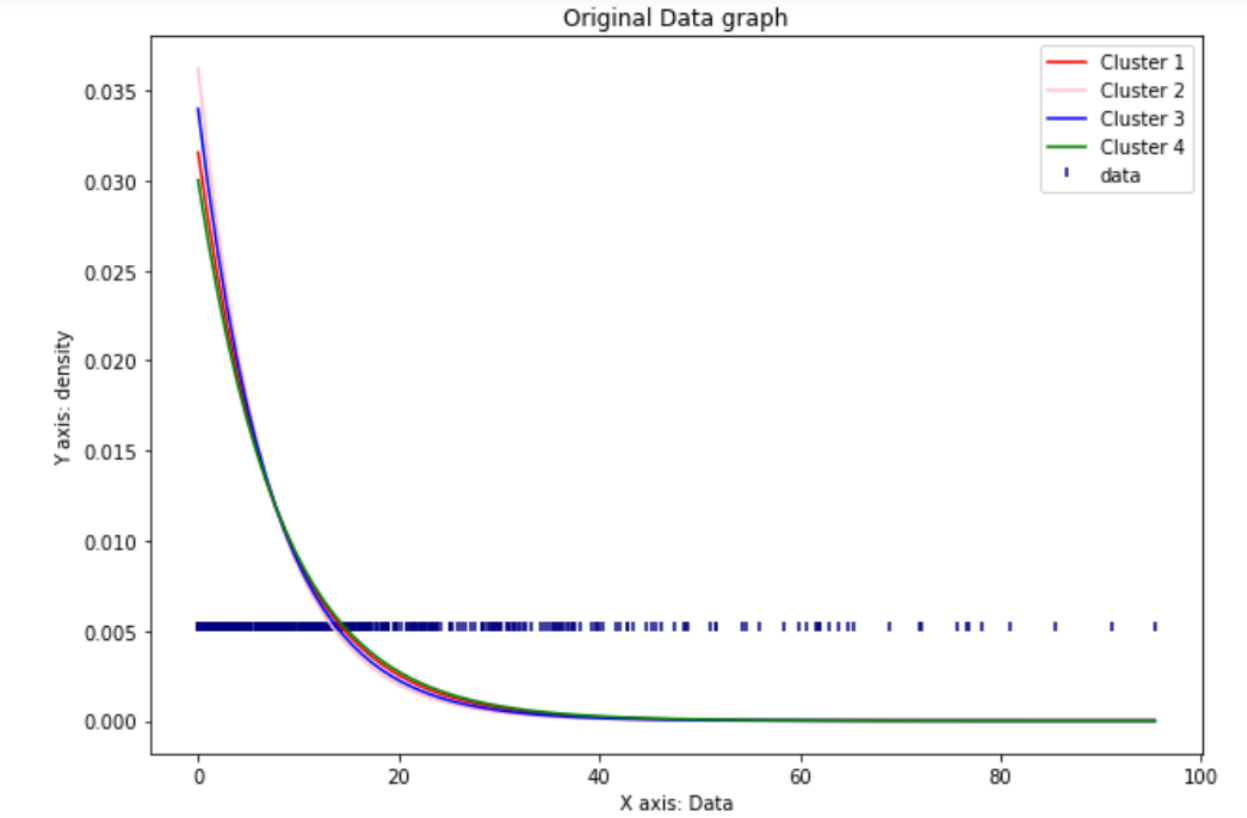
NAME : SNEHA MAGANAHALLI
RAJENDRANATH

ROLL NO : CSR1M522

Machine Learning, Assignment : 2

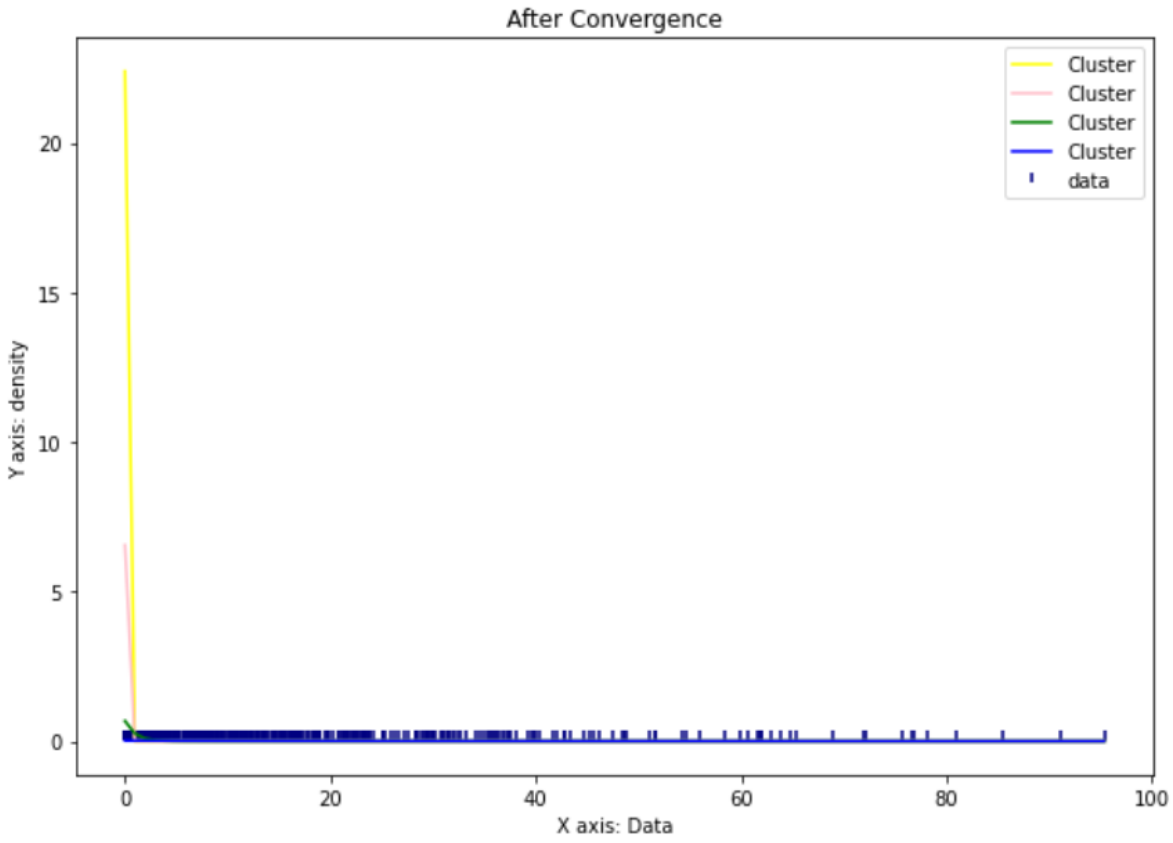
1) i)

Fig 1:


Original Data graph
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: density
X axis: Data
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100

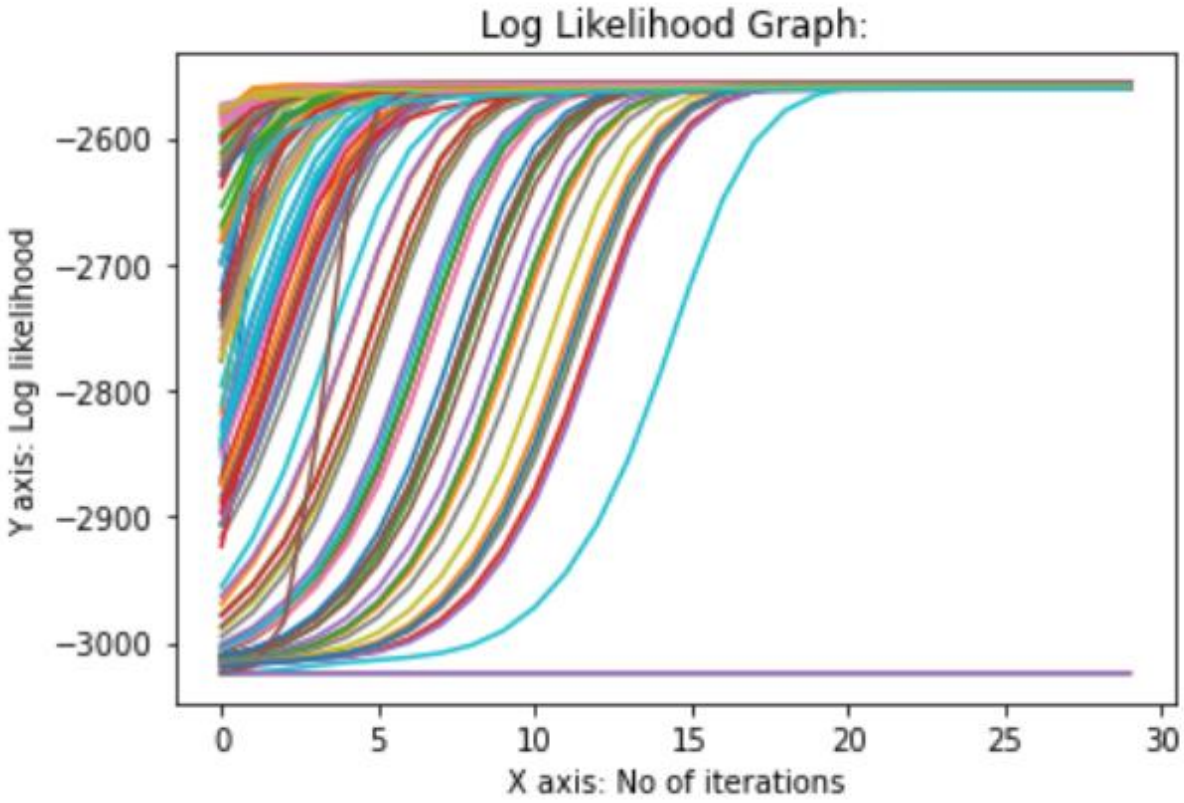
1) i)

