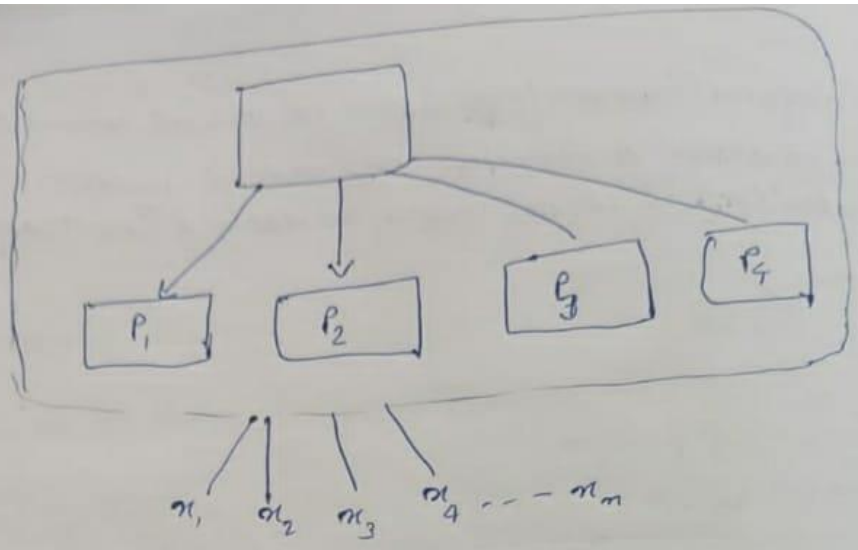
(1) You are given a data-set with 400 data points in {0, 1} 50 generated from a mixture of some distribution in the file A2Q1.csv. (Hint: Each datapoint is a flattened version of a {0, 1} 10×5 matrix.)

i. **(i) Determine which probabilisitic mixture could have generated this data (It is not a Gaussian mixture). Derive the EM algorithm for your choice of mixture and show your calculations. Write a piece of code to implement the algorithm you derived by setting the number of mixtures K = 4. Plot the log-likelihood (averaged over 100 random initializations) as a function of iterations.**

Solution

I have assumed that the data is the mixure of beroulli

The derivation of of this model is below.



Parameters which is need to be estimate

$\pi_1, \pi_2, \ldots \pi_K$

$P_1 = \{P_{11}, P_{12}, \ldots P_{1d}\}$

$P_2 = \{P_{21}, P_{22}, \ldots P_{2d}\}$

$P_K = \{P_{K1}, P_{K2}, \ldots P_{Kd}\}$

$$L(\theta) = \prod_{i=1}^{m} \left(\sum_{j=1}^{K} \left[\prod_{l=1}^{d} \left(P_{jl}^{x_{il}} (1-P_{jl})^{(1-x_{il})} \right) \right] \pi_j \right)$$

$$\log L(\theta) = \sum_{i=1}^{m} \log \left(\sum_{j=1}^{K} \left[\prod_{l=1}^{d} \left(P_{jl}^{x_{il}} (1-P_{jl})^{(1-x_{il})} \right) \right] \pi_j \right)$$

$$\text{mod-}\log L(\theta) = \sum_{i=1}^{m} \sum_{j=1}^{K} \lambda_j^i \log \left(\frac{\prod_{l=1}^{d} \left(P_{jl}^{x_{il}} (1-P_{jl})^{(1-x_{il})} \right) \pi_j}{\lambda_j^i} \right)$$

$$= \sum_{i=1}^{m} \sum_{j=1}^{K} \left[\lambda_j^i \log \pi_j + \lambda_j^i \log \left(\prod_{l=1}^{d} \left(P_{jl}^{x_{il}} (1-P_{jl})^{(1-x_{il})} \right) - \lambda_j^i \log \lambda_j^i \right] \right]$$

differentiate mod-$\log L(\theta)$ w.r.t P_{jl} and setting it to zero we obtain

$$\sum_{i=1}^{n} \left[\frac{\lambda_j^i x_{il}}{p_{jl}} - \frac{\lambda_j^i (1 - x_{il})}{(1 - p_{jl})} \right] = 0$$

