Leading University

Department of Computer Science and Engineering

Course Title:	Machine Learning
Course Code:	CSE-4233
Assignment Title:	Final Assignment

Submitted To:

Name:	Shafkat Kibria
Designation:	Associate Professor
Faculty:	Dept. of CSE

Submitted By:

Name:	Emad Ahmed
Department:	Dept. of CSE
Batch:	50th
Section:	C
ID:	1912020106

Submission Date: 2022-12-30

Ans to the Qn: 1

Forward Propagation: Calculate y_1 and y_2

Given that, $V_{11} = -1$, $V_{21} = 0$, $V_{31} = 0$

$V_{12} = 0$, $V_{22} = 1$, $V_{32} = 1$

$V_{10} = 1$, $V_{20} = 1$, $V_{30} = 1$

And $x_1 = 0$, $x_2 = 0$.

$$z_1 = V_{11} \times x_1 + V_{12} \times x_2 + V_{10}$$

$$= (-1) \times 0 + (0 \times 1) + 1 = 1$$

$$z_2 = V_{21} \times x_1 + V_{22} \times x_2 + V_{20}$$

$$= (0 \times 0) + (1 \times 1) + 1 = 2$$

$$z_3 = V_{31} \times x_1 + V_{32} \times x_2 + V_{30}$$

$$= (0 \times 0) + (1 \times 1) + 1 = 2$$

Now, Given, $W_{11} = 1$, $W_{21} = 1$

$W_{12} = 0$, $W_{22} = 1$

$W_{131} = 1$, $W_{23} = 0$

$W_{10} = 1$, $W_{20} = 1$

$$\therefore y_2 = W_{21} \times Z_1 + W_{22} \times Z_2 + W_{23} \times Z_3 + W_{20}$$

$$= (-1) \times 1 + (1 \times 2) + (0 \times 2) + 1$$

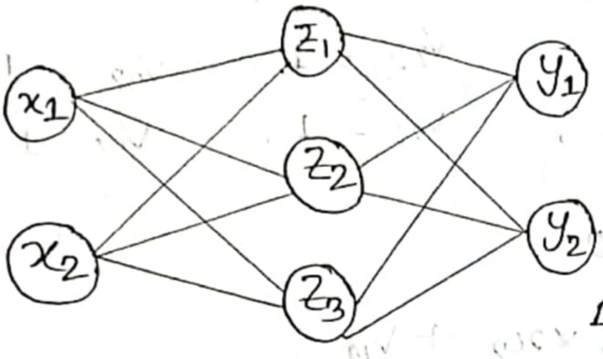
$$= 2$$

$$y_1 = w_{11} \times z_1 + w_{12} \times z_2 + w_{13} \times z_3 + w_{10}$$

$$= (1 \times 1) + (0 \times 2) + (1 \times 2) + 1$$

$$= 4$$

$\Delta_1 = -3$



4

2

$\Delta_2 = -2$

Cost / Error :

$$\Delta_1 = (d_1 - d\, y_1)$$

$$= (1 - 4) = -3$$

$$\Delta_2 = (d_2 - y_2)$$

$$= (1 - 2) = -1$$

Back propagation :

$$w_{11}^* = w_{11} + \eta \Delta_1 z_1 = (1 + (0.1 \times -3)) = +0.7$$

$$w_{12}^* = w_{12} + \eta \Delta_1 z_2 = (0 + (0.1 \times -6)) = -0.6$$

$$w_{13}^* = w_{13} + \eta \Delta_1 z_3 = (1 + (0.1 \times -6)) = 0.4$$

And,

$$w_{21}^* = w_{21} + \eta \Delta_2 z_1 = (-1 + (0.1 \times -1)) = -1.1$$

$$w_{22}^* = w_{22} + \eta \Delta_2 z_2 = (1 + (0.1 \times -2)) = 0.8$$

$$w_{23}^* = w_{23} + \eta \Delta_2 z_3 = (0 + (0.1) \times -2) = -0.2$$

For update bias value :

$$W_{10}^{*} = b_1^{*} = b_1 + \eta \Delta_1 z_0 = 1 + (0.1 \times -3 \times 1) = 0.7$$

