COMP24112 Lab-2 Report

Chittesh Kumar Singore

Experiment-1: Regularised Least Squares Model

Q1. 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. The Regularised Least Squares Model implementation uses two functions l2_rls_train() and l2_rls_predict(). The weight matrix is calculated using l2_rls_train() and the labels vector (predicted_y) is calculated using l2_rls_predict(). Further explanation of the implementation of these functions are given below:

(a) l2_rls_train() function takes in three arguments: data, labels, and, lmbd. The function then computes the regularized weights w using the formula given in the lectures. If the regularization parameter lmbd is zero, then the function uses the pseudo-inverse of X_tilde to compute w. Otherwise, it computes w using the regularized closed-form solution. Finally, it returns the trained weights w.

Expanded feature matrix (X_tilde) is calculated in this function by adding columns of ones to the input data matrix to account for the bias term. Also, transpose of this matrix-X_tilde is used in the formula of w as shown in the piece of code below.

Snippet of code:

```
…
if(lmbd > 0):
      w = np.linalg.inv(X_tilde_transpose @ X_tilde + lmbd*I) @ X_tilde_transpose @ y
…
```

(b)The l2_rls_predict function takes in two arguments: w and data. The function uses X_tilde (expanded feature matrix) and value of w (trained weight matrix) to compute the predicted output values by multiplying matrix w and X_tilde.

Experiment-2: Face Recognition Model

Q1. 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 analyse your results.

Ans. The code implements a classification task using the L2-regularized least squares model. The approach for classification is explained below:

(a) The data is split into a training set and a test set using partition_data function.
(b) one-hot encoding is used to encode the labels.
(c) The KFold cross-validation technique is used to select the best lambda value. The hyper-parameter lambda is selected from a set of values ranging from 10^-4 to 10^3 (lmbd = np.logspace(-4, 3, num=8)).
(d) For each lambda value, the model is trained using the l2_rls_train() function on the training set and evaluated on the validation set. The lambda value with the highest average accuracy on the validation sets is selected.
(e) Finally, the model is trained on the entire training set using the selected lambda value and evaluated on the test set.

The chosen hyper-parameter varies at every execution (however, in between 0 and 1) due to random selection of the training set. The results on the test set show an accuracy of te_acc = 0.908.

The easiest classes are found by selecting highest diagonal values in the confusion matrix, and the test images corresponding to these classes are plotted. The most difficult subjects are found by selecting lowest accuracies of subjects in the confusion matrix.

Observations: While observing the easy and difficult subjects which were classified, it was noticed that output for easy subjects have better image quality and significant facial expression which can be classified easily. On the other hand, difficult subjects have low quality as well as similar facial expressions which makes it difficult to classify.

Experiment-3: Face Completion Model

Q1. 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. The face completion model uses a training set and a test set (splitted using left and right input data) to train the model using the training set, and evaluate the model's performance using mean absolute percentage error (MAPE) on the test set. The MAPE of the model is 21.01%. Three random test samples are selected, and the ground truth faces and completed faces are concatenated and displayed using the show_single_face() function.

Observations: The MAPE of 21.01% suggests that the face completion model performs accurately most of the time, but it still makes errors that need to be addressed.
Suggestions: To improve the model, we can increase the amount of training data. A larger training set may enable the model to learn more representative features and improve its accuracy on the test set.

Experiment-4: Training linear least squares model via gradient descent

Q1. 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. Over each iteration in the experiment, the learning rate determines the step size during gradient descent optimization. The impact of different values of learning rate can be observed from the graphs, for instance, for learning rate of 0.001 the sum-of-squares error loss decreases very steeply in the initial interactions (around 5 iterations), while decreases slowly for the rest of the iterations.

Looking at the graph depicting the learning ability of the model, we observe that after training and testing the model there is a gradual increase in the learning aspect of the model. When the learning rate is 0.001, the model performs well enough to classify subjects easily as compared to the rate of 0.01 where the training and testing accuracies are the same.

Experiment-5: Compare with stochastic gradient descent training (Option-1)

Q1.Explain in the report your experiment design, comparative result analysis and interpretation of obtained results. Try to be thorough in your analysis.

Ans. The implementation compares two algorithms for training a linear least squares model using gradient descent and stochastic gradient descent. The experiment aims to compare the performance of the two approaches in terms of accuracy and cost over iterations. The following steps are taken:

1. The code partitions the dataset into training and testing data.
2. Two models are trained using gradient descent and stochastic gradient descent algorithms, respectively.
3. The accuracy of each model is computed using the training and testing data.
4. The cost of each model is plotted against the number of iterations to compare their convergence.
5. The accuracy of each model is plotted against the number of iterations to compare their performance.

In the SGD function, the model is trained using a single-output linear model by minimizing the sum of squares loss using stochastic gradient descent. The function takes as input the training data, labels, the number of iterations, learning rate, and batch size (here the batch size is set to be 1 to get standard SGD). it returns the cost as well as the weights.
From the graphs, we can observe that in GD the cost function is stable as compared to the SGD algorithm. As per the learning rate of the model, GD performs better with lesser fluctuations in the graph while SGD has a lot of fluctuations in its accuracy.