does not give us optimum results.
- They require huge amounts of data. Other algorithms (e.g. decision trees, logistic regression, naive Bayes) can perform well with much less data.
- They are very computationally expensive to train. It is only the advancements in GPUs that has made training neural networks viable.

1.3.4 Neural Network for OR gate

- The logic gate OR returns 1 if any input is 1 and will only return 0 if both inputs are 0. The truth table is given below.



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

Figure 3: Truth Table for OR gate

- Consider the Perceptron Rule:

$$y = \begin{cases} 0, if \sum_{i=1}^{n} w_i x_i + b \leq 0 \\ 1, if \sum_{i=1}^{n} w_i x_i + b > 0 \end{cases} \tag{6}$$

- Substituting the input values in the perceptron rule, the inequalities which must be satisfied are:

$$(a) w_1 * 0 + w_2 * 0 + b \leq 0 \equiv b \leq 0$$
$$(b) w_1 * 1 + w_2 * 0 + b > 0 \equiv w_1 + b > 0$$
$$(c) w_1 * 0 + w_2 * 1 + b > 0 \equiv w_2 + b > 0$$
$$(d) w_1 * 1 + w_2 * 1 + b > 0 \equiv w_1 + w_2 + b > 0$$

- Initialise the values. Let b = -1, $w_1 = 1$ and $w_2 = 1$.
- Now substitute each input in the perceptron equation and check if the predicted output is equal to the actual output.

  (a) Input 1: $1 * 0 + 1 * 0 + -1 = -1 \leq 0$ is true. So output would be 0 which is equal to the required output.

  (b) Input 2: $1 * 0 + 1 * 1 + -1 = 0 > 0$ is false. Output would be 0 which is not equal to the desired output. Update the weight matrix. Since x_1 is 0 (implying $w_1 * x_1 = 0$), w_2 is increased. Now, $w_1 = 1$, $w_2 = 2$ and b = -1. On substituting: $1 * 0 + 2 * 1 + -1 = 1 > 0$ is true.

  (c) Input 3: $1 * 1 + 2 * 0 + -1 = 0 > 0$ is false. So, w_1 is incremented. Now, $w_1 = 2$, $w_2 = 2$ and b = -1. On substituting: $2 * 1 + 2 * 0 + -1 = 1 > 0$ is true.

  (d) Input 4: $2 * 1 + 2 * 1 + -1 = 3 > 0$ is true.

  (e) Since weight matrix was updated, it must be checked for the other inputs.

  Input 1: $2 * 0 + 2 * 0 + -1 = -1 \leq 0$ is true

  Input 2: $2 * 0 + 2 * 1 + -1 = 1 > 0$ is true
- So, for $w_1 = 2$, $w_2 = 2$ and b = -1, all the linear inequalities are satisfied. The final network is:

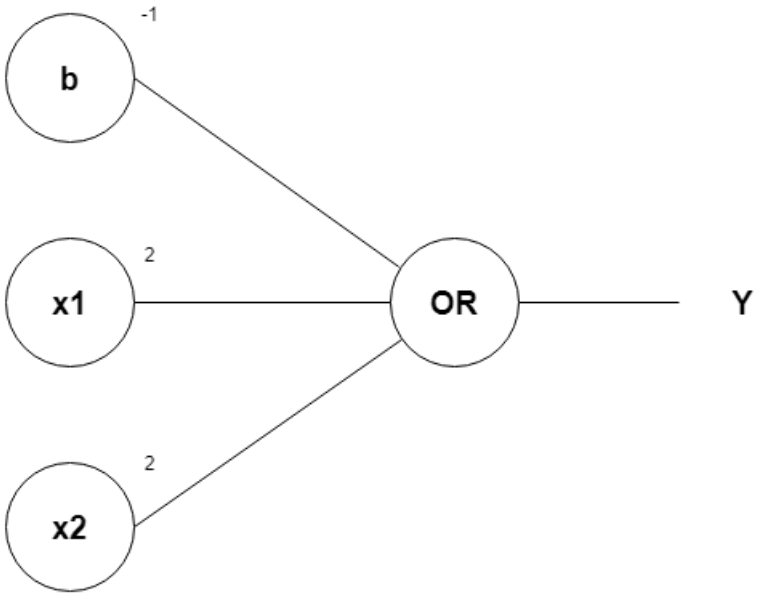
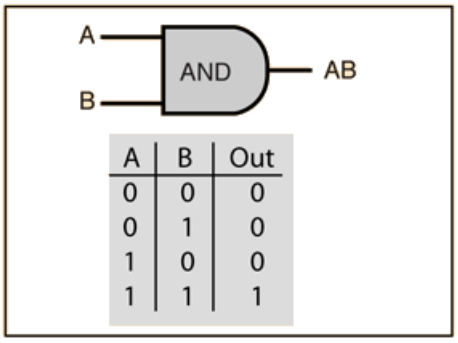


Figure 4: Network for OR gate

1.3.5 Neural Network for AND gate

- The logic gate AND returns 0 if any input is 0, and will only return 1 if both inputs are 1. The truth table is given below.