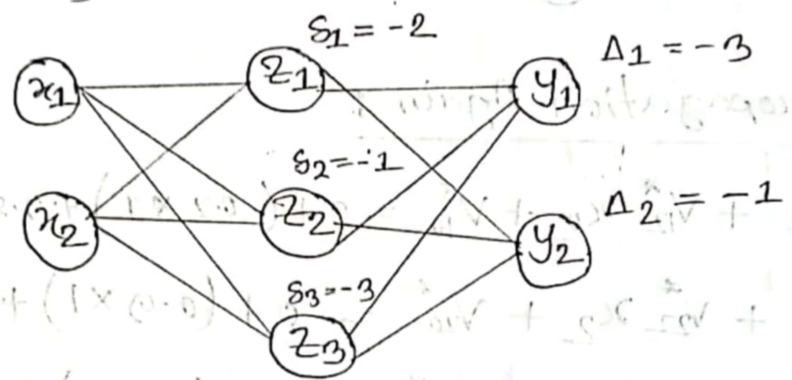
$$W_{20}^{*} = b_2^{*} = b_2 + \eta \Delta_2 z_0 = 1 + (0.1 \times -1 \times 1) = 0.9$$

Now update hidden layers node values :

$$\delta_1 = z_1 = \Delta_1 w_{11} + \Delta_2 w_{21} = -3 \times 1 + (-1 \times -1) = -2$$

$$\delta_2 = z_2 = \Delta_1 w_{12} + \Delta_2 w_{22} = -3 \times 0 + (-1 \times 1) = -1$$

$$\delta_3 = \Delta_1 z_3 = \Delta_1 w_{13} + \Delta_2 w_{23} = -3 \times 1 + (-1 \times 0) = -3$$



Now updating input layers of weight :

$$V_{11}^{*} = V_{11} + \eta \delta_1 x_1 = -1 + (0.1 \times -2 \times 0) = -1$$

$$V_{12}^{*} = V_{12} + \eta \delta_1 x_2 = 0 + (0.1 \times -2 \times 1) = -0.2$$

$$V_{21}^{*} = V_{21} + \eta \delta_2 x_1 = 0 + (0.1 \times -1 \times 0) = 0$$

$$V_{22}^{*} = V_{22} + \eta \delta_2 x_2 = 1 + (0.1 \times -1 \times 1) = 0.9$$

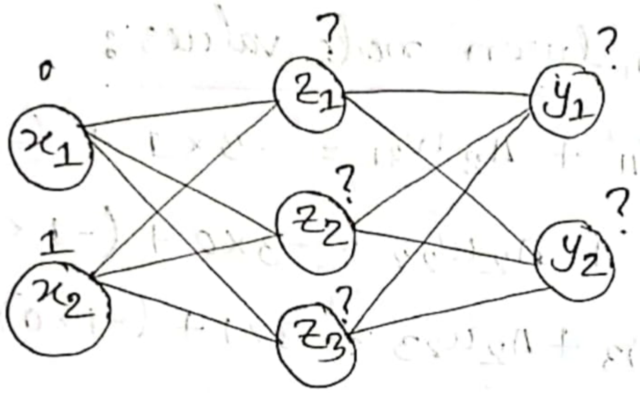
$$V_{31}^{*} = V_{31} + \eta \delta_3 x_1 = 0 \cancel{x} + (0.1 \times -3 \times 0) = 0$$

$$V_{32}^{*} = V_{32} + \eta \delta_3 x_2 = 1 + (0.1 \times -3 \times 1) = 0.7$$

$V_{10}^{*} = B_1^{*} = 1 + \eta \delta_1 b_1 = 1 + (0.1 \times -2 \times 1) = 0.8$

$V_{20}^{*} = B_2^{*} = 1 + \eta \delta_2 b_2 = 1 + (0.1 \times -1 \times 1) = 0.9$

$V_{30}^{*} = B_3^{*} = 1 + \eta \delta_3 b_3 = 1 + (0.1 \times -3 \times 1) = 0.7$