$$\Rightarrow \boxed{\hat{p}_{jl} = \frac{\sum_{i=1}^{n} \lambda_j^i x_{il}}{\sum_{i=1}^{n} \lambda_j^i}}$$

to obtain π_j, we use method of lagrange multiplier

$$g(\pi) = \sum_{j=1}^{k} \pi_j = 1 = 0 \longrightarrow \text{constraint}$$

$$(\nabla g(\pi))_j = 1$$

$$(\nabla f(\pi))_j = \frac{\sum_{i=1}^{n} \lambda_j^i}{\pi_j} \qquad f \rightarrow \text{mod-log } L$$

Let λ^* be the lagrange multiplier then

$$(\nabla f(\pi))_j = -\lambda^* (\nabla g(\pi))_j \qquad \lambda^* \in R$$

$$\Rightarrow \frac{\sum_{i=1}^{n} \lambda_j^i}{\pi_j} = -\lambda^* \times 1$$

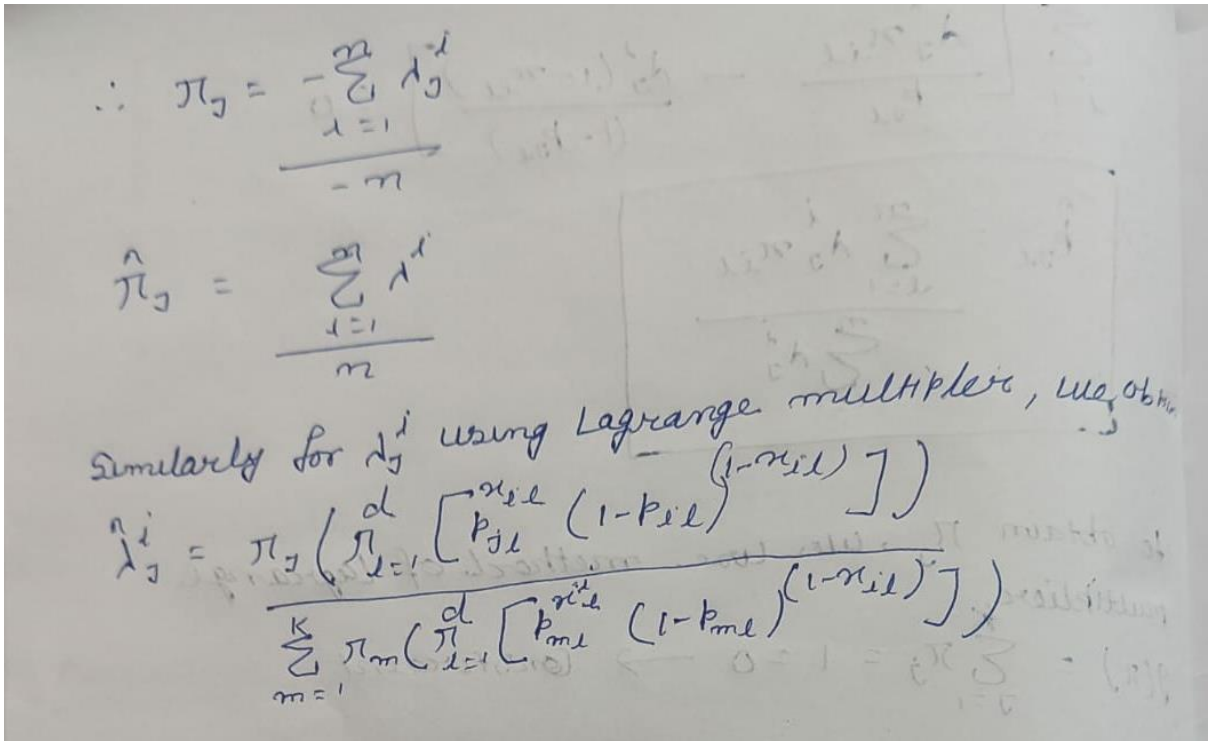
$$\Rightarrow \pi_j = \frac{-\sum_{i=1}^{n} \lambda_j^i}{\lambda^*}$$

