Analytics and Systems of Big Data

Reading Assignment

CED16I010

Kruttika Bhat

Contents

1 Classifiers

Understand the working of the following classifier algorithms and trace the same for a sample dataset (min 10 records) which involves 2 classes (Binary Classifier) eg: 'Yes' or 'No' , 'True' or 'False'

1.1 Decision Tree Induction

Give pseudo code and trace decision tree algorithm. Understand attribute selection measures such as information gain, gain ratio (use anyone for the trace).

Entropy: Entropy is a measure of randomness. If all outcomes are positive or all are negative then entropy would be zero. If half are positive and half are negative then entropy is 1. Entropy multiplies the probability of the class times the log (base=2) of that class probability. For a data set S with n classes, the entropy $E(S)$ is defined as:

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

Entropy only computes the quality of a single subset of examples which corresponds to a single value. But we wish to compute the quality of the entire split which corresponds to an entire attribute. The solution is to use Average Entropy (or) Information which is the weighted average over all sets resulting from the split.

$$I(S, A) = \sum_{i} \frac{|S_i|}{|S|} E(S_i) \tag{2}$$

where A is an attribute, S is a set of values to split and S_i is a subset.

Information Gain: Using entropy we can calculate information gain. Information gain for an attribute is the difference between the entropy of target and entropy of the attribute, i.e between the parent node and the child nodes. Information Gain favors smaller partitions with many distinct values. When constructing decision tree, a variable split with maximum information gain is considered. The information gain for attribute A is:

$$Gain(S, A) = E(S) - I(S, A) = E(S) - \sum_{i} \frac{|S_i|}{|S|} E(S_i) \tag{3}$$

Gain Ratio: A problem with using information gain is that it favors splits which are small in size and have many distinct values. Gain ratio is a modification of the information gain that reduces this bias towards multi-valued attributes. It takes number and size of branches into account when choosing an attribute and corrects the information gain by taking the intrinsic information of a split into account. Intrinsic information of a split is entropy of distribution of instances into branches i.e. it tells how much information we need to

tell which branch an instance belongs to. The intrinsic information of an attribute A, set of values S and subsets S_i is:

$$IntI(S, A) = -\sum_i \frac{|S_i|}{|S|} \log\left(\frac{|S_i|}{|S|}\right) \tag{4}$$

Gain ratio would then be:

$$GR(S, A) = \frac{Gain(S, A)}{IntI(S, A)} \tag{5}$$

Gini Index: This is a popular alternative to information gain. The Gini index is used in the classic CART algorithm. It is a measure of how often an element would be incorrectly identified. It favors larger partitions. Perfectly classified Gini index would be zero. A variable split with a low Gini Index is ideal. The Gini Index is calculated by subtracting the sum of the squared probabilities of each class from one.

$$Gini(S) = 1 - \sum_i p_i^2 \tag{6}$$

Average Gini index:

$$Gini(S, A) = \sum_i \frac{|S_i|}{|S|} Gini(S_i) \tag{7}$$

Gini index is analogous to Entropy and average Gini index is analogous to Information. Similarly, Gini gain could be defined analogous to Information gain. But typically the average Gini index is minimized rather than maximizing Gini gain.

Pseudocode: This is the pseudocode for the ID3 algorithm for constructing decision tree. Other than this, a divide and conquer method also exists for constructing decision tree.

- Input: Example set S
- Output: Decision Tree DT
- If all examples in S belong to the same class c, return a new leaf and label it with c. Else,
    1. Select an attribute A according to some heuristic function.
    2. Generate a new node DT with A as test.
    3. For each Value v_i of A,
        (a) Let S_i = all examples in S with $A = v_i$
        (b) Repeat the algorithm to construct a decision tree DT_i for example set S_i
        (c) Generate an edge that connects DT and DT_i

Trace: Consider the weather data table given below. Based on the weather we should make a decision on playing golf. Decision tree will be constructed using information gain.

Temperature	Outlook	Humidity	Windy	Play Golf?
hot	sunny	high	false	no
hot	sunny	high	true	no
hot	overcast	high	false	yes
cool	rain	normal	false	yes
cool	overcast	normal	true	yes
mild	sunny	high	false	no
cool	sunny	normal	false	yes
mild	rain	normal	false	yes
mild	sunny	normal	true	yes
mild	overcast	high	true	yes
hot	overcast	normal	false	yes
mild	rain	high	true	no
cool	rain	normal	true	no
mild	rain	high	false	yes

1. Consider all attributes at root node. Find the gain for each of the 4 attributes. From equation 3, we see that first we should find the entropy for the set i.e $E(S)$. Overall, there are 9 yes and 5 no. So,

$$\text{E(S)}= -(\tfrac{9}{14}\log_2\tfrac{9}{14} + \tfrac{5}{14}\log_2\tfrac{5}{14}) = 0.940$$

Now the information gain for each attribute should be found.

(a) Humidity: Here there are 2 possibilities, high and normal.

- For Humidity=high, there are 3 yes and 4 no.

$$E(Humidity = high) = -(\tfrac{3}{7}\log_2\tfrac{3}{7} + \tfrac{4}{7}\log_2\tfrac{4}{7}) = 0.985$$

- For Humidity=normal, there are 6 yes and 1 no.

$$E(Humidity = normal) = -(\tfrac{6}{7}\log_2\tfrac{6}{7} + \tfrac{1}{7}\log_2\tfrac{1}{7}) = 0.592$$

On substitution,

$$Gain(S, Humidity) = E(S) - I(S, Humidity) = 0.940 - (\tfrac{7}{14} * 0.985 + \tfrac{7}{14} * 0.592) = 0.151$$

(b) Wind: Do the same procedure. Here there are 2 possibilities, weak and strong.

- $E(Wind = weak) = -(\tfrac{6}{8}\log_2\tfrac{6}{8} + \tfrac{2}{8}\log_2\tfrac{2}{8}) = 0.811$

- $E(Wind = strong) = -(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{3}) = 1$

On substitution,

$$Gain(S, Wind) = E(S) - I(S, Wind) = 0.940 - (\tfrac{8}{14} * 0.811 + \tfrac{6}{14} * 1) = 0.048$$

(c) Outlook: Here there are 3 possibilities, sunny, overcast and rain.

- $E(Outlook = sunny) = -(\frac{3}{5}\log_2\frac{3}{5} + \frac{2}{5}\log_2\frac{2}{5}) = 0.971$
- $E(Outlook = overcast) = -(1\log_2 1 + 0\log_2 0) = 0$
- $E(Outlook = rainy) = -(\frac{2}{5}\log_2\frac{2}{5} + \frac{3}{5}\log_2\frac{3}{5}) = 0.971$

On substitution,

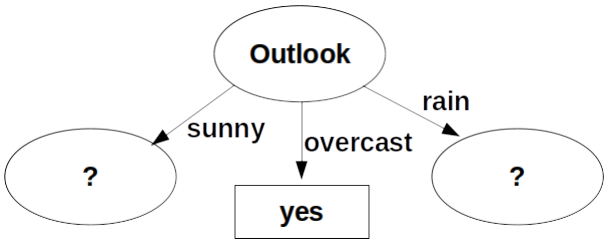
$$Gain(S, Wind) = E(S) - I(S, Wind) = 0.940 - (\tfrac{5}{14} * 0.971 + \tfrac{4}{14} * 0 + \tfrac{5}{14} * 0.971) = 0.246$$

(d) Temperature: Here there are 3 possibilities, hot, cool and mild.

- $E(Temperature = hot) = -(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}) = 1$
- $E(Temperature = cool) = -(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}) = 0.811$
- $E(Temperature = mild) = -(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}) = 0.918$

$$Gain(S, Temperature) = E(S) - I(S, Temperature) =$$
$$0.940 - (\tfrac{4}{14} * 1 + \tfrac{4}{14} * 0.811 + \tfrac{6}{14} * 0.918) = 0.029$$

Out of the 4, Outlook has the highest gain value, so Outlook is chosen as the root node.



2. Now consider each branch of Outlook. First consider overcast. It's entropy is 0 (since all options are yes), so it will end in a leaf node.

3. Next consider sunny. First get the entropy of the node. For sunny, we see that there are 2 yes and 3 no.

$$E(sunny) = -(\tfrac{2}{5}\log_2\tfrac{2}{5} + \tfrac{3}{5}\log_2\tfrac{3}{5}) = 0.971$$

Now consider each attribute, given that the outlook is sunny and find the information gain. For understanding purpose, the complete calculation for information gain of temperature is shown.

(a) Here there are 3 possibilities, hot, cool and mild.

- $E(Temperature = hot) = -(\frac{0}{2}\log_2\frac{0}{2} + \frac{2}{2}\log_2\frac{2}{2}) = 0$
- $E(Temperature = cool) = -(\frac{1}{1}\log_2\frac{1}{1} + \frac{0}{1}\log_2\frac{0}{1}) = 0$
- $E(Temperature = mild) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) = 1$

$Gain(S, Temperature) = E(S) - I(S, Temperature) = 0.971 - (\frac{2}{5} * 0 + \frac{1}{5} * 0 + \frac{2}{5} * 1) = 0.571$

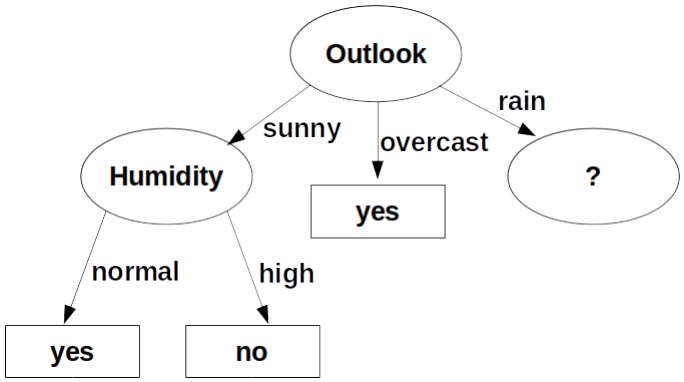
In the same way, information gain is found for the other 2 attributes.

