(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 $\{0,1\}^{10\times 5}$ matrix.)

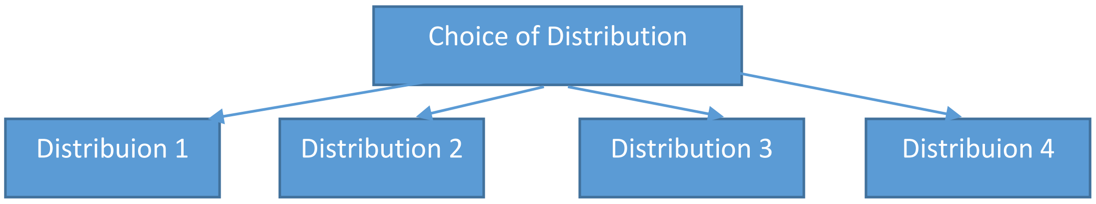
i. (i) Determine which probabilisitic *mixture* could have generated this data (It is not a Gaussian mixture). Derive the EM algorithm for your choice of mixture and show your calculations. Write a piece of code to implement the algorithm you derived by setting the number of mixtures $K = 4$. Plot the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

Ans: Question1(i).py

Model assummed: Two level estimation model with:

i. Level one choosing out of mixture of bernoulli and
ii. the other level is generating the data from that bernoulli.

Now, each of the bernoulli distribution functions as toss of 50 coins to generate each data.



For our notation, we denote the probabilities of each coins to fall on heads to be: p_k^d

Where, k ranges from 0 to 3(corresponding to the 4 clusters). And d ranges from 0 to 49 (corresponding each dimension for every data)

For instance, the distribution 1 comprises of (p_1^1 to p_1^{50}) and varying the subsctript k for every other distributions.

Now, we have to obtain:

i. Probability of the generation of the data: P(data)
ii. Probability of a particular distribution gets chosen: $P(Z_i=k) = \pi_k$
iii. Probability of each of the bernoulli variables: p_k^d
iv. Also the constants that we will be imposing in modified log likelihood, for each data in every cluster: λ_k^i

We begin with defining our likelihood:

L(parameters;data) = P(data/parameters)

$$= \Pi_{i=1}^{400} \Sigma_{k=1}^{4} \pi_k \Pi_{d=1}^{50} (p_k^d)^{(data[i][d])}(1-p_k^d)^{(1-data[i][d])}$$

Now, to estimate parameters, we take the logarithm of the likelihood.

Log(L(parameters;data)) =

$$\Sigma_{i=1}^{400} Log(\Sigma_{k=1}^{4} \pi_k \Pi_{d=1}^{50} (p_k^d)^{(data[i][d])}(1-p_k^d)^{(1-data[i][d])})$$

Hereafter, we introduce λ_k^i, to be able to get a lower bound which we can then differentiate partially and obtain estimations for parameters.

Therefore, Modified Log(L(parameters;data)) =

$$\Sigma_{i=1}^{400} Log(\Sigma_{k=1}^{4} \lambda_k^i (\pi_k \Pi_{d=1}^{50} (p_k^d)^{(data[i][d])}(1-p_k^d)^{(1-data[i][d])})/ \lambda_k^i)$$

$$= \Sigma_{i=1}^{400} \Sigma_{k=1}^{4} \lambda_k^i Log(\pi_k (\Pi_{d=1}^{50} (p_k^d)^{(data[i][d])}(1-p_k^d)^{(1-data[i][d])})/ \lambda_k^i)$$

Differentiating this above equation partially with respect to λ_k^i fixing p_k^d & π_k we obtain the equations. Here is the complete derivation as follows:

$$L = \prod_{i=1}^{m} \sum_{K=1}^{K} \pi_K \prod_{d=1}^{50} (p_K^d)^{f_d^i} (1-p_K^d)^{(1-f_d^i)}$$

$$\log L = \sum_{i=1}^{m} \log \sum_{K=1}^{K} \pi_K \prod_{d=1}^{50} (p_K^d)^{f_d^i} (1-p_K^d)^{(1-f_d^i)}$$

$$= \sum_{i=1}^{m} \sum_{K=1}^{K} \lambda_K^i \log \left(\frac{\pi_K \prod_{d=1}^{50} (p_K^d)^{f_d^i} (1-p_K^d)^{(1-f_d^i)}}{\lambda_K^i} \right)$$