Since $\sum_{j=1}^{k} \pi_j = 1$

$$\Rightarrow -\sum_{j=1}^{k} \sum_{i=1}^{n} \frac{\lambda_j^i}{\lambda^*} = 1$$

$$\Rightarrow -\sum_{i=1}^{n} \left(\sum_{j=1}^{k} \lambda_j^i \right) = \lambda^*$$

$$\Rightarrow \lambda^* = -\sum_{i=1}^{n} 1 = -n \qquad \left[\text{Since } \sum_{j=1}^{k} \lambda_j^i = 1 \right]$$



first I initialize p for each mixture and pi for each mixture and then calculate lemda for each data of each mixture and calculate the log likelihood and again from lemda I update theta(p and pi).

And calculate log likelihood so after each updation I am calculating log likelihood.

What I observed that- in each iteration the log likelihood is increasing till convergence. And ideally it should increase because we are maximizing the log likelihood function

And after convergence we will get our cluster. The data which have maximum probability for a cluster will be assigned to that cluster.

The fig-1(a) is the plotting of log likelihood with respect to iteration

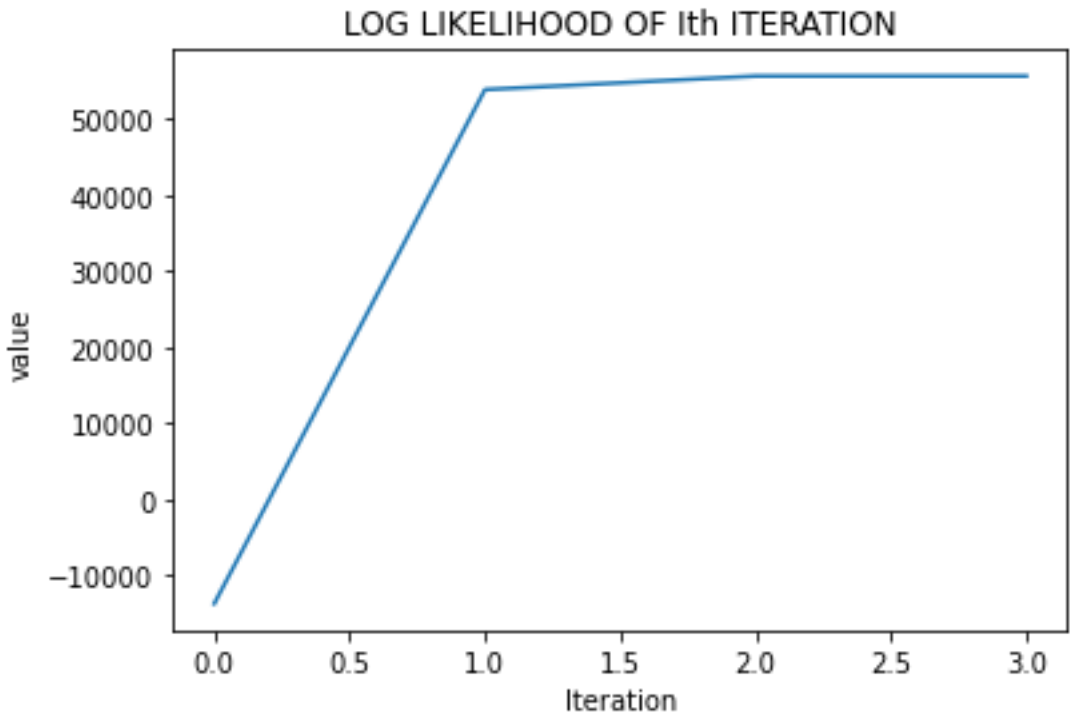


Fig-1(a)

ii. **(ii) Assume that the same data was infact generated from a mixture of Gaussians with 4 mixtures. Implement the EM algorithm and plot the log-likelihood (averaged over 100 random initializations of the parameters) as a function of iterations. How does the plot compare with the plot from part (i)? Provide insights that you draw from this experiment.**

