Analysis and Systems of Big Data

Pranav Parameshwaran

COE17B036

I. **Classifiers**

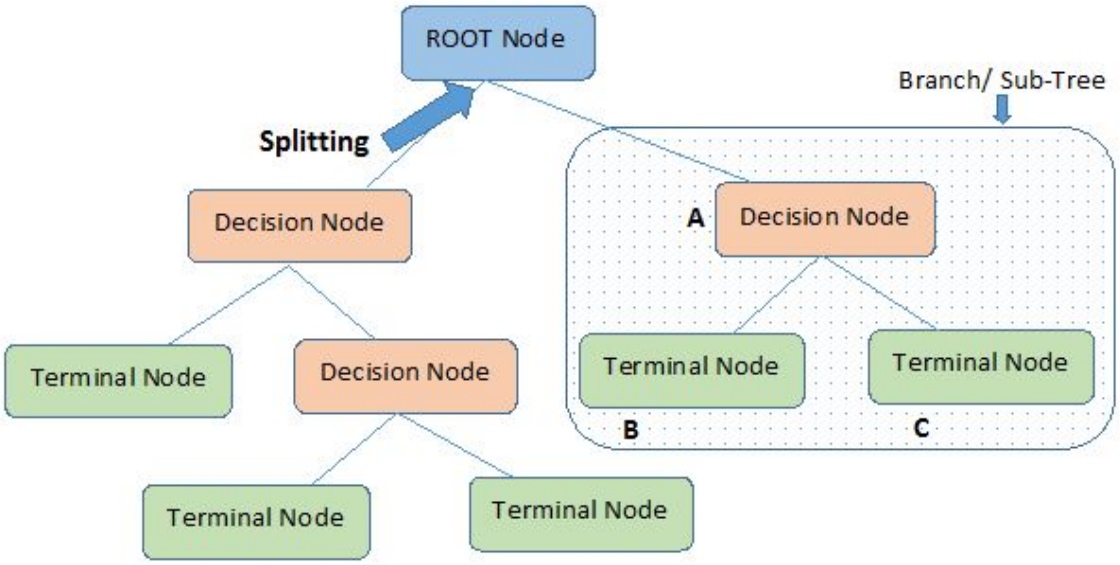
Toy Dataset

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

This is the dataset which we will be using for all the following classifiers.
This dataset is used to predict if given the information like outlook,temperature etc, if we can play football or not.
The number of Yes (1) cases are 9 and No(0) is 5 and total number of training data is 14.

Decision Tree Induction:

A decision tree is a classification and prediction tool having a tree like structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Entropy :

Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy, the harder it is to draw any conclusions from that information.

Suppose S is a set of instances, A is an attribute, then

E(S,A) = $\sum_{i=1}^{j} P(A(i)) * E(yes(i), no(i))$, where j εA (j is all subparts of A)

E(yes,no) = -[P(yes)*$\log_2$(P(yes)) + P(no)*$\log_2$(P(no))]

** Yes,No corresponds to the 1 and 0 (Target Class) respectively.

Information Gain :

Information gain can be defined as the amount of information gained about a random variable from observing another random variable.It can be considered as the difference between the entropy of a parent node and weighted average entropy of child nodes.

Root Node is chosen to be the one with High Information Gain. Following which its child nodes are also chosen in the same fashion.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log_b \mathrm{P}(x_i)$$

$$IG(S, A) = H(S) - H(S, A)$$

Alternatively,

$$IG(S, A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

Pseudo Code

ID3 (examples, Target_Attribute, Attributes)

Create a root node for the tree

if all examples are positive/negative , **then Return** the single-node tree Root, with label = +/- respectively.

if number of predicting attributes is empty, **then Return** the single-node tree Root, with label = most common value of the target attribute in the examples.

Otherwise Begin

A ← The attribute that best classifies examples. i.e; which has Max. Info Gain.

Decision tree attribute for Root = A

For each possible value, v_i, of A,

Add a new tree branch below Root, corresponding to the test A = v_i

Let Examples(v_i) be the subset of examples that have the value v_i for A

if Examples(v_i) is empty

Then below this branch add a leaf node with label = most common target value in the examples.

else below this new branch add the subtree ID3 (Examples(v_i), Target_Attribute, Attributes - {A})

End

Return Root

Trace :

Find the entropy of a class variable.

E(S) = -[(9/14)log(9/14) + (5/14)log(5/14)] = 0.94

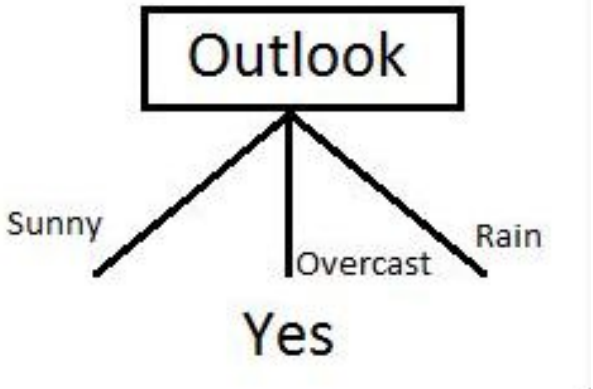
Now we have to calculate average weighted entropy. ie, we have found the total of weights of each feature multiplied by probabilities.

E(S, outlook) = (5/14)*E(3,2) + (4/14)*E(4,0) + (5/14)*E(2,3) = (5/14)(-(3/5)log(3/5)-(2/5)log(2/5))+ (4/14)(0) + (5/14)((2/5)log(2/5)-(3/5)log(3/5)) = 0.693

Next step is to find the information gain.It is the difference between parent entropy and average weighted entropy we found above.

IG(S, outlook) = 0.94 - 0.693 = 0.247
Similarly find Information gain for Temperature,Humidity and Windy.
IG(S, Temperature) = 0.940 - 0.911 = 0.029
IG(S, Humidity) = 0.940 - 0.788 = 0.152
IG(S, Windy) = 0.940 - 0.8932 = 0.048

Now select the feature having the largest entropy gain.Here it is Outlook.So it forms the first node(root node) of our decision tree.