(b) Gain(Humidity)=0.971

(c) Gain(Windy)=0.020

Humidity is selected.

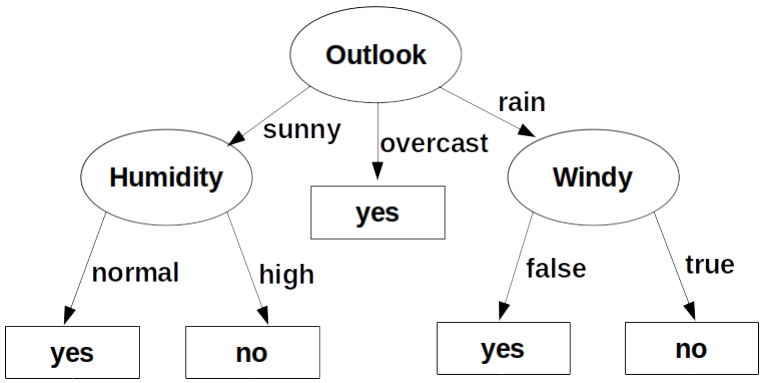
4. Further expand this branch. But it seen that entropy is 0, so the branches end with leaf nodes.



5. Next consider rainy. Check information gain for each of the 3 attributes. It is found that windy has the best information gain.

6. Further expand this branch. Here both branches have entropy 0 and will end in leaf nodes.

7. No further branching is possible so the algorithm stops. The final decision tree is:



1.2 Naive Bayesian Classifier

Read about Bayes theorem, conditional class independence, prior and posterior probabilities. How to handle zero probability scenario (Laplacian Estimator). Give a short pseudo code and trace.

- 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.
- Class conditional 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.
- Prior probability: Prior probability is the probability of an event before new data is collected.
- Posterior probability: Posterior probability is the revised or updated probability of an event occurring after taking into consideration new information.
- Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{8}$$

$$P(c|X) = P(x_1|c) * P(x_2|c) * ... * P(x_d|c) * P(c) \tag{9}$$

  where,

  - $P(c|x)$ is the posterior probability of class (target) given predictor (attribute)
  - $P(c)$ is the 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 (8) is for a single variable x while equation (9) is for multivariable $X = (x_1, ..., x_d)$.

- 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.

  A small-sample correction, or pseudo-count, will be incorporated in every probability estimate. Consequently, no probability will be zero. This is a way of regularizing Naive Bayes, and when the pseudo-count is zero, it is called Laplace smoothing. While in the general case it is often called Lidstone smoothing.

Given an observation $x = (x_1, ..., x_d)$ from a multinomial distribution with N trials and parameter vector $\theta = (\theta_1, ..., \theta_d)$, a smoothed version of the data gives the estimator:

$$\hat{\theta} = \frac{x_i + \alpha}{N + \alpha d} \tag{10}$$

where the pseudocount $\alpha > 0$ is the smoothing parameter ($\alpha = 0$ corresponds to no smoothing). Additive smoothing is a type of shrinkage estimator, as the resulting estimate will be between the empirical estimate $\frac{x_i}{N}$, and the uniform probability $\frac{1}{d}$. Using Laplace's rule of succession, some authors have argued that α should be 1 (in which case the term add-one smoothing is also used), though in practice a smaller value is typically chosen.

Pseudocode:

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.

Trace: Consider the table given below.

Temperature	Weather	Play
hot	sunny	no
hot	sunny	no
hot	overcast	yes
mild	rainy	yes
cool	rainy	yes
cool	rainy	no
cool	overcast	yes
mild	sunny	no
cool	sunny	yes
mild	rainy	yes
mild	sunny	yes
mild	overcast	yes
hot	overcast	yes
mild	rainy	no

Input: If we have an input where the Weather is overcast and the Temperature is mild, then using Naive Bayes classifier we'll decide whether to play golf or not.

Probability of playing: If you consider equation (9), the class c would be Play=yes and conditions x_1 is Weather=overcast and x_2 is Temperature=mild. So, substituting in equation (9) (as it is multivariable):

$$P(yes|overcast, mild) = P(overcast, mild|yes) * P(yes)$$

$$P(overcast, mild|yes) = P(overcast|yes) * P(mild|yes)$$

1. Calculate prior probability:

$$P(yes) = \frac{9}{14} = 0.64$$

2. Calculate class conditional probability for each attribute:

$$P(overcast|yes) = \frac{4}{9} = 0.44$$

$$P(mild|yes) = \frac{4}{9} = 0.44$$

3. Multiply the class conditional probabilities to get likelihood.

$$P(overcast, mild|yes) = 0.44 * 0.44 = 0.1936$$

4. Multiply likelihood and prior probability to get posterior probability.

$$P(yes|overcast, mild) = 0.1936*0.64 = 0.124$$

Probability of not playing: If you consider equation (9), the class c would be Play=no and conditions x_1 is Weather=overcast and x_2 is Temperature=mild. So, substituting in equation (9):

$$P(no|overcast, mild) = P(overcast, mild|no) * P(no)$$

$$P(overcast, mild|no) = P(overcast|no) * P(mild|no)$$

1. Calculate prior probability:

$$P(no) = \frac{5}{14} = 0.36$$

2. Calculate class conditional probability for each attribute:

$$P(overcast|no) = \frac{0}{5} = 0$$

$$P(mild|no) = \frac{2}{5} = 0.4$$

3. Multiply the class conditional probabilities to get likelihood.

$$P(overcast, mild|yes) = 0 * 0.4 = 0$$

4. Multiply likelihood and prior probability to get posterior probability.

$$P(yes|overcast, mild) = 0*0.36 = 0$$

The probability of playing is higher. So you can say here that if the weather is overcast and temperature is mild then players will play the sport.

1.3 Neural Network Classifier (Back Propagation Network)

Understand basic terminologies - topology of a network, input layer, hidden layer, output layer, activation functions, weight, bias. List the strengths and limitations of a neural network. Take a sample network to learn the basics. For instance, learning the behaviour of OR gate, AND gate. Take a network involving atleast one hidden layer and trace the BPN algorithm assuming a target class for one epoch.

Terminologies:

- Topology: The way in which the neurons are connected together determines the topology of the neural network. Sometimes referred to as architecture or paradigm.
- Input layer: The first layer of a neural net, that accepts certain input patterns and generates output values to the succeeding weight matrix.
- Hidden layer: A type of neuron layer that lies between a neural net's input and output layers. Called hidden, because its neuron values are not visible outside the net. The usage of hidden layers extends a neural net's abilities to learn logical operations.
- Output layer: The last layer of a neural net, that produces the output value of the net.
- 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: A pseudo input of a neural net with any value except zero. Its purpose is to generate different inputs for different input patterns given to the net.
- Weight: An element of a weight matrix. A connection between two neurons with a value that is dynamically changed during a neural net's learning process.

Strengths:

- Can handle very large datasets.
- Can be used in a variety of fields including speech, image recognition, image captioning, natural language processing, handwriting recognition, etc.
- Can 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.
- 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 learn: 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.

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.

1.3.1 Logic Gates

To understand the working of neural networks, we'll see the trace for some logic gates.
Consider the Perceptron Rule:

$$y = \begin{cases} 0, & \text{if } \sum_{i=1}^{n} w_i x_i + b \leq 0 \\ 1, & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \end{cases} \tag{11}$$

where x_i is the input, w_i is the weight, b is the bias term, y is the output and there are n input terms.

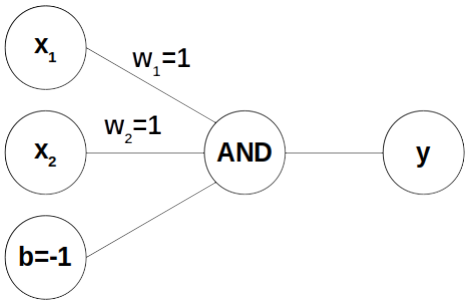
AND Gate: Consider the truth table of AND gate.

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

1. 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) $w_1 * 1 + w_2 * 0 + b \leq 0 \equiv w_1 + b \leq 0$
   (c) $w_1 * 0 + w_2 * 1 + b \leq 0 \equiv w_2 + b \leq 0$
   (d) $w_1 * 1 + w_2 * 1 + b > 0 \equiv w_1 + w_2 + b > 0$
2. Initialise the values. Let $b = -1$ (as b cannot be 0), $w_1 = 1$ and $w_2 = 1$.
3. Now substitute the input values in the perceptron equation and check if the predicted output is equal to the actual output.

x_1	x_2	$\sum_{i=1}^{n} w_i x_i + b$	y_{pred}	y_{actual}
0	0	$1*0+1*0+-1=-1$	0	0
0	1	$1*0+1*1+-1=0$	0	0
1	0	$1*1+1*0+-1=0$	0	0
1	1	$1*1+1*1+-1=1$	1	1

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



OR Gate: Consider the truth table of OR gate.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

1. 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) $w_1 * 1 + w_2 * 1 + b > 0 \equiv w_1 + w_2 + b > 0$
2. Initialise the values. Let $b = -1$, $w_1 = 1$ and $w_2 = 1$.
3. Now substitue 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.

