CS5691: Assignment 2

Sukriti Shukla

ED20B067

Question 1

i) I loaded the train dataset and stored first 100 components in a variable x, and last component in y. Then I used the below code snippet to compute the least squares solution using the closed form expression.
$\hat{\mathbf{w}}_{ML} = (\mathbf{X}^\top \mathbf{X})^{-1}\mathbf{X}^\top \mathbf{y}$

ii) I made functions in the code namely:

1. mse_cost computes the mean squared error between the predicted values and the true values of the output, using the formula

   $J(\mathbf{w}) = \frac{1}{2m}\sum_{i=1}^{m}(\mathbf{x}_i^\top \mathbf{w} - y_i)^2$

2. mse_gradient computes the gradient of the cost function with respect to the weights using the formula

   $\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{m}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y})$

3. A third function, w_diff, is defined to compute the difference between the weights estimated using gradient descent and the maximum likelihood estimate of the weights.

4. The norm function from numpy.linalg is used to compute the Euclidean distance between the two weight vectors.

   $\|\mathbf{a} - \mathbf{b}\|_2 = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$

5. The weights are initialized to zero using np.zeros (initializing weights to random numbers is a better choice because then weights are not forced to start with equal values but it gave more error for me so I kept weights as 0 for initialization), and the learning rate and number of iterations are defined.

6. Two arrays are initialized to store the cost and weight differences at each iteration.

7. Gradient descent is performed using a for loop that iterates over the number of iterations. At each iteration, the weights are updated using the gradient of the cost function with respect to the weights. The cost_history and diff_history arrays are updated with the current cost and weight difference, respectively.

8. Finally, two plots are generated using matplotlib.pyplot. The first plot shows the cost as a function of iterations, while the second plot shows the difference between the weights estimated using gradient descent and the maximum likelihood estimate of the weights as a function of iterations.

Observations:

- Learning rate of 0.01 was used and the results are as follows

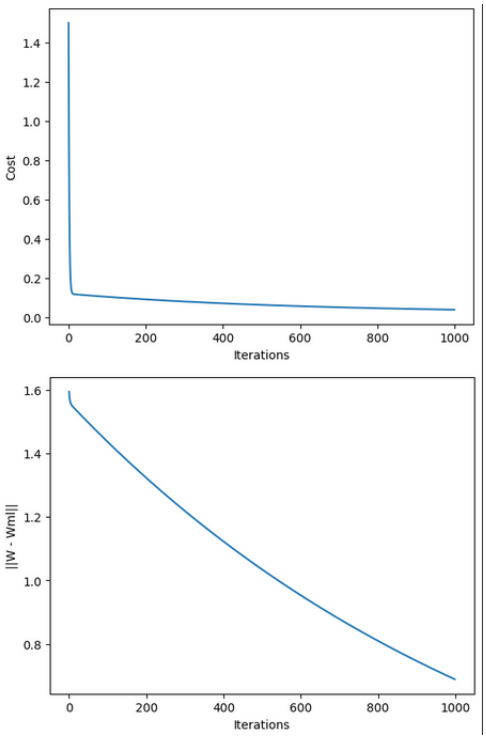

Figure 1: cost and $\|\mathbf{w} - \mathbf{wML}\|$ with iterations

- This shows that w^T becomes closer to analytical solution(wML) with the number of iterations.

- As the number of iterations approach infinity, w^T becomes exactly equal to analytical solution.

iii) Stochastic gradient descent The updated step is :
$w_{t+1} = w_t - \eta_t \mathbf{X'}^\top (\mathbf{X'w} - \mathbf{y'})$ X' and y' are sampled from X and y with a batch size of 100.

Observations: We see the following plot for stochastic gradient descent: As

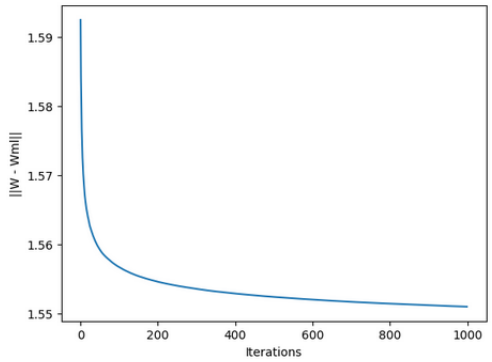

Figure 2: $\|\mathbf{w} - \mathbf{wML}\|$ with iterations

the number of iterations increases w^T becomes exactly equal w_{ml}

Question 2

i) My code has 3 functions , functions **ridge_cost** and **ridge_gradient** to compute the cost and gradient of the Ridge cost function, respectively, given input data x and y, weights w, and the regularization parameter lambda_.

Ridge cost function: $J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^{m} (\mathbf{x}_i^\top \mathbf{w} - y_i)^2 + \frac{\lambda}{2} \sum_{j=1}^{n} w_j^2$
Gradient of the Ridge cost function: $\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{m} \mathbf{X}^\top (\mathbf{Xw} - \mathbf{y}) + \lambda \mathbf{w}$

The function **ridge_regression** performs Ridge Regression on the input data x and y, using the specified regularization parameter lambda_. It initializes the weights to zero, and performs gradient descent for a fixed number of iterations (num_iterations). At each iteration, it updates the weights using the Ridge gradient and a fixed learning rate (alpha), and stores the cost and weight difference in arrays (cost_history and diff_history, respectively).

ii) I used K-fold cross validation to choose the best value of λ

- The best results were obtained for $\lambda = 0.05$
- Training set MSE: 0.13881
- Test set MSE: 0.23980

Table 1: MSE error comparison on training and test sets for Analytical solution and ridge regression

Training set		
	Analytical solution	**Ridge Regression**
MSE	0.03969	0.13881
Test set		
	Analytical solution	**Ridge Regression**
MSE	0.37073	0.23980
Observations		
The results show that Ridge Regression has lower MSE error on both training and test sets compared to analytical solution . This is because Ridge Regression introduces a regularization term that helps to reduce overfitting, analytical solution MSE shows that the model is overfitted thus performs poorly on the test set.		

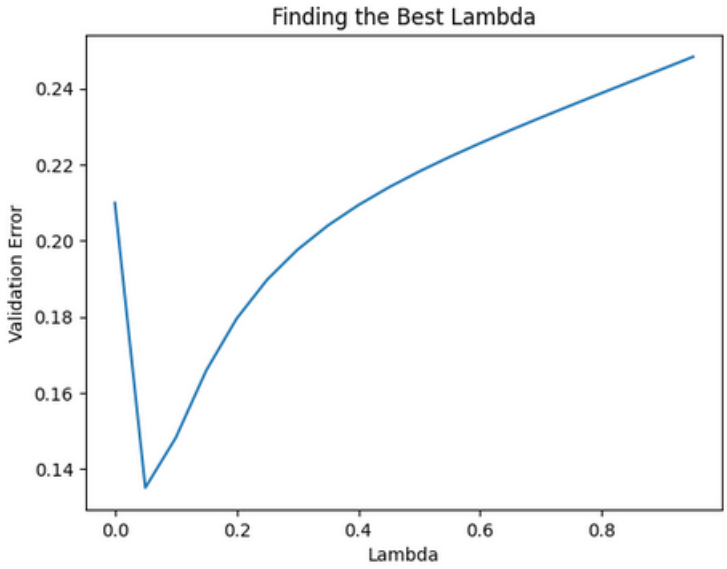

Figure 3: $\|\mathbf{w} - \mathbf{wML}\|$ with iterations