Machine Learning 2 Report

Usman Ali

3.2: The first function defined is the ***l2_rls_train*** function which trains a linear model by minimising the L_2-regularised sum of squares loss through zeroing the loss gradient. Firstly, the input data is expanded with a column of ones which allows for an intercept term to become present in the model. It then computes the coefficient vector using L_2-regularised least squares. The parameter ***lmbd*** is a float that represents the regularisation parameter. If ***lmbd*** is zero, it uses the pseudoinverse of the expanded data matrix to compute the coefficient vector. Otherwise, it uses the closed-form solution for ridge regression because it provides a computationally efficient way to estimate the coefficients of the linear regression model.

The second function defined is the ***l2_rls_predict*** function which predicts target values for new input samples using a learned weight vector. The function first expands the input data with a column of ones and then computes the prediction for the input samples by multiplying the expanded data matrix with the learned weight vector.

4.2: The first step involved in the classification was the splitting of the data into training and testing data. The labels were then extracted and encoded. By doing this, the category labels are transformed into a format that the linear regression model can use. Using a variety of hyperparameters lambda, the code trains an L_2-regularized least squares linear regression model on the training data before choosing the hyperparameter that produces the maximum accuracy on the test set. The model is trained using the ***l2_rls_train*** function, and the accuracy is calculated using the test set's true labels and the predicted labels. The code evaluates the trained classifier using the chosen hyperparameter lambda on the test set. In order to achieve this, a new linear regression model is trained on the training data using the chosen lambda, and the labels for the test data are then predicted using the ***l2_rls_predict*** function. The confusion matrix compares the true labels to the predicted labels and counts the amount of true positives, true negatives, false positives, and false negatives, the confusion matrix is calculated.

It is noticeable that faces belonging to the most difficult subjects to classify express exaggerated expressions or the picture may be angled. The reason for these being amongst the most difficult subjects to classify involves the training data not having many representations of such images. Meanwhile, the easier subjects display consistency when it comes to facial expressions and the camera angle of the image. One reason this is easier to classify is due to the training data having more representation of such images.

5.2: The reported mean absolute percentage error was 23.26% which shows that there is still a good amount of error within the model. The reasons for this could be due to a lack of training data or poor choice of hyperparameter. Observations from the images show a stark difference between the ground truth image and the predicted images; this is extremely clear from the middle image where the glasses are not completely present in the predicted image unlike the ground truth image.

6.3: When a small learning rate is used (in this case $1x10^{-9}$), the sum of squares error loss graph shows a decrease in error loss. This shows that the model is learning from the training data and is improving in its ability to fit the training data. There is also an increase in both the training and testing accuracy which is also further indication that the model is improving to fit the data. A small increase in the learning rate may result in faster convergence to the optimal solution.

However, the effect of increasing the learning rate too much is that the algorithm overshoots the optimal solution and leads to divergence; this occurs due to the parameters changing more significantly at each iteration. The accuracies for both training and testing remain at 50% showing no change in the accuracies; this is representative of the model not improving when iterating over the data.

7.3: The objective of the experiment is to compare the performance and behaviour of gradient descent and stochastic gradient descent for linear least squares regression. The hypothesis of the experiment is that stochastic gradient descent will converge faster than gradient descent, but with a lower accuracy and a higher variance. In gradient descent, the weights are adjusted depending on the cost function's average gradient across all training samples. SGD, on the other hand, updates the weights depending on the gradient of the cost function with respect to a batch or subset of the training samples that is randomly chosen. The reason for SGD being faster is that each iteration only requires a small subset of data but this can make SGD susceptible to noisy data. Convergence time is used a measure to test the speed of both models and it can be observed from the results that SGD is faster at each learning rate supporting the hypothesis. The average squared difference between the anticipated values and the actual values is what the MSE measures. The accuracy of the model's predictions increases with decreasing MSE. However, from the results it can be observed that the MSE is generally significantly larger for SGD compared to gradient descent; this too is further support of the hypothesis that SGD converges with a lower accuracy and higher variance.