The formula for parameter of gaussian mixture is given below.

for multivariate Gaussian distribution we have.

$$\log L(\theta) = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{K} \pi_j \frac{e^{-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j)}}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \right)$$

$$\hat{\pi}_j = \frac{\sum_{i=1}^{n} \lambda_j^i}{n}$$

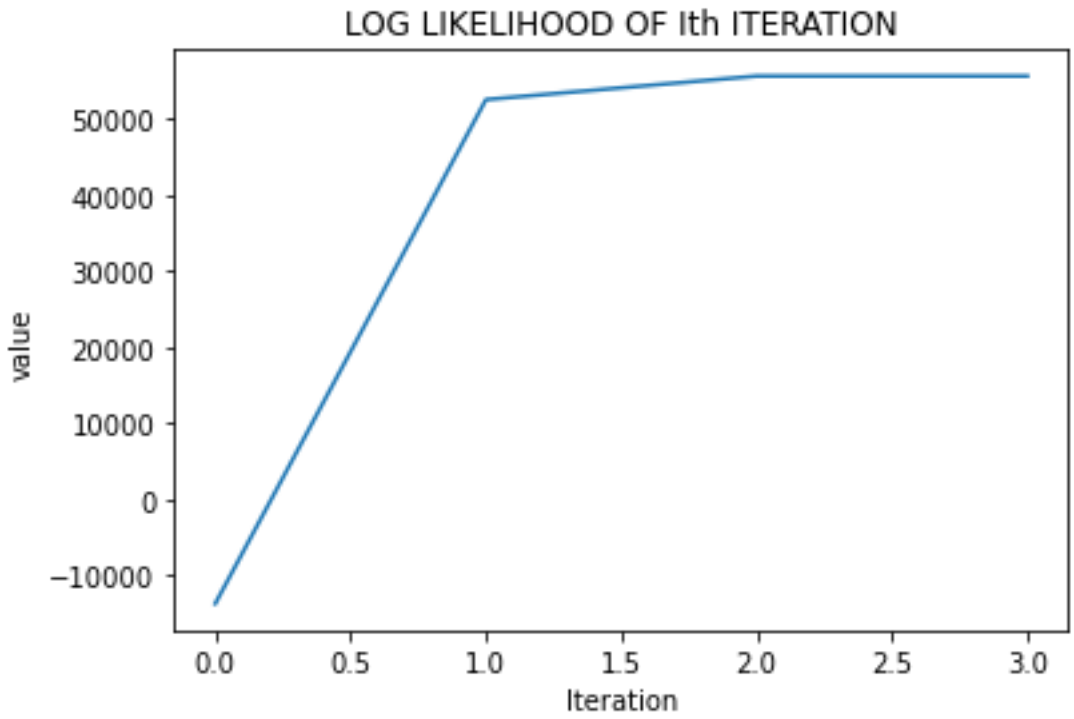
$$\hat{\mu}_j = \frac{\sum_{i=1}^{n} \lambda_j^i x_i}{\sum_{i=1}^{n} \lambda_j^i}$$

$$\hat{\Sigma}_j = \frac{1}{\sum_{i=1}^{n} \lambda_j^i} \cdot \sum_{i=1}^{n} \lambda_j^i (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T$$

$$\lambda_j^i = \frac{\frac{1}{(2\pi)^{d/2} |\Sigma_j|^{1/2}} \cdot e^{-\frac{1}{2}(x_i - \mu_j)^T \Sigma_j^{-1} (x_i - \mu_j)}}{\sum_{l=1}^{K} \left(\frac{1}{(2\pi)^{d/2} |\Sigma_l|^{1/2}} \cdot e^{-\frac{1}{2}(x_i - \mu_l)^T \Sigma_l^{-1} (x_i - \mu_l)} \right)}$$

(10)

ideally we should get the log likelihood vs iteration like the figure shown below.



iii. Run the K-means algorithm with K = 4 on the same data. Plot the objective of K – means as a function of iterations.