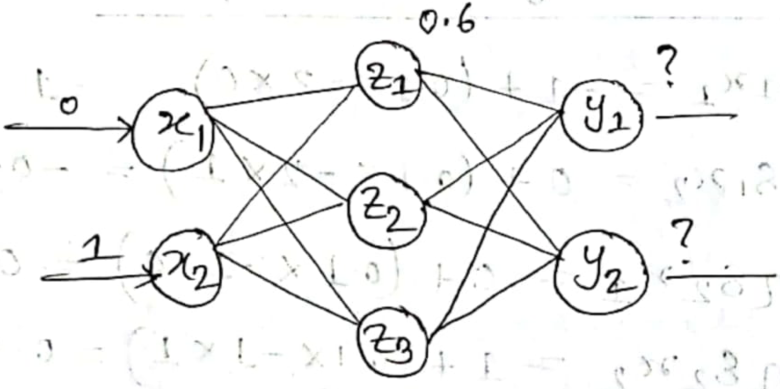


Forward Propagation Again :

$Z_1 = V_{11}^{*} x_1 + V_{12}^{*} x_2 + V_{10}^{*} = 0 + (-0.2 \times 1) + 0.8 = 0.6$

$Z_2 = V_{21}^{*} x_1 + V_{22}^{*} x_2 + V_{20}^{*} = 0 + (0.9 \times 1) + 0.9 = 1.8$

$Z_3 = V_{31}^{*} x_1 + V_{32}^{*} x_2 + V_{30}^{*} = 0 + (0.7 \times 1) + 0.7 = 1.4$



$$y_1 = W_{11}^* z_1 + W_{12}^* z_2 + W_{13}^* z_3 + W_{10}^*$$

$$= (0.7 \times 0.6) + (0.6 \times 1.8) + (0.4 \times 1.4) + 0.7$$

$$= 0.6$$

$$y_2 = W_{21}^* z_1 + W_{22}^* z_2 + W_{23}^* z_3 + W_{20}^*$$

$$= (-1.1 \times 0.6) + (0.8 \times 1.8) + (-0.2 \times 1.4) + 0.9$$

$$= 1.4$$

$\therefore$ $y_1 = 0.6$ and $y_2 = 1.4$

And our target value is $[1, 1]$, this value is close to y_1 and y_2 Ans.

Ans to the Qn: 2

Let,

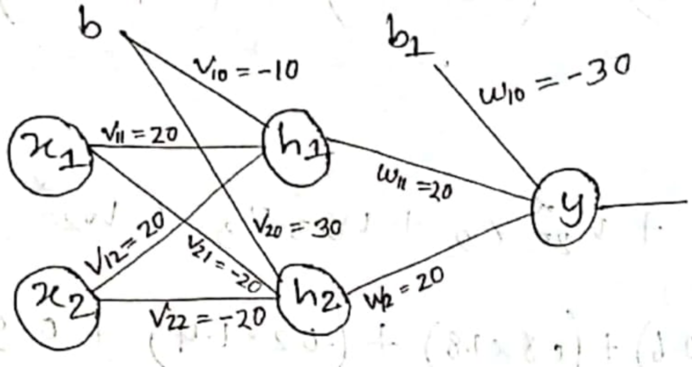


Fig-1

We know that X-OR Truth table

x_1	x_2	F
0	0	0
0	1	1
1	0	1
1	1	0

For $x_1 = 0$ & $x_2 = 1$, X-OR output is 1.