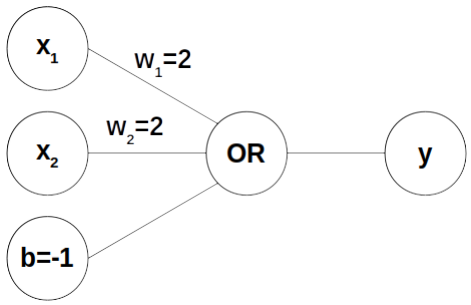
(b) Input 2: $1*0+1*1+-1=0>0$ is false. Output would be 0 which is not equal to the desired output. Update the weight matrix. Since x_1 is 0 (implying $w_1*x_1=0$), w_2 is increased. Now, w_1=1, w_2=2 and b=-1. On substituting: $1*0+2*1+-1=1>0$ is true.

(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

4. So, for w_1=2, w_2=2 and b=-1, all the linear inequalities are satisfied. The final network is:



1.3.2 Hidden Layer Trace

The code for this can be found in trace_bpn.py. If you run this program, the values will be different from what is shown in the report, as the weights are initialised randomly. So, a pdf of the jupyter notebook is included, trace_bpn.pdf, which shows the intermediate values for the given set of weights. The dataset is the file, golf.csv.

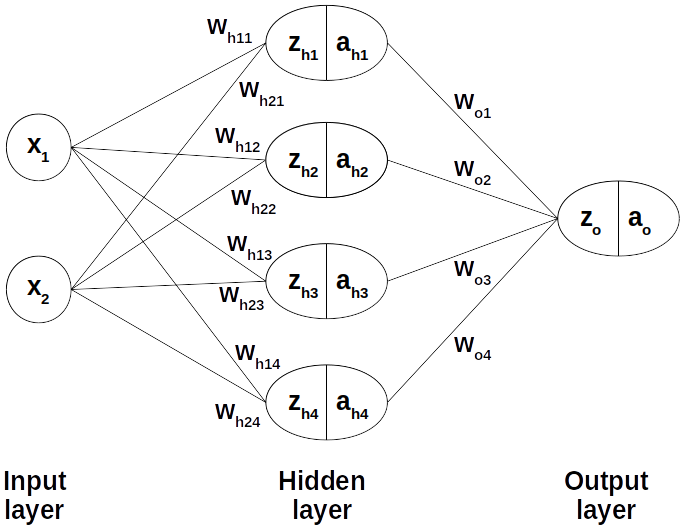
Dataset: We'll consider the dataset given in section 1.2 after doing encoding with normalization (and removing duplicate rows). The dataset is:

Temperature	Weather	Play
0.5	0.0	1.0
1.0	0.5	1.0
0.0	0.5	1.0
0.0	0.5	0.0
0.0	0.0	1.0
1.0	1.0	0.0
0.0	1.0	1.0
1.0	0.5	1.0
1.0	1.0	1.0
1.0	0.0	1.0
0.5	0.0	1.0
1.0	0.5	0.0

Initialisation: There are 2 input nodes (x_1 and x_2), 1 output node which is a binary classifier. Let there be 4 hidden nodes. The weights between input layer and hidden layer (w_h, a 2x4 matrix) and weights between hidden layer and output layer (w_o, a 4x1 matrix) are initialised randomly from a uniform distribution over the range [0,1). We'll also define a learning rate.

$$w_h = \begin{bmatrix} 0.89 & 0.22 & 0.66 & 0.14 \\ 0.44 & 0.19 & 0.87 & 0.43 \end{bmatrix}, \; w_o = \begin{bmatrix} 0.36 \\ 0.25 \\ 0.66 \\ 0.95 \end{bmatrix}, \text{ Learning rate= } lr = 0.5$$

Network Topology:



Feed Forward: In the forward pass we need to do sum of products and then apply the activation function.

1. Perform dot product of input (x, a 13x2 matrix, since there are 13 rows in the dataset) and weights (w_h, a 2x4 matrix). Let this be z_h (a 13x4 matrix).

$$z_h = x \cdot w_h = \begin{bmatrix} 0.88 & 0.30 & 1.20 & 0.50 \\ 0.44 & 0.11 & 0.33 & 0.07 \\ 1.11 & 0.31 & 1.09 & 0.35 \\ 0.22 & 0.09 & 0.43 & 0.21 \\ 0.22 & 0.09 & 0.43 & 0.21 \\ 0 & 0 & 0 & 0 \\ 1.33 & 0.41 & 1.53 & 0.57 \\ 0.44 & 0.19 & 0.87 & 0.43 \\ 1.11 & 0.31 & 1.09 & 0.35 \\ 1.33 & 0.41 & 1.53 & 0.57 \\ 0.89 & 0.22 & 0.66 & 0.14 \\ 0.44 & 0.11 & 0.33 & 0.07 \\ 1.11 & 0.31 & 1.09 & 0.35 \end{bmatrix}$$

2. Apply the activation function on z_h to get a_h. Sigmoid function is used as the activation function.

$$a_h = \frac{1}{1+e^{-z_h}} = \begin{bmatrix} 0.71 & 0.57 & 0.76 & 0.62 \\ 0.61 & 0.52 & 0.58 & 0.51 \\ 0.75 & 0.57 & 0.74 & 0.58 \\ 0.55 & 0.52 & 0.60 & 0.55 \\ 0.55 & 0.52 & 0.60 & 0.55 \\ 0.5 & 0.5 & 0.5 & 0.5 \\ 0.79 & 0.60 & 0.82 & 0.63 \\ 0.60 & 0.54 & 0.70 & 0.60 \\ 0.75 & 0.57 & 1.74 & 0.58 \\ 0.79 & 0.60 & 1.82 & 0.63 \\ 0.70 & 0.55 & 0.65 & 0.53 \\ 0.60 & 0.52 & 0.58 & 0.51 \\ 0.75 & 0.57 & 0.74 & 0.58 \end{bmatrix}$$

3. Now perform dot product between a_h(a 13x4 matrix) and w_o(a 4x1 matrix) to get z_o(a 13x1 matrix).

$$z_o = a_h \cdot w_o = \begin{bmatrix} 1.50 \\ 1.23 \\ 1.47 \\ 1.26 \\ 1.26 \\ 1.11 \\ 1.59 \\ 1.40 \\ 1.47 \\ 1.59 \\ 1.34 \\ 1.23 \\ 1.47 \end{bmatrix}$$

4. Apply the activation function on z_o to get a_o.

$$a_o = \frac{1}{1+e^{-z_o}} = \begin{bmatrix} 0.81 \\ 0.77 \\ 0.81 \\ 0.77 \\ 0.77 \\ 0.75 \\ 0.83 \\ 0.80 \\ 0.81 \\ 0.83 \\ 0.79 \\ 0.77 \\ 0.81 \end{bmatrix}$$

Back Propagation: Here, we need to find the cost and update the weights to minimise this cost.

1. First find the error cost. We will take mean squared error as our loss function.
   $cost = \frac{1}{n}\sum_{i=1}^{n}(a_o - labels)^2 = 1.51$

2. Gradient descent will be used to minimise the cost function. First $\frac{dcost}{dw_o}$ will be found, which will tell how much w_o should be changed to minimise the cost.

   $$\frac{dcost}{dw_o} = \frac{dcost}{da_o} * \frac{da_o}{dz_o} * \frac{dz_o}{dw_o} = (a_o - labels) * sigmoid(z_o) * (1 - sigmoid(z_o)) * a_h = \begin{bmatrix} 0.14 \\ 0.11 \\ 0.16 \\ 0.12 \end{bmatrix}$$

3. Now we need to find $\frac{dcost}{dw_h}$ to propagate the error to the input layer and it will tell how much w_h should be changed to minimise the cost.
   $\frac{dcost}{dw_h} = \frac{dcost}{da_h} * \frac{da_h}{dz_h} * \frac{dz_h}{dw_h}$

   - $\frac{dcost}{da_h} = \frac{dcost}{da_o} * \frac{da_o}{dz_o} * \frac{dz_o}{da_h} = (a_o - labels) * sigmoid(z_o) * (1 - sigmoid(z_o)) * w_o$
   - $\frac{da_h}{dz_h} = sigmoid(z_h) * (1 - sigmoid(z_h))$
   - $\frac{dz_h}{dw_h} = x$

   On evaluation, (please refer to the pdf titled trace_bpn to see step by step output values)

$$\frac{dcost}{dw_h} = \begin{bmatrix} 0.0087 & 0.0088 & 0.0130 & 0.0320 \\ 0.0187 & 0.0162 & 0.0306 & 0.0595 \end{bmatrix}$$

4. Final step is to update the weight matrices.

$$w_h = w_h - lr * \frac{dcost}{dw_h} = \begin{bmatrix} 0.88 & 0.21 & 0.65 & 0.12 \\ 0.43 & 0.18 & 0.85 & 0.40 \end{bmatrix}$$

$$w_o = w_o - lr * \frac{dcost}{dw_o} = \begin{bmatrix} 0.29 \\ 0.19 \\ 0.57 \\ 0.88 \end{bmatrix}$$

This completes 1 epoch. For simplicity only a few records have been taken but normally neural network requires a very large dataset.

1.4 Nearest Neighbor Classifier

Understand the algorithm/working and give the pseudocode and the trace.

Introduction: K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g: distance functions). The KNN algorithm assumes that similar things exist in close proximity.

Pseudocode: The algorithm for KNN or nearest neighbor classification is:

Input: Training data samples. Let p be an unknown point.

Output: Class label of point p.

1. Initialize k to your chosen number of neighbors
2. For each data point x_i in the training sample, calculate the distance (using any suitable distance measure) of the point p from that point x_i denoted as $d(x_i, p)$.
3. Sort the training points in increasing order of distance.
4. Select the first k points from this list.
5. Return the class label which occurs the maximum times among these k points.