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

Figure 5: Truth Table for AND gate

- Consider the Perceptron Rule:

$$y = \begin{cases} 0, if \sum_{i=1}^{n} w_i x_i + b \leq 0 \\ 1, if \sum_{i=1}^{n} w_i x_i + b > 0 \end{cases} \tag{7}$$

- Substituting the input values in the perceptron rule, the inequalities which must be satisfied are:

$$(a) w_1 * 0 + w_2 * 0 + b \leq 0 \equiv b \leq 0$$
$$(b) w_1 * 1 + w_2 * 0 + b > 0 \equiv w_1 + b \leq 0$$
$$(c) w_1 * 0 + w_2 * 1 + b > 0 \equiv w_2 + b \leq 0$$
$$(d) w_1 * 1 + w_2 * 1 + b > 0 \equiv w_1 + w_2 + b > 0$$

- Initialise the values. Let b = -1, $w_1 = 1$ and $w_2 = 1$.

- Now substitute each input in the perceptron equation and check if the predicted output is equal to the actual output.

  (a) Input 1: $1 * 0 + 1 * 0 + -1 = -1 \leq 0$ is true. So output would be 0 which is equal to the required output.

  (b) Input 2: $1 * 0 + 1 * 1 + -1 = 0 \leq 0$ is true. So output would be 0 which is equal to the required output.

  (c) Input 3: $1 * 1 + 1 * 0 + -1 = 0 \leq 0$ is true. So output would be 0 which is equal to the required output.

  (d) Input 4: $1 * 1 + 1 * 1 + -1 = 1 > 0$ is true. So output would be 1 which is equal to the required output.

  (e) The predicted values are matching with the actual values. So weight matrix does not need to be updated. The final network is:

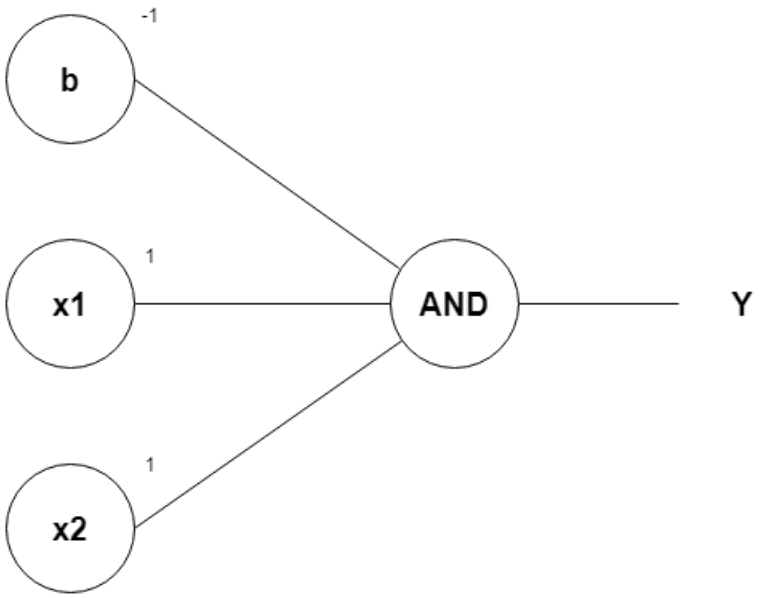


Figure 6: Network for OR gate

1.3.6 Backpropagation Algorithm Trace

- Backpropagation learns by iteratively processing a data set of training tuples, comparing the network's prediction for each tuple with the actual known target value.
- The target value may be the known class label of the training tuple (for classification problems) or a continuous value (for numeric prediction).
- For each training tuple, the weights are modified so as to minimize the mean-squared error between the network's prediction and the actual target value.
- These modifications are made in the "backwards" direction (i.e., from the output layer) through each hidden layer down to the first hidden layer (hence the name backpropagation).

