COMP 24112 Lab 2 Report(Machine Learning)
Vansh Goenka

3.2 Explain briefly the knowledge supporting your implementation and your design step by step. Explicitly comment on the role of any arguments you have added to your functions.

Ans: Initially the function def l2_rls_train() had two arguments: data and labels. In order to complete the function I added the lmbd argument which is the hyperparameter that needs to be selected. As per the notes, I loaded the data in the variable X and ground truth labels in y. I then calculate the expanded feature vector X_tilde using the sklearn function hstack. Now, by comparing the value of lambda, I calculate the weights. If the lambda was 0, the weight is calculated simply using the pseudo-inverse of X_tilde multiplied by label y. if it's greater than 0 then the weight is calculated using the formula :

$$\mathbf{w} = \left(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}} + \lambda\mathbf{I}\right)^{-1}\tilde{\mathbf{X}}^T\mathbf{y}$$

given in the notes where I is the Identity matrix of the size of (1025, 1025) and X_tilde ^ T is the transpose of X_tilde.

The l2_rls_predict(w, data) function takes in the weight and returns the multiplication of the expanded feature matrix for the passed data and the given weight.

4.2 Explain the classification steps, and report your chosen hyper-parameter and results on the test set. Did you notice any common features among the easiest and most difficult subjects to classify? Describe your observations and analyze your results.

Ans: The code performs a classification task using L2-regularized least squares on data. It starts by partitioning the data into training and test sets using the partition_data function. It then encodes the categorical labels using one-hot encoding from sklearn and initializes variables for storing results.

I used k-fold cross-validation to select the best lambda value. It initializes a list of possible lambda values and then loops over each value, performing k-fold cross-validation to compute the average accuracy for that value of lambda. It selects the lambda value with the highest average accuracy on the validation sets as the best lambda value. When I ran the code the best lambda value I got was 0.010 with a Training accuracy of 0.905.

The chosen hyper-parameter is the lambda value, which is used for L2 regularization. The possible values for lambda are set in the lmd_vals list. The code performs k-fold cross-validation to select the best lambda value and reports the best lambda value and training accuracy on the test set.

The code then trains the model using the selected lambda value and all training data and evaluates it on the test set. It computes the accuracy score and confusion matrix for the test set.

The code selects the easiest and most difficult subjects by finding the highest accuracy row in the confusion matrix and selecting the 2 easiest subjects, and finding the subjects with the lowest values on the diagonal of the confusion matrix and selecting their misclassified test images.

The easiest subjects have better image quality and significant facial expressions, while the most difficult subjects have low quality and similar facial expressions.

5.2 Report the MAPE and make some observations regarding the results of the face completion model. How well has your model performed? Offer one suggestion for how it can be improved.

Ans: When I ran the code the MAPE I got was 0.21464671396490903 or 21.465%.
It calculates the mean absolute percentage error (MAPE) of a face completion model. The data is split into left and right-face images. The left pixels are used as input, and the right pixels are used as labels for training and testing.

The model is trained using regularized least squares with L2 regularization. The weight is calculated using the left and right pixels partitioned for training. The model's performance is evaluated on the unseen left pixels by predicting the right pixels using the trained weight.

The MAPE is then calculated as the mean absolute percentage difference between the predicted and actual right pixels. The MAPE value is printed. A lower MAPE value indicates better performance.

Since I got a low MAPE error, I can be sure that my model has performed well and that the predictions made by the prediction function are accurate. However, it could be more accurate and give better results by improving the model with other regularization techniques or by using a different model altogether.

6.3 Analyse the impact that changing the learning rate has on the cost function and obtained testing accuracies over each iteration in experiment 6.2. Drawing from what you observed in your experiments, what are the consequences of setting the learning rate and iteration number too high or too low?

Ans: The impact of changing the learning rate (eta) on the cost function and the obtained testing accuracies over each iteration is shown in the three plots in the code. In the plots, we can observe that the sum-of-squares error cost decreases as the iteration number increases, which is expected in Gradient Descent.

In the first plot, we can observe that the cost decreases slowly with an eta of 0.001, and it takes about less than 10 iterations to converge to a stable minimum. We can observe that the cost dips steeply from a maximum value of 225 to 125 and then has a stable decrease in cost over the next iterations. On the other hand, in the second plot, with an eta of 0.01, we observe faster convergence of the cost function to a stable minimum, which takes around 180 to 190 iterations where it shoots up to 1.75.

We can observe that setting the learning rate too high may cause the GD algorithm to overshoot the optimal solution, and it may cause divergence or oscillation around the minimum point. On the other hand, setting it too low may result in very slow convergence and longer training time.

Explain in the report your experiment design, comparative result analysis and interpretation of obtained results. Try to be thorough in your analysis.

Ans: As you can clearly see in the graph that with the use of normal gradient descent(GD), the cost function is very stable after iteration 125 and then dips to its minimum value. However, with the stochastic gradient descent(SGD), we can see that the cost is not stable at all and cannot find a trend in the graph. This might be because of the random selection of samples each time we iterate.

As for as training accuracy is concerned, we can see that accuracy for normal GD is constant until iteration 50 and then get stable at iteration 75. On the other hand SGD, there are only some values for which the accuracy is stable.

As for as testing accuracy is concerned, we can see that accuracy for normal GD is constant until iteration 50 and then get stable at iteration 75. On the other hand SGD, there are only some values for which the accuracy is stable and the testing accuracy for SGD is more stable than the training.

These all values were when I ran the code and the values might fluctuate a little bit, but the trend is almost the same.