Trace : Consider a very simple example with only 1 feature, Age and the output is, 'Likes Pineapple'.

Age	22	23	21	18	19	25	27	29	31	45
Likes Pineapple	yes	yes	yes	yes	yes	no	no	no	no	no

Consider an input point p, which has Age 33.

1. Let k=3.

2. 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
Likes Pineapple	yes	yes	yes	yes	yes	no	no	no	no	no
Distance from p	11	10	12	15	14	8	6	4	2	12

3. After sorting, the distance values are:

Age	31	29	27	25	23	22	21	45	19	18
Likes Pineapple	no	no	no	no	yes	yes	yes	no	yes	yes
Distance	2	4	6	8	10	11	12	12	14	15

4. The first k=3 points are:

Age	31	29	27
Likes Pineapple	no	no	no

5. 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 does not like pineapple.

2 Genetic Algorithm

Understand the working of a simple genetic algorithm involving operators of selection, cross-over, mutation. Apply these operators to an optimisation function such as max $f(x) = x^3 - 2x^2 + x$ within a range of (0,31).

Working: Genetic algorithms are based on the concept of natural selection. They try to find the optimal solution for a problem.

1. This is done by starting with a set population of chromosomes where each chromosome is a string of 0's and 1's.

2. This population is evaluated using a fitness function to get the fitness score for each individual.

3. In order to get the next population:

   (a) Selection: Select parents. There are various techniques to do this like roulette wheel of selection, elitism, etc. An individual having higher fitness is more likely to be selected.

   (b) Perform crossover for selected parents. The crossover point is selected at random. Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. This process of selection and crossover are repeated till the size of the new population is equal to the size of the old population.

   (c) On some of the offspring mutation of one gene (bit) is performed based on a low mutation probability. Mutation occurs to maintain diversity within the population and prevent premature convergence.

4. Repeat this process of selection, crossover and mutation until a termination criteria is satisfied or until convergence.

Trace: The problem is to find x such that $f(x) = x^3 - 2x^2 + x$ is maximised, where $x \in [0, 31]$. Python function, random.randint() was used to generate random numbers wherever mentioned.

Let's assume the population size n=6.

1. The population is initialised randomly and their fitness score is obtained. To get the fitness score, the integer value of the chromosome is substituted in the function f(x).

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]

2. 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

3. 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

4. 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]

5. Offspring are found.

S no	Parent 1	Parent 2	Crossover point	Offspring1	Offspring 2
1	59=C_4=10110	20=C_1=11111	2	10111	11110
2	11=C_1=11111	38=C_2=10110	4	11110	10111
3	93=C_6=11110	69=C_5=10110	3	11110	10110

6. Mutation is done.

S no	Chromosome	Mutation point	Output	Decoded Integer
1	10111	-	10111	23
2	11110	5	11111	31
3	11110	-	11110	30
4	10111	-	10111	23
5	11110	-	11110	30
6	10110	2	11110	30

7. This procedure should be repeated until the new population is not significantly different from the old population. The population now is not exactly matching with the previous population but there are

many repeating values and they are similar to the values obtained in the previous population. So we shall stop the algorithm here, although ideally it should be run for more iterations till there is convergence.

S no	Chromosome	Decoded Integer	Fitness Score
1	10111	23	11132
2	11111	31	27900
3	11110	30	25230
4	10111	23	11132
5	11110	30	25230
6	11110	30	25230

In the current population we can see that the chromosome 11111 or the value 31 will maximise the given function $f(x)$.

3 Bucket Brigade Classifier

Understand the working of Bucket Brigade Classifier[BBC]. Concepts to focus - Credit Apportionment, Fitness and Strength of classifier. Understand the trace given in the Goldberg textbook.

Working: Points on Bucket Brigade Classifier as discussed in Genetic Algorithms in Search, Optimization and Machine Learning, David E. Goldberg:

- Method for ranking individual classifiers.
- It is an economy where information is bought and sold by classifiers.
- Classifiers are like the middlemen from infomation manufacturer (environment) to information consumer (effector)
- This economy has 2 main components namely auction and clearinghouse.
- Each classifier also has 2 components. One is the classifier condition and another is a message that it will send.
- A message needs to match with the classifier condition for the classifier to become active. Then it may participate in the auction.
- The activated classifier must make a bid and then post it's message.
- Strength S: The net worth of a classifier

- A classifier must clear their payment (bid) through the clearinghouse. The clearinghouse will distribute the total amount collected and distribute it among the classifiers who sent a message which resulted in a match.
- A classifier may be subject to tax or reward.
- BBC helps ensure the formation of an appropriately sized subpopulation of rules.
- An equation can be written for the ith classifier's strength at time t+1.

$$S_i(t+1) = S_i(t) - P_i(t) - T_i(t) + R_i(t) \tag{12}$$

  where,

  - S is strength
  - P is payment (bid)
  - T is tax (incase it is subjected)
  - R is receipt from previous activity (bid made by another classifier) or reward from environment

- The bid made is proportional to a classifier's strength at time t.

$$B_i = C_{bid} S_i \tag{13}$$

- A random noise can be introduced to prevent status quo. This will give the effective bid.

$$EB_i = B_i + N(\sigma_{bid}) \tag{14}$$

  where N is a function of σ_{bid}

- Tax is collected proportional to classifiers strength.

$$T_i = C_{tax} S_i \tag{15}$$

- The algorithm ends when there are no more matches and it cannot proceed further.
- Auction winners are chosen by selecting the k best classifiers.

Trace: C_{bid}=0.1 and all the classifiers are initialised with strength=200.

t=0:

1. Environmental message=0111 is posted. Out of the 4 classifiers, classifier 1 matches (check the message with a classifier's condition). Classifier 1 bids 20 units (=200*0.1). Classifier 1 strength reduces by 20 units.
2. The classifier's bid is paid to the party responsible for its activation. Here since the Environment activated classifier 1, the Environment gets the bid and it's strength increases by 20 units.
3. The change in strength is seen in the next time step.
4. In subsequent time steps, the activated classifier makes it's payment to previously activated classifiers.

S no	Classifier(Condition:Message)	Strength	Messages	Match	Bid
1	01##:0000	200		E	20
2	00#0:1100	200			
3	11##:1000	200			
4	##00:0001	200			
	Environment	0	0111		

t=1:

1. Now classifier 1 posts its message. This matches with classifier 2 and classifier 4. Each classifier bids 20.
2. Since classifier 1 was previously activated, it will receive the total bid amount of 40 (look at the match column to understand which classifier should receive the bid amount).

S no	Classifier(Condition:Message)	Strength	Messages	Match	Bid
1	01##:0000	180	0000		
2	00#0:1100	200		1	20
3	11##:1000	200			
4	##00:0001	200		1	20
	Environment	20			

t=2:

1. Now classifiers 2 and 4 post their messages. Classifier 2 message matches with classifier 3 and 4. Classifier 4 message doesn't match with any of the classifiers.
2. So, classifier 3 and 4 make a bid proportional to their strength. Classifier 3 strength decreases by 20. Classifier 4 strength decreases by 18.

3. Classifier 2 and 4 were previously activated, but match was only for classifier 2 so it will get total bid amount of 38 units. Classifier 2 strength increases by 38.

S no	Classifier(Condition:Message)	Strength	Messages	Match	Bid
1	01##:0000	220			
2	00#0:1100	180	1100		
3	11##:1000	200		2	20
4	##00:0001	180	0001	2	18
	Environment	20			

t=3:

1. Since 3 and 4 matched, they will send their messages. Classifier 3 message matches with classifier 4.
2. So classifier 4 is activated and will send a bid of 16. Strength of classifier 4 decreases by 16.
3. The bid is sent to the matched classifier. Strength of classifier 3 increases by 16.

S no	Classifier(Condition:Message)	Strength	Messages	Match	Bid
1	01##:0000	220			
2	00#0:1100	218			
3	11##:1000	180	1000		
4	##00:0001	162	0001	3	16
	Environment	20			

t=4: Since classifier 4 had matched in the previous time step, it will send it's message. There are no matches, so we stop. (In the textbook, at this time step it says that classifier 4 has strength 156 but this should be 146 as 16 was deducted from 162.)

S no	Classifier(Condition:Message)	Strength	Messages	Match	Bid
1	01##:0000	220			
2	00#0:1100	218			
3	11##:1000	196			
4	##00:0001	146	0001		
	Environment	20			

t=5: A reward of 50 units comes into the system and is paid to the last active classifier which was classifier 4. Closing strengths are:

S no	Classifier(Condition:Message)	Strength
1	01##:0000	220
2	00#0:1100	218
3	11##:1000	196
4	##00:0001	196
	Environment	20

4 BBC in Decision Tree

Explore the scope of applying the bucket brigade classifier in the context of decision tree induction. This question doesn't require the trace. You will have to discuss how the optimization, quicker convergence supported by a BBC can be useful for decision tree induction.

- The main problem in decision tree is deciding the splitting criterion. For this, information gain, gain ratio or gini index is used to decide which split would be best.
- It will look at all possibilities and choose the suitable one based on the heuristic function. This is quite time consuming especially if there are a lot of split possibilities.
- Instead of evaluating each possible split, we could use a combination of genetic algorithm and bucket brigade algorithm.
- Genetic algorithm would generate possible splits where the algorithm would aim to maximize information gain or minimize gini index (depending on which one you're using). The result would be a set of the best classifier conditions i.e encoded splitting criterion.
- Since bucket brigade is used to evaluate classifier systems, we could input the different classifiers (performing a split is nothing but classification) from the genetic algorithm and get the best possible split.
- Here, messages would be different input values which would be matched with the classifier condition (encoded splitting criterion).
- Since BBC has quicker convergence, it will help in choosing the best splitting criterion faster.
- The main lag would come from the genetic algorithm.
- Although, to improve the speed, this step could be removed completely and BBC could directly be applied on all the different possible splitting criterion (classifiers).

