CSE 574 Machine Learning
Programming Assignment 3

Yesh Kumar Singh - 50247208
Sneha Mehta - 50245877

1. Logistic Regression:

In this part of the project we had to implement a logistic regression classifier to classify the hand written digits into their respective classes. As logistic regression is a binary classifier we will use the one vs all strategy and implement 10 classifiers to classify the hand written digits in their respective classes.

The following functions were implemented as a part of the logistic regression:

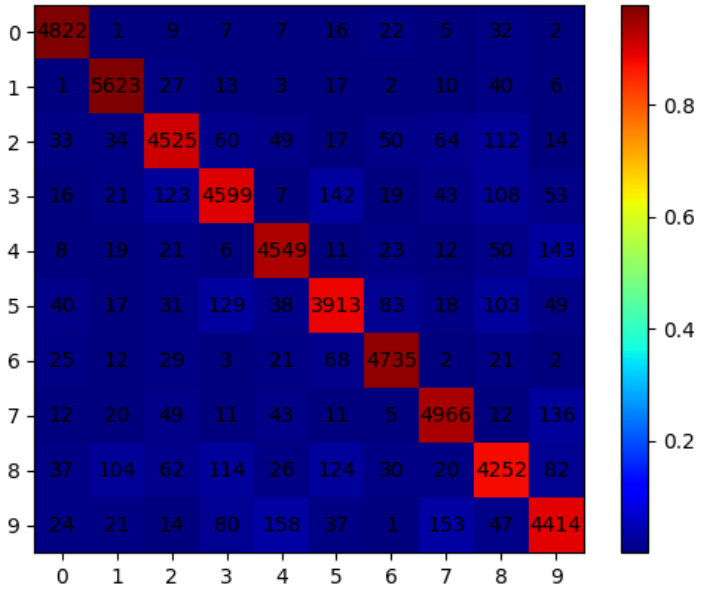
1. blrObjFunction() to train the classifier
2. blrPredict() to build the actual classifier

Accuracy Reported:

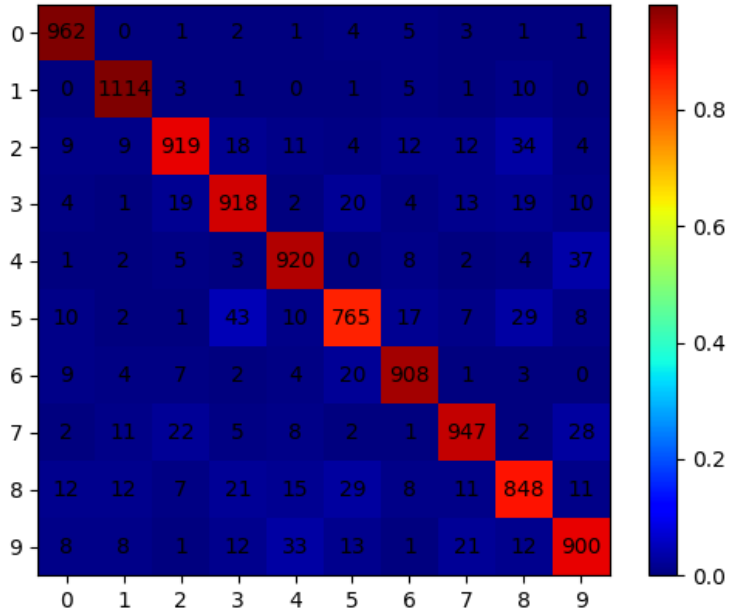
Training Set Accuracy	**92.796%**
Validation Set Accuracy	**91.5%**
Testing Set Accuracy	**92.01%**

Logistic regression is a technique that works well with data having low input features as it tries to linearly separate the data. In this case it does okay work considering the accuracy.

The Confusion matrix for the training set is as follows:



The Confusion matrix for the testing set is as follows:



There is a difference between the train error and the test error but it is not very big because the logistic regression has created a hyperplane and fit it as closely to the training data and using this fitting to make predictions, now as the plane is fit on the training set the model will and should give a lesser amount of error for the same.

2. Multi Class Logistic Regression

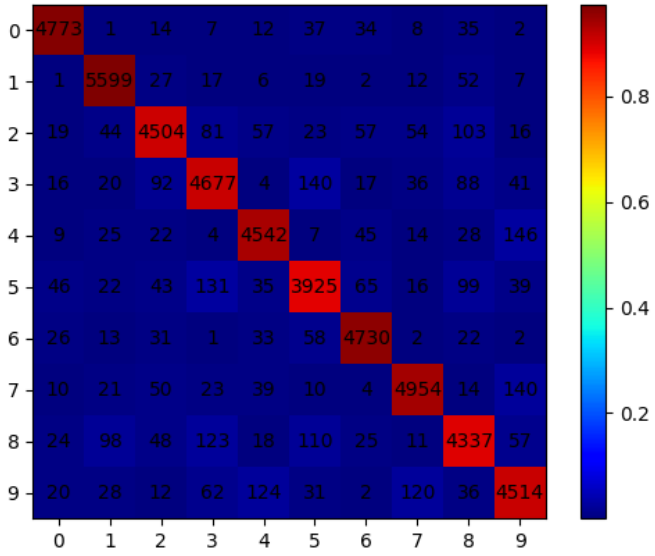
The strategy now is to design a Multi Class logistic regression technique
And we implement the following functions as a part of doing it

1. mlrObjFunction() to train the classifier
2. mlrPredict() to build the actual classifier