Fig 2:


After Convergence
Cluster
Cluster
Cluster
Cluster
data
Y axis: density
X axis: Data
0
5
10
15
20
0
20
40
60
80
100

1) i)

Fig 3:


Log Likelihood Graph:
Y axis: Log likelihood
−2600
−2700
−2800
−2900
−3000
0
5
10
15
20
25
30
X axis: No of iterations

(1) (c)

Determine which probabilistic mixture could have generated this data.

Exponential Distribution has generated this data.

Fig 1: (attached)

It shows the original data (It is one-dimensional) 4 clusters, has been identified with four different colours. The initial probability of each cluster is 0.25

cluster 1 + cluster 2 + cluster 3 + cluster 4 = 1

0.25 + 0.25 + 0.25 + 0.25 = 1

It shows before convergence (initial state)

Fig 2:

It shows after converging.

Here we see that all the clusters are overlapped on each other at the beginning.

Fig 3:

It shows the log-likelihood graph for 100- different random initialization.

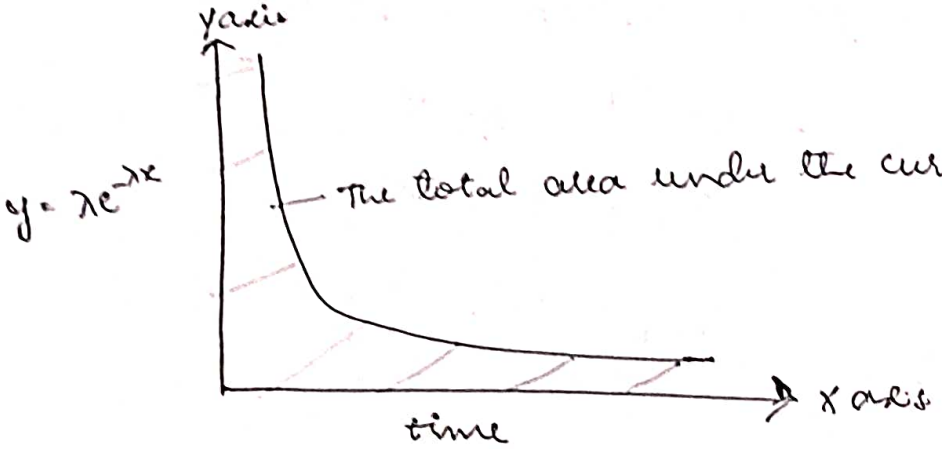
we see that likelihood is 0 initially & raises slowly as the number of iterations increases. And reaches maximum to give the maximum likelihood.

Each color in the Graph shows one random-initialzation of data

What is exponential distribution?

It is a statistical distribution that models the time between events.

e.g. How long you wait before you get another text message



(amount of time between events)

probability of an event, (like person waiting for text message) happening within 0 to 5 seconds ⇒ check the area under the curve from $x=0$ to $x=5$ seconds

formula : $y = \lambda e^{-\lambda x}$

λ : lambda is called a rate parameter, & it is proportional to how quickly things happen

Goal of Maximum likelihood :

Given a set of measurements, to find an optimal value for λ

$L(\lambda | x_1) = \lambda e^{-\lambda x_1}$ --- What is the likelihood of λ given our first measurement is x_1

IIIrd $L(\lambda|x_2) = \lambda e^{-\lambda x_2}$

What is the likelihood of λ given x_1 and x_2

$L(\lambda|x_1) = \lambda e^{-\lambda x_1}$ $\quad$ $L(\lambda|x_2) = \lambda e^{-\lambda x_2}$

$$L(\lambda|x_1 \text{ and } x_2) = L(\lambda|x_1)\, L(\lambda|x_2)$$

$$= \lambda e^{-\lambda x_1} \quad \lambda e^{-\lambda x_2}$$

$$= \lambda^2 \left[e^{-\lambda x_1} \quad e^{-\lambda x_2} \right]$$

$$= \lambda^2 \left[e^{-\lambda (x_1 + x_2)} \right]$$

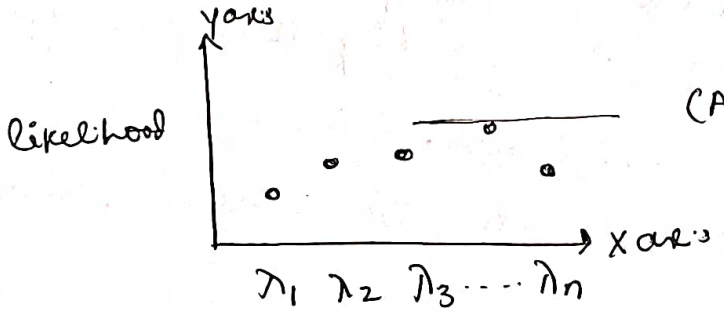
IIIrd

$$L(\lambda|x_1, x_2, x_3 \cdots x_n) = \lambda^n \left[e^{-\lambda (x_1 + x_2 + \cdots x_n)} \right] \quad (\star)$$

To find the Maximum likelihood

Steps:

1. Take derivative of $(\star)$
2. solve for λ when the derivative is set to be equal to 0



(At Maximum likelihood, the slope (& thus the ~~Maximum~~ derivative) will be 0

Step 1:

$$\frac{\partial}{\partial \lambda} L(\lambda|x_1, x_2 \cdots x_n) = \frac{\partial}{\partial \lambda} \left[\lambda^n \left[e^{-\lambda (x_1 + x_2 + \cdots x_n)} \right] \right]$$

$$= \frac{\partial}{\partial \lambda} \log \left(\lambda^n \left[e^{-\lambda (x_1 + x_2 + \cdots x_n)} \right] \right)$$

[$\because$ derivative of a function & the derivative of the log of a function is equal to 0 at at same place]

$$= \frac{\partial}{\partial \lambda} \log(\lambda^n) + \log\left[e^{-\lambda(x_1+x_2+\cdots x_n)}\right]$$

$$= \frac{\partial}{\partial \lambda} n\log\lambda - \lambda(x_1+x_2+\cdots+x_n) = \boxed{n\frac{1}{\lambda} - (x_1+x_2+\cdots x_n)}$$

Step 2

Set the derivative to be 0 & solve for λ.

$$0 = n\frac{1}{\lambda} - (x_1+x_2+\cdots x_n)$$

$$\frac{n}{\lambda} = x_1+x_2+\cdots+x_n$$

$$1/\lambda = \frac{x_1+x_2+\cdots+x_n}{n}$$

$$\therefore \lambda = \frac{n}{x_1+x_2+\cdots+x_n}$$ ... Max likelihood estimate for λ

EM Algorithm for exponential distribution:

2 steps

1. Find expected log likelihood.
2. Adjust the parameters in the likelihood to maximize this.
   Iterate using new parameter values

Algorithm:

Detailed steps

i) Initialize the initial probability.

$\pi = [0.25, 0.25, 0.25, 0.25]$

ii) Compute the probability (posterior) that a point in feature space comes from a given class

$\gamma_k(x) = P(T=t \mid y=k)$ and equal to the Bayes rule as $\gamma_k(t) = \frac{\pi_k f_{T|Y=k}(t;\lambda_k)}{\sum_{k\in K} \pi_k f_{T|Y=k}(t;\lambda_k)}$

→ Bayes classifi

E-step:

Responsibilities at iteration m for data point n in class k is

$$r_{n,k}^{m} = \frac{\pi_k^m \, f_{T_n \mid Y_n = k}(t_n ; \lambda_k^m)}{\sum_{k \in K} \pi_k^m \, f_{T_n \mid Y_n = k}(t_n ; \lambda_k^m)}$$

Where the probability that data point n occurs given that it is in class k is

$$f_{T_n \mid Y_n = k}(t_n ; \lambda_k^m) = \lambda_{k,1}^m \, e^{-t_{n,1} \lambda_{k,1}^m} \; \lambda_{k,2}^m \, e^{-t_{n,2} \lambda_{k,2}^m}$$

M-step:

M-step updates rules for class l

$$\pi_l^{m+1} = \frac{\sum_{n \in N} r_{n,l}^m}{\sum_{n \in N, l \in K} r_{n,l}^m}$$

$$\lambda_{l,i}^{m+1} = \frac{\sum_{n \in N} r_{n,l}^m}{\sum_{n \in N} t_{n,i} \, r_{n,l}^m}$$

Iterate, until the algorithm converges.