- In this case, no heurisitc function would be used (as compared to genetic algorithm). Instead messages would directly be matched with each classifier condition.
- In the end the best splitting criterion would emerge as the one which has the maximum strength.
- The computational complexity is removed as we are not using information gain, etc.
- Time complexity would depend on the number of possible splits.

5 Performance measures

Understand confusion matrix and various performance measures associated with classification tasks such as accuracy, sensitivity, specificity, precision, recall, F1 score etc. Consider a sample binary classifier results (TP, FP, TN, FN) and compute various measures.

- Confusion matrix: A table which summarises the number of correct and incorrect predictions with count values by each class. Actual classes are veritically listed while predicted classes are horizontally listed. Essentially it gives n(A/B) i.e the number of cases that class A was predicted when the actual class was B. If there are 2 classes (yes and no i.e positive and negative) then we can define:
  - 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.
- Accuracy: Percent of correct predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{16}$$

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

$$Sensitivity = \frac{TP}{TP + FN} \tag{17}$$

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

$$Specificity = \frac{TN}{TN + FP} \tag{18}$$

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

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

- Recall: Fraction of positives that were correctly identified (recall and sensitivity are the same).

$$Recall = \frac{TP}{TP + FN} \tag{20}$$

- 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.

$$f1score = \frac{2 * Recall * Precision}{Recall + Precision} \tag{21}$$

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. $Accuracy = \frac{3+4}{3+4+1+2} = \frac{7}{10} = 0.7$
2. $Sensitivity = Recall = \frac{3}{3+2} = \frac{3}{5} = 0.6$
3. $Specificity = \frac{4}{4+1} = \frac{4}{5} = 0.8$
4. $Precision = \frac{3}{3+1} = \frac{3}{4} = 0.75$
5. $f1 - score = \frac{2 * 0.6 * 0.75}{0.6 + 0.75} = 0.667$

6 Clustering Algorithms

To make obtaining the trace faster, small programs were written. This can be found in trace_clustering.py. These programs assisted in getting the values for the traces. All the traces for the clustering algorithms are included in this file, each one commented out separately. Uncomment one by one and substitute the required values to check the trace.

6.1 K Means

Understand the working of k-means clustering algorithm. Give a pseudo code for the same and trace it for a sample dataset of your choice, clearly showing the centroid updates.

Working: 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 intial partitioning.

Pseudocode:

1. Partition objects into k nonempty subsets (or) clusters.
2. Compute the centroids of the clusters. The centroid is the mean of all points belonging to a cluster.
3. Assign each object to the nearest cluster by taking distance between the object and the centroid of a cluster. Distance may be calculated using any suitable distance measure.
4. Go back to step 2 and repeat until there is no change in clusters (convergence).

Trace: Consider the table given below.

Number	1	2	3	4	5	6	7	8	9	10
x	5	3	7	3	3	4	4	5	7	8
y	1	4	3	8	6	7	5	5	5	5

Iteration 1:

1. Consider k=2. Let the datapoints be assigned as:

Class	Datapoints	Centroid (k_i)
C_1	1,2,3,4,5	$k_1 = (\frac{5+3+7+3+3}{5}, \frac{1+4+3+8+6}{5})$=(4.2,4.4)
C_2	6,7,8,9,10	$k_2 = (\frac{4+4+5+7+8}{5}, \frac{7+5+5+5+5}{5})$=(5.6,5.4)

2. Euclidean distance is used to find the distance between a point and a centroid. The class which has less distance is assigned to a data point.

Number	k_1 distance	k_2 distance	Assigned class
1	3.49	4.44	C_1
2	1.26	2.95	C_1
3	3.13	2.78	C_2
4	3.79	3.68	C_2
5	2.00	2.67	C_1
6	3.61	2.26	C_2
7	0.63	1.65	C_1
8	1.00	0.72	C_2
9	2.86	1.46	C_2
10	3.85	2.43	C_2

Iteration 2:

1. Get the new centers.

Class	Datapoints	Centroid (k_i)
C_1	1,2,5,7	k_1=(3.75,4.0)
C_2	3,4,6,8,9,10	k_2=(5.67,5.5)

2. Get the distance of each point with the new centers.

Number	k_1 distance	k_2 distance	Assigned class
1	3.25	4.55	C_1
2	0.75	3.06	C_1
3	3.40	2.83	C_2
4	4.07	3.66	C_2
5	2.14	2.71	C_1
6	3.01	2.24	C_2
7	1.03	1.74	C_1
8	1.60	0.83	C_2
9	3.40	1.42	C_2
10	4.37	2.39	C_2

The clusters don't change, so we stop here.

6.2 K Medoids

Understand the working of k-medoids clustering algorithm. Give a pseudo code for the same and trace it for the sample dataset used in section 6.1 clearly showing the centroid updates.

Working: K Medoids algorithm, like K Means, is also partitional, which involves breaking the dataset into groups. K medoids attempts to minimize the sum of dissimilarities between objects labeled to be in a cluster and one of the objects designated as the representative of that cluster. These representatives are called medoids. In contrast to the K Means algorithm that the centroids are central (average positions that might not be data points in the set), K Medoids chooses medoids from the data points in the set. It may give a different result compared to K Means.

Pseudocode:

1. 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 :

$$c = \sum_{C_i} \sum_{P_i \epsilon C_i} |P_i - C_i| \tag{22}$$

   which is the total sum of the distance between the medoid and the non-medoid objects belonging to each cluster.

   While the cost decreases, for each medoid m, for each data point o which is not a medoid:

   (a) Swap m and o, associate each data point to the closest medoid, recompute the cost.

   (b) If the total cost is more than that in the previous step, undo the swap.

Trace: Consider the table from Section 6.1 and let k=2.

Number	1	2	3	4	5	6	7	8	9	10
x	5	3	7	3	3	4	4	5	7	8
y	1	4	3	8	6	7	5	5	5	5

1. Initialise 2 random points as medoids: k_1=Point 2=(3,4); k_2=Point 10=(8,5).
2. Using Euclidean distance, assign data points to clusters with medoids as centers.

Number	k_1 distance	k_2 distance	Assigned class
1	3.61	5.00	C_1
2	0	5.10	-
3	4.12	2.24	C_2
4	4.00	5.83	C_1
5	2.00	5.10	C_1
6	3.16	4.47	C_1
7	1.41	4	C_1
8	2.24	3.00	C_1
9	4.12	1.00	C_2
10	5.10	0	-

The resultant clusters are (bold data points are the medoids):

C_1=1,**2**,4,5,6,7,8

C_2=3,9,**10**

Cost=sum of all distances within cluster

$=d(1,2)+d(4,2)+d(5,2)+d(6,2)+d(7,2)+d(8,2)+d(3,10)+d(9,10)$

=3.61+2.24+4+2+3.16+1.41+2.24+1

=19.65

3. Now let's try swapping data point 1 (non-medoid) with 2 (medoid). Now, k_1=Point 1=(5,1); k_2=Point 10=(8,5).

Number	k_1 distance	k_2 distance	Assigned class
1	0	5.00	-
2	3.61	5.10	C_1
3	2.83	2.24	C_2
4	7.28	5.83	C_2
5	5.39	5.10	C_2
6	6.08	4.47	C_2
7	4.12	4	C_2
8	4.00	3.00	C_2
9	4.47	1.00	C_2
10	5.00	0	-

The resultant clusters are (bold data points are the medoids):

C_1=**1**,2

C_2=3,4,5,6,7,8,9,**10**

Cost=sum of all distances within clusters=29.24

This is more than the cost before swapping. So, we cannot swap points 1 and 2.

4. Now let's try swapping data point 4 (non-medoid) with 2 (medoid). Now, k_1=Point 4=(3,8); k_2=Point 10=(8,5).

Number	k_1 distance	k_2 distance	Assigned class
1	7.28	5.00	C_2
2	4.00	5.10	C_1
3	6.40	2.24	C_2
4	0	5.83	-
5	2.00	5.10	C_1
6	1.41	4.47	C_1
7	3.16	4	C_1
8	3.61	3.00	C_2
9	5.00	1.00	C_2
10	5.83	0	-

The resultant clusters are (bold data points are the medoids):

C_1=2,**4**,5,6,7

C_2=1,3,8,9,**10**

Cost=sum of all distances within clusters=21.81

This is more than the cost before swapping. So, we cannot swap points 4 and 2.

5. Now let's try swapping data point 5 (non-medoid) with 2 (medoid). Now, k_1=Point 5=(3,6); k_2=Point 10=(8,5).

Number	k_1 distance	k_2 distance	Assigned class
1	5.39	5.00	C_2
2	2.00	5.10	C_1
3	5.00	2.24	C_2
4	2.00	5.83	C_1
5	0	5.10	-
6	1.41	4.47	C_1
7	1.41	4	C_1
8	2.24	3.00	C_1
9	4.12	1.00	C_2
10	5.10	0	-

The resultant clusters are (bold data points are the medoids):

C_1=2,4,**5**,6,7,8

C_2=1,3,9,**10**

Cost=sum of all distances within clusters=17.30

This is less than the cost before swapping(=19.65). So, we can swap points 5 and 2. Point 5 is updated as the medoid for cluster 1.