let

$$\omega = \pi_K \prod_{d=1}^{50} (p_K^d)^{f_d^i} (1-p_K^d)^{(1-f_d^i)}$$

$$f(\lambda_K^i) = \lambda_K^i \log(\omega) - \lambda_K^i \log \lambda_K^i$$

$$g(\lambda^i) \Rightarrow \sum_{K=1}^{K} \lambda_K^i - 1 = 0$$

let, $\lambda_K^i = e^{\theta_K^i}$

$$\therefore f(\theta_K^i) = e^{\theta_K^i} \log(\omega) - e^{\theta_K^i} \theta_K^i$$

$$\Rightarrow f'(\theta_K^i) = e^{\theta_K^i} \log(\omega) - e^{\theta_K^i} - \theta_K^i e^{\theta_K^i}$$

$$\& \quad \nabla(g(\theta_K))_K = e^{\theta_K}$$

by <u>lagrange</u>:

$$\cancel{e}^{\theta^i_k}(\log w) - \cancel{e}^{\theta^i_k} - \theta^i_k \cancel{e}^{\theta^i_k} = \cancel{e}^{\theta^i_k}(-y)$$

or, $-y = \log c - 1 - \theta^i_k$

or, $\theta^i_k = (y-1) + \log w$

or, $e^{\theta^i_k} = e^{y-1} \cdot w$

or, $\sum_{k=1}^{K} e^{\theta^i_k} = \sum_{k=1}^{K} e^{(y-1)} \cdot w$

or, $e^{(y-1)} \sum_{k=1}^{K} w = 1$

or, $e^{y-1} = \dfrac{1}{\sum_{k=1}^{K} w}$

Now,

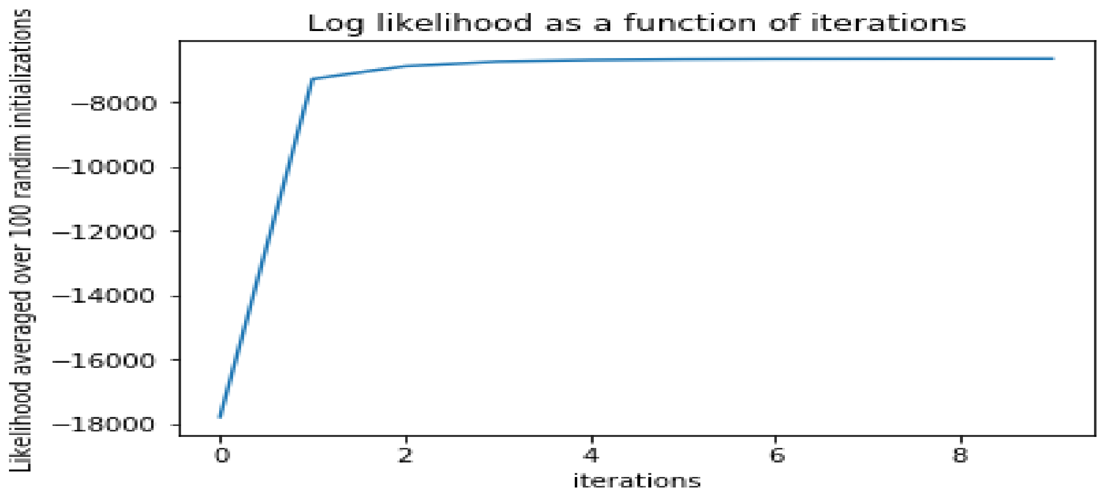
$$\log w - 1 - \vartheta_k^i = -y$$

$$\Rightarrow \vartheta_k^i - \log w = y - 1$$

$$\Rightarrow \frac{e^{\vartheta_k^i}}{w} = e^{y-1}$$

$$\Rightarrow e^{\vartheta_k^i} = \frac{w}{\sum_{k=1}^{K} w}$$

$$\Rightarrow \vartheta_k^i = \frac{\pi_k \prod_{d=1}^{50} (p_k^d)^{f_d^i} (1-p_k^d)^{(1-f_d^i)}}{\sum_{l=1}^{K} \pi_l \prod_{d=1}^{50} (p_k^d)^{f_d^i} (1-p_k^d)^{(1-f_d^i)}}$$



The plot of log likelihood as a function of iterations is presented above.

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.

Ans: Question1(ii).py

We use a multivariate gaussian distribution to generate the data in the following model.