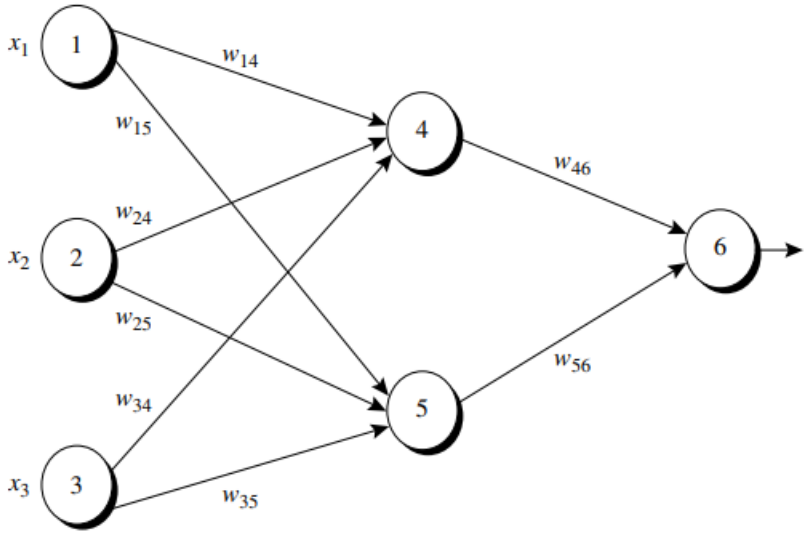


Figure 7: Example of a multilayer feed-forward neural network.

- Each unit in the hidden and output layers takes its net input and then applies an activation function to it. The function symbolizes the activation of the neuron represented by the unit. The logistic, or sigmoid, function is used. Given the net input I_j to unit j, then O_j , the output of unit j, is computed as:

$$O_j = \frac{1}{1 + e^{-I_j}} \tag{8}$$

- Given a unit, j in a hidden or output layer, the net input, I_j, to unit j is:

$$I_j = \sum_i w_{ij} O_i + \theta_j \tag{9}$$

- **Backpropagate the error:** The error is propagated backward by updating the weights and biases to reflect the error of the network's prediction. For a unit j in the output layer, the error Err_j is computed by:

$$Err_j = O_j(1 - O_j)(T_j - O_j) \tag{10}$$

where O_j is the actual output of unit j, and T_j is the known target value of the given training tuple. Note: $O_j(1 - O_j)$ is the derivative of the logistic function.

- To compute the error of a hidden layer unit j, the weighted sum of the errors of the units connected to unit j in the next layer are considered. The error of a hidden layer unit j is:

$$Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk} \tag{11}$$

where w_{jk} is the weight of the connection from unit j to a unit k in the next higher layer, and Err_k is the error of unit k.

- The weights and biases are updated to reflect the propagated errors. Weights are updated by the following equations, where $\triangle w_{ij}$ is the change in weight w_{ij}:

$$\triangle w_{ij} = (l) Err_j O_i \tag{12}$$

$$w_{ij} = w_{ij} + \triangle w_{ij} \tag{13}$$

The variable l is the learning rate, a constant typically having a value between 0.0 and 1.0.

- **Training epoch**

First training tuple, X = (1, 0, 1)

Class label = 1

Learning rate (l) = 0.9

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

Figure 8: Initial Input, Weight, and Bias Values

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	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

Figure 9: Net Input and Output Calculations

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	$(0.332)(1-0.332)(0.1311)(-0.3)=-0.0087$

Figure 10: Calculation of the Error at Each Node

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	$-0.4+(0.9)(-0.0087)=-0.408$

Figure 11: Calculations for Weight and Bias Updating

1.4 Nearest Neighbour Classifier

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well-

- **Lazy learning algorithm:** KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm:** KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

K-nearest neighbors (KNN) algorithm uses **'feature similarity'** to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set.

1.4.1 Pseudo Code

1. Load the data.
2. Initialize K to your chosen number of neighbors.

3. For each example in the data:
   (a) Calculate the distance between the query example and the current example from the data.
   (b) Add the distance and the index of the example to an ordered collection.
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances.
5. Pick the first K entries from the sorted collection.
6. Get the labels of the selected K entries.
7. For classification, return the mode of the K labels

1.4.2 Trace

- Consider a simple example with only 1 feature, Age and the output is, 'Plays Football'.

Age	22	23	21	18	19	25	27	29	31	45
Plays Football	yes	yes	yes	yes	yes	no	no	no	no	no

- Consider an input point p, which has Age 33.
- Let k = 3.
- As there is only 1 feature, we can take difference as the distance measure.

Age	22	23	21	18	19	25	27	29	31	45
Plays Football	yes	yes	yes	yes	yes	no	no	no	no	no
Distance from p	11	10	12	15	14	8	6	4	2	12

- After sorting, the distance values are:

Age	31	29	27	25	23	22	21	45	19	18
Plays Football	no	no	no	no	yes	yes	yes	no	yes	yes
Distance from p	2	4	6	8	10	11	12	12	14	15

- The first k=3 points are:

Age	31	29	27
Plays Football	no	no	no
Distance from p	2	4	6

- The class label count is:

  yes = 0

  no = 3

- The class label 'no' is returned as the output as it has the maximum count. So, the output is that the person aged 33 years does not play football.

1.4.3 Advantages

- The algorithm is simple and easy to implement.
- There's no need to build a model, tune several parameters, or make additional assumptions.
- The algorithm is versatile. It can be used for classification as well as regression.

