Computer Vision Homework 3
Trisha P Malhotra

LRModule :-

- Given the skeleton code and the instructions, I thoroughly read the pdf of instructions, then the helper.py functions , and finally the LR Module.py file
- Started by working in the Main method,
- Using getBinaryfer13Data from helper.py I loaded the given training dataset : fer3and4train.csv. This loaded the data samples along with their class labels.

- Next, called the LRModule train function to learn the weights and bias units
- In the train function, validation data has been extracted from the original dataset.
- Initialized the weights and bias using init_weight_and_bias
- Called forward function for the given number of epochs,
- Where in forward function calls sigmoid(X.dot(self.W) + self.b)
- Here Sigmoid is the activation function.
- Next I performed Gradient descent using the following equation:

```
diff = pY - Y

self.W -= step_size.dot(X.dot(diff))
self.b -= step_size.dot(diff)
```

- Using the validation data, next step was to compute P(Y|X_valid) using the forward algorithm.
- Also, stored the costs returned from function: sigmoid_costs,
- Next for range (pY_valid), I computed the best_validation_error, by comparing it be the last saved value of validation error.

- The graph needed to be plotted using matplotlib library.
- Train function return best_validation_error

- Predict function is called to now test our Logistic Regressor on a test dataset.
- As these X values are unseen by the network , it is interesting to see how well the system fare in correctly predicting TestY values as per its learned weights and biases.

- I could not achieve this task without errors, which is why I did not get any graphs.
- But had I gotten them, the next step was to get the accuracy report from this Testing.

- To calculate the accuracy score, for the test, I compared TrueY with predictedY,to keep a count of the matches that were correct.
- Next the score is converted into an accuracy percentage.

NNModule :-

- For the module containing Neural Networks classification
- I thoroughly went through the given instructions and the skeleton code.
- Again, the file dataset was read using getBinaryfer13Data function from the helper file.
- And then called the train function from the NNModule class by passing the file to it.
- Inside this class, the train function, validation set has been created that contains 1000 records.
- Randomly initialized weights and biases.
- For 10000 epochs, I call forwad function that returns back pYtrain, and Ztrain for the training data , and pYvalid and Zvalid for the validation data.
- For conducting Gradient Descent, I used the below given equation:

```
diff = pYtrain - Y
result = Ztrain.dot(diff)
self.W2 -= step_size * (result)
self.b2 -= step_size * (diff)
```

- Next to store the results from back propagation

```
result2 = (1 - (Ztrain ** 2))
result3 = W2.dot(result2)
dJ = diff.dot(result3)
```

- Computed the training and validation errors using cross_entropy function which does the following:
- Cross entropy uses the following function to register error or loss,

```
- np.sum(np.multiply(Y, np.log(pY)) + np.multiply((1 - Y), np.log(1 - pY)))
```

- Next, the errors returned by the cross entropy are appended in the arrays, namely, train_costs[] and valid_costs[].
- For range of pYvalid and pYtrain, I then computed the corresponding best validation errors, which are returned by this Train function.
- Train function also plots the graphs of the cost arrays for training and validation data, but I did not get any graphs plotted due to some error.
- After checking the validation errors, I then decide that themodel, has been trained and the Neural network is ready to be tested.

- Next the test file fer3and4test, is read and fed into the Predict function, that inturn calls the forward function.

- The activation chosen was Sigmoid function.

- Forward function returns back the result from the activation function and the predicted values from the Softmax classifier.

- Next Accuracy or Classification Rate is calculated by using the score function,

- This score function compares every predicted Y to True Y, and returns the final accuracy score.