As per our assumption, the individual mixtures has means as: (μ_1 ,μ_2,μ_3 & μ_4)

Also the covariances are assummed to be: (Σ_1, Σ_2, Σ_3, Σ_4)

The likelihood is as follows:

$$\prod_{i=1}^{n} \sum_{K=1}^{K} \pi_K \frac{1}{(2\pi)^{d/2} |\Sigma_K|^{1/2}} e^{-1/2(data_i - \mu_K)^T \Sigma_K^{-1} (data_i - \mu_K)}$$

Therefore the Log Likelihood becomes:

$$\sum_{i=1}^{n} \log\left(\sum_{K=1}^{K} \frac{\pi_K}{(2\pi)^{d/2} |\Sigma_{K}|^{1/2}} e^{-1/2 (data_i - \mu_K)^T \Sigma_{K}^{-1} (data_i - \mu_K)}\right)$$

Converting it into modified log likelohood is:

$$\sum_{i=1}^{n} \sum_{K=1}^{K} \lambda_K^i \log\left(\frac{\frac{\pi_K}{(2\pi)^{d/2} |\Sigma_{K}|^{1/2}} e^{-1/2 (data_i - \mu_K)^T \Sigma_{K}^{-1} (data_i - \mu_K)}}{\lambda_K^i}\right)$$

Partially differentiating the modified log likelihood to obtain different parameters gives:

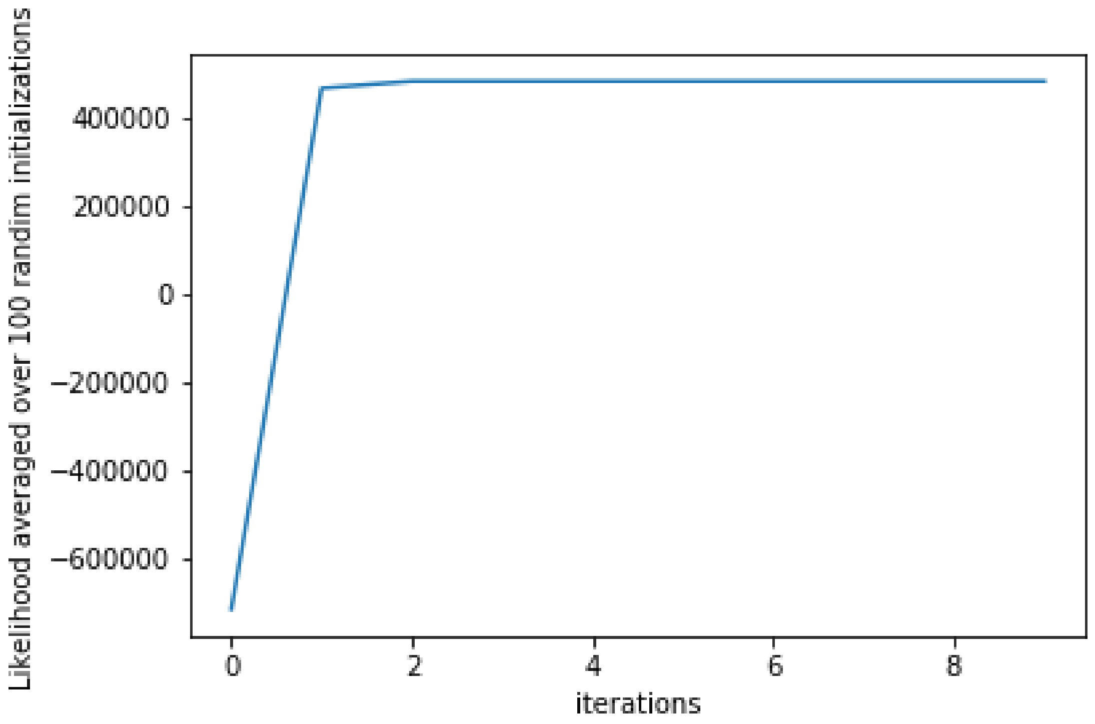
$$\hat{\pi}_K^{MML} = \frac{\sum_{i=1}^{n} \lambda_K^i \, data_i}{\sum_{i=1}^{n} \lambda_K^i}$$

$$\hat{\Sigma}_K^{MML} = \frac{\sum_{i=1}^{n} \lambda_i^K (data_i - \hat{\mu}_K^{MML}) (data_i - \hat{\mu}_K^{MML})^T}{\sum_{i=1}^{n} \lambda_i^K}$$

$$\hat{\mu}_K^{MML} = \frac{\sum_{i=1}^{n} \lambda_i^K \, data_i}{\sum_{i=1}^{n} \lambda_i^K}$$

$$\hat{\lambda}^{i}_{k}{}^{MML} = \frac{\pi_k \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} e^{-1/2 (data_i - \mu_k)^T \Sigma_k^{-1} (data_i - \mu_k)}}{\sum_{l=1}^{K} \frac{1}{(2\pi)^{d/2}|\Sigma_l|^{1/2}} e^{-1/2 (data_i - \mu_l)^T \Sigma_k^{-1} (data_i - \mu_l)}}$$