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

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

Since overcast contains only examples of class 'Yes' we can set it as yes. That means If outlook is overcast football will be played. Now our decision tree looks as follows.

Next step is to find the next node in our decision tree. Now we will find one under sunny. We have to determine which of the following Temperature ,Humidity or Wind has higher information gain.

Calculate parent entropy E(sunny)
E(sunny) = (-(3/5)log(3/5)-(2/5)log(2/5)) = 0.971.

E(sunny, Temperature) = (2/5)*E(0,2) + (2/5)*E(1,1) + (1/5)*E(1,0)=2/5=0.4

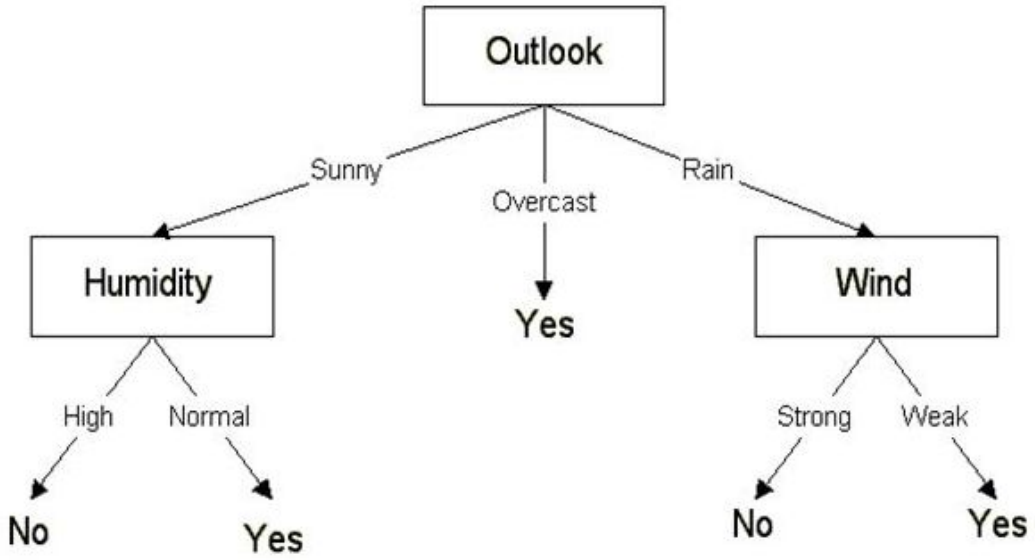
Now calculate information gain.

IG(sunny, Temperature) = 0.971–0.4 =0.571
Similarly we get
IG(sunny, Humidity) = 0.971
IG(sunny, Windy) = 0.020

Here IG(sunny, Humidity) is the largest value.So Humidity is the node which comes under sunny.

	play	
Humidity	yes	no
high	0	3
normal	2	0

For humidity from the above table, we can say that play will occur if humidity is normal and will not occur if it is high. Similarly, find the nodes under rainy.



Our final decision tree looks like the above. This method was using Entropy and Information Gain as a metric. It is called ID3 (Iterative Dichotomiser 3).

Naive Bayesian Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks. The crux of the classifier is based on the 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.

The fundamental Naive Bayes assumption is that each feature makes an:

- independent
- equal

contribution to the outcome.

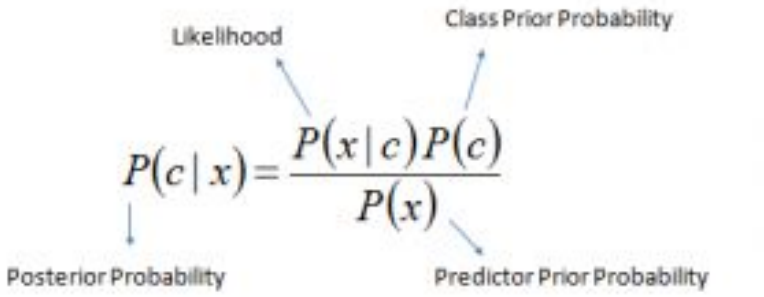
Zero probability scenario

When there is an attribute that just has only probability 1 , an event that cannot possibly happen has a probability of zero. If there is a chance that an event will happen, then its probability is between zero and 1.

Solution：An approach to overcome this 'zero frequency problem' in a Bayesian setting is to add one to the count for every attribute value-class combination when an attribute value doesn't occur with every class value.

This ensures that $P(x_i|y) \neq 0$ hence we will never get a zero probability scenario

Formula



$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Above,

- P(x) is the evidence. The event which has occured, we use this to predict event c given this.
- P(c) is the prior probability of class (probability before the event x has occurred)
- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(x|c) is the likelihood which is the probability of a predictor given class.

Since X is an independent vector of attributes, we can simplify and we get,

$$P(C_k \mid x_1,x_2,x_3....,x_n) = P(C_k) * \prod_{i=1}^{n} P(x_i \mid C_k)$$

Here C_k represents the target Class 'k'. If we are doing a classifier model then C_k = 'yes' (or) 'no'.

From this we can construct the classifier model, we find the probability of a given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:

$$\hat{y} = \text{argmax}_{k \in \{1,\ldots,K\}} \; P(C_k) * \prod_{i=1}^{n} P(x_i \mid C_k)$$

Gaussian naive bayes classifier

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values.

The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{\sigma_y^2}\right)$$

Trace

We now trace this algorithm on the Toy dataset which we used before. We need to compute P(X|C) and also P(C) which we have computed in the table below.