1.4.4 Disadvantages

- It is computationally a bit expensive algorithm because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

2 Genetic Algorithm

- A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution.
- This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.
- Genetic Algorithms (GAs) are a subset of a much larger branch of computation known as Evolutionary Computation.
- In GAs, we have a pool or a population of possible solutions to the given problem.
- These solutions then undergo recombination and mutation (like in natural genetics), producing new children, and the process is repeated over various generations.
- Each individual (or candidate solution) is assigned a fitness value (based on its objective function value) and the fitter individuals are given a higher chance to mate and yield more "fitter" individuals.
- Five phases are considered in a genetic algorithm.
    1. Initial population
    2. Fitness function
    3. Selection
    4. Crossover
    5. Mutation

2.1 Population

- Each individual (solution in search space) in the population is coded as a finite length vector analogous to chromosome of components.
- These variable components are analogous to Genes.
- Thus a chromosome (individual) is composed of several genes (variable components).

2.2 Fitness Function

- The fitness function determines how fit an individual is.
- A fitness score is given to each individual.
- The probability that an individual will be selected for reproduction is based on its fitness score.

2.3 Selection

- The idea of selection phase is to select the fittest individuals and let them pass their genes to the next generation.
- Two pairs of individuals (parents) are selected based on their fitness scores.
- Individuals with high fitness have more chance to be selected for reproduction.

2.4 Crossover

- This represents mating process between individuals.
- By mutating the old generation parents, the new generation offspring comes by carrying genes from both parents.
- For every two parents, crossover takes place by selecting a random point in the chromosome.
- Then the genes at these crossover sites are exchanged thus creating a completely new individual.
- The resulting chromosomes are offspring. Thus, this operator is called single-point crossover.
- Other types are two-point crossover, uniform crossover and crossovers for ordered lists.
- In the absence of the crossover operator, the offspring will be identical to the parent.

2.5 Mutation

- Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next.
- Mutation alters one or more gene values in a chromosome from its initial state.
- Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search.
- A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be flipped.
- This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are uniform, non-uniform, Gaussian, inversion and floating point mutation.

2.6 Pseudo Code

1. START
2. Generate the initial population
3. Compute fitness
4. REPEAT
   Selection
   Crossover
   Mutation
   Compute fitness
5. UNTIL population has converged
6. STOP

2.7 Optimization problem

The problem chosen for optimization is:

$max(f(x) = x^3 - 2x^2 + x)$, where $x \in [0, 31]$

1. Population size is chosen to be 6 and the single point crossover and bit string mutation operators are used.

2. Population size is chosen to be 6 and the single point crossover and bit string mutation operators are used. The Range of number for use in Roulette selection is calculated by using the percentage contribution of each individual chromosome to the fitness of the generation.

S no	Chromosome	Decoded Integer	Fitness Score	Fitness (%)	Range
1	00101	5	80	0	0
2	11110	30	25230	56	[1,56]
3	01111	15	2940	7	[57,63]
4	01011	11	1100	2	[64,65]
5	10111	23	11132	25	[66,90]
6	10001	17	4352	10	[91,100]

3. To select the parents, roulette wheel method was used. To simulate this, a random number from [0,100] was generated, which is then mapped to a chromosome (C_1,...,C_6) based on the chromosome's range. A crossover point was selected at random from the range [1,4] and offspring were generated.

S no	Parent 1	Parent 2	Crossover point	Offspring1	Offspring 2
1	19=C_2=11110	73=C_5=10111	4	11111	10110
2	43=C_2=11110	95=C_6=10001	2	11001	10110
3	18=C_2=11110	25=C_2=11110	1	11110	11110

4. Some of the offspring undergo mutation to form the next population.

S no	Chromosome	Mutation point	Output	Decoded Integer
1	11111	-	11111	31
2	10110	-	10110	22
3	11001	5	11000	24
4	10110	-	10110	22
5	11110	2	10110	22
6	11110	-	11110	30

5. Repeat the procedure.

S no	Chromosome	Decoded Integer	Fitness Score	Fitness(%)	Range
1	11111	31	27900	29	[0,28]
2	10110	22	9702	10	[29,38]
3	11000	24	12696	13	[39,51]
4	10110	22	9702	10	[52,61]
5	10110	22	9702	10	[62,71]
6	11110	30	25230	27	[72,99]

6. The above steps are repeated until the termination criteria is achieved. Here since one of the chromosomes has the highest achieveable value for the function, we terminate the algorithm. The maximum value of the function is found at x = 31.

