For this assignment, I have used Google Colab environment for running the programs. I have used Python as the programming language and Numpy library extensively as it offers better computational advantages over traditional containers of python3 and Matplotlib to plot diagrams.

The following versions are used -

Python 3.9.16

Numpy 1.22.4

1. In this bit of question, we are given a data set that contains (10000 X 101) data points, and the last column contains the labeled data for the previous 100 dimensions. In simple words, the dimension of our input dataset will be (10000 X 100), 10000 data points of 100 dimensions each, and the output dimension will be (10000 X 1), 10000 points of each one dimension.

1.1 The question expects to obtain the least square solution or the closed-form solution of linear regression obtained from the maximum likelihood operator.

Our objective function

$$\text{arg min of w, } \sum_{i=1}^{n} (w^T x_i - y_i)^2 \text{ ; where } w \in R^d$$

$$= min \ \ || X^T w - Y ||^2$$

$$\Rightarrow 2(XX^T)w - 2XY = 0$$

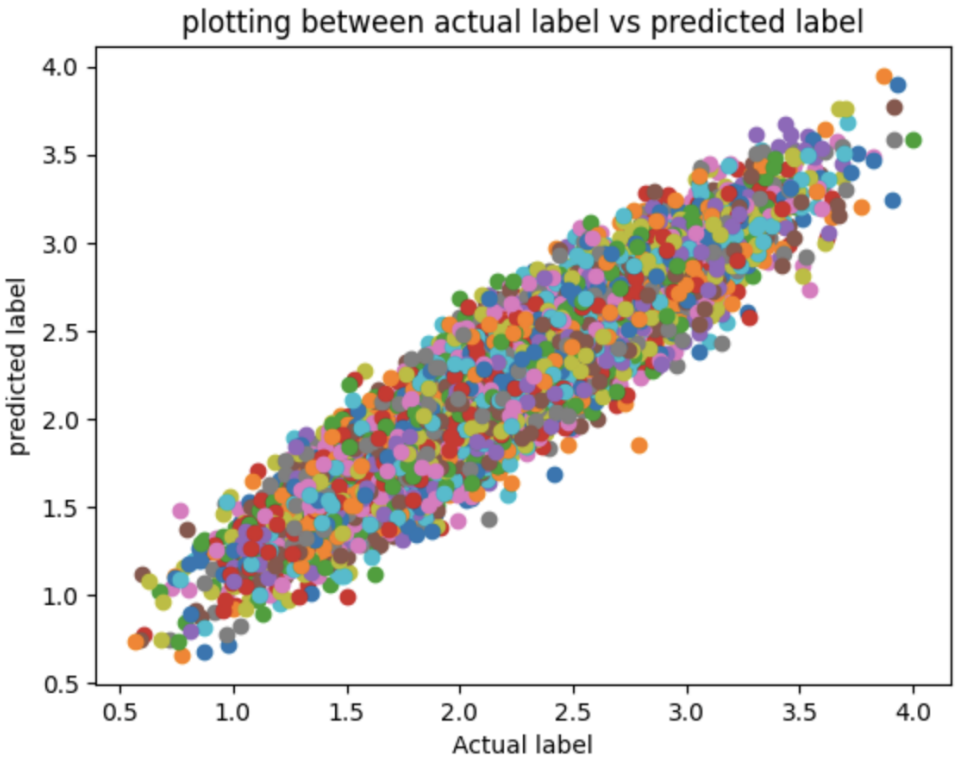
$$\Rightarrow (XX^T)w^* = XY$$

$$\Rightarrow w^* = (XX^T)^{-1}XY$$

At first I have imported all data from CSV file to numpy array and assigned the input to X numpy vector and labeled data to Y numpy vector. Then calculated XX^T and assigned to the COV matrix, the calculated inverse of the COVmatrix, and multiplied that with the product of the XY matrix. Then we obtained a closed-form solution of linear regression.

We have also plotted the projection of $W^T X$ and Y mean the plot between the actual label and the predicted label after we fit the data, the diagram is attached below.

The predicted weights are also attached in the submission folder as $wml.csv$.



1.2 Here we are asked to solve the linear regression least square problem with the gradient descent algorithm and plot the euclidian distance of $w^t - w_{ML}$ with respect to t. Generally, we apply gradient descent when the matrix XX^T inverse computation becomes hard for us, as inverse computation is of order d^3. As our objective function is an unconstrained quadratic optimization problem we can apply gradient descent.