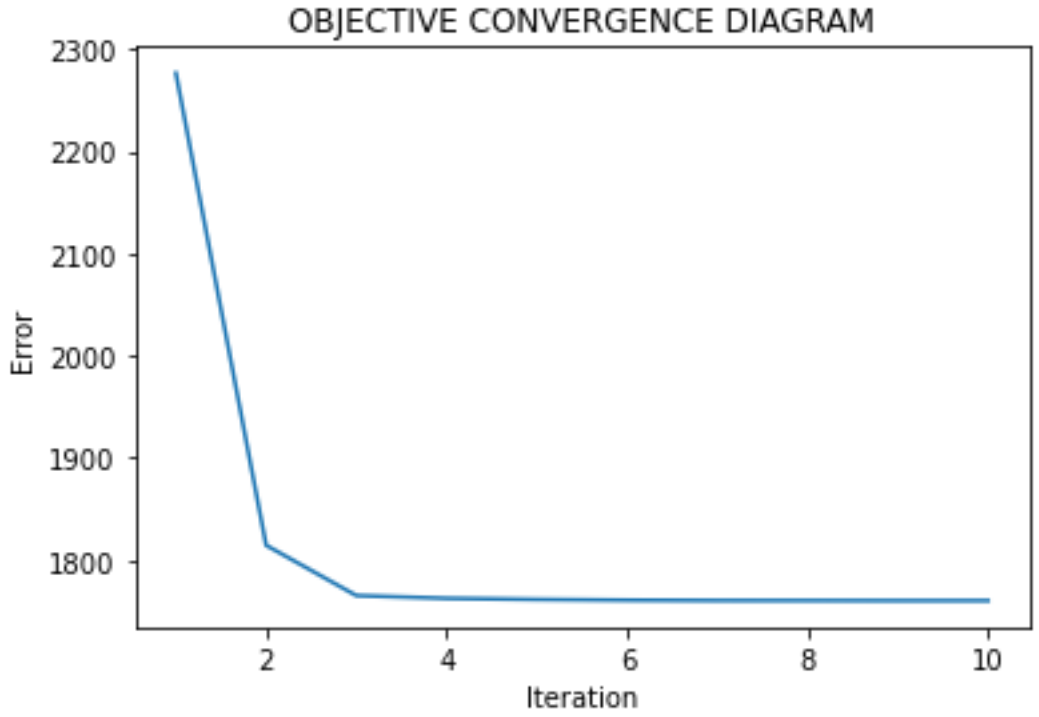
Solution

```
Objective function value is 1760.405994695705 after convergence
```

the below diagram is the objective function convergence diagram.

On the x-axis, I have taken the iteration number and on the y-axis, I have taken the objective function value of that iteration.

Here we can clearly see that the objective function value is decreasing at each iteration till convergence.



iv. Among the three different algorithms implemented above, which do you think you would choose to for this dataset and why?

For this data set I will prefer mixture of Bernoulli which I have used in first question.

First if we analyse the data then we see that each data is made by 0,1.

Gaussian mixture is suiltable for real number generated data but in this case data is generated either by 1 or 0. So I think gaussian mixture is not suitable for this dataset.

Now coming to k-means, k-means do the hard clustering means it directly assign the data to a cluster without any probability. But in case of Bernoulli mixture with Em algorithm we are doing soft clustering means the data is assigned to a cluster with some probability and hence it is more

useful . Further we can use it for hard clustering also by assigning a data to a cluster for which it has more probability.

Hence we can say that Bernoulli is best suited algorithm for this data set

(2) You are given a data-set in the file A2Q2Data train.csv with 10000 points in (R 100 , R) (Each row corresponds to a datapoint where the first 100 components are features and the last component is the associated y value).

i. Obtain the least squares solution wML to the regression problem using the analytical solution.