The plot of log-likelihood (averaged over 100 random initializations of parameters) as a function of iterations is as follows:



Comparision with the plot from part (i):

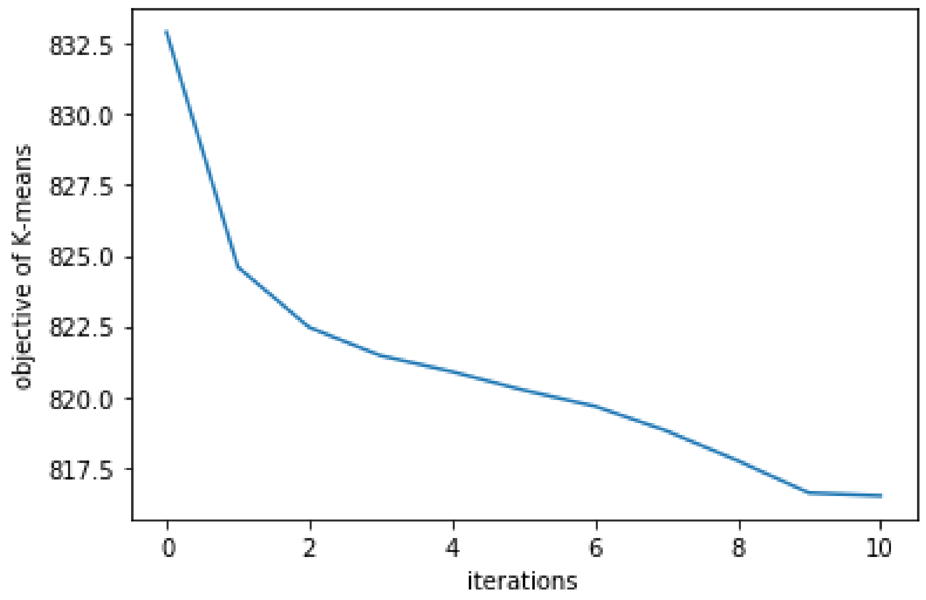
a. The log likelihood reaches to a higher value in the second model than the first.
b. The increase in the log likelihood is monotonus in this case, which was not quite there in the first.
c. The convergence in the log likelihood is comparatively faster in the second case.

<u>Insights:</u>

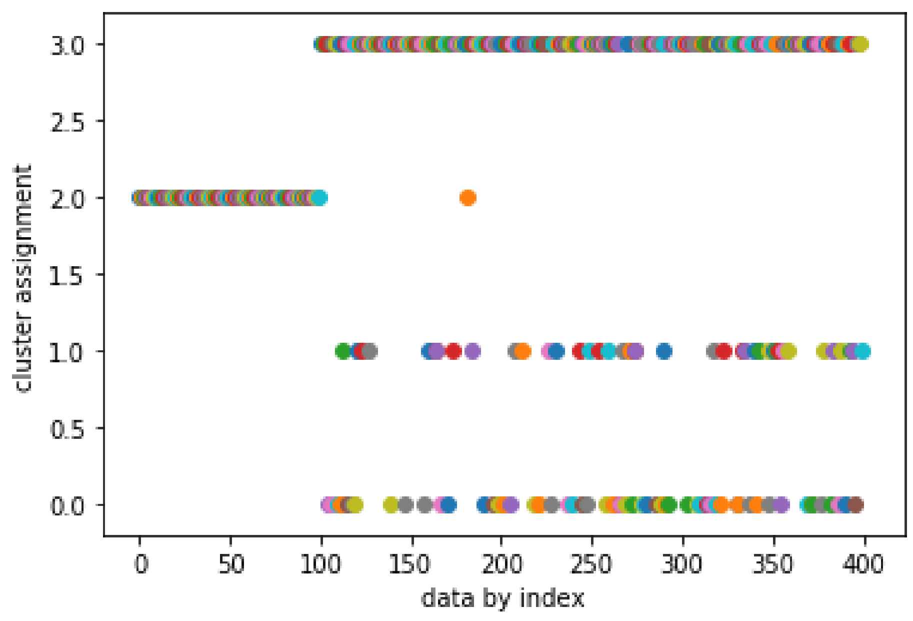
iii. Run the K-means algorithm with $K = 4$ on the same data. Plot the objective of $K - means$ as a function of iterations.