Now we will check using figure-1

$$\therefore h_1 = V_{10} + V_{11}x_1 + V_{12}x_2$$

$$= -10 + (20 \times 0) + (20 \times 1)$$

$$= 10$$

Using Sigmoid function, $h_1 = \dfrac{1}{1+e^{-h_1}}$

$$= \frac{1}{1+e^{-10}}$$

$$= 0.99 \cong 1$$

$\therefore\ h_2 = V_{20} + V_{21}x_1 + V_{22}x_2$

$= 30 + (-20\times 0) + (-20\times 1)$

$= 10$

Using Sigmoid function, $h_2 = \frac{1}{1+e^{-h}}$

$= \frac{1}{1+e^{-10}}$

$= 0.99 \cong 1$

Now we calculate output layers

y. value $\longrightarrow$

$y = W_{10} + W_{11}h_1 + W_{22}h_2$

$= -30 + (20\times 1) + (20\times 1)$

$= 10$

Using Sigmoid function, $y = \frac{1}{1+e^{-y}}$

$= \frac{1}{1+e^{-10}}$

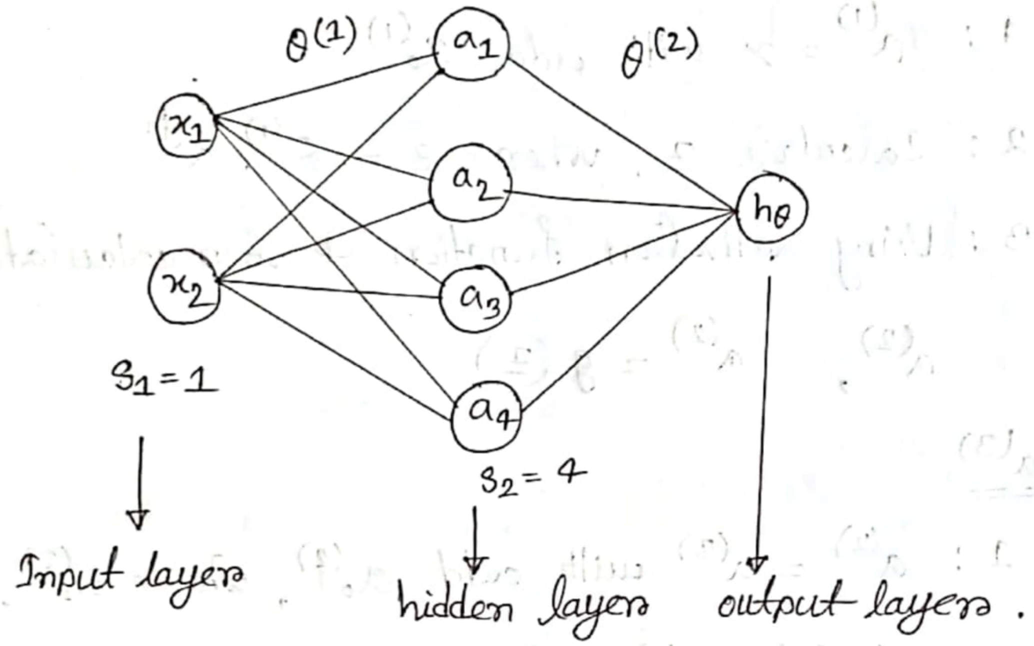
$= 0.99 \cong 1$

So we can see that, Our target value is 1.

and our hypothesis value is 1. [Proved]

Ans to the Qn: 3 (a)

(i)



(ii)

We know that, if network has S_j units in layer j, $S_j + 1$ units in layer $j+1$, then $\theta(j)$ will be of dimension $- \quad S_{j+1} \times (S_j + 1)$.

The dimension of $\theta^{(1)}$ matrices is $= 4 \times (2+1)$

$= 4 \times 3 = 12$

dimension of $\theta^{(2)}$ matrices is $= 1 \times (4+1)$

$= 1 \times 5 = 5$

Ans

(iii)

For $a^{(2)}$

Step-1 : $a^{(1)} = x$ with add $a_0^{(1)}$.

Step-2 : Calculate z, where $z = \theta^{(1)} . a^{(1)}$

Step-3 : Using activation function g for calculating $a^{(2)}$, $a^{(2)} = g(z)$

For $a^{(3)}$

Step-1 : $a^{(2)} = a^{(2)}$ with add $a_0^{(2)}$, where $a_0^{(2)}$ is bias.

Step-2 : Calculate $z^{(3)} = \theta^{(2)} . a^{(2)}$

Step-3 : Using activation function for calculating $a^{(3)}$

$$a^{(3)} = g\left(z^{(3)}\right)$$

Ans to the Qn: 4 (a)

SM SVM algorithms use a set of mathematical functions that are defined as the kernel. So kernel function generally transforms the training set of data so that a non-linear dicision surface is able to transform to a linear equation in a higher numbers of dimension spaces.

Different types of kernel function are —

1. Polynomial kernel function :

The polynomial kernel is a general representation of kernel with a degree of more than one.

$$k(x, y) = \tanh(\gamma \cdot x^T y + r)^d, \quad r > 0$$

OR, $$k(x_i, x_j) = (x_i \cdot x_j + c)^d$$

2. Gaussion kernel :

It is used to perform transformation when there is no prior knowledge about data.

$$k(x, y) = e^{-\left(\frac{\|x-y\|^2}{2a^2}\right)}$$

3. Sigmoid kernel :

This function is equivalent to two-layer perceptron model of the neural network. Which is used as an activation function.