3 Bucket Brigade Classifier

- The first major learning task facing any rule-based system operating in a complex environment is the credit assignment task, Somehow the performance system must determine both the rules responsible for its successes and the representativeness of the conditions encountered in attaining the successes.
- The bucket brigade algorithm is designed to solve the credit assignment problem for classifier systems.
- To implement the algorithm, each classifier is assigned a quantity called its strength.
- The bucket brigade algorithm adjusts the strength to reflect the classifier's overall usefulness to the system. The strength is then used as the basis of a competition.
- Each matched classifier makes a bid B proportional to its strength.
- In this way, rules that are highly fit are given preference over the other rules.
- This auction allows the appropriate classifiers to post their messages i.e, each time step, each satisfied classifier makes a bid based on its strength, and only the highest bidding classifiers get their messages on the message list for the next time step.
- A matched and activated classifier sends its bid B to those classifiers responsible for sending the messages that matched the bidding classifier's condition.
- The bid payment is divided in some manner among matching classifiers.
- There are no consistency requirements on posted messages; the message list can hold any set of messages, and any such set can direct further competition. The only point at which consistency enters is at the output interface.
- Here, different sets of messages may specify conflicting responses. Such conflicts are again resolved by competition. For example, the strengths of the classifiers advocating each response can be assumed so that one of the conflicting actions is chosen with a probability proportional to the sum of its advocates.

4 Confusion Matrix and Performance Measures

4.1 Confusion Matrix

- A confusion matrix gives the summary of the predictions or the results of a classifier.
- It depicts the errors as well as the types of errors made by a classifier.
- A confusion matrix consists of a table which give the true positive and negative values along with the predicted positive and negative values.

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

Figure 12: Confusion Matrix

where,

  - **True Positive(TP):** Actual class is positive and predicted class is positive.
  - **True Negative(TN):** Actual class is negative and predicted class is negative.
  - **False Positive(FP):** Actual class is negative and predicted class is positive.
  - **False Negative(FN):** Actual class is positive and predicted class is negative.

4.2 Performance Measures

- **Accuracy:** Accuracy refers to closeness of the measurements to a specific value.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

- **Sensitivity (Recall):** Sensitivity of a classifier is the ratio between how much were correctly identified as positive to how much were actually positive.

$$Sensitivity = Recall = \frac{TP}{TP + FN} \tag{15}$$

- **Specificity:** Specificity of a classifier is the ratio between how much were correctly classified as negative to how much was actually negative.

$$Specificity = \frac{TN}{TN + FP} \tag{16}$$

- **Precision:** Accuracy of positive predictions i.e the ratio that positive predictions are correct.

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

- **F1 Score:** The harmonic mean of precision and recall gives a score call F1 score which is a measure of performance of the models classification ability.

$$F1Score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{18}$$

4.3 Example

- Consider some sample results of a binary classifier.

Expected	yes	yes	no	yes	no	no	no	yes	yes	no
Predicted	no	yes	no	yes	yes	no	no	yes	no	no

- Confusion Matrix:

Actual|Predicted	yes	no
yes	3	2
no	1	4

- From the confusion matrix we see that, TP = 3, TN = 4, FP = 1 and FN = 2. Using these values we can get the performance measures.

  1. Accuracy = $\frac{7}{10} = 0.7$
  2. Sensitivity = Recall = $\frac{3}{5} = 0.6$
  3. Specificity = $\frac{4}{5} = 0.8$
  4. Precision = $\frac{3}{4} = 0.75$
  5. F1 Score = $\frac{2*0.6*0.75}{0.6+0.75} = 0.667$

5 Clustering

- Clustering can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different.
- Similarity between the data points is calculated based on similarity measures such as Euclidean-based distance or Correlation based distance, depending on the application.

5.1 K-Means Clustering

- K means algorithm is an iterative algorithm that tries to partition the dataset into K pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one cluster.
- It tries to minimise the intra-cluster distance while maximising the inter-cluster distance.
- K-means aims to minimize the total squared error from a central position in each cluster.
- The less variation we have within clusters, the more homogeneous the data points are within the same cluster. This algorithm gives many different solutions. The output clusters will depend on the initial partitioning.

5.1.1 Pseudo Code

1. Specify the number of clusters K.
2. Initialise the centroids by selecting K data points at random for the centroids, without replacement.
3. Repeat till there is no change to the centroids and the assignment of points to clusters.
4. Compute the sum of the squared distance between data points and all centroids.
5. Assign each data point to the closest cluster (centroid).
6. Compute the centroids for the clusters by taking the average of all data points that belong to each cluster.