Outlook

	Yes	No	P(yes)	P(no)
Sunny	2	3	2/9	3/5
Overcast	4	0	4/9	0/5
Rainy	3	2	3/9	2/5
Total	9	5	100%	100%

Temperature

	Yes	No	P(yes)	P(no)
Hot	2	2	2/9	2/5
Mild	4	2	4/9	2/5
Cool	3	1	3/9	1/5
Total	9	5	100%	100%

Humidity

	Yes	No	P(yes)	P(no)
High	3	4	3/9	4/5
Normal	6	1	6/9	1/5
Total	9	5	100%	100%

Wind

	Yes	No	P(yes)	P(no)
False	6	2	6/9	2/5
True	3	3	3/9	3/5
Total	9	5	100%	100%

Play		P(Yes)/P(No)
Yes	9	9/14
No	5	5/14
Total	14	100%

Since we have computed the required probabilities, let us test on new features;
today = (Sunny, Hot, Normal, False)

So, probability of playing golf is given by:

$$P(Yes|today) = \frac{P(SunnyOutlook|Yes)P(HotTemperature|Yes)P(NormalHumidity|Yes)P(NoWind|Yes)P(Yes)}{P(today)}$$

and probability to not play golf is given by:

$$P(No|today) = \frac{P(SunnyOutlook|No)P(HotTemperature|No)P(NormalHumidity|No)P(NoWind|No)P(No)}{P(today)}$$

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

$$P(Yes|today) \propto \frac{2}{9}.\frac{2}{9}.\frac{6}{9}.\frac{6}{9}.\frac{9}{14} \approx 0.0141$$

And

$$P(No|today) \propto \frac{3}{5}.\frac{2}{5}.\frac{1}{5}.\frac{2}{5}.\frac{5}{14} \approx 0.0068$$

Now, since

P(Yes | today) + P(No | today) = 1

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

P(Yes | today) = $\frac{0.0141}{0.0141 + 0.0068}$ = 0.67

And

P(No | today) = $\frac{0.0068}{0.0141 + 0.0068}$ = 0.33

Since

P(Yes | today) > P(No | today)

So, the prediction that golf would be played is 'Yes'.

Neural Network Classifier:

Topology of the Network:

It refers to the way the neurons are connected in the network, and it is an important factor in network functioning and learning.

Input Layer:

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. The input layer is the very beginning of the workflow for the artificial neural network.

Hidden Layer:

A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.The hidden layers perform nonlinear transformations of the inputs entered into the network.

Output Layer:

The output layer in an artificial neural network is the last layer of neurons that produces given outputs for the program.

Activation Function:

Activation functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated or not, based on whether each neuron's input is relevant for the model's prediction.

Weight:

A weight represents the strength of the connection between units. A weight brings down the importance of the input value.

Bias:

Bias is like the intercept added in a linear equation. It is an additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron.

Strengths and Limitations of Neural Network:

Strength:

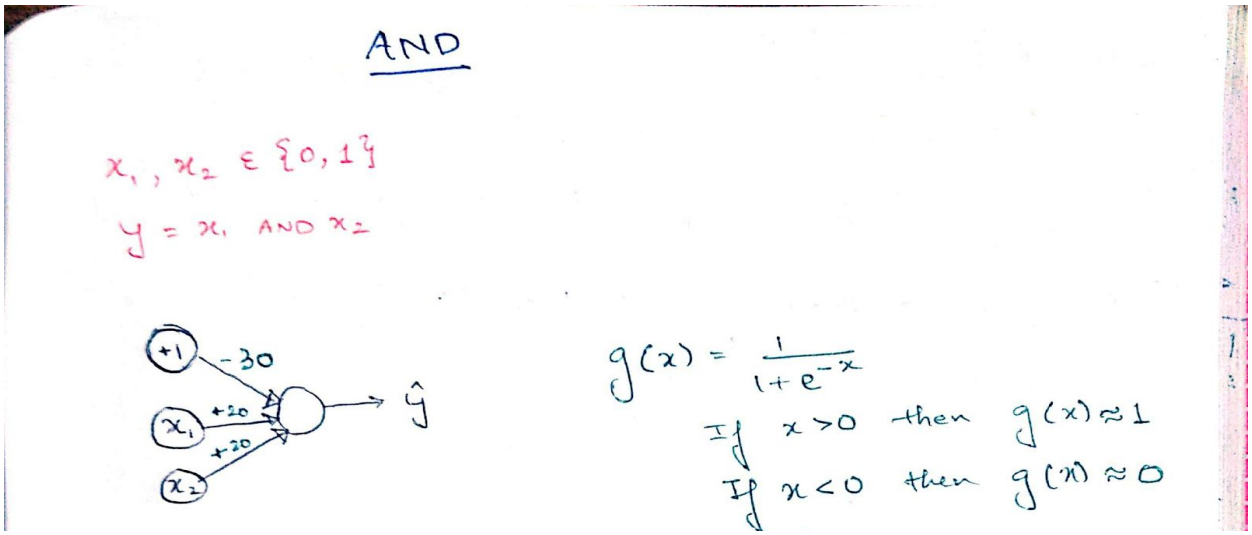
- Storing information on the entire network
- It has the ability to work with inadequate knowledge: After ANN training, the data may produce output even with incomplete information.

- It has fault tolerance: Corruption of one or more cells of ANN does not prevent it from generating output. This feature makes the networks fault-tolerant.
- Having a distributed memory
- Ability to train machines: Artificial neural networks learn events and make decisions by commenting on similar events.
- Parallel processing ability

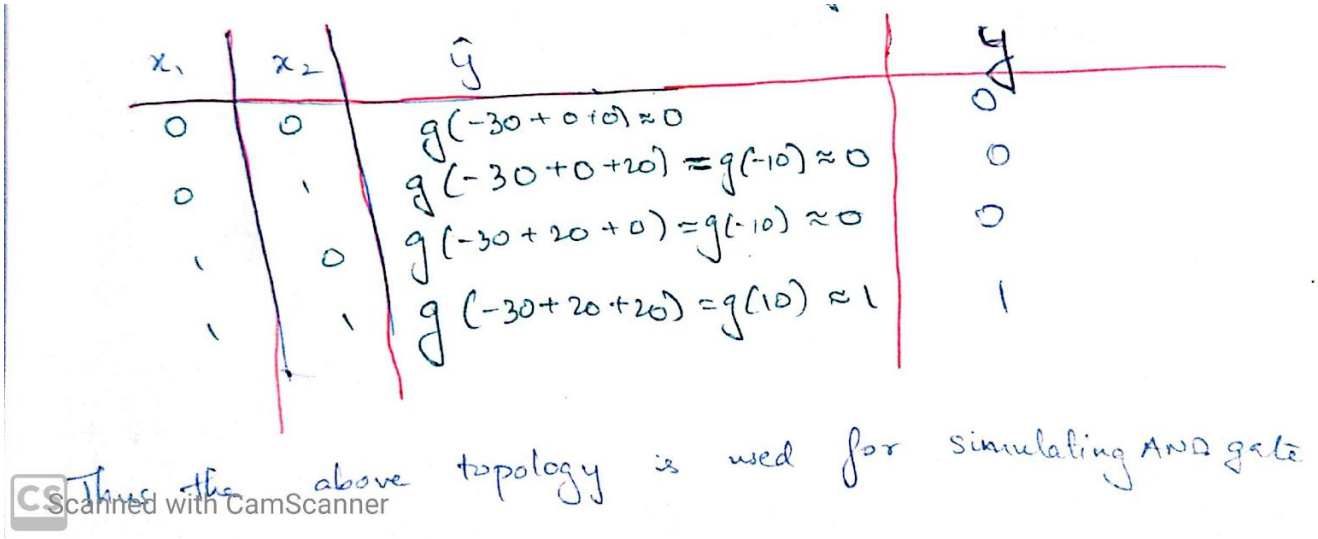
Limitations:

- Hardware dependence: Artificial neural networks require processors with parallel processing power, by their structure. For this reason, the realization of the equipment is dependent.
- Unexplained functioning of the network:Generally neural networks are considered as Black-Box as we are not aware of the functioning about what is happening.
- Assurance of proper network structure: The appropriate network structure is achieved through experience and trial and error.
- The 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 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.

Simulation of AND operation using Neural Network

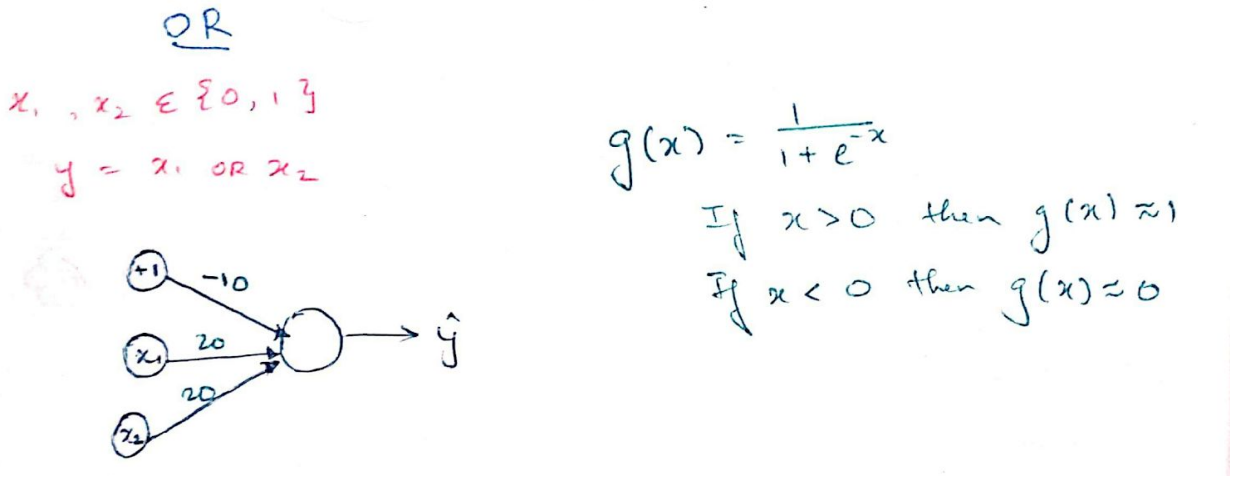


We design the Network to be as the above and initialize the weights as above.



We see that when we initialized the weights as (+20,+20) and bias as (-30) we were able to replicate the AND gate.

Simulation of OR operation using Neural Network



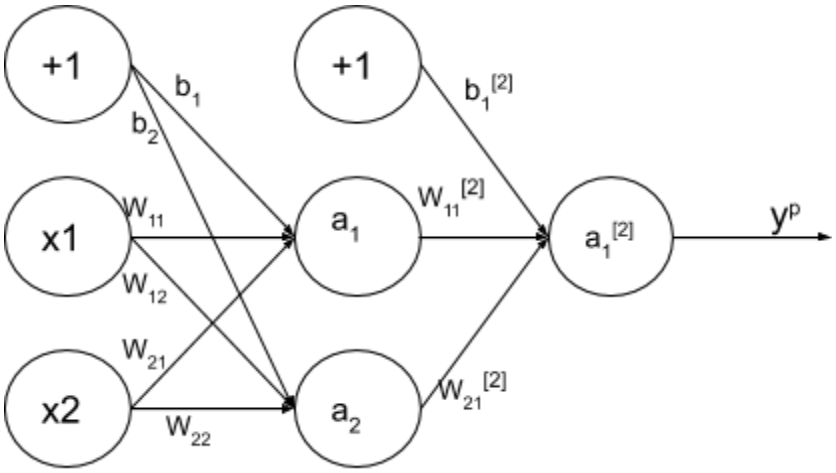
We design the Network to be as the above and initialize the weights as above.

x_1	x_2	$\hat{y}$	y
0	0	$g(-10+0+0) \approx 0$	0
0	1	$g(-10+0+20) \approx g(10) \approx 1$	1
1	0	$g(-10+20+0) \approx g(10) \approx 1$	1
1	1	$g(-10+20+20) = g(30) \approx 1$	1

Thus the above topology is used for simulating OR gate

We see that when we initialized the weights as (+20,+20) and bias as (-10) we were able to replicate the OR gate.