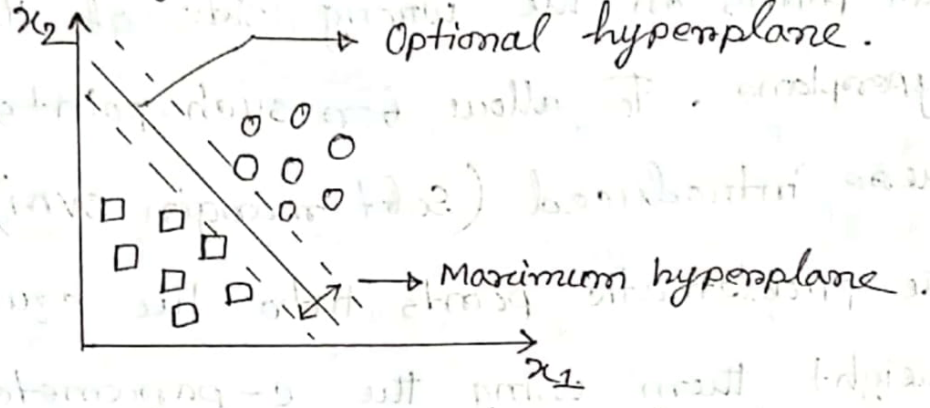
$$k = \tanh(\gamma \cdot x^T \cdot y + r)$$

4. String kernel function :

This kernel operates on the basis of string. It is mainly used in areas like text classification. They are very useful in text mining, genome analysis etc.

Ans. to the Qn: 4 (b)

We call SVM is a large margin classifier. Explained below :—



SVM is a type of classifier which classifies positive and negetive examples, here blue & red data points. As shown in the figure, the largest margin is found in order to avoid overfiting. i.e. The optional hyperplane in at the maximum distance from the positive and negetive examples. To satisfy this constraint and also to classify the data points accurately. The margin is maximised, that is why this is called the large margin classifier.

Ans to the Qn: 4 (c)

If our data is not perfectly separable there will points on the wrong side of the separating hyperplane. To allow for such points slack variable were introduced (soft margin SVM). They include the problematic points into the equation and weight them using the c-parameter. This parameters is a tradeoff between maximizing the margin and minimizing error.

Yes, It affect bias / variance.

If c is large → Lower bias, high variance.

If c is small → Higher bias, Low variance.

Ans to the Qn : 4 (d)

The most commonly used kernel function of Support Vector Machine (SVM) in non-linear separable data set in machine learning is gaussian kernel. Also known as radial basis function. The gaussian kernel putreby exponentially in the input feature space and uniformly in all directions around the support vector. Causing hyper spherical contours of kernel function.

Yes it affects bias / variance

If σ^2 Large : Feature f_i vary more smoothly.
And high bias, Low variance.

If σ^2 Small : Feature f_i vary less smoothly.
With Lower bias, Higher variance.

Ans to the Qn: 4 (e)

n = Number of features ($x \in R^{n+1}$)

m = Number of training examples.

↳ If n is small, m is intermediate:

→ Use SVM with Gaussian kernel.

↳ If n is large (Relative to m):

→ Use Logistic Regression OR

→ Use SVM without a kernel ("Linear kernel")

↳ If n is small, m is large:

→ Create or add more features, then use Logistic Regression OR SVM without a kernel.

Ans to the Qn: 4 (b)

Linear kernel is used when the data is Linearly saparable, that is, it can be separated using a single line. It is one of the most common kernel to be used. It is mostly used when there are a large numbers of features in a particular Data Set.

Linear SVM :

Linear SVM is less prone to overfitting then non-linear. If number of features is really large compared to the training sample just use linear kernel in Support Vector Machine. If numbers of features are small, but the Training sample is large. We may also need linear kernel then we will add more features.

Ans to the Qn: 5(a)

k-means is a very popular clustering algorithm. Below given, Some example applications —

1. **Customer Segmentation:**

Subdivision of customers into group / segments. Such that each customer segment consist of customers with similar market characteristics — pricing, loyality, spending behaviours etc.

2. **Anomaly or Fraud Detection:**

- Separate valid activity group from bots.
- Detect fraudulent claims.

3. **Inventory Categorization:**

Based on sales or other manufacturing metrics.

4. **Creating Newsfeeds:**

k-means can be used to cluster articles by their Similarity — it can separate documents into disjoint clusters.