5.1.2 Trace

1. K = 2

   Centroids = X3 (3,4), X6 (4.5,5)

2. Calculating distance from each centroid:

$$Dist(X(a,b), Y(c,d)) = |a-c| + |b-d| \tag{19}$$

	x	y	Distance From C1	Distance from C2
X1	1	1	5	7.5
X2	1.5	2	3.5	6
X3	3	4	0	2.5
X4	5	7	5	2.5
X5	3.5	5	1.5	1
X6	4.5	5	2.5	0
X7	3.5	4.5	1	1.5

Figure 13: Distance from Centroids - Iteration 1

3. Partition the data into clusters based on minimum distance from Centroids, and find mean of each cluster.

	C1	
X1	1	1
X2	1.5	2
X3	3	4
X7	3.5	4.5
Mean	2.25	2.875

Figure 14: Cluster 1 - Iteration 1

	C2	
X4	5	7
X5	3.5	5
X6	4.5	5
Mean	4.333333	5.666667

Figure 15: Cluster 2 - Iteration 1

New Centroids - (2.25,2.875),(4.33,5.66)

4. Calculating distance from each new centroid:

	x	y	Distance From C1	Distance from C2
X1	1	1	3.125	8
X2	1.5	2	1.625	6.5
X3	3	4	1.875	3
X4	5	7	6.875	2
X5	3.5	5	3.375	1.5
X6	4.5	5	4.375	0.84
X7	3.5	4.5	2.875	2

Figure 16: Distance from Centroids - Iteration 2

5. Partition the data into clusters based on minimum distance from Centroids, and find mean of each cluster.

	C1	
X1	1	1
X2	1.5	2
X3	3	4
X7	3.5	4.5
Mean	2.25	2.875

Figure 17: Cluster 1 - Iteration 2

	C2	
X4	5	7
X5	3.5	5
X6	4.5	5
Mean	4.333333	5.666667

Figure 18: Cluster 2 - Iteration 2

6. Optimum clusters have been obtained as there is no further change in the clusters and centroids and the algorithm is terminated.

5.2 K - Medoids Clustering

- The k-medoids or partitioning around medoids (PAM) algorithm is a clustering algorithm reminiscent of the k-means algorithm.
- The k-medoids algorithm chooses data points as centers(medoids or exemplars) and can be used with arbitrary distances, while in k-means the centre of a cluster is not necessarily one of the input data points (it is the average between the points in the cluster).
- A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal, that is, it is a most centrally located point in the cluster.

5.2.1 Pseudo Code

1. Initialize: Select k random points out of the n data points as the medoids.
2. Associate each data point to the closest medoid by using any common distance metric methods.

3. Cost is computed as :

$$c = \sum_{C_i} \sum_{P_i \in C_i} |P_i - C_i| \tag{20}$$

4. While the cost decreases:

   - For each medoid m, for each data o point which is not a medoid:
     - Swap m and o, associate each data point to the closest medoid , recompute the cost.
     - If the total cost is more than that in the previous step, undo the swap.

5.2.2 Trace

1. K = 2
   Centroids = X3 (3,4), X6 (4.5,5)

2. Calculating dissimilarity from each centroid:

$$Dist(X(a,b), Y(c,d)) = |a - c| + |b - d| \tag{21}$$

	x	y	Dissimilarity From C1	Dissimilarity From C2
X1	1	1	5	7.5
X2	1.5	2	3.5	6
X3	3	4	0	2.5
X4	5	7	5	2.5
X5	3.5	5	1.5	1
X6	4.5	5	2.5	0
X7	3.5	4.5	1	1.5

Figure 19: Dissimilarity from Centroids - Iteration 1

3. Each point goes to the cluster with lesser distance. Calculate the total cost.

	C1		Cost
X1	1	1	5
X2	1.5	2	3.5
X3	3	4	0
X7	3.5	4.5	1
		Total	8.5

Figure 20: Cluster 1 - Iteration 1

	C2		Cost
X4	5	7	2.5
X5	3.5	5	1
X6	4.5	5	0
		Total	3.5

Figure 21: Cluster 2 - Iteration 1

C (Total Cost) + 9.5 + 3.5 = 13

4. Select another point at random from the non-medoids and recalculate the cost. Let the point be(3.5,4.5)

5. Calculating dissimilarity from each centroid:

   The centroids for calculation are (3,4) and (3.5,4.5).

	x	y	Dissimilarity From C1	Dissimilarity From C2
X1	1	1	5	6
X2	1.5	2	3.5	4.5
X3	3	4	0	1
X4	5	7	5	4
X5	3.5	5	1.5	0.5
X6	4.5	5	2.5	1.5
X7	3.5	4.5	1	0

Figure 22: Dissimilarity from Centroids - Iteration 2

6. Each point goes to the cluster with lesser distance. Calculate the total cost.

	C1		Cost
X1	1	1	5
X2	1.5	2	3.5
X3	3	4	0
		Total	8.5

Figure 23: Cluster 1 - Iteration 2

	C2		Cost
X4	5	7	4
X5	3.5	5	0.5
X6	4.5	5	1.5
X7	3.5	4.5	0
		Total	6

Figure 24: Cluster 2 - Iteration 2

Total Cost, C = 8.5 + 6 = 14.5

Swap Cost = Present Cost – Previous Cost = 14.5 – 13 = 1.5 > 0

Swap cost is not less than 0 ⇒ Undo the swap

7. Optimum clusters have been obtained as there is no further change in the clusters and centroids and the algorithm is terminated.