6. Now let's try swapping data point 6 (non-medoid) with 5 (medoid). Now, k_1=Point 6=(4,7); k_2=Point 10=(8,5).

Number	k_1 distance	k_2 distance	Assigned class
1	6.08	5.00	C_2
2	3.16	5.10	C_1
3	5.00	2.24	C_2
4	1.41	5.83	C_1
5	1.41	5.10	C_1
6	0	4.47	-
7	2.00	4	C_1
8	2.24	3.00	C_1
9	3.61	1.00	C_2
10	4.47	0	-

The resultant clusters are (bold data points are the medoids):

C_1=2,4,5,**6**,7,8

C_2=1,3,9,**10**

Cost=sum of all distances within clusters=18.46

This is more than the cost before swapping. So, we cannot swap points 6 and 5.

7. Now let's try swapping data point 7 (non-medoid) with 5 (medoid). Now, k_1=Point 7=(4,5); k_2=Point 10=(8,5).

Number	k_1 distance	k_2 distance	Assigned class
1	4.12	5.00	C_1
2	1.41	5.10	C_1
3	3.61	2.24	C_2
4	3.16	5.83	C_1
5	1.41	5.10	C_1
6	2.00	4.47	C_1
7	0	4	-
8	1.00	3.00	C_1
9	3.00	1.00	C_2
10	4.00	0	-

The resultant clusters are (bold data points are the medoids):

C_1=1,2,4,5,6,**7**,8

C_2=3,9,**10**

Cost=sum of all distances within clusters=16.35

This is less than the cost before swapping (=17.30). So, we can swap points 7 and 5. Point 7 is updated as the medoid of cluster 1.

8. Now let's try swapping data point 8 (non-medoid) with 7 (medoid). Now, k_1=Point 8=(5,5); k_2=Point 10=(8,5).

Number	k_1 distance	k_2 distance	Assigned class
1	4.00	5.00	C_1
2	2.24	5.10	C_1
3	2.83	2.24	C_2
4	3.61	5.83	C_1
5	2.24	5.10	C_1
6	2.24	4.47	C_1
7	1.00	4	C_1
8	0	3.00	-
9	2.00	1.00	C_2
10	3.00	0	-

The resultant clusters are (bold data points are the medoids):

C_1=1,2,4,5,6,7,**8**

C_2=3,9,**10**

Cost=sum of all distances within clusters=18.55

This is more than the cost before swapping. So, we cannot swap points 8 and 7.

9. We have checked all the data points for cluster 1. Now we'll check for cluster 2. Let's try swapping data point 3 (non-medoid) with 10 (medoid). Now, k_1=Point 7=(4,5); k_2=Point 3=(7,3).

Number	k_1 distance	k_2 distance	Assigned class
1	4.12	2.83	C_2
2	1.41	4.12	C_1
3	3.61	0	-
4	3.16	6.40	C_1
5	1.41	5.00	C_1
6	2.00	5.00	C_1
7	0	3.61	-
8	1.00	2.83	C_1
9	3.00	2.00	C_2
10	4.00	2.24	C_2

The resultant clusters are (bold data points are the medoids):

C_1=2,4,5,6,**7**,8

C_2=1,**3**,9,10

Cost=sum of all distances within clusters=16.06

This is less than the cost before swapping(=16.35). So, we can swap points 3 and 10. Point 3 is updated as the medoid of cluster 2.

10. The procedure is repeated by swapping all the non medoid points with the current medoids (i.e k_1=Point 7=(4,5); k_2=Point 3=(7,3)), from the clusters output of the previous step.

k_1	k_2	Swapped	Cost
7	1	3 with 1	18.82
7	9	3 with 9	16.11
7	10	3 with 10	16.35
2	3	7 with 2	19.88
4	3	7 with 4	20.47
5	3	7 with 5	16.13
6	3	7 with 6	17.29
8	3	7 with 8	18.38

It can be seen that swapping doesn't improve the cost further. So, the algorithm ends with final clusters as (bold data points are the medoids):

C_1=2,4,5,6,**7**,8

C_2=1,**3**,9,10

6.3 Hierarchical

Understand the working of hierarchical clustering algorithm- Agglomerative, Divisive and trace them for the dataset used in section 6.1. You may trace the algorithm for both the approaches and use the dendrogram to represent the clustering process pictorially as well.

Definition: 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.

6.3.1 Agglomerative

In agglomerative or bottom-up clustering method we assign each observation to it's own cluster. Then, compute the similarity (e.g., distance) between each of the clusters and join the two most similar clusters.

This is repeated till there is only one cluster left.

Pseudocode: Given a dataset $(d_1, d_2, d_3, ...d_N)$ of size N.

1. Initially each data point is a singleton cluster.
2. 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.
3. Merge the 2 clusters having minimum distance. The new center is the average of the 2 old centers.
4. Repeat from step 2 till only a single cluster remains.

Trace: Consider the dataset from section 6.1.

Number	0	1	2	3	4	5	6	7	8	9
x	5	3	7	3	3	4	4	5	7	8
y	1	4	3	8	6	7	5	5	5	5

1. Find the distance matrix for the 10 clusters (data points). Euclidean distance is used.

	0	1	2	3	4	5	6	7	8	9
0	0	3.6	2.82	7.28	5.38	6.08	4.12	4	4.47	5
1	3.6	0	4.12	4	2	3.16	1.41	2.23	4.12	5.09
2	2.82	3.12	0	6.4	5	5	3.6	2.82	2	2.23
3	7.28	4	6.4	0	2	1.41	3.16	3.6	5	5.83
4	5.38	2	5	2	0	1.41	1.41	2.23	4.12	5.09
5	6.08	3.16	5	1.41	1.41	0	2	2.23	3.6	4.47
6	4.12	1.41	3.6	3.16	1.41	2	0	**1**	3	4
7	4	2.23	2.82	3.6	2.23	2.23	**1**	0	2	3
8	4.47	4.12	2	5	4.12	3.6	3	2	0	1
9	5	5.09	2.23	5.83	5.09	4.47	4	3	1	0

2. Closest points are 6 and 7. New cluster is (6,7) with center (4.5,5).

Number	0	1	2	3	4	5	(6,7)	8	9
x	5	3	7	3	3	4	4.5	7	8
y	1	4	3	8	6	7	5	5	5

Distance matrix:

	0	1	2	3	4	5	(6,7)	8	9
0	0	3.6	2.82	7.28	5.38	6.08	4.03	4.47	5
1	3.6	0	4.12	4	2	3.16	1.8	4.12	5.09
2	2.82	3.12	0	6.4	5	5	3.2	2	2.23
3	7.28	4	6.4	0	2	1.41	3.35	5	5.83
4	5.38	2	5	2	0	1.41	1.8	4.12	5.09
5	6.08	3.16	5	1.41	1.41	0	2.06	3.6	4.47
(6,7)	4.03	1.8	3.2	3.35	1.8	2.06	0	2.5	3.5
8	4.47	4.12	2	5	4.12	3.6	2.5	0	**1**
9	5	5.09	2.23	5.83	5.09	4.47	3.5	**1**	0

3. Closest clusters are 8 and 9. New cluster is (8,9) with center (7.5,5).

Number	0	1	2	3	4	5	(6,7)	(8,9)
x	5	3	7	3	3	4	4.5	7.5
y	1	4	3	8	6	7	5	5

Distance matrix:

	0	1	2	3	4	5	(6,7)	(8,9)
0	0	3.6	2.82	7.28	5.38	6.08	4.03	4.71
1	3.6	0	4.12	4	2	3.16	1.8	4.60
2	2.82	3.12	0	6.4	5	5	3.2	2.06
3	7.28	4	6.4	0	2	1.41	3.35	5.4
4	5.38	2	5	2	0	**1.41**	1.8	4.6
5	6.08	3.16	5	1.41	**1.41**	0	2.06	4.03
(6,7)	4.03	1.8	3.2	3.35	1.8	2.06	0	3
(8,9)	4.71	4.60	2.06	5.4	4.6	4.03	3	0

4. Closest clusters are 4 and 5. New cluster is (4,5) with center (3.5,6.5).

Number	0	1	2	3	(4,5)	(6,7)	(8,9)
x	5	3	7	3	3.5	4.5	7.5
y	1	4	3	8	6.5	5	5

Distance matrix:

	0	1	2	3	(4,5)	(6,7)	(8,9)
0	0	3.6	2.82	7.28	5.7	4.03	4.71
1	3.6	0	4.12	4	2.54	1.8	4.60
2	2.82	3.12	0	6.4	4.94	3.2	2.06
3	7.28	4	6.4	0	**1.58**	3.35	5.4
(4,5)	5.7	2.54	4.94	**1.58**	0	1.8	4.27
(6,7)	4.03	1.8	3.2	3.35	1.8	0	3
(8,9)	4.71	4.60	2.06	5.4	4.27	3	0

5. Closest clusters are 3 and (4,5). New cluster is (3,(4,5)) with center (3.25,7.25).

Number	0	1	2	(3,(4,5))	(6,7)	(8,9)
x	5	3	7	3.25	4.5	7.5
y	1	4	3	7.25	5	5

Distance matrix:

	0	1	2	(3,(4,5))	(6,7)	(8,9)
0	0	3.6	2.82	6.49	4.03	4.71
1	3.6	0	4.12	3.25	**1.8**	4.60
2	2.82	3.12	0	5.66	3.2	2.06
(3,(4,5))	6.49	3.25	5.66	0	2.57	4.8
(6,7)	4.03	**1.8**	3.2	2.57	0	3
(8,9)	4.71	4.60	2.06	4.8	3	0

6. Closest clusters are 1 and (6,7). New cluster is (1,(6,7)) with center (3.75,4.5).

Number	0	(1,(6,7))	2	(3,(4,5))	(8,9)
x	5	3.75	7	3.25	7.5
y	1	4.5	3	7.25	5

Distance matrix:

	0	(1,(6,7))	2	(3,(4,5))	(8,9)
0	0	3.71	2.82	6.49	4.71
(1,(6,7))	3.71	0	3.57	2.79	3.78
2	2.82	3.57	0	5.66	**2.06**
(3,(4,5))	6.49	2.79	5.66	0	4.8
(8,9)	4.71	3.78	**2.06**	4.8	0

7. Closest clusters are 2 and (8,9). New cluster is (2,(8,9)) with center (7.25,4).

Number	0	(1,(6,7))	(3,(4,5))	(2,(8,9))
x	5	3.75	3.25	7.25
y	1	4.5	7.25	4

Distance matrix:

	0	(1,(6,7))	(3,(4,5))	(2,(8,9))
0	0	3.71	6.49	3.75
(1,(6,7))	3.71	0	**2.79**	3.53
(3,(4,5))	6.49	**2.79**	0	5.15
(2,(8,9))	3.75	3.53	5.15	0

8. Closest clusters are (1,(6,7)) and (3,(4,5)). New cluster is ((1,(6,7)),(3,(4,5))) with center (3.5,5.875).

Number	0	((1,(6,7)),(3,(4,5)))	(2,(8,9))
x	5	3.5	7.25
y	1	5.875	4

Distance matrix:

	0	((1,(6,7)),(3,(4,5)))	(2,(8,9))
0	0	5.1	**3.75**
((1,(6,7)),(3,(4,5)))	5.1	0	4.19
(2,(8,9))	**3.75**	4.19	0

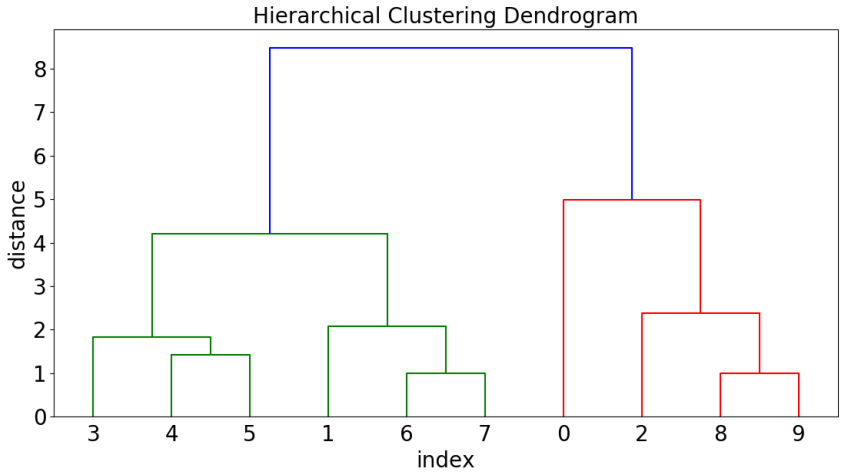
9. Closest clusters are 0 and (2,(8,9)). New cluster is (0,(2,(8,9))) with center (6.125,2.5).

Number	((1,(6,7)),(3,(4,5)))	(0,(2,(8,9)))
x	3.5	6.125
y	5.875	2.5

Distance matrix:

	((1,(6,7)),(3,(4,5)))	(0,(2,(8,9)))
((1,(6,7)),(3,(4,5)))	0	4.27
(0,(2,(8,9)))	4.27	0

10. Merge ((1,(6,7)),(3,(4,5))) and (0,(2,(8,9))). New center will be (4.8125,4.1875). The cluster hierarchy is represented by the dendrogram:



6.3.2 Divisive

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.

Pseudocode:

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. We start with the dataset given in Section 6.1.

Number	0	1	2	3	4	5	6	7	8	9
x	5	3	7	3	3	4	4	5	7	8
y	1	4	3	8	6	7	5	5	5	5

   Initially all the datapoints are a single cluster (0,1,2,3,4,5,6,7,8,9).

2. K Means algorithm is applied (using python package KMeans). This results in clusters:
   C_1=(1,3,4,5,6,7)
   C_2=(0,2,8,9)

3. C_1 is split further using k means. It results in:

   C_{11}=(1,6,7)

   C_{12}=(3,4,5)

4. C_{11} is split further resulting in:

   C_{111}=(1)

   C_{112}=(6,7)

5. This branch is finished, so we split C_{12}.

   C_{121}=(3)

   C_{122}=(4,5)

6. C_1 branch is finished, so now we'll split cluster C_2. It results in:

   C_{21}=(0)

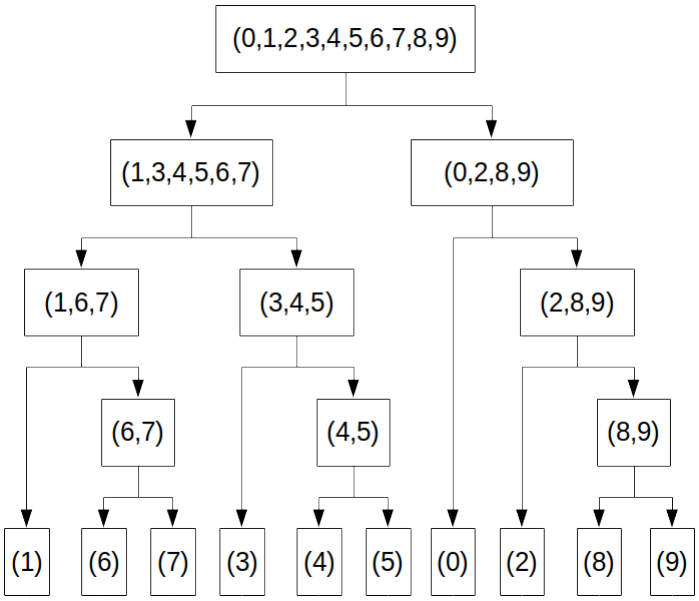
   C_{22}=(2,8,9)

7. C_{22} is split further resulting in:

   C_{211}=(2)

   C_{212}=(8,9)

8. Nothing can be split further, so the algorithm ends. The cluster hierarchy was the same as that obtained in Agglomerative approach. A pictorial representation of the splitting is given below:



7 Distance Measures

Survey the various distance measures used by clustering algorithms eg: Cosine, Jaccard similarity measures etc. Explore for a minimum of 5 measures (non-Euclidean distance measures) and trace them to measure distance between 2 data points.

1. Manhatttan distance: It is also known as City block distance or L_1 distance. Manhattan distance follows a route along the non-hypotenuse sides of a triangle. The name refers to the grid-like layout of most American cities which makes it impossible to go directly between two points. It is obtained by substituting p=1 in the Minkowski distance formula (L_p norm). The Minkowski distance of order p between 2 points $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_n) \in \mathbb{R}$ is defined as:

$$D(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{\frac{1}{p}} \tag{23}$$

   So, Manhattan distance is given by:

$$D(X,Y) = \sum_{i=1}^{n} |x_i - y_i| \tag{24}$$

2. Chebyshev: 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

$$D(x,y) = \max_i(|x_i - y_i|) \tag{25}$$

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

3. Jaccard Distance: The Jaccard index, also known as the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets. 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.

$$JaccardIndex = J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{26}$$

$$JaccardDistance = 1 - J(A,B) \tag{27}$$

4. Cosine Distance: Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The term cosine distance is often

used for the complement in positive space.

$$CosineSimilarity = S_C(A,B) = \cos\theta = \frac{A \cdot B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{28}$$

$$CosineDistance = D_C(A,B) = 1 - S_C(A,B) \tag{29}$$

5. Bray Curtis Distance: Bray Curtis distance is also called Sorensen distance. It is a normalization method that is commonly used in botany, ecology and environmental science field. It views the space as a grid similar to the city block distance. The Bray Curtis distance has a property that if all coordinates are positive, the value of the distance is between zero and one. Zero Bray Curtis distance represent exact similar coordinate. If both objects are in the zero coordinates, the Bray Curtis distance is undefined. The normalization is done using absolute difference divided by the summation.

$$d_{ij} = \frac{\sum_{k=1}^{n} |x_{ik} - y_{jk}|}{\sum_{k=1}^{n} (x_{ik} + y_{jk})} \tag{30}$$

6. Canberra Distance: Canberra distance examines the sum of series of a fraction difference between coordinates of a pair of objects. Each term of fraction difference has value between 0 and 1. The Canberra distance itself is not between 0 and 1. If one of the coordinate is zero, the term becomes unity regardless of the other values. This distance is very sensitive to a small change when both coordinates are near to zero. The Canberra distance is found using equation:

$$d_{ij} = \sum_{k=1}^{n} \frac{|x_{ik} - y_{jk}|}{|x_{ik}| + |y_{jk}|} \tag{31}$$

Trace: Consider the data points (1,5) and (3,4).

1. Manhattan distance=$|1-3| + |5-4| = 3$
2. Chebyshev distance=$\max(|1-3|, |5-4|) = 2$
3. Consider the sets $s_1 = \{1,2,3,4,5\}$, $s_2 = \{4,5,6,7,8,9,10\}$.
   Jaccard index=$\frac{|\{4,5\}|}{|\{1,2,3,4,5,6,7,8,9,10\}|} = 0.2$ and Jaccard distance $= 0.8$
4. Cosine Similarity=$\frac{1*3+5*4}{\sqrt{1^2+5^2}\sqrt{3^2+4^2}} = \frac{23}{25.495} = 0.9$, Cosine Distance $= 0.1$
5. Bray Curtis Distance=$\frac{|1-3|+|5-4|}{(1+3)+(5+4)} = \frac{3}{13} = 0.23$
6. Canberra Distance=$\frac{|1-3|}{1+3} + \frac{|5-4|}{5+4} = \frac{2}{4} + \frac{1}{9} = 0.61$

8 Implementaion

Test drive/implement Decision tree, Naive Bayes, BPN, k-means, hierarchical clustering in a platform of your choice.