Ans: Question1(iii).py

The plot of the objective of K-means as a function of iterations is as follows:



Assignment of each data points to their clusters is depicted as follows:



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

Ans: I would choose the mixture of bernoulli model as a generative model for this data. Since the data is a vector of either 1 or zero, it makes better sense to assume a bernoulli generative story. Also here if we try to assign hard-clustering of the data points with those clusters for which it has highest lambda, we can obtain our objective as the sum of the distances of the datas from their means. That value averaged over 100 random initializations comes as: 823.378. Which is pretty close to the naïve K-Means (817.84). We get a Lambda, that is soft clustering by our generative model for each point, which gives preference to this model over K-Means. In the other model(Gaussian Mixture) we get the sum of distances as: 930.368 which is significantly larger than the two.

(2) You are given a data-set in the file A2Q2Data_train.csv with 10000 points in $(\mathbb{R}^{100}, \mathbb{R})$ (Each row corresponds to a datapoint where the first 100 components are features and the last component is the associated y value).

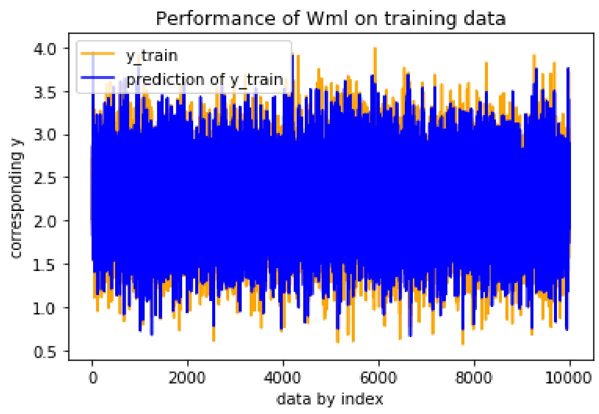
i. Obtain the least squares solution $\mathbf{w}_{ML}$ to the regression problem using the analytical solution.

Question2.py

Ans: The least squares solution W_{ML} to the regression problem is calculated by the following equation(analytical solution):

$$\hat{w}^{ML} = (XX^T)^{-1} Xy$$

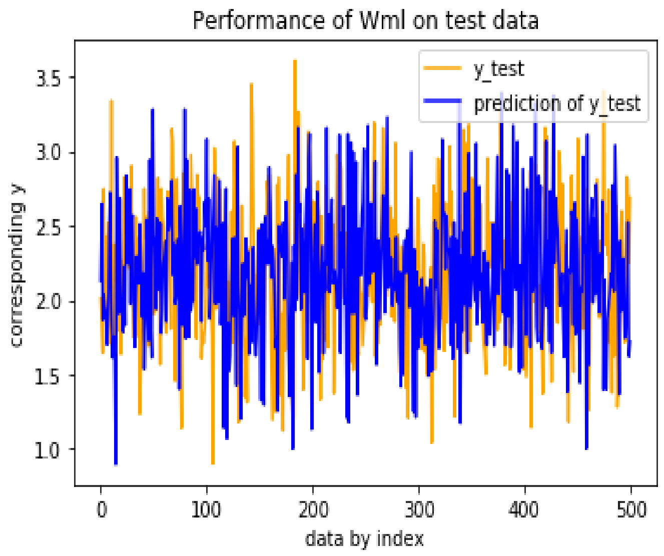
Here, X corresponds to the data in (d*n) format where d is the dimension and n is the number of data points. Y is the corresponding dependent variable.



Here, our observation is the performance of Wml on train data is very close to the corresponding y_train, i.e the approximation of y value is very close to the actual y value for train data with Wml.

Also, averaging the train error over the number of data points we can find the norm to be:

0.001992145623761605

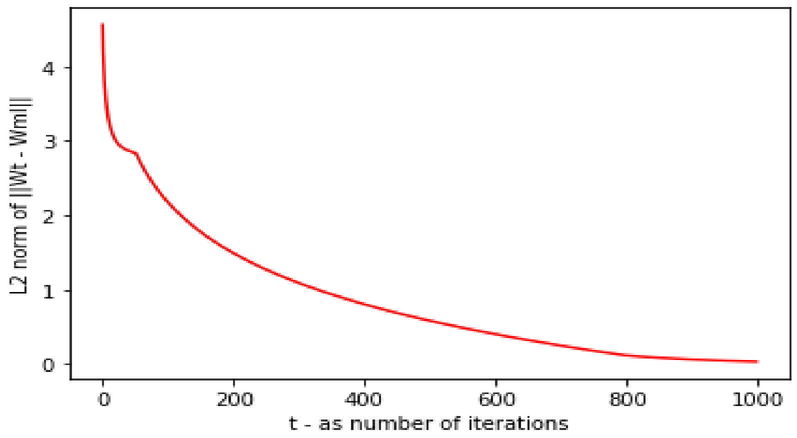


Here, we observe the performance of Wml on <u>test data</u>. This gives us an average error of:

0.02722966438169207

ii. Code the gradient descent algorithm with suitable step size to solve the least squares algorithms and plot $\|\mathbf{w}^t - \mathbf{w}_{ML}\|_2$ as a function of t. What do you observe?