5.3 Hierarchial Clustering

- Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters and seeks to build a hierarchy of these clusters.
- In general, the merges and splits are determined in a greedy manner.
- The results of hierarchical clustering are usually presented in a dendrogram.

5.3.1 Agglomerative Clustering

- Agglomerative clustering is a bottom-up clustering algorithm in which the number of clusters need not be specified in advance.
- Each object is initially considered as a single-element cluster (leaf).

- At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (nodes).
- This procedure is iterated until all points are member of just one single big cluster (root).

Pseudo Code

1. Given a dataset (X1,X2,X3,....Xn) of size n.
2. Initially each data point is a singleton cluster.
3. Compute the distance matrix for the clusters. Distance is found using a suitable distance measure and is the distance between the centers of 2 clusters.
4. Merge the 2 clusters having minimum distance. The new center is the average of the 2 old centers.
5. Repeat from step 2 till only a single cluster remains.

Trace

1. Calculating distance matrix
   - Each point is a single cluster.
   - Number of Clusters = 7

	X1	X2	X3	X4	X5	X6	X7
X1	0						
X2	1.12	0					
X3	3.61	2.5	0				
X4	7.21	6.10	3.61	0			
X5	4.72	3.61	1.12	2.5	0		
X6	5.32	4.24	1.80	2.06	1	0	
X7	4.30	3.20	0.71	2.92	0.5	1.12	0

Figure 25: Distance Matrix - Iteration 1

2. Updating distance matrix
   - Merged - X5 and X7
   - Number of Clusters = 6

	X1	X2	X3	X4	X6	X5,X7
X1	0					
X2	1.12	0.00				
X3	3.61	2.50	0.00			
X4	7.21	6.10	3.61	0.00		
X6	5.32	4.24	1.80	2.06	0.00	
X5,X7	4.30	3.20	0.71	2.92	1.12	0.00

Figure 26: Distance Matrix - Iteration 2

3. Updating distance matrix
    - Merged - X3, X5 and X7
    - Number of Clusters = 5

	X1	X2	X4	X6	X3,X5,X7
X1	0.00				
X2	1.12	0.00			
X4	7.21	6.10	0.00		
X6	5.32	4.24	2.06	0.00	
X3,X5,X7	4.30	3.20	2.92	1.12	0.00

Figure 27: Distance Matrix - Iteration 3

4. Updating distance matrix
    - Merged - X3, X5, X6 and X7
    - Number of Clusters = 4

	X1	X2	X4	X3,X5,X6,X7
X1	0			
X2	1.12	0.00		
X4	7.21	6.10	0.00	
X3,X5,X6,X7	4.30	3.20	2.92	0

Figure 28: Distance Matrix - Iteration 4

5. Updating distance matrix
    - Merged - X1 and X2
    - Number of Clusters = 3

	X1, X2	X4	X3,X5,X6,X7
X1, X2	0		
X4	6.10	0	
X3,X5,X6,X7	3.20	2.92	0

Figure 29: Distance Matrix - Iteration 5

6. Updating distance matrix
    - Merged - X3, X4, X5, X6 and X7
    - Number of Clusters = 2

	X1, X2	X3,X4,X5,X6,X7
X1, X2	0	
X3,X4,X5,X6,X7	3.201562	0

Figure 30: Distance Matrix - Iteration 6

7. The remaining two clusters are merged to form the final cluster. A dendogram is made to represent the hierarchical clustering.

5.3.2 Divisive Clustering

- In divisive or top-down clustering method we assign all of the observations to a single cluster and then partition the cluster to two least similar clusters.
- We proceed recursively on each cluster until there is one cluster for each observation.

Pseudo Code

1. Given a dataset of size N.
2. At the top we have all data in one cluster.
3. The cluster is split using a flat clustering method (eg: K-Means).
4. Choose the best cluster among all the clusters to split.
5. Go back to step 2 and repeat till each data is in its own singleton cluster.

Trace

1. Consider the dataset used in subsection 5.3.1.

	x	y
X1	1	1
X2	1.5	2
X3	3	4
X4	5	7
X5	3.5	5
X6	4.5	5
X7	3.5	4.5

2. Initially all the data points are a single cluster (X1, X2, X3, X4, X5, X6, X7).
3. K-Means algorithm is applied (using python package KMeans). This results in clusters:
   C_1=(X1,X2)
   C_2=(X3, X4, X5, X6, X7)
4. C_1 can be divided into singleton clusters.
5. C_2 is split further using K-means. It results in:
   C_{21}=(X3)
   C_{22}=(X5,X2,X4,X6)

6. C_{22} is split further resulting in:
   C_{221}=(X5)
   C_{222}=(X2,X4,X6)
7. C_{222} is split further resulting in:
   C_{2221}=(X2)
   C_{2222}=(X4,X6)
8. C_{2222} can be divided into singleton clusters.
9. Nothing can be split further, so the algorithm ends. The cluster hierarchy was the same as that obtained in Agglomerative approach.