Simulation of XOR operation using Neural Network (consists of one hidden layer)



Initialization of weights

$W^{[1]}$ = | 0.886 | 0.612 |
|---|---|
| 0.372 | 0.871 |

$b^{[1]}$ = | 0.191 | 0.163 |
|---|---|

$W^{[2]}$ = | 0.052 |
|---|
| 0.029 |

$b^{[2]}$ = | 0.912 |
|---|

X = | 0 | 0 |
|---|---|
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

Y = | 1 |
|---|
| 0 |
| 0 |
| 1 |

Forward Propagation

$A^{[1]} = [[a_1 \ a_2]] =$

0.547	0.540
0.637	0.738
0.746	0.685
0.810	0.838

$\hat{y} = A^{[2]} = [[a^{[2]}]] =$

0.722
0.724
0.725
0.726

Error =

1-0.722
0-0.724
0-0.725
1-0.726

=

0.278
-0.724
-0.725
0.274

Backward Propagation

$dA^{[2]} = Error^* (\hat{y} * (1- \hat{y})) =$

0.0556
-0.1446
-0.1445
0.0542

$dA^{[1]} = (dA^{[2]}.(W^{[2]})^T)*A^{[1]}*(1-A^{[1]}) =$

0.0007	0.0004
-0.0017	-0.0008
-0.0014	-0.0009
0.0004	0.0002

Update Weights and Biases

$W^{[2]} += lr*((A^{[1]})^T . dA^{[2]}) =$

0.040
0.016

$b^{[2]} += lr* \sum dA^{[2]} =$

0.894

$W^{[1]} += lr*((X)^T . dA^{[1]}) =$

0.885	0.611
0.371	0.871

$b^{[1]} += lr* \sum dA^{[1]} =$

0.191	0.163

We see that after doing back propagation we update our weights and we see that these values have changed. Hence repeating over a lot of iterations will get us to the minima, this method of updating the weights is known as gradient descent.

The predicted values after 10,000 iterations is

0.947
0.048
0.048
0.948

which is very close to the actual value. Hence NN can be used for even non-linear functions as well.

Nearest Neighbour Classifier:

Algorithm

Let m be the number of training examples.Let p be an unknown point which we want to classify.

1. For all the training examples: Compute the euclidean distance between each point and point p.
2. Make a set S of K smallest distances obtained. Each of these distances corresponds to an already classified datapoint.
3. Return the majority label among S.

We want to find out which class ‘p’ belongs and we identify using the K closest points and the max(class-num) in that list.

Trace

Let us consider the following dataset

S.No.	x1	x2	Class
1	1	12	0
2	2	5	0
3	5	3	1
4	3	2	1
5	1.5	9	1
6	3	6	0
7	3	10	0
8	3.5	8	0
9	7	2	1
10	6	1	1
11	3.8	1	1
12	5.6	4	1

13	2	11	0
14	2	9	0
15	1	7	0
16	4	2	1
17	2	5	0

Let p = (2.5,7), K=4

We compute the euclidean distance between all these points with p. We get the following (arranged in sorted order):

Class	Euclidean dist.
0	1.11
0	1.41
0	1.5
0	2.06
0	2.06
1	2.06
1	2.23
0	3.04
0	4.03
1	4.31
1	4.71
1	5.02
0	5.22

1	5.22
1	6.13
1	6.72
1	6.94

Hence we see that the top K points all have class 0. Hence p also belongs to class 0.

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.

Process of Natural Selection

The process of natural selection starts with the selection of fittest individuals from a population. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. This process keeps on iterating and at the end, a generation with the fittest individuals will be found.
There are 5 phases involved in Genetic Algorithm:

1. Initial Population
2. Fitness function
3. Selection
4. Crossover

5. Mutation

1. Initial Population

The process begins with a set of individuals which is called a **Population**. Each individual is a solution to the problem you want to solve.

An individual is characterized by a set of parameters (variables) known as **Genes**. Genes are joined into a string to form a **Chromosome** (solution).

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Usually, binary values are used (string of 1s and 0s). We say that we encode the genes in a chromosome.

2. Fitness Function

The **fitness function** determines how fit an individual is (the ability of an individual to compete with other individuals). It gives a **fitness score** to each individual. The probability that an individual will be selected for reproduction is based on its fitness score.

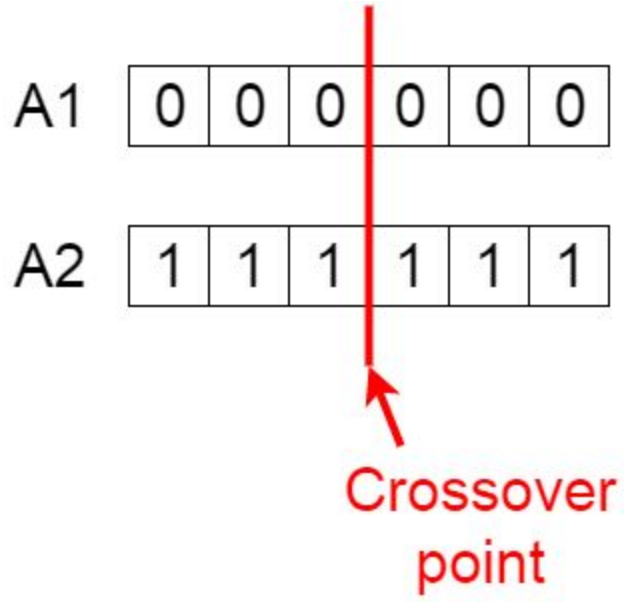
3. Selection

The idea of the **selection** phase is to select the fittest individuals and let them pass their genes to the next generation.

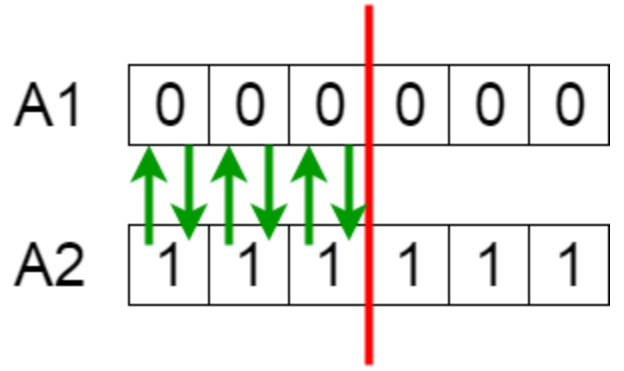
Two pairs of individuals (**parents**) are selected based on their fitness scores. Individuals with high fitness have more chances to be selected for reproduction.