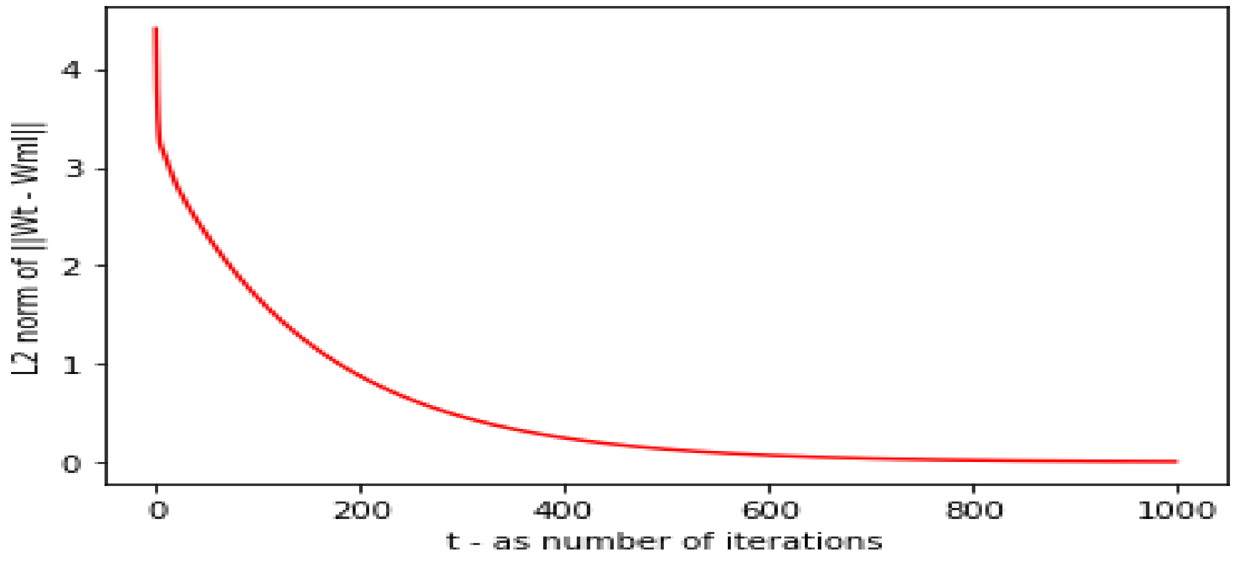
Ans: <u>Step Size</u> = (1/t) where t is taken in the range of 1 to 1000 as per the iterations. The plot of $||Wt – Wml||_2$ as a function of t is presented below:



Here, the optimality of the Wt (i.e the solution of gradient descent at each step) is achieved gradually. As per the iteration progresses the difference with Wml gets minimised. This suits our assumption.

The least L2 norm achieved by the $W_{(gradient\ descent)}$ from Wml is 0.0347325 in 1000th iteration, taken (1/t) as step size. Also, tried with increased step size as (2/t) as t is the number of iteration. And, with 1000 iterations we observe the following plot of L2 norm of ||Wt – Wml||

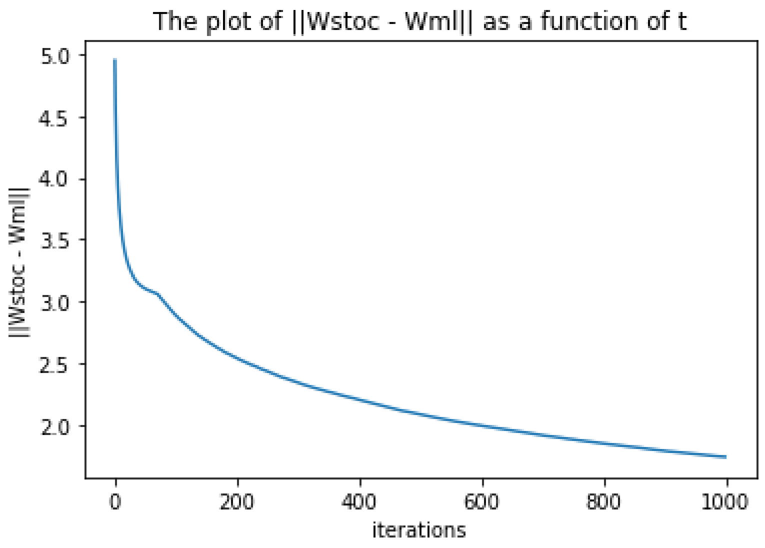
as a function of t.