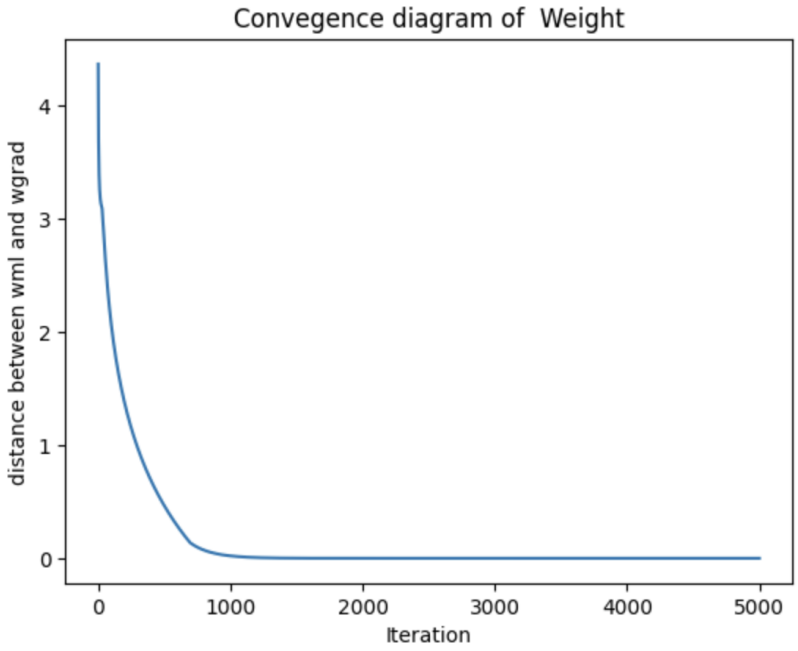
$$objective\ function = arg\ min\ \ ||\ X^T w - Y\ ||^2$$

taking derivative to get the gradient, $gradient = 2(XX^T)w - 2XY$

updating w, $\Rightarrow\ w^{t+1} = w^t - \eta^t * gradient$

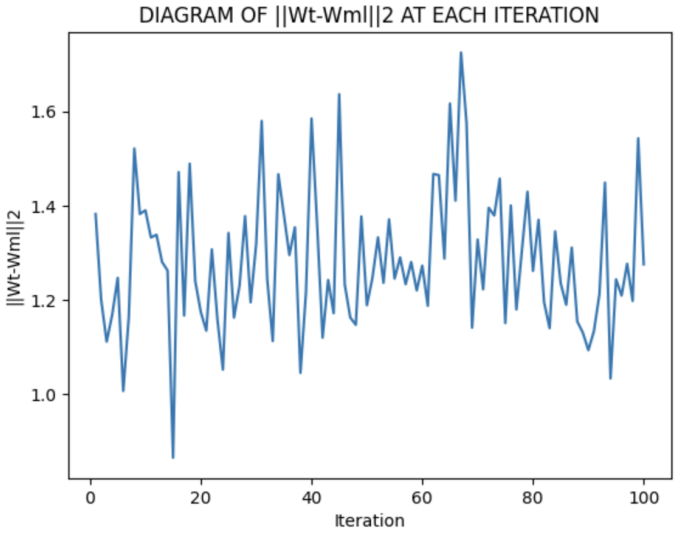
$$\eta^t = 1/t$$

Initially, I randomly initialized the weight vector, and to calculate the distance between two vectors I used $np.linalg.norm$. At each iteration, I have taken the distance between w^t and w_{ml} and plotted. Till the 5000th iteration, I have run the algorithm and near 1000 it converges. The plot of convergence of weight is attached below.



1.3 Here we have to code stochastic gradient descent and in question, it is specified that we have to take 100 batch size and we have to plot the euclidian distance of $w^t - w_{ML}$ with respect to t. Normally we apply stochastic gradient descent when the number of data points is high. When the number of data points is high XX^T calculation itself becomes hard, so we apply stochastic gradient descent multiple times. Here convergence to minima is not guaranteed but we get a reasonable value of weights.

Since I have randomly sampled 100 data points each time, sometimes I get a good sample and sometimes a bad one. Depending on that the euclidian distance of $w^t - w_{ML}$ increases a little or decreases a little. More over it fluctuates depending on the sample.



2.1 Ridge regression objective function $arg\ min\ \ ||X^T w - Y||^2 + \lambda||w||^2$. Ridge regression pushes the weights to near zero because the nature of objective function. Objective function equivalent to $arg\ min\ \ ||X^T w - Y||^2$ and $||w||^2 < \theta$.

2.2 In the code I have checked for lambda between 0 to 100. And for each lambda, the W0(initial random W for gradient descent) is the same so that I can compare all lambda for minimum error. Since each time the code is executed, W0 is random hence we might get different best lambda(lambda for which the corresponding W gives minimum error on training data). When I last executed my code I got the best lambda at 18. The mean square error corresponds to W_r and W_{ml} is 181.70162 and 185.3757. From here we can observe that W_{ml} gives the squared error more on test data. So we can say Ridge regression performs better.