4. Crossover

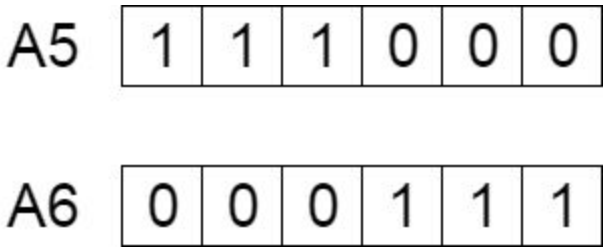
Crossover is the most significant phase in a genetic algorithm.
For each pair of parents to be mated, a **crossover point** is chosen at random from within the genes.
For example, consider the crossover point to be 3 as shown below.



Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached.

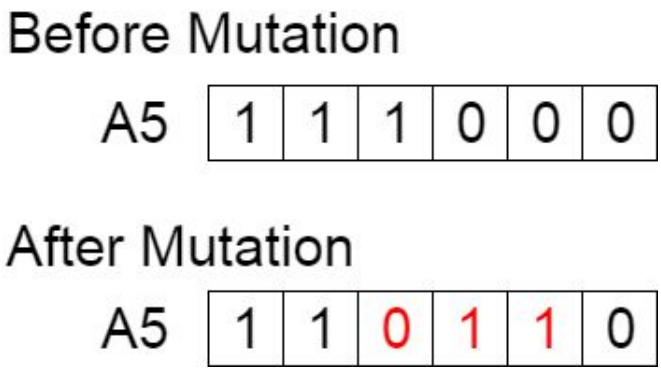


The new offspring are added to the population.



5. Mutation

In certain new offspring formed, some of their genes can be subjected to a **mutation** with a low random probability. This implies that some of the bits in the bit string can be flipped.



Mutation occurs to maintain diversity within the population and prevent premature convergence.

6. Termination

The algorithm terminates if the population has converged (does not produce offspring which are significantly different from the previous generation). Then it is said that the genetic algorithm has provided a set of solutions to our problem.

NOTE: The population has a fixed size. As new generations are formed, individuals with least fitness die, providing space for new offspring.
The sequence of phases is repeated to produce individuals in each new generation which are better than the previous generation.

Pseudocode

START
Generate the initial population
Compute Fitness
REPEAT
    Selection
    Crossover
    Mutation
    Compute Fitness
UNTIL population has converged
STOP

Trace

The optimization function given = max f(x) = $x^3 - 2x^2 + x$, within a range of (0, 31)

S.No	Initial Population	x	f(x)	Probability	Expected Count	Actual count
1	01101	13	1872	0.06	0.24	0
2	11110	30	25230	0.8	3.17	3

3	01000	8	392	0.013	0.05	0
4	10001	17	4352	0.127	0.54	1

$\sum f(x) = 31846$

avg(f(x)) = 7961.5
max(f(x)) = 25230
It is observed that after updating the mating pool, the average fitness value has increased drastically. For the element 11110, if we perform mutation in the bit 0, we get 11111 which is 31 which gives a fitness value of 27900. We stop the gene selection algorithm at this stage as we have reached the end of the interval given in the question.
Hence max value of = $x^3 - 2x^2 + x = 27900$, occurs at x = 31

Bucket Bridge Classifier:

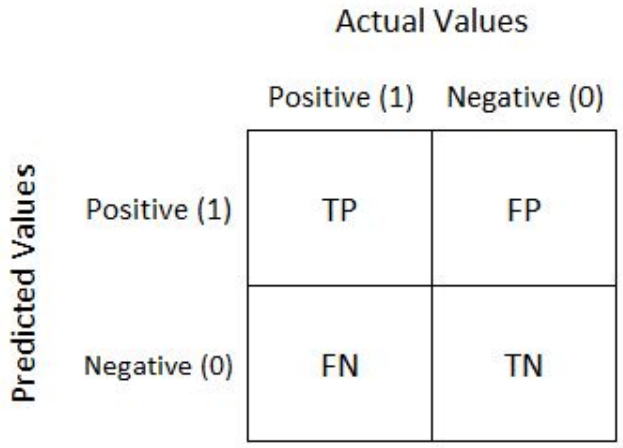
The first major learning task facing any rule-based system operating in a complex environment is the credit assignment task, Somehow the performance system must determine both the rules responsible for its successes and the representativeness of the conditions encountered in attaining the successes. 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 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. It is worth recalling that 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 19enters 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 summed so that one of the conflicting actions is chosen with a probability proportional to the sum of its advocates.

Performance measures:

Confusion Matrix

It is a performance measurement for machine learning classification problems where output can be two or more classes. It is a table with 4 different combinations of predicted and actual values.



- True Positive - How much has the model predicted as True and how much of it is actually true.

- False Positive - How much has the model predicted as True but the actual value is false.

- False Negative- How much has the model predicted as False but the actual value is true.

- True Negative- How much has the model predicted as False and how much of it is actually False.

Precision - Out of all the positive classes we have predicted correctly, how many are actually positive. It should be as high as possible.
Precision = $\frac{TP}{(TP+FP)}$
Sensitivity/Recall - Out of all the positive classes, how much we predicted correctly. It should be as high as possible.
Recall = $\frac{TP}{(TP+FN)}$

Accuracy - Out of all the classes, how much we predicted correctly. It should be as high as possible.
Accuracy = $\frac{1}{m} * \sum_{i=1}^{m} 1\{\hat{y} == y\}$, where m is all the test examples, $\hat{y}$ is our predicted class and y is the actual class.

Specificity - Out of all the positive classes, how much we predicted correctly.
Specificity = $\frac{TN}{(TN+FP)}$

F1-Score - It is difficult to compare a model using two evaluation metrics i.e Precision and Recall, because we expect that for a model

to be the best it should have both Precision and Recall as high as possible. Suppose there are 3 models A,B,C and it has the following Precision and Recall.

Model	Precision	Recall	F1-Score
A	0.98	0.83	0.89878453
B	0.77	0.97	0.858505747
C	0.94	0.93	0.934973262

We see that Both A,B have high Precision and Recall respectively, but their F1-Score is less than C. So C is the best model.