Here the least L2 norm we are able to get is: 0.0065722 at our 1000th iteration.

iii. Code the stochastic gradient descent algorithm using batch size of 100 and plot $\|\mathbf{w}^t - \mathbf{w}_{ML}\|_2$ as a function of t. What are your observations?

Ans:

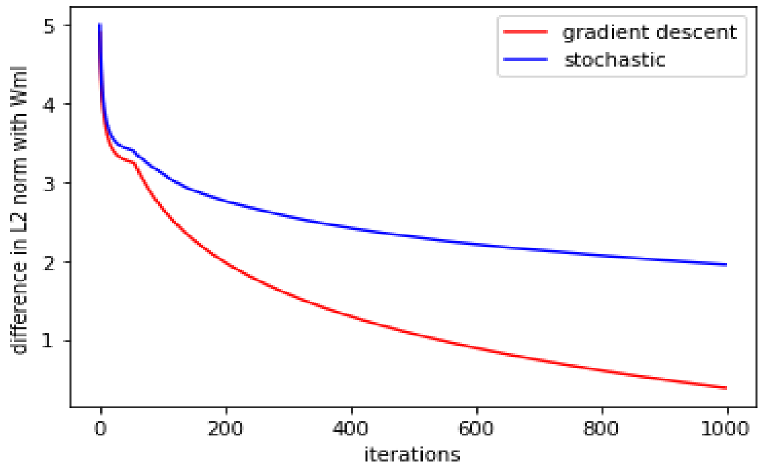


Here, the convergence of the L2 Norm with Wml of the Wstochastic is comparatively slower than the **gradient descent** that we performed earlier. L2 norm from Wml at 1000th iteration is around 2.23048, with step size (1/t) where t is the number of iterations.

So quite expectedly it does not perform as good as Gradient Descent performed using the same step size following the same number of iteartions.

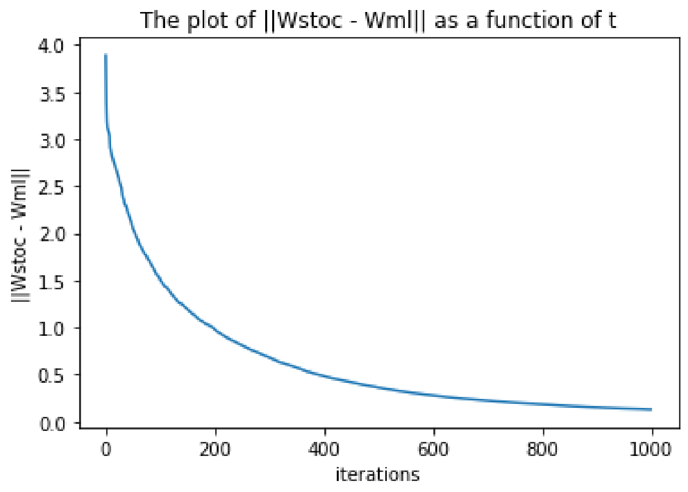
Also the Stochastic Gradient descent approaches to its minimum difference with Wml monotonically because the objective function is quadratic. So with progress every updation is leading to lower the L2 norm, i.e getting the Wstochastic nearer to Wml.

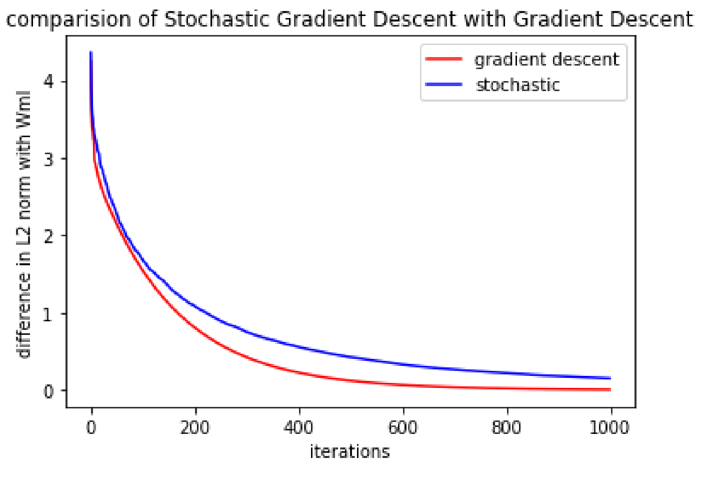
Comparision with respecet to gradient descent is found to be as follows, plotted the L2 norm with Wml as a function of iterations:



Here, we observe the difference in the final norm along with its propagation throughout the 100 iterations.