We used the new π and λ and iterate again in every step to check whether the algorithm has converged.

Each iteration, likelihood is found with new π, λ. After few iterations, when the algorithm converges. The likelihood will be Maximum!

The log of the likelihood is taken to plot the likelihood graph for each iteration.

Calculation :-

Initial pi & mu i.e; (Initial probabilities & centroid)

pi, mu
0.25 7.9 → cluster1
0.25 6.9 → cluster2
0.25 7.3 → cluster3
0.25 8.3 → cluster4

Run Expectation Maximization

~~for~~ 30 iterations (It can be more or less than 30)
{ calculate exponential pdf] → E step (use result from M step)
calculate new pi
calculate new mu → M step (use result from E step)
calculate log_likelihood & using new pi & new mu
plot log_likelihood graph

3 stop once it converges.

Eg

Estep:

log_gamma = $\begin{bmatrix} 2.7 & \cdots & 1.9 \\ \vdots & & \\ 1.1 & \cdots & 7.9 \end{bmatrix}_{1000 \times 4}$ matrix

result from exponential distribution

Formula:

$f(x,\lambda) = \begin{cases} \lambda e^{-\lambda x} & \cdots x \geq 0 \\ 0 & \cdots x < 0 \end{cases}$

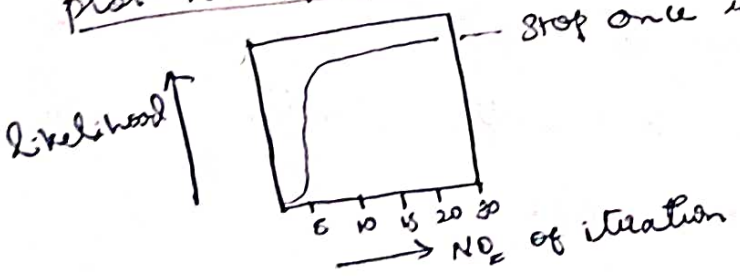
Mstep:

new_pi = [1.1 1.2 1.3 1.9]

new_mu = [3.0 0.25 0.8 1.9]

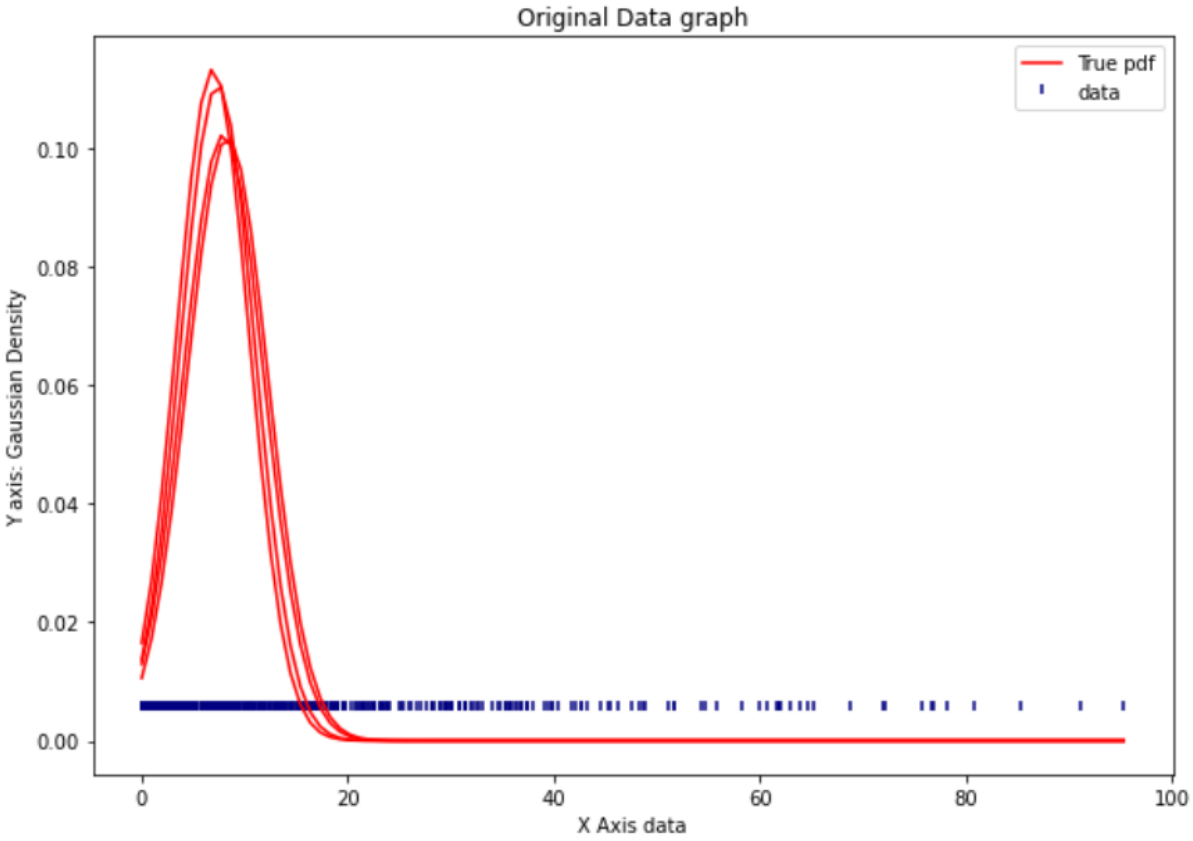
log_likelihood [-2985, 2870, ... -2556] 30 values ↓ No of iteration

plot the graph of likelihood



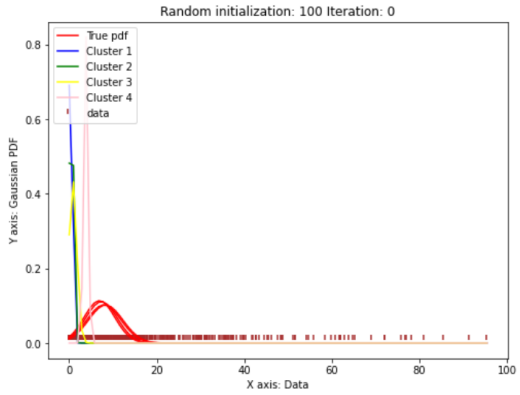
1) ii)

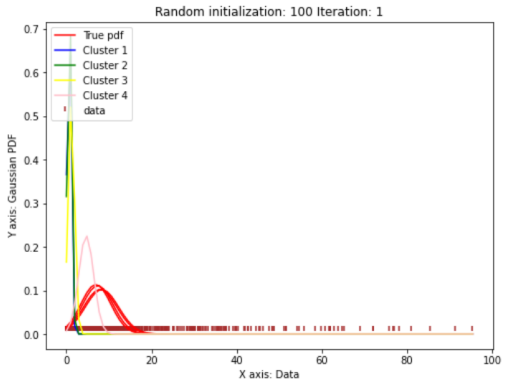
Fig a:

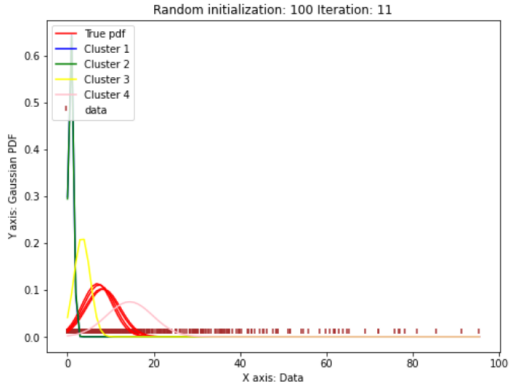

Original Data graph
True pdf
data
Y axis: Gaussian Density
X Axis data
0.00
0.02
0.04
0.06
0.08
0.10
0
20
40
60
80
100

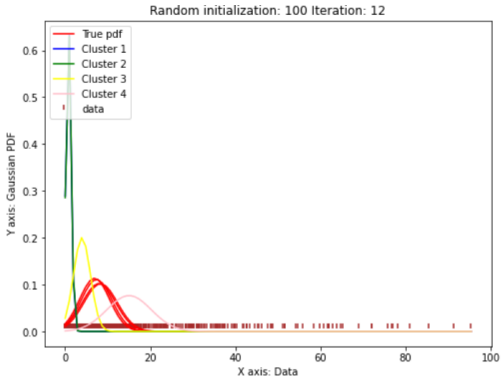
1) ii)