Sample Confusion Matrix

	Actual True	Actual False
Predicted True	320	43
Predicted False	20	538

TP = 320, TN = 538, FP = 43, FN = 20.

Precision= 320/(320+20) = 0.941
Recall= 320/(320+43) = 0.882
Accuracy = (320+538)/(320+538+20+43) = 0.932

Specificity = 538/(538+43) = 0.926
F1-Score = 0.911

Clustering Algorithms

1. K-means Clustering

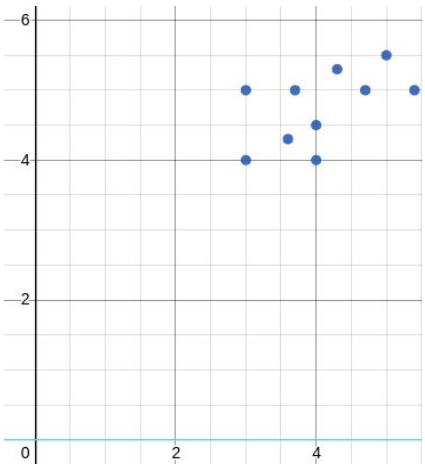
Algorithm

1. Specify number of clusters K.
2. Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
3. Repeat until convergence
   {
   a. Compute the sum of the squared distance between data points and all centroids, assign each data point to the closest cluster (centroid).
   b. Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.
   }

Trace
Dataset

We are taking the dataset in a 2D coordinate system.

Point Value	x1	x2
1	3	4
2	4.7	5
3	3	5
4	4	4
5	5.4	5
6	4	4.5
7	3.7	5
8	3.6	4.3
9	4.3	5.3
10	5	5.5



Plot of the above dataset. We see that K=2 seems to be the optimum value of K.

Let us choose two cluster points C1 = (4,4.5), C2 = (4.3,5.3)
After computing the squared distance between all points we see that
The following points have least distance with C1 :
p1,p3,p4,p6,p7,p8
The following points have least distance with C2 : p2,p5,p9,p10

Now we take average of the the points that belong to C1 and C2 respectively:
$C1_{new}$ = ((3+3+4+4+3.7+3.6)/6,(4+5+4+4.5+5+4.3)/6) = (3.55,4.67)
$C2_{new}$ = ((4.7+5.4+4.3+5.0)/4,(5+5+5.5+5.3)/4) = (4.85,5.2)

We see that now the centroid values have updated. We repeat the above process till convergence.

2. K-medoids Clustering

Algorithm

1. Specify number of clusters K.
2. Initialize medoids by first shuffling the dataset and then randomly selecting K data points for the medoids without replacement.
3. Repeat until convergence
   {
   a. Compute the Minkowski distance between data points and all medoids, assign each data point to the closest medoid.

b. For each medoid *j* and each data point *i* associated with *j*, swap *j* and *i* and compute the total cost of the configuration (which is, the average dissimilarity(Minkowski) of *i* to all the data points associated to *j*). Select the medoid *j* with the lowest cost of the configuration. Iterate between steps 2 and 3 until there is **no change** in the assignments.

}

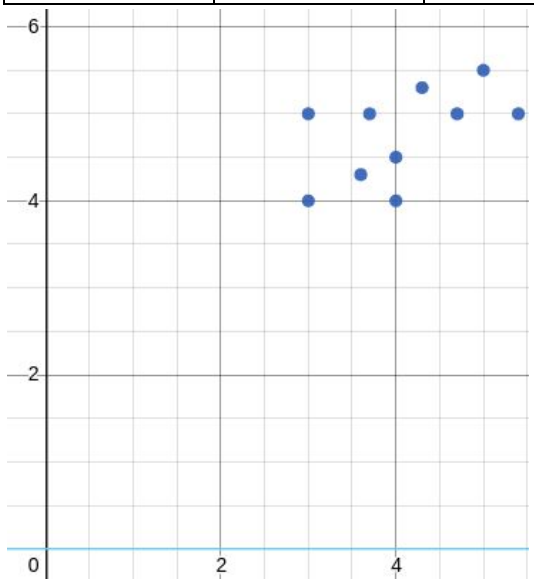
Trace

Dataset

We are taking the dataset in a 2D coordinate system.

Point Value	x1	x2
1	3	4
2	4.7	5
3	3	5
4	4	4
5	5.4	5
6	4	4.5
7	3.7	5
8	3.6	4.3
9	4.3	5.3

10	5	5.5



Plot of the above dataset. We see that K=2 seems to be the optimum value of K.

Let us initialize two medoids M1 = (3.7,5)(p3) ; M2 = (5,5.5)(p10)
Each item is assigned to whichever cluster has least dissimilarity
So, M1 = p2,p3,p4,p5,p7,p8,p9 ; M2 = p1,p6,p10
After applying Euclidean Distance as measure to find which point is the best medoid within that cluster,
Average Dissimilarity = 10.19 [computed for C1 medoid wrt to other points] ; 22.37 [computed for C2 medoid wrt to other points]
New Dissimilarity = 8.97 [for the point(4,4.5) hence this becomes our new medoid.] ; 12.79 [for the point(4.7,5) hence this becomes our new medoid.]
We update the medoids
M1 = (4,4.5)(p2) ; M2 = (4.7,5)(p6)
Repeat the process until there is no change to the assignments.
(This update is done for just one iteration)

3. Hierarchical Clustering

Agglomerative Clustering

This is a "bottom-up" approach, each data point starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy until one cluster or K clusters are formed.

Algorithm

1. Compute the Proximity matrix
2. Let each data point be a cluster
3. Repeat until only a single cluster remains {
    a. Merge the two closest clusters based on similarity and update the proximity matrix

}

To merge clusters we use similarity between two clusters, e.g, Min,Max,Group Average,Ward's Method.

For our trace we will go through Ward's Method.