With step size (2/t) where t is the number of iterations, we produce the following plot of the norm as a function of the iteartions, here we observe that the L2 difference becomes lower as compared to step size (1/t) such that at 1000th iteration, the minimum L2 difference we get is 0.12888. Though this is still not as good as Gradient Descent's performance using same parameters but this is a significant improvement over the previous step size on Stochastic Gradient Descent. Also from the comparision plot it is also notable that the difference is very minimum between the two with this step size.





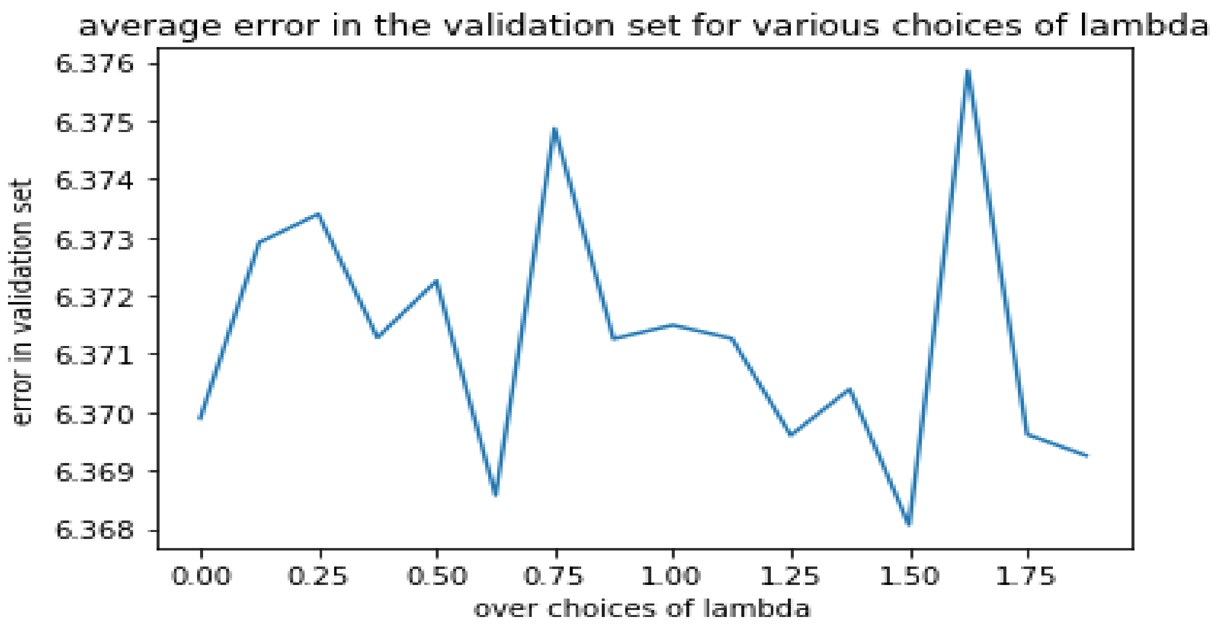
iv. Code the gradient descent algorithm for ridge regression. Cross-validate for various choices of λ and plot the error in the validation set as a function of λ. For the best λ chosen, obtain $\mathbf{w}_R$. Compare the test error (for the test data in the file A2Q2Data_test.csv) of $\mathbf{w}_R$ with $\mathbf{w}_{ML}$. Which is better and why?

Ans: Choices of labmda: Lambda has been chosen between the range of 0 to 2 with a 0.125 difference between each.

Cross-validation: Here, K-fold cross validation is applied with k = 10.

Also, the step size is chosen to be (2/t), with t as the number of iterations in the gradient descent step.

Following is the plot of the error in the validation set as a function of λ.



Test error of Wridge with Wml: (averaged over number of data points in test data)

0.02720635668276448

Comparing with test error of Wml:

(Error obtained by Wridge – Error obtained by Wridge) = [-2.33076989e-05]

Therefore our observation is that the error we obtain by Wridge is lower in test data in comparision with Wml. Also our choice of lambda does not improve the Wridge by a wide margin, the difference is in the order of 10^{-5}.