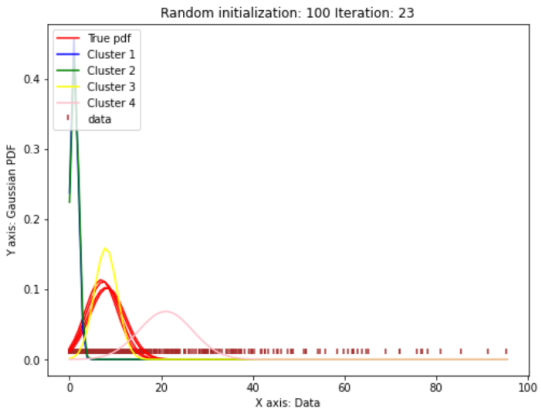
Fig 2:

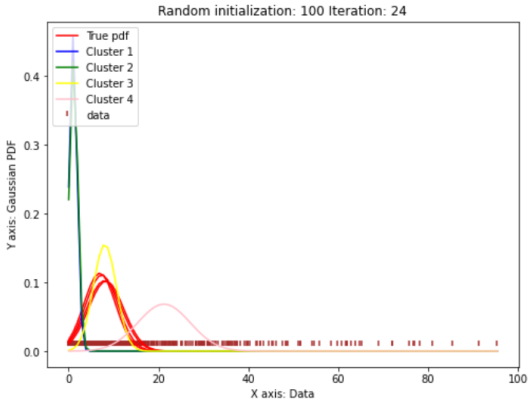

Random initialization: 100 Iteration: 0
True pdf
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: Gaussian PDF
X axis: Data


Random initialization: 100 Iteration: 1
True pdf
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: Gaussian PDF
X axis: Data


Random initialization: 100 Iteration: 11
True pdf
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: Gaussian PDF
X axis: Data

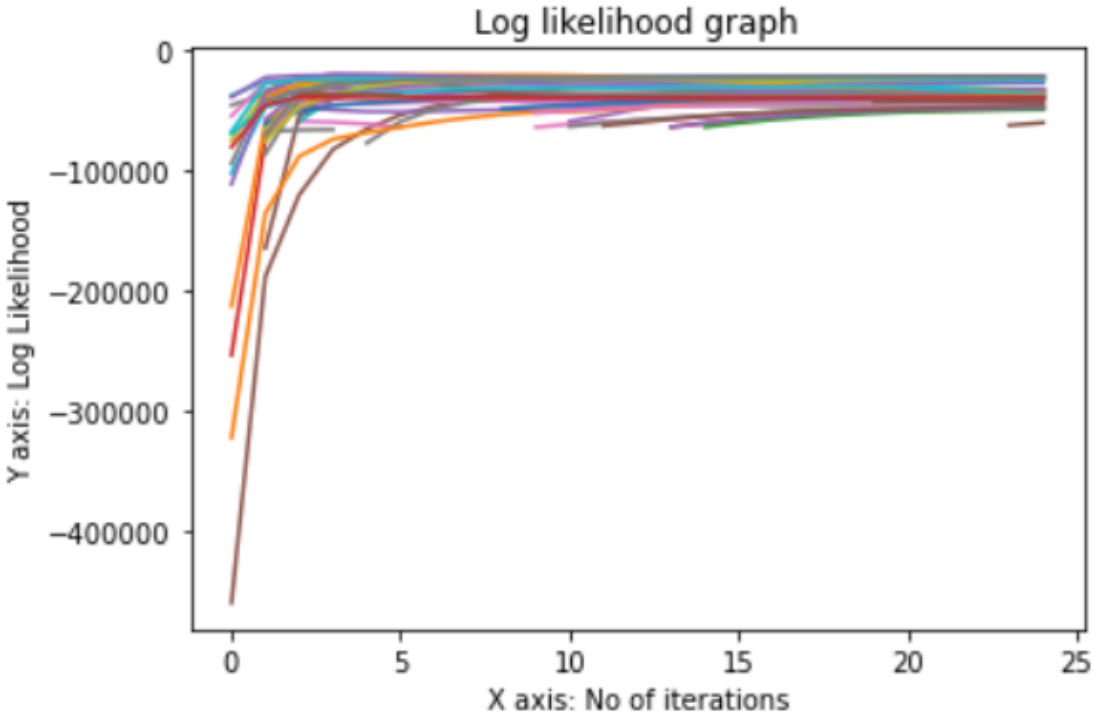

Random initialization: 100 Iteration: 12
True pdf
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: Gaussian PDF
X axis: Data


Random initialization: 100 Iteration: 23
True pdf
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: Gaussian PDF
X axis: Data


Random initialization: 100 Iteration: 24
True pdf
Cluster 1
Cluster 2
Cluster 3
Cluster 4
data
Y axis: Gaussian PDF
X axis: Data

1) ii)

Fig 3:


Log likelihood graph
0
−100000
−200000
−300000
−400000
Y axis: Log Likelihood
0
5
10
15
20
25
X axis: No of iterations

(1)(ii) Apply Gaussian distribution with 4 mixtures

Fig 1:

- Shows the original data [blue color, 1 dimension]
- Shows 4-Gaussian bell curve [Red color].
  Each as initial probability as 0.25

$$0.25 + 0.25 + 0.25 + 0.25 = 1$$

$$\begin{bmatrix}\text{cluster 1}\\ \text{probability}\end{bmatrix} + \begin{bmatrix}\text{cluster 2}\\ \text{probability}\end{bmatrix} + \begin{bmatrix}\text{cluster 3}\\ \text{probability}\end{bmatrix} + \begin{bmatrix}\text{cluster 4}\\ \text{probability}\end{bmatrix} = 1$$

Fig 2:

- It shows one of the random initialization of the data [here 100th random initialization]
- Iteration 0 & 1
  we see that all the four clusters are very near to 'y' axis. (overlapping on each other)
- Iteration 11 & 12
  we see that clusters are moving to their appropriate positions
- Iteration 23 & 24
  we see that cluster 1 (green), cluster 2 (blue) overlap on each other. cluster 3, cluster 4 are spread on the x axis to cover all the data points.
  we see that at the end of the x axis, there is no cluster, since density of the data is less.
  we see more clusters at the beginning since the density is higher at the beginning.

Fig 3

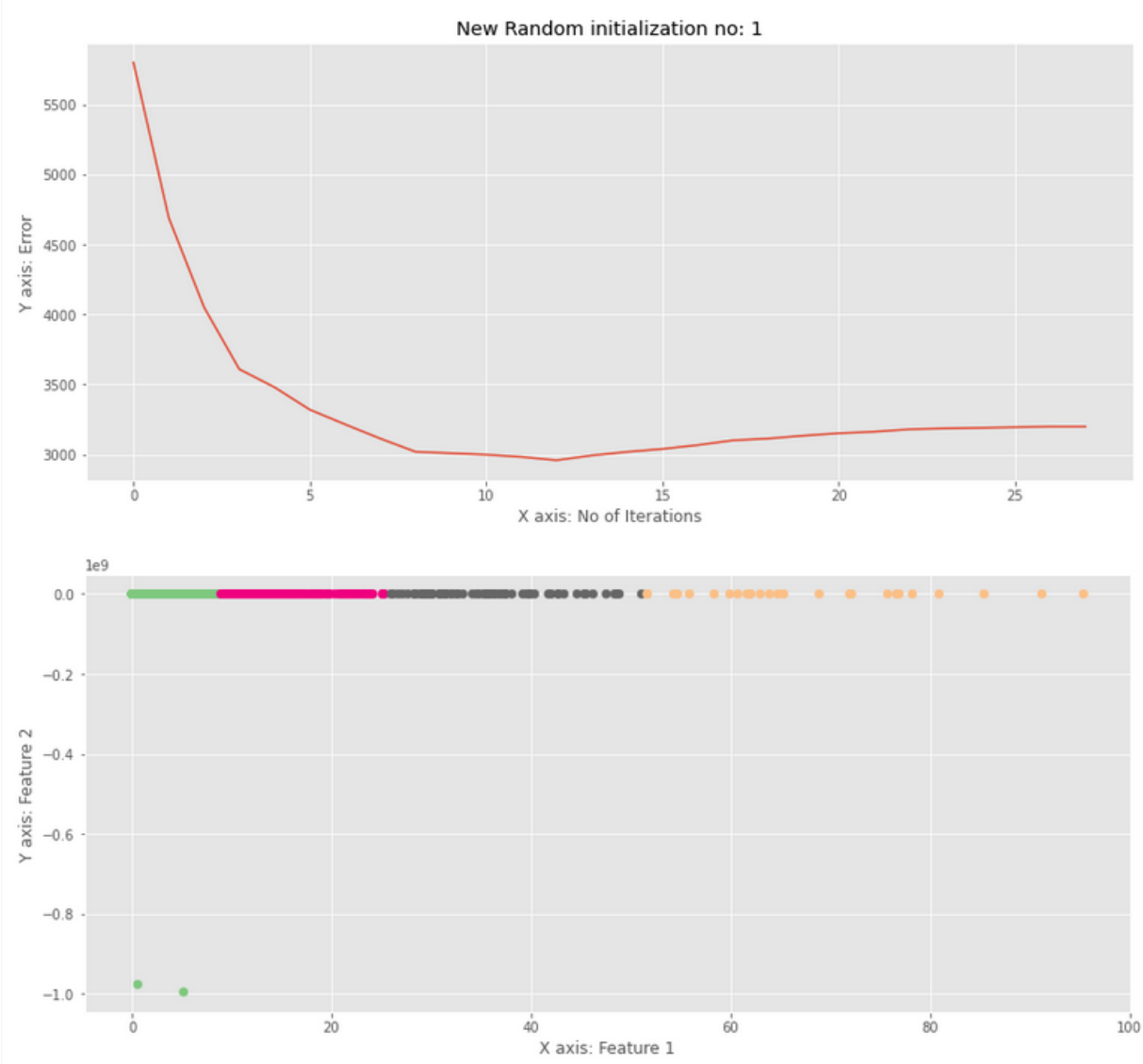
It shows log-likelihood graph for 100 random initializa. each color shows, one random initialization. we see that likelihood almost starts from ∞ a lesser-value & goes on increasing as the no of iterations increases. finally we get the maximum likelihood.

