Project Report

Name: Ipsa Mishra
Student ID: 1001759616

1. Implement your own logistic regression. For your training data, generate 1000 training instances in two sets of random data points (500 in each) from multi-variate normal distribution with mu1 = [1,0], mu2 = [0,1], sigma1 = [[1,0.75],[0.75,1]], sigma2 = [[1,-0.5],[0.5,1]] for each group and label them 0 and 1. Generate testing data in the same manner but sample 500 instances for each class, i.e., 1000 in total. Use sigmoid function for your activation function and cross entropy for your objective function, and perform batch training. The stopping criteria would be 1) when the gradient vanishes (i.e., norm of the gradient is too small) and 2) when the loss function converges. Set your maximum number of iterations to 10000 so that you don't wait forever for the model to converge. Plot a ROC curve and compute Area Under the Curve (AUC) in the end to evaluate your implementation, and write up a brief summary in your report. All the code base for the training, testing and plotting the ROC curve should be on your own. How many iterations does it require to train when you change your learning rate as 0.0001, 0.001, 0.01, 0.1, and 1? Plot the number with respect to the learning rate.

   When learning rate = 0.0001, the number of iterations required = 10000, Accuracy: 0.643 and AUC-ROC value is 0.7363239999999998

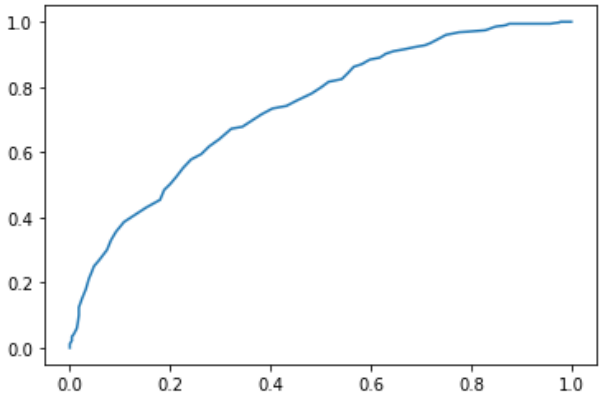
   When learning rate = 0.001, the number of iterations required = 10000, Accuracy: 0.849 and AUC-ROC value is 0.8871459999999999

   When learning rate = 0.01, the number of iterations required = 2641, Accuracy: 0.796 and AUC-ROC value is 0.861606

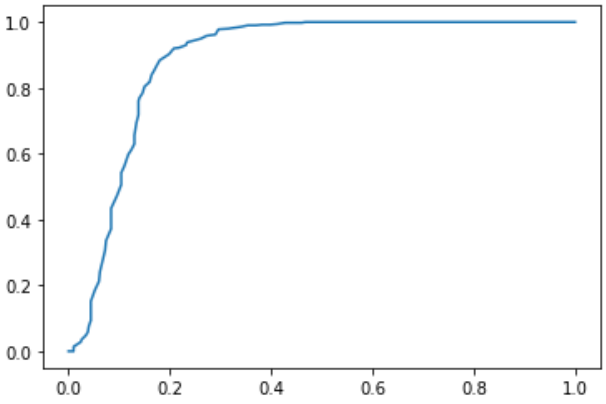
   When learning rate = 0.1, the number of iterations required = 273, Accuracy: 0.836 and AUC-ROC value is 0.8864520000000001

   When learning rate = 1, the number of iterations required = 29, Accuracy: 0.854 and AUC-ROC value is 0.9047259999999999

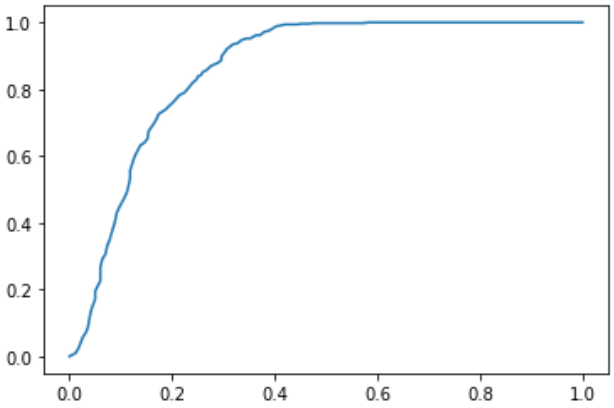
   ROC curve when learning rate = 0.0001:



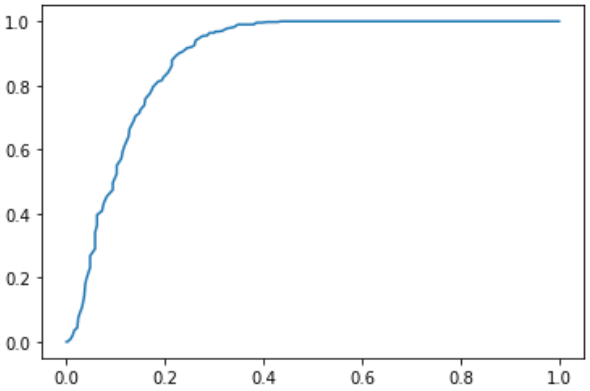
ROC curve when learning rate = 0.001:



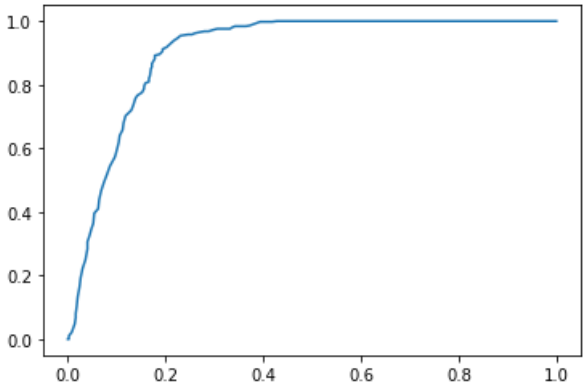
ROC curve when learning rate = 0.01:



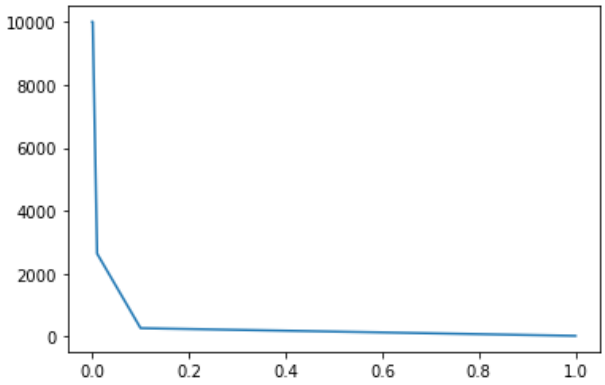
ROC curve when learning rate = 0.1:



ROC curve when learning rate = 1:



Plot of number of iterations w.r.t learning rate:



I have implemented logistic regression from scratch by using sigmoid function as activation function and cross entropy as objective function. The stopping criteria will be when the gradient would be too less or when the loss function converges (i.e when there is minimal change or it won't change anymore). I had previously tried the loss function converging thing by using statement (if np.abs(le - l) <= L1diff) but then I saw it stopped after few iterations when the learning rate was too small. That's why I changed it to statement (if np.abs(le - l) <= lr * L1diff) so that it can go for some iterations even for small learning rate. Then I have plotted ROC curve for each learning rates. ROC curve is the relationship between TPR (total positive rate) and FPR (false positive rate) with FPR on X-axis and TPR on Y-axis. Then I have found out the number of iterations w.r.t each of the learning rates given and plotted it accordingly.

2. Implement multi-class logistic regression using a soft-max function and cross entropy. Use MNIST data for your training and testing data. The dataset has 60000 training and 10000 test samples that contain images of hand-written numbers in 28x28 pixels. There are various sources you can find online to fetch the data.

   Please use the images within class 0 to 4 (i.e., 5 classes) only, you can do this by checking the labels given to each image. Set the maximum number of iterations to 10000. Print out accuracy, precision and recall at the end to evaluate your implementation, and write up a brief summary in your report.

   Number of epochs: 259

Accuracy = 0.9420120646040085

Precision for class 0 is 0.9626639757820383

Precision for class 1 is 0.9646246764452114

Precision for class 2 is 0.916751269035533

Precision for class 3 is 0.9213372664700098

Precision for class 4 is 0.9412360688956434

Recall for class 0 is 0.9734693877551021

Recall for class 1 is 0.9850220264317181

Recall for class 2 is 0.875

Recall for class 3 is 0.9277227722772278

Recall for class 4 is 0.9460285132382892

Here, I have implemented multiclass logistic regression using softmax function and cross entropy. The mnist dataset is used here for training and testing. This gives us the number of iterations, accuracy , precision and recall value for each of the classes.

3. For this part, you may use existing libraries. Implement an artificial neural network (ANN) by stacking up your perceptron. You may use one hidden layer and the number of hidden nodes are up to you. Use the same MNIST data as in P2 for training and testing. Try out both sigmoid and ReLU for activation functions. Print out accuracy, precision and recall at the end. Did the performance get better or worse compared to the result from P2? It can be either so justify your answer in your report.

Accuracy = 0.9945514691574237

Precision for class 0 is 0.9938837920489296
Precision for class 1 is 0.9982332155477032
Precision for class 2 is 0.989351403678606
Precision for class 3 is 0.996039603960396
Precision for class 4 is 0.9949135300101729

Recall for class 0 is 0.9948979591836735
Recall for class 1 is 0.9955947136563876
Recall for class 2 is 0.9903100775193798
Recall for class 3 is 0.996039603960396
Recall for class 4 is 0.9959266802443992

ANN model can better learn the class boundaries and they have larger parameter set to capture that information. Also logistic model are normally better at detecting classes with more linear boundaries. That's why it gives better results for ANN model.