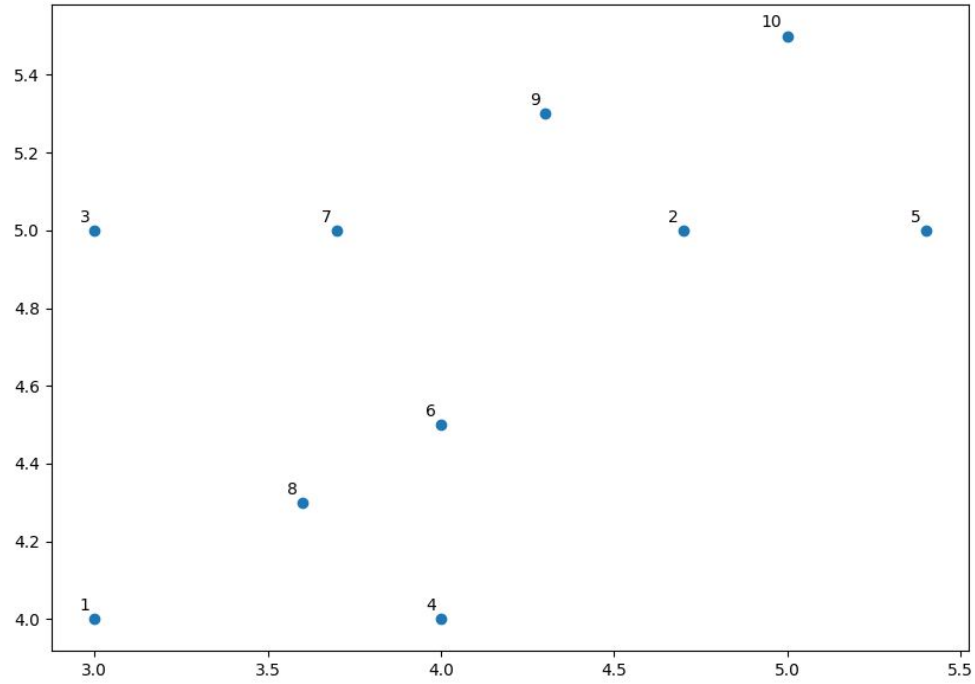
Trace

Dataset

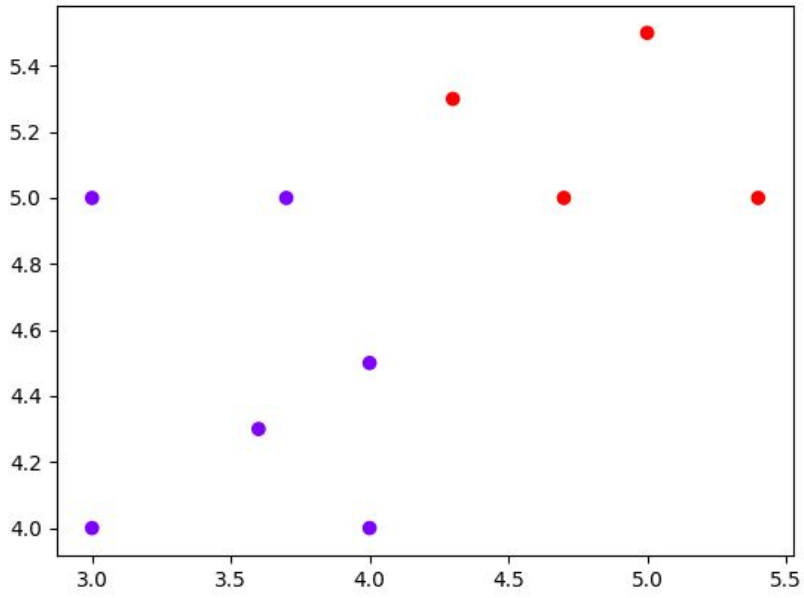
We are taking the dataset in a 2D coordinate system.

Point Value	x1	x2
1	3	4
2	4.7	5
3	3	5
4	4	4

5	5.4	5
6	4	4.5
7	3.7	5
8	3.6	4.3
9	4.3	5.3
10	5	5.5



We see that points 6(4,4.5) and 8(3.6,4.3) are closer so merge them into one cluster. Similarly points 2(4.7,5) and 9(4.3,5.3) are closer so we form another cluster. By repeating the above process based on the Euclidean distance and Ward's method as a similarity measure, we get 2 clusters.



Distance Measures

1. Minkowski Distance

$d(x,y) = \left(\sum_{i=0}^{n-1} \left| x_i - y_i \right|^p \right)^{1/p}$ where x,y are n-dimensional vectors.

Special Case

When p=1, the distance is known as Manhattan distance.
When p=2, the distance is known as Euclidean Distance.
When p-> ∞ the distance is known as Chebyshev Distance.

Example

X = (2,2) ; Y = (3,5) ; p = 3
$d(X,Y) = ((|2-3|)^3 + (|2-5|)^3)^{⅓} = 3.03658897188$

2. Cosine Distance

similarity(x,y) = $\frac{\sum_{i=1}^{n} x_i * y_i}{\sqrt{\sum_{i=1}^{n} (x_i)^2} * \sqrt{\sum_{i=1}^{n} (y_i)^2}}$ where x,y are n-dimensional vectors.

Example

X = (4,1) ; Y = (6,2)

s(X,Y) = (4*6 + 1*2)/((17)$^{0.5}$*(40)$^{0.5}$) = 0.9970544855

3. Jaccard Distance

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

JaccardDistance(A,B) = 1 - JaccardIndex(A,B)

Example

X = {1,2,3,4,5,6} ; Y = {3,4,6,7,8}

JI(X,Y) = 3/8

JaccardDistance(X,Y) = 1 - (3⁄8) = 5⁄8 = 0.625

4. Bray-Curtis Distance

D(x,y) = $\frac{\sum_{i=1}^{n} |x_i - y_i|}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$ where x,y are n-dimensional vectors.

Example

X = (4,1) ; Y = (6,2)

D(X,Y) = (|4-6| + |1-2|)/((1+4) + (6+2)) = 0.23076923076

5. Chebyshev Distance

d(x,y) =$\max_{0<i<n}(|x_i - y_i|)$ where x,y are n-dimensional vectors and 'i' runs from 1 till n, we want to find max over those dimension

Example

X = (5,3) ; Y = (4,7)

D(X,Y) = max(|5-4| , |3-7|) = max(1,4) = 4