Difference between Exponential distribution mixture & Gaussian distribution mixture.

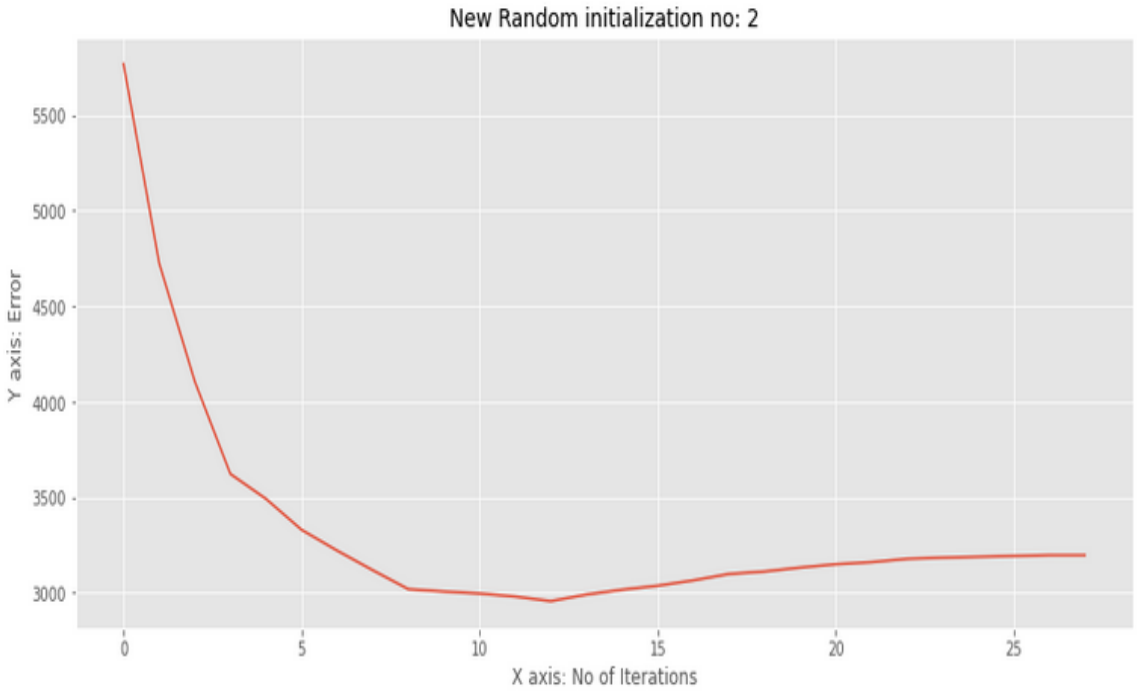
Exponential distribution mixture	Gaussian distribution mixture.
Compare: Fig (1) (i) Fig (2) All the clusters are touching y axis after convergence since the data-density is higher at the beginning of the x axis. If data density is too high at one point & too low at other point, exponential distribution mixture suits. ⇒ Good & better than Gaussian if data density is high at one point & decreases drastically at other point. ⇒ For current data, exponential distribution is better than Gaussian	(1) (ii) Fig (2) The clusters are ~~spread~~ spread along the x-axis. It is Good if data density is spread-out evenly. Good than exponential if data density is spread out evenly ⇒ For current data fairly does the job! not bitter than exponential
Fig (1) (i) Fig 3 ⇒ We see that log likelihood starts from minimum & raises slowly for each iteration. This is true for all random initialization of data. Better than Gaussian Infact best !!!	(1) (ii) Fig 3 ⇒ For some random initialization of data, the likelihood starts at the maximum itself or even crosses maximum point !! Many random initialization should be considered & averaged to get the best maximum likelihood Not very good like Exponential :C

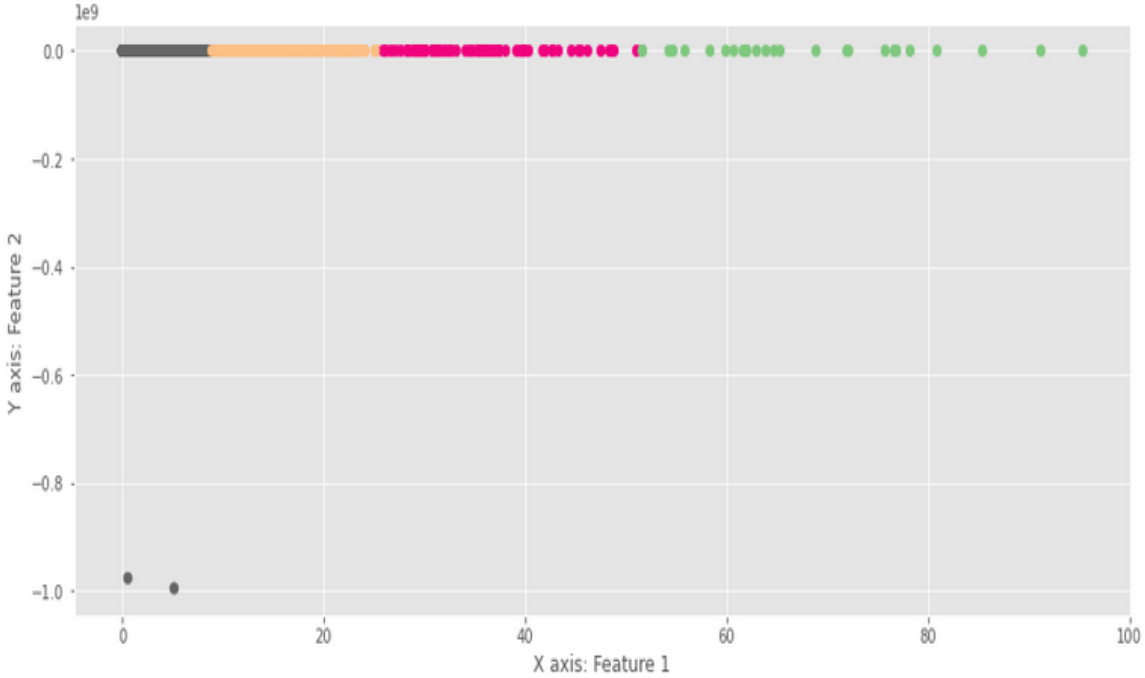
1) iii)

Fig 1:

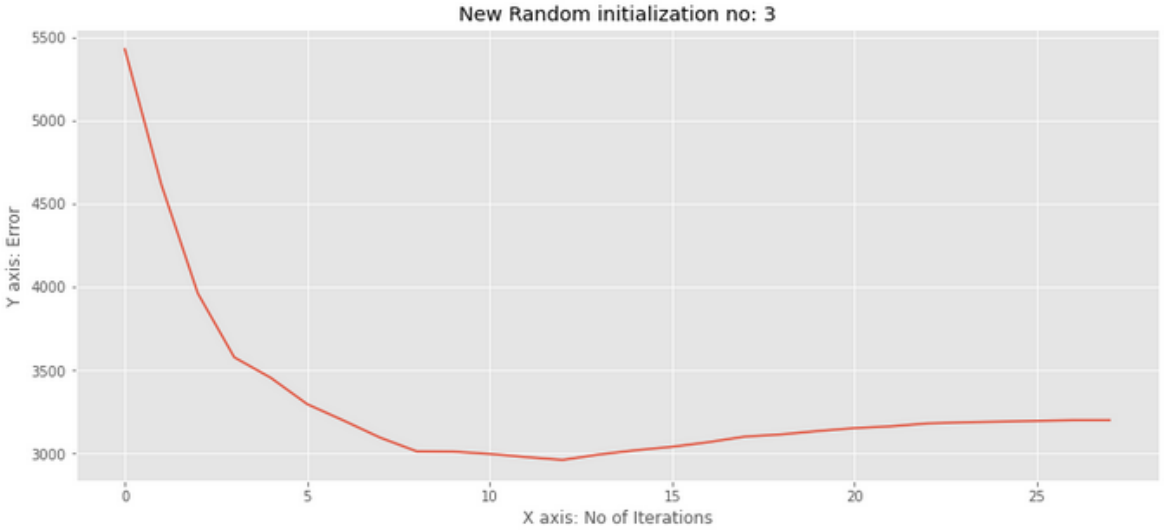

New Random initialization no: 1
Y axis: Error
X axis: No of Iterations
1e9
Y axis: Feature 2
X axis: Feature 1