Details: For all algorithms iris dataset was used. All the programs are included in the folder as .py files.

8.1 Decision Tree

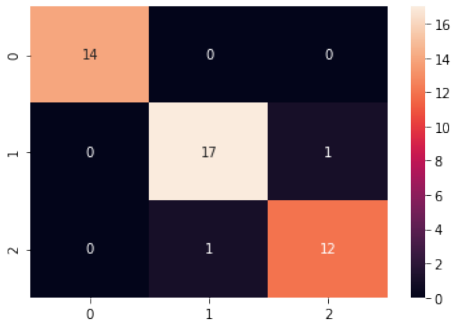
The DecisionTreeClassifier package in sklearn was used to build the decision tree and test the model.

Results:

- Accuracy: 0.95
- Number of mislabeled point out of 45: 2
- Classification report:

Class Label	Precision	Recall	f1-score	support
0	1.00	1.00	1.00	14
1	0.94	0.94	0.94	18
2	0.92	0.92	0.92	13

- Confusion matrix:



8.2 Naive Bayes

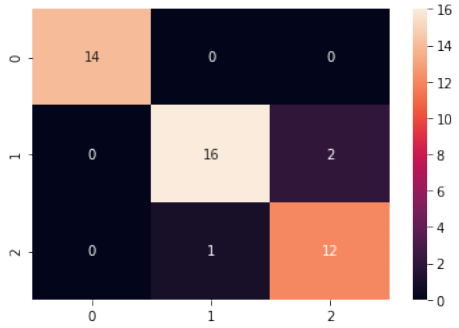
The GaussianNB package in sklearn was used to build and test the model.

Results:

- Accuracy: 0.93
- Number of mislabeled point out of 45: 3
- Classification report:

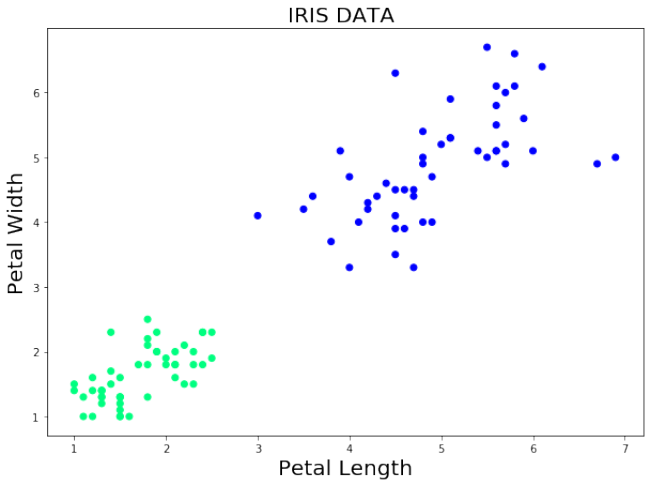
Class Label	Precision	Recall	f1-score	support
0	1.00	1.00	1.00	14
1	0.94	0.89	0.91	18
2	0.86	0.92	0.89	13

- Confusion matrix:



8.3 BPN

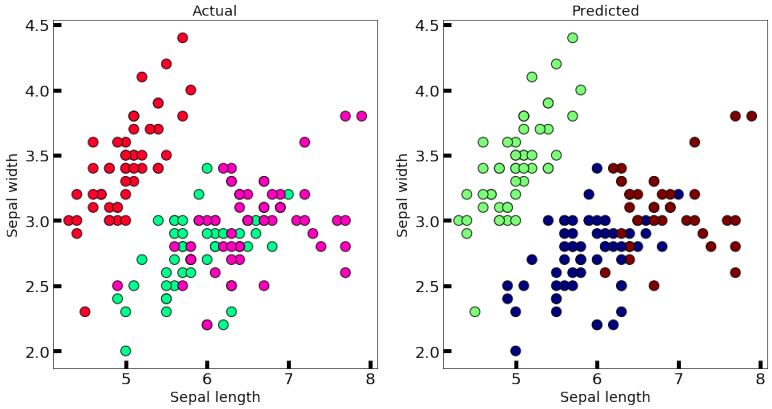
The algorithm was built using numpy package. 2 classes, versicolor and virginica, and 2 features, petal length and petal width were considered. There were 100 records in the training dataset. Scatter plot of the records:



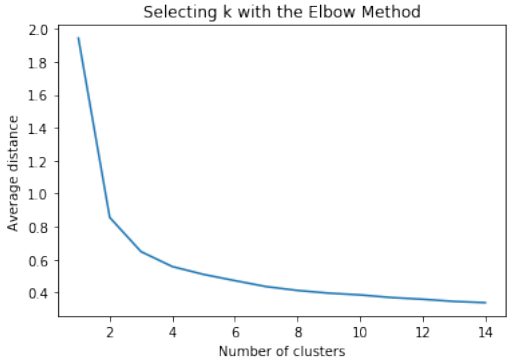
1000 epochs were performed. At the end of which the mean squared error was approx 12.4999. Although, it hits this value at around 700 epochs and stays at this point even if the number of epochs increases.

8.4 K Means

The KMeans package in sklearn was used to perform the clustering. 3 clusters were considered. The actual classification is on the left while the clusters generated by the algorithm is on the right.

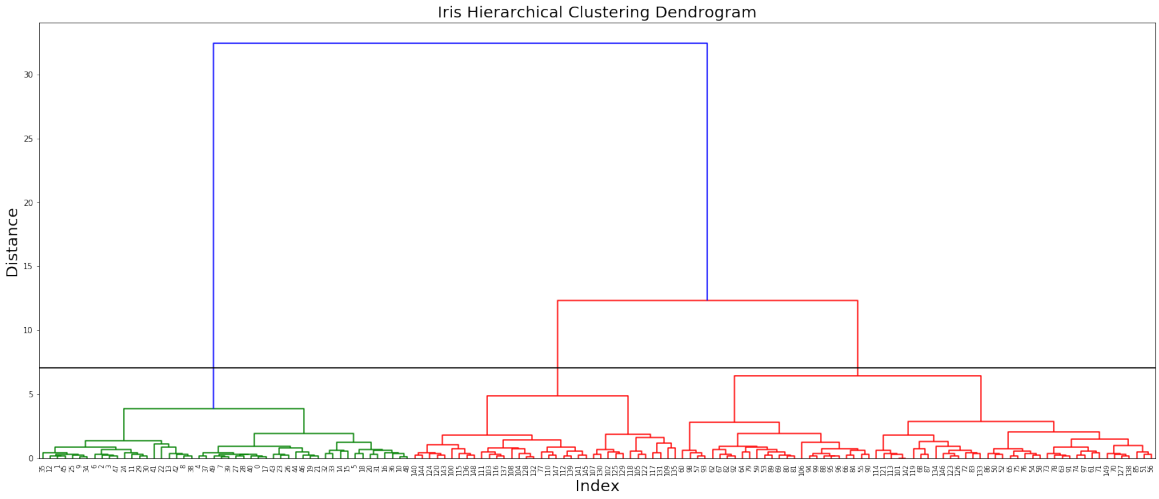


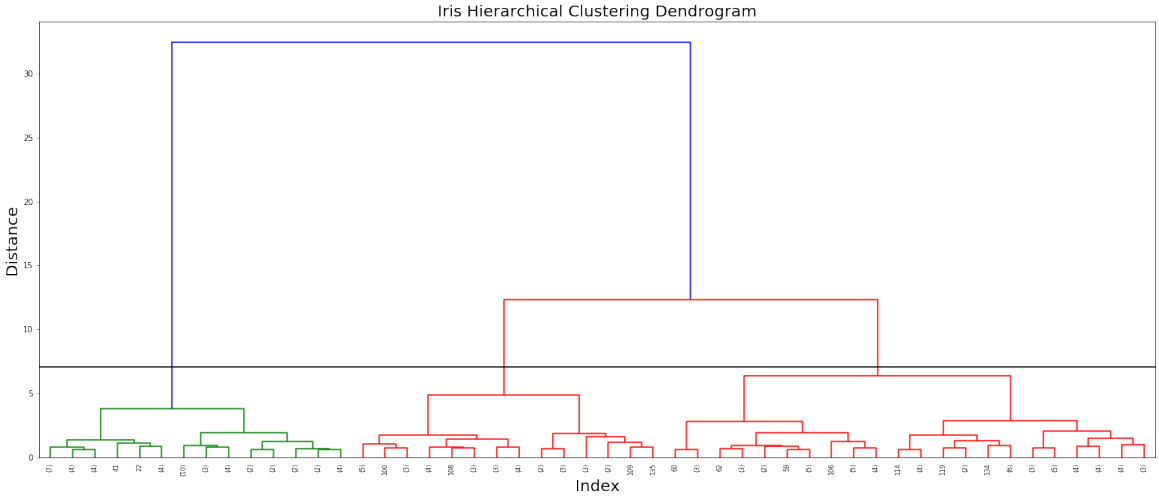
Different number of clusters were taken and the average distance of the cluster points with their centers was plotted. Generally the best k value is found this way and will be at the 'elbow' of the plot.



8.5 Heirarchical clustering

The linkage function in scipy.cluster.hierarchy was used to perform the clustering. The clustering was done and the dendograms are shown for the last 150 and 50 merged clusters respectively. A max distance of 7.08 is defined and a black horizontal line is drawn at this distance on the dendrograms. It demarcates the 3 different classes (clusters) before they get merged.


Iris Hierarchical Clustering Dendrogram
Distance
Index


Iris Hierarchical Clustering Dendrogram
Distance
Index