Solution

the least squares solution wML to the regression problem using the analytical solution is given by

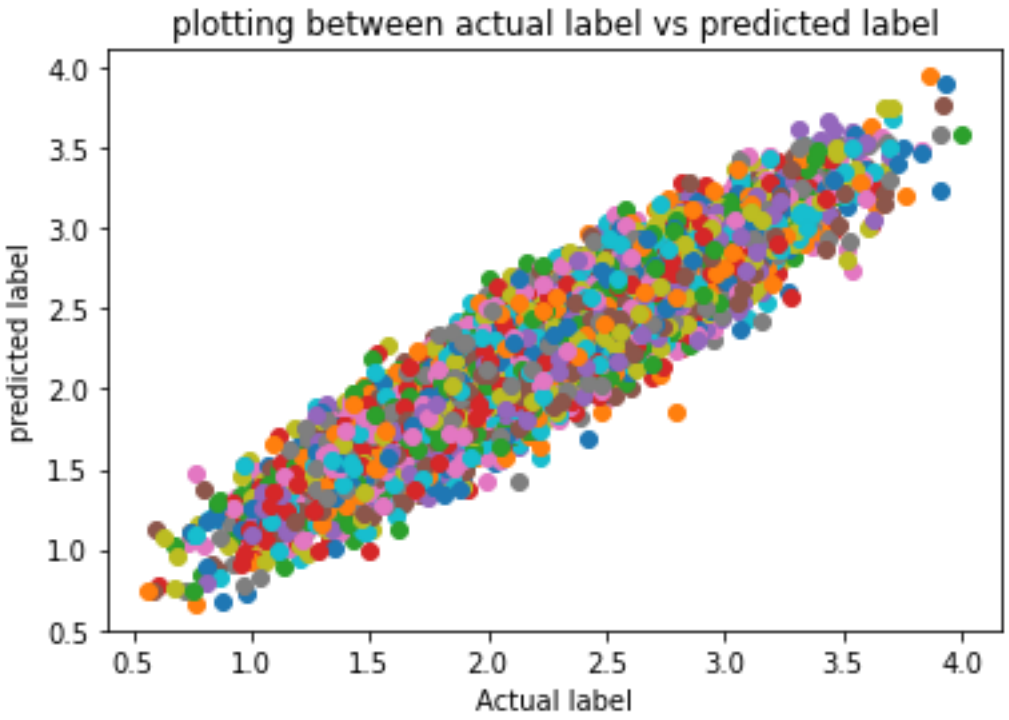
$$w = (X^T X)^{-1} X^T y$$

I have added an extra feature with value=1 to every data point. And the reason is to fit the intercept.

And hence the W we get is of 101 dimesion.

The below diagram is between the predicted Actual label and predicted label of the each data.

The X-axis is represented as Actual label and the Y-axis is represented as the predicted label



ii. Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot ||wt – wML||2 as a function of t. What do you observe?

Solution

First I take a W which is randomly initialilized. Each time my code will run then it will initialized randomly

I took step size=0.1

At each iteration ||Wt-Wml|| is decreasing that means the Wt is going near to Wml at each iteration and after some iteration it will reach very near to Wml.

The diagram for ||Wt-Wml|| with iteration is shown in below diagram 2(b)

In diagram also we can see that the ||Wt-Wml|| at iteration number 1000 is approx 0 that means the Wt is very close to Wml after 1000 iteration