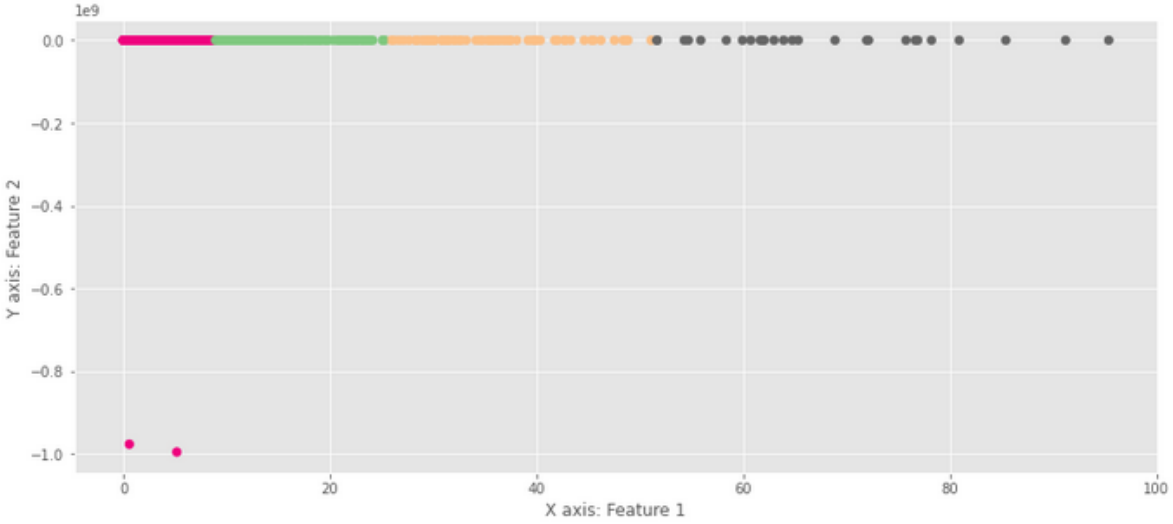
1) iii)


New Random initialization no: 2
Y axis: Error
X axis: No of Iterations
5500
5000
4500
4000
3500
3000
0
5
10
15
20
25

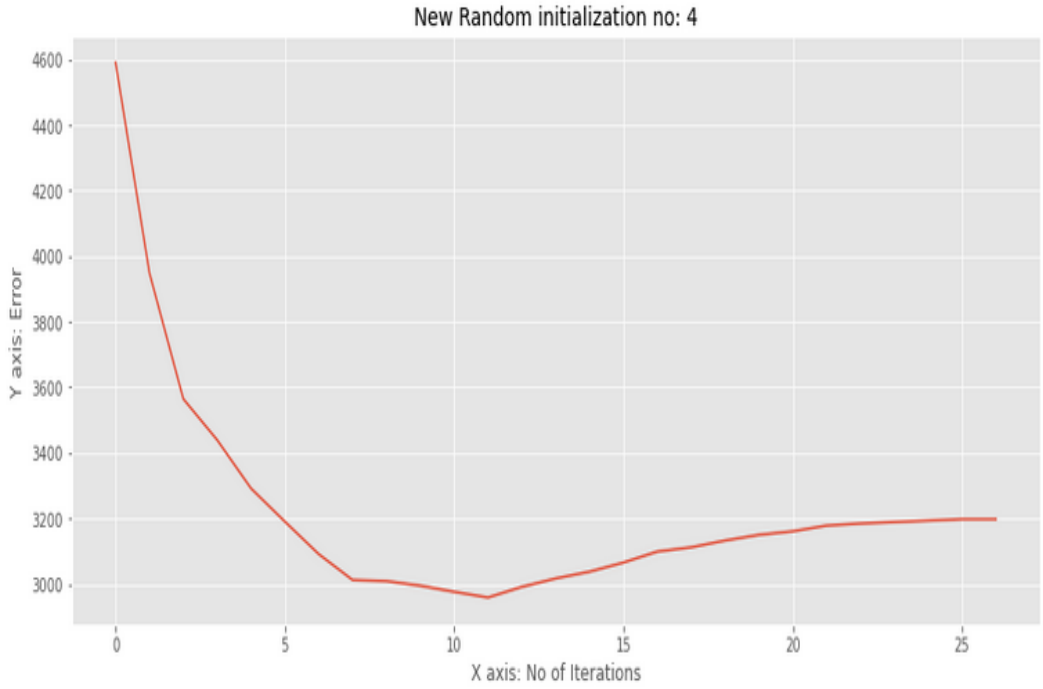

1e9
Y axis: Feature 2
X axis: Feature 1
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
20
40
60
80
100

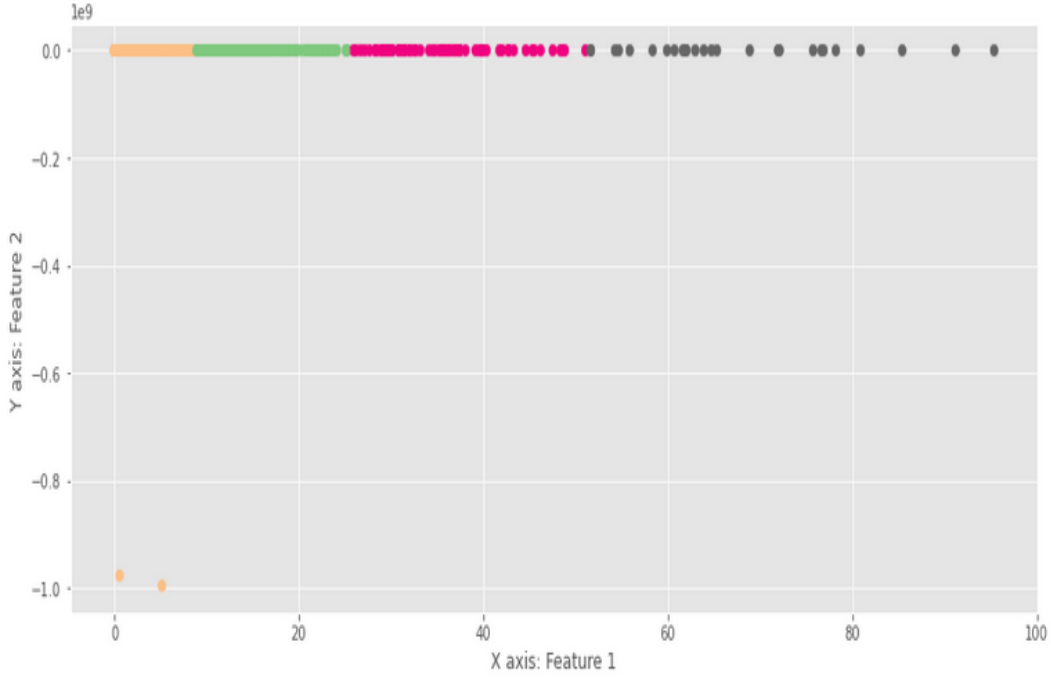
1) iii)