Ans to the Qn: 5 (b)

The basic steps of the k-means algorithm :-

Step-1 : Select the k to decide the number of clusters.

Step-2 : Select random k points on centroids. (It can be other from the input data set)

Step-3 : Measure the distance (Euclidean distance) between each pair and the centroid.

Step-4 : Assign each point to the nearest cluster.

Step-5 : Calculate the mean of each cluster as the new centroid.

Step-6 : Repeat steps 3 to 5 with the new center of the clusters.

Step-7 : Calculate the varient of each other

Step-8 : Repeat step 2 to 7 until getting the lowest sum of variance. And we get the lowest sum of variance and pick those as our result.

Ans to the Qn: 5 (c)

The performance of the k-means clustering algorithm depens upon highly efficient clusters that it forms. But choosing the optimal number of cluster is a challenging task. Here we are explaining the most appropriate method to find the number of clusters or value of k.

Elbow Method :

The elbow method is one of the most popular ways to find the optimal number of clusters. This method uses the concept of wcss value. wcss stands for within cluster sum of square which defines the total variations within a cluster.

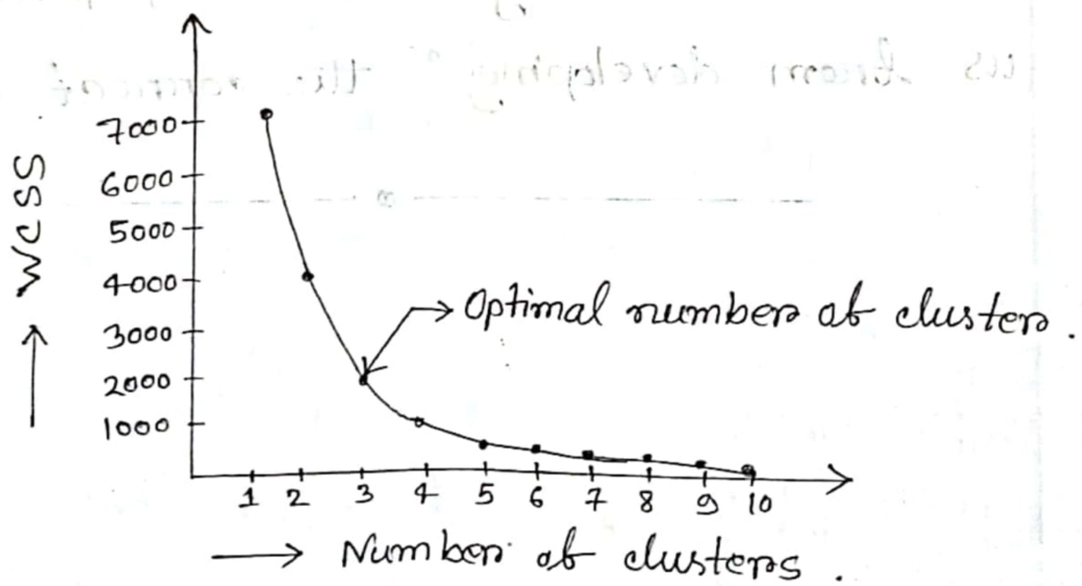
Formula :

$$\text{wcss} = \sum P_i \text{ in cluster}_1 \times \text{distance}(P_i C_1)^2 + \sum P_i \text{ in cluster}_2 \times \text{distance}(P_i C_2)^2 \ldots$$

$\sum P_i$ in $\text{cluster}_1 \times \text{distance}(P_i C_1)^2$: It is the sum of the square of the distance between each data point & each centroid within a cluster 1 and the same for the other terms.

◫ To find the optional value of clusters the elbow method follows the below steps :—

- It executes the k-means clustering on a given dataset for different k values (range from 1-10)
- For each value of k, calculates the WCSS value.
- Plots a curve between calculated WCSS values and the number of clusters k.
- The sharp point of bend or a point of the plot looks like an arm, then that points is considerd as the best value of k.



Clearly the elbow is forming at $k=3$. So the optimal value will be 3 for performing k-means.

Ans to the Qn: 5 (d)

Random Initialization trap is a problem that occurs in the k-means algorithm. In random Initilization, when the condroids of the clusters to be generated are explicitly defined by the user then Inconsistency may be created and this may sometimes lead to generating wrong clusters in the dataset. So random initialization may sometimes prevent us from developing the correct clusters.