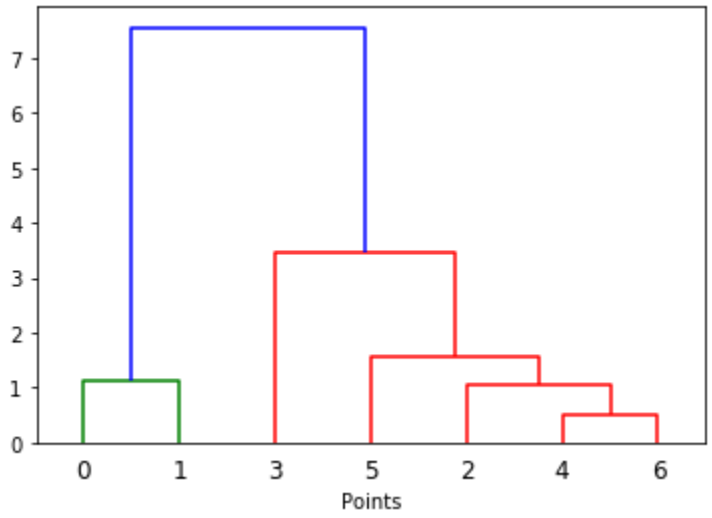


Figure 31: Dendogram for hierarchical clustering

6 Distance Measures

6.1 Jaccard Distance

- Jaccard index or jaccard similarity index is known as intersection over union.
- It is formally defined as size of intersection divided by size of union of two sets.

$$JaccardIndex = J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{22}$$

- The Jaccard distance, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from one.

$$JaccardDistance = 1 - J(A,B) \tag{23}$$

6.2 Cosine Distance

- Cosine similarity is calculated by measuring the cosine of angle between two vectors.

$$CosineSimilarity = cos(\theta) = \frac{A.B}{|A||B|} \tag{24}$$

- The cosine of 0° is 1, and it is less than 1 for any angle in the interval $(0, \pi]$ radians. It is thus a judgment of orientation and not magnitude:
  - two vectors oriented in same direction have a similarity of 1,
  - two vectors oriented at 90° relative to each other have a similarity of 0,
  - and two vectors which are diametrically opposed will have cosine similarity of -1.
- The term cosine distance is often used for the complement in positive space.

$$CosineDistance = 1 - CosineSimilarity \tag{25}$$

6.3 Minkowski Metric/L_p Norm

- The popularly used distance metric, called the Minkowski Metricis of the form:

$$L_p(X, Y) = (\sum_{i=1}^{d} |x_i - y_i|^p)^{\frac{1}{p}} \tag{26}$$

  where p = 1,2,3,...∞ and d is the dimension.
- This is also called the L_p norm. Depending on the value of p, we get different distance measures.

6.4 Manhattan Distance/L_1 Norm

- When p = 1 is substituted in equation(26), we get the Manhattan or the city block distance. It is also called the L_1norm.
- It can be written as:

$$D(X, Y) = \sum_{i=1}^{d} |x_i - y_i| \tag{27}$$

6.5 Euclidean Distance/L_2 Norm

- When p = 2 is substituted in equation(26) it is Euclidean distance. It is also called the L_2 norm.
- It can be written as:

$$D(X, Y) = (\sum_{i=1}^{d} |x_i - y_i|^2)^{\frac{1}{2}} \tag{28}$$

6.6 Chebyshev Distance

- Chebyshev distance, chessboard distance or L_∞ metric is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension.
- The Chebyshev distance between two vectors or points x and y, with standard coordinates x_i and y_i, respectively, is

$$D(X, Y) = max(|x_i - y_i|) \tag{29}$$

- This is equivalent to L_p norm if p $\rightarrow \infty$.

6.7 Trace

Consider the data points (2,6) and (4,5).

1. Cosine Distance = 1 - $\frac{(2*4)+(6*5)}{\sqrt{4+36}\sqrt{16+25}}$ = 1 - $\frac{38}{40.49}$ = 1 - 0.938 = 0.62

2. Manhattan Distance = $|2-4| + |6-5| = 3$

3. Euclidean Distance = $(|2-4|^2 + |6-5|^2)^{\frac{1}{2}} = \sqrt{5} = 2.236$

4. Chebyshev distance = = $\max(|2-4|, |6-5|) = 2$

5. Consider the sets s1 = 1,2,3, s2 = 2,3,4,5,6.
   Jaccard Index = $\frac{|2,3|}{|1,2,3,4,5,6|} = \frac{2}{6} = 0.34$
   Jaccard Distance = 1 - Jaccard Index = 0.66

7 Implementation

Python implementation of Decision tree, Naive Bayes, BPN, K-Means, and Hierarchical Clustering are attached.