New Random initialization no: 3
Y axis: Error
X axis: No of Iterations

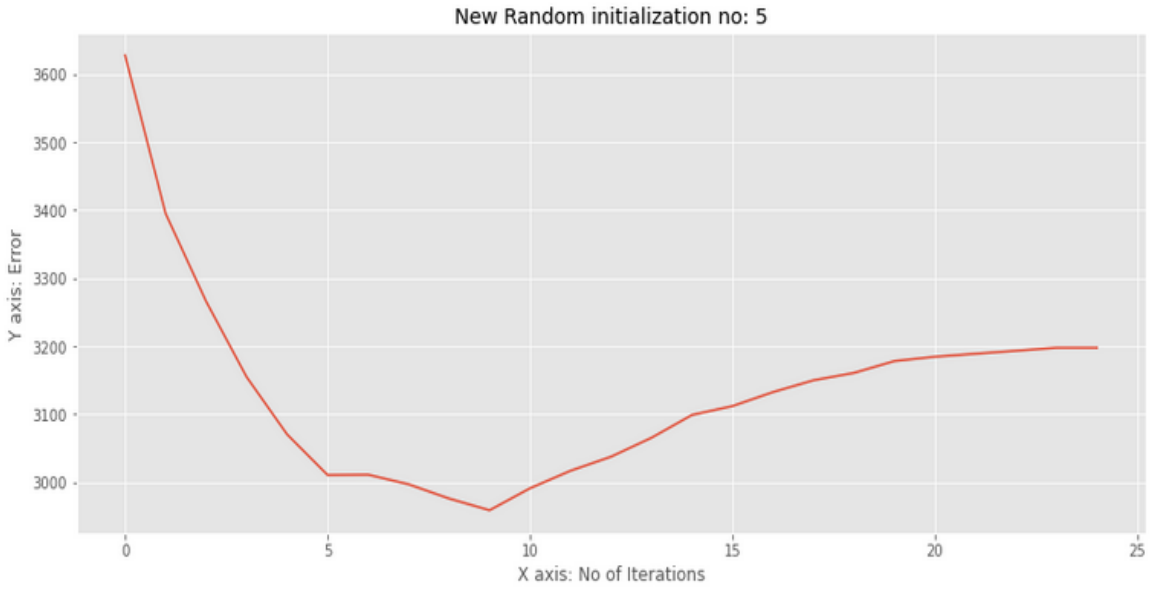

1e9
Y axis: Feature 2
X axis: Feature 1

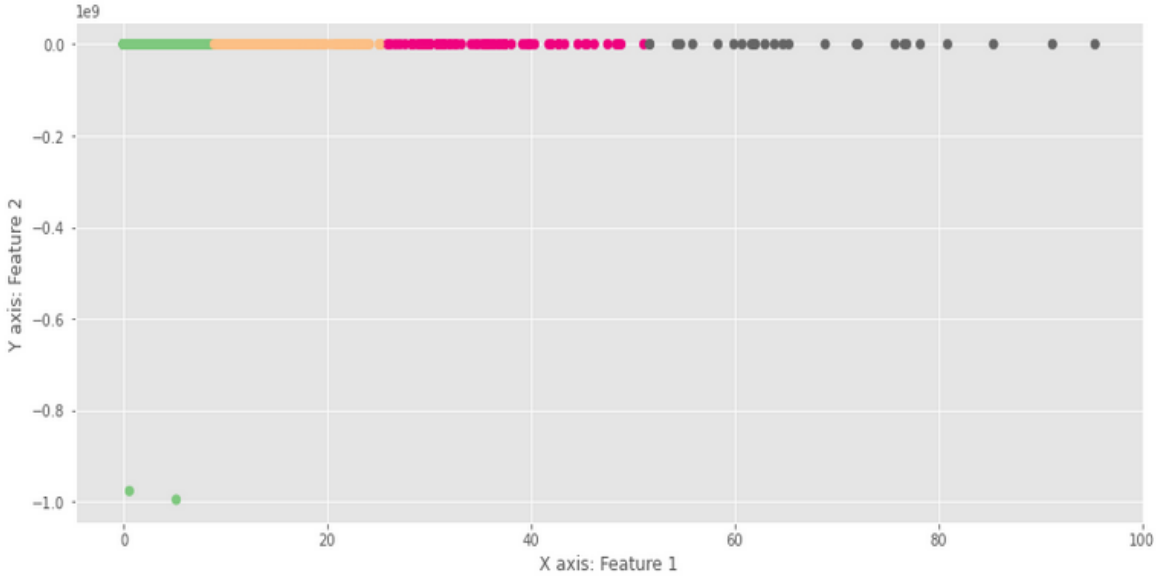
1) iii)


New Random initialization no: 4
Y axis: Error
X axis: No of Iterations


1e9
Y axis: Feature 2
X axis: Feature 1

1) iii)


New Random initialization no: 5
Y axis: Error
X axis: No of Iterations
3600
3500
3400
3300
3200
3100
3000
0
5
10
15
20
25


1e9
Y axis: Feature 2
X axis: Feature 1
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
0
20
40
60
80
100

(1)
(iii)

K-means clustering :

K-means clustering does a hard clustering. i.e; it assigns every point to a specific cluster (& we color it) unlike Gaussian or other mixture models.

The mixture models does soft clustering. i.e; at each iteration, it will calculate the probability of the data-point belonging to a particular cluster.

Fig (1) : (error function Vs no of iterations

It shows, as the number of iteration increases, the error decreases. And finally the algorithm converges. Each data-point will be assigned a particular best (nearest) cluster.

We can also see the Graph, to see that how each data-point is grouped into the best clusters.

(i) (ii)

Comparision between Exponential / Gaussian / K-Means for given data set.

Exponential Distribution	Gaussian Distribution	K-means
○ Best	Better	Good.
	a) Find k to minimize $\frac{(x-\mu_k)^2}{\sigma^2}$ ⇒ denominator is different. It will consider the variance also while clustering.	Find k to minimize $(x-\mu_k)^2$ Does not consider variance for clustering
	b)	b) No mechanism to handle uncertainity when a data point is close to more than 1 cluster centroid.
	c) Not sensitive to initial guess of centroids	c) Sensitive to initial guess of centroids. Different initializa's will give different clusters.
	d) It will analyze more complex data.	d) cannot analyze more complex data
	e) can handle outliers	e) outliers misrepresent the centroid
	✱f) Since in current example data is not in circular shape, hence K-means is not Good & Gaussian is good.	f) Data is in ⊙lar shape ⇒ K-means suits.
	g) Handles any shaped data.	g) clusters created are in ⊙lar shape [∵ centroid is updated using mean value]

Exponential distribution	Gaussian distribution	K means
h) Since the original data has a good exponential curve, exponential mixture model suits	All the cluster come at one location itself Hence not suited if the data density falls exponentially.	
i) For each random-initialization, the likelihood starts from minimum & increases as the number of iterations increases	Some random-initializ has the higher likelihood at the beginning itself or may also cross the maximum likelihood point.	
∴ No of random-initializ needed will be less when calculating maximum likelihood. (look at Graph attached)	⇒ No of random initialization needed is more when calculating maximum likelihood.	
j) converges early	j) converges little late	
k) The initial plot of the data & after the convergence are almost similar. It says exponential distribution has done good job!	The initial plot of the data & after the convergence differ (Cluster position) Gaussian distribution has done not pretty good job!	

∴ I Rank: Exponential distribution mixture model

II Rank: Gaussian distribution mixture model

III Rank: K-means model.

(2)

(a) Fitting a line to data / least squares / Linear regression.

Mean squared error on traing data is 0.03968

Mean squared error on testing set is 0.370

[Mean squared error on traing data] ≈ approximately equal [Mean squared error on testing data]

Therefor we can say that a line has been fitted to the data well. MSE should be less ⇒ line fitted to the data is having less error.

Let us see what is MSE?

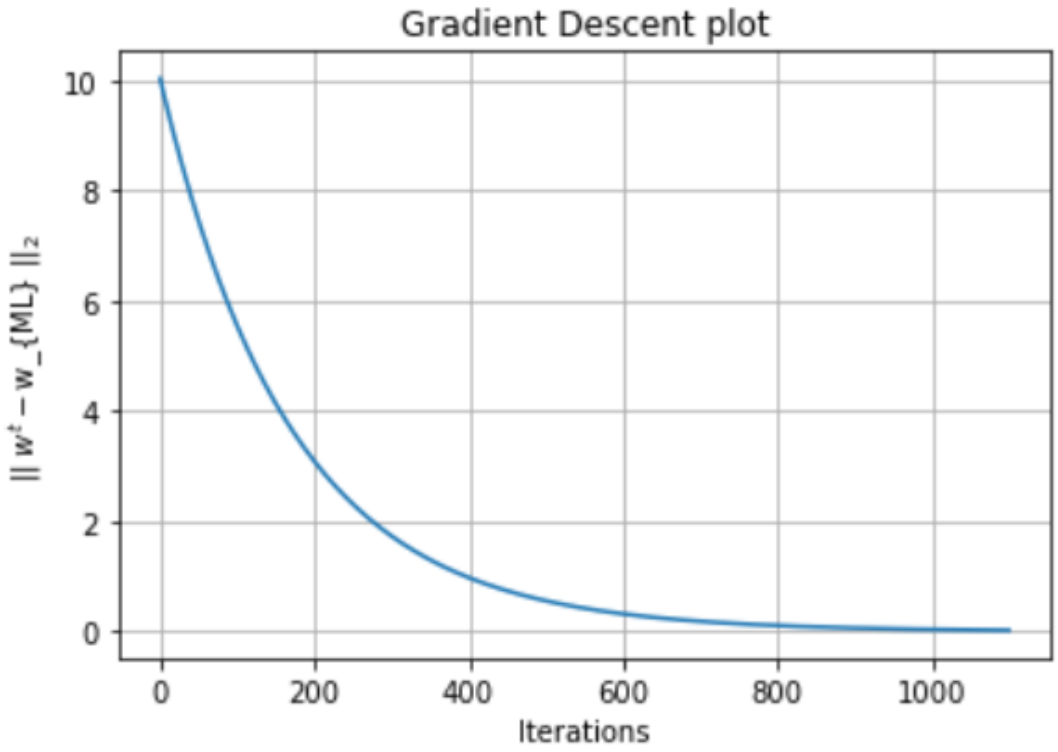
[Mean squared error] $MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$