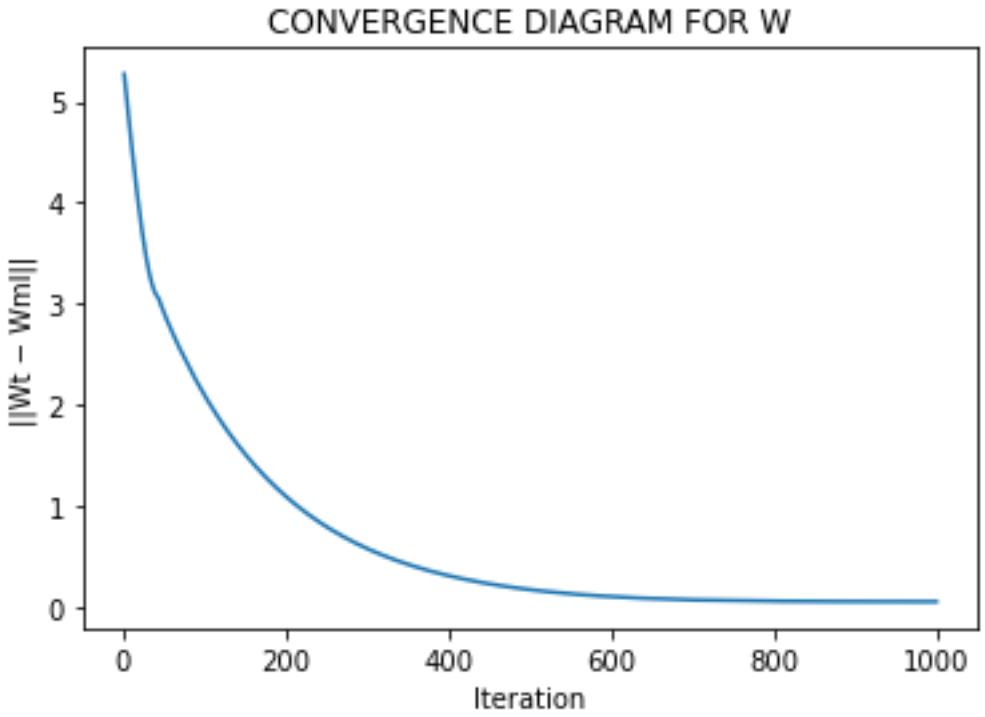


Fig-2(b)

iii. Code the stochastic gradient descent algorithm using batch size of 100 and plot ||wt – wML||2 as a function of t. What are your observations?

Solution

since batch size is 100 hence at each iteration I am taking 100 random data from dataset and calculating Wt, ||Wt-Wml|.

Since sometime we get good sample and sometime we will get bad sample and hence we will Sometime get Wt which is very near to Wml and sometimes we will get Wt which is very far Away from Wml.

If we take average of all the Wt then it will be near Wml

In the Fig-2(c) we can see that Wt is fluctuating in each iteration, as at each iteration Wt

Depends upon the random sample

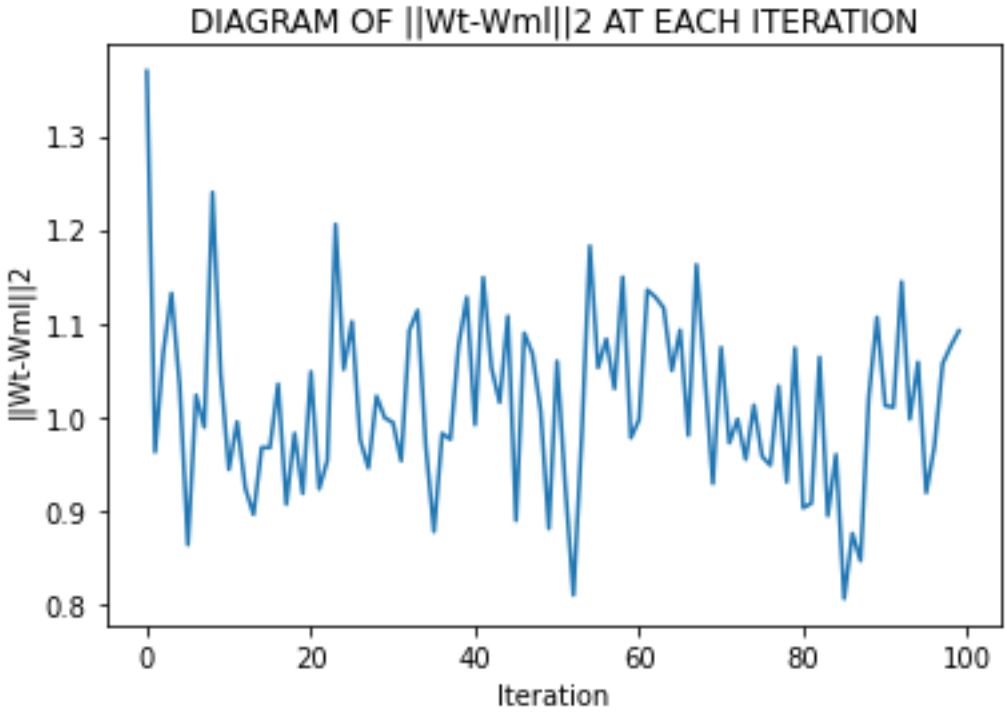


Fig-2(c)

iv. Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of λ and plot the error in the validation set as a function of λ. For the best λ chosen, obtain wR. Compare the test error (for the test data in the file A2Q2Data test.csv) of wR with wML. Which is better and why?

Solution

In the code I have checked for lemda between 0 to 100. And for each lemda the W0(initial random W for gradient descent) is same so that I can compare all lemdas for minimum error.

Since each time the code is executed, W0 is random hence we might get different best lemda(lemda for which the corresponding W gives minimum error on training data)

An example of best lemda----

When I last executed my code i got best lemda=6

Below figure 2(d) is the ploting of the error in the validation set as a function of λ

This is same plot for which lemda=6 gives minimum error.

So we assume this as best lemda and compare with Wml

And after checking error for Wml and Wr on test data I got the below result

```
square error of W_ml on test data is  185.3757511443317

square error of W_r on test data is  184.12446358170067
```

so from the result we can observe that even if the Wml gives least error on training data but Wr gives lesser error on test data
hence I can say that Wr perform better than Wml on test data.

One more thing I observe is that when I got lemda=0 as best lemda then in that case I am getting Test error of Wml is equal to test error of Wr.

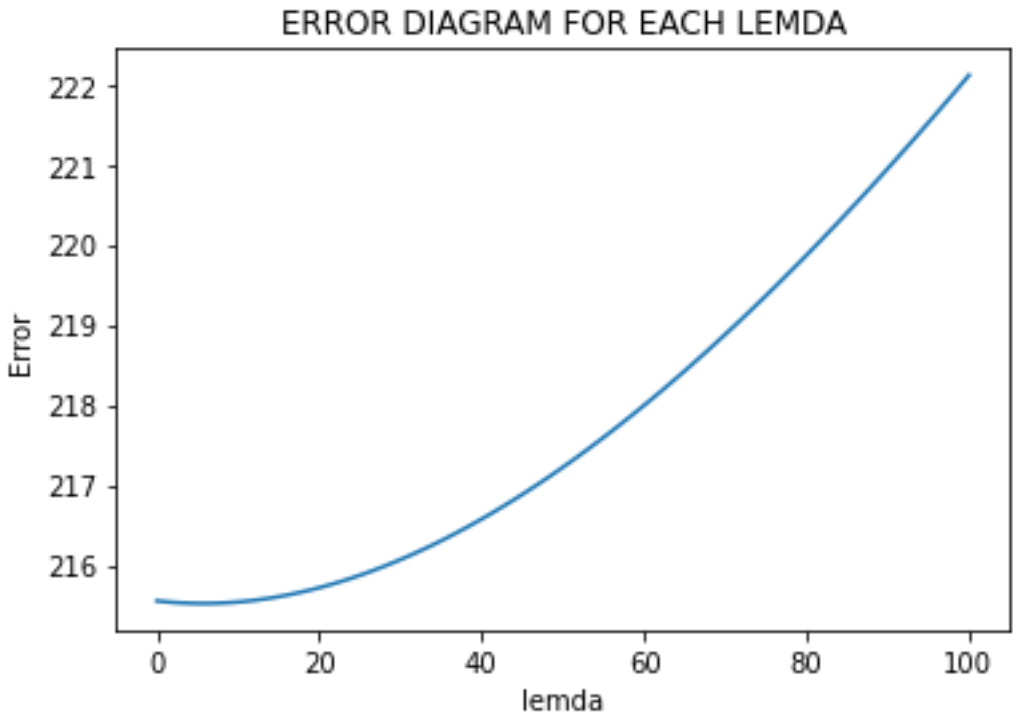


Fig-2(d)