N : The number of samples we are testing against

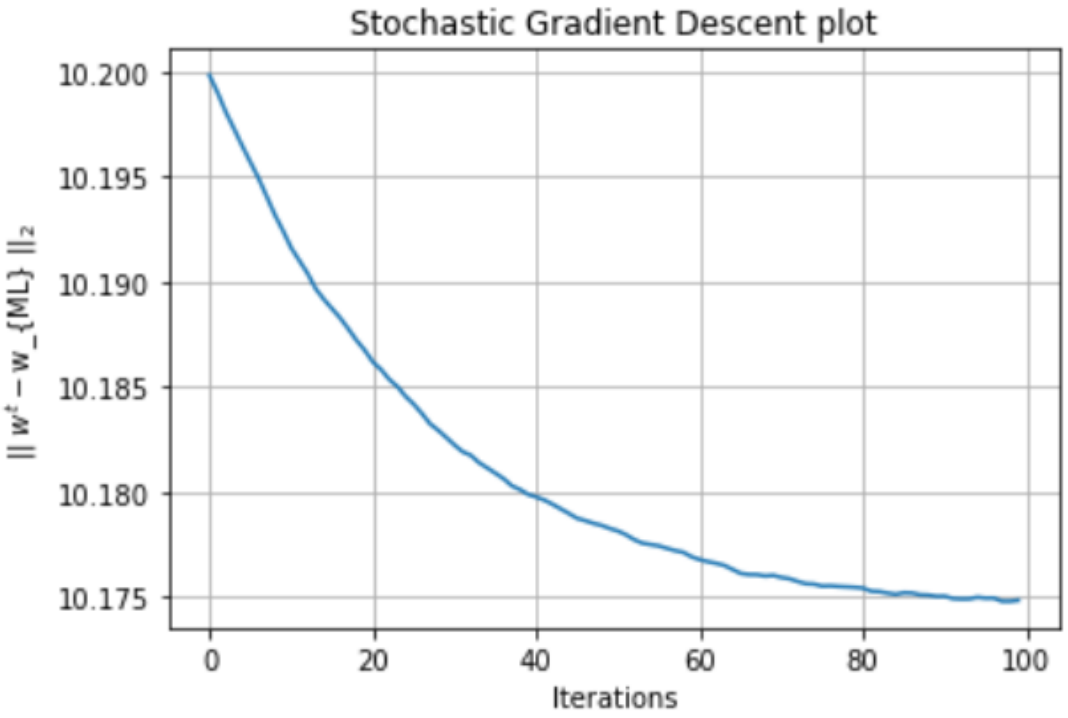
Y_i : predicted y value

Y_i : Actual y value

2) ii)


Gradient Descent plot
10
8
6
4
2
0
0
200
400
600
800
1000
Iterations
|| w^t − w_{ML} ||_2

2) iii)


Stochastic Gradient Descent plot
10.200
10.195
10.190
10.185
10.180
10.175
|| w^t − w_{ML} ||_2
0
20
40
60
80
100
Iterations

(2) (ii)

By looking at the gradient descent plot, we can say that the sum of squared residuals decreases as the number of iterations increases. At a particular point [sweet spot] we see that the squared residuals is very less [almost equal to 0] At this point we should stop calculating the sum of squared residuals.

$$\left[\begin{array}{l}\text{MSE on training}\\ \text{data set}\end{array}\right]_{\text{Gradient descent.}} = 0.370$$

Steps:

1. Define the step size.
2. calculate the MSE. If it is not equal to 0 calculate the step size. If MSE is approaching near to 0 then reduce the step size & calculate MSE again [Iterate]
3. If MSE is almost equal to 0 ⇒ stop

(2) (iii) By looking at the stochastic Gradient descent plot, we can say that the sum of squared residuals decreases as the number of iteration increases.

$$\left\{\begin{array}{l}\text{MSE on training data-set}\\ \text{for Gradient descent}\\ = 0.370\end{array}\right\} < \left\{\begin{array}{l}\text{MSE on training data-set}\\ \text{for stochastic Gradient}\\ \text{descent} = 6.688\end{array}\right\}$$

∴ Gradient descent does a good job! than stochastic Gradient descent. It does not do very bad also (not too much difference) 6.6 - 0.3 = 6.3

- Stochastic Gradient descent algorithm should be used when there is lot of redundancy in the data.
- Instead of calculating squared residuals for each data point. It calculates on batches (here 100 data points = 1 batch)

Steps for stochastic gradient descent

i) Make the data into batches.

ii) Define the step size.

iii) Calculate MSE on 1 batch.

iv) Check the MSE.

if MSE is approaching zero, then decrease the step size. i.e; update step size appropriately

v) Go to step iii) Iterate

vi) [MSE] $\approx$ [0] $\Rightarrow$ Stop.

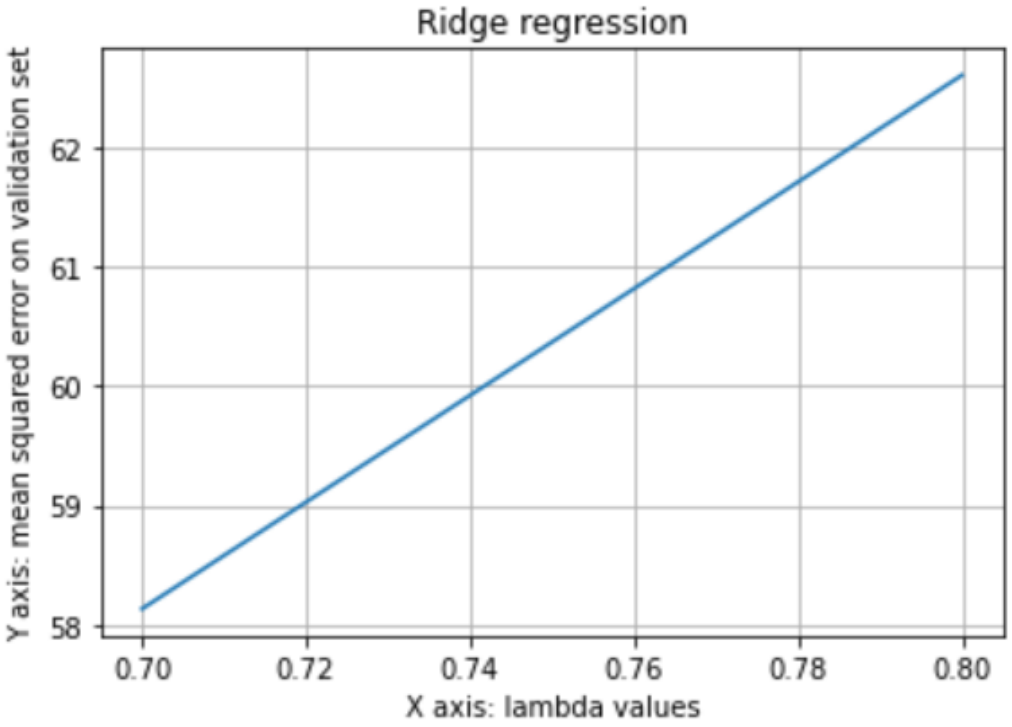
approximately equal to 0

Advantages:

i) stochastic gradient descent algorithm reduces lot of computation.

ii) Very good when there is lot of data redundancy. Or when data-points are very near to each other

2) iv)


Ridge regression
Y axis: mean squared error on validation set
62
61
60
59
58
0.70
0.72
0.74
0.76
0.78
0.80
X axis: lambda values

(2)
(iv)

Ridge Regression :-

By looking at the ridge regression graph :
as the λ-value increases, the MSE (mean squared error) also increases.

∴ Ridge regression for the current data set is of no-use.
Without applying ridge regression algorithm, the MSE was good.

$$\left[\begin{array}{l}\text{MSE without}\\ \text{Ridge regression}\end{array}\right] < \left[\begin{array}{l}\text{MSE with ridge}\\ \text{regression}\end{array}\right]$$

$$[0.3] < [8.86]$$

When Ridge Regression should be used???

- When the MSE is higher for the data set, then ridge regression should be used.

When the MSE will be higher???

- Sometimes, we may adjust & fit a line properly & thus decrease the MSE on the training data-set. ⇒ [over fitting]
- But after using the testing data-set, the MSE may increase abruptly. To reduce this abrupt increase & fit the line properly for both training & testing data, we use ridge regression.
- we try to tune the λ-value so that (MSE ≈ 0) MSE is almost equal to 0.

PTO

Steps for ridge regression:

(i) Define ~~step size~~ initial λ value.

(ii) Calculate MSE

$$[MSE]_{Ridge} = MSE + [\text{Ridge penalty (includes } \lambda \text{ parameter)}]$$

If MSE_{ridge} is decreasing as the lambda (λ) values increase, then ridge-regression is working bbb

if $[MSE_{ridge}] > 0$ continue calculating MSE_{ridge} for different λ values. (Ridge regression penalty)

Once the $[MSE_{ridge}] \approx [0]$ i.e., MSE_{ridge} is approximately equal to zero. ⇒ Stop.

(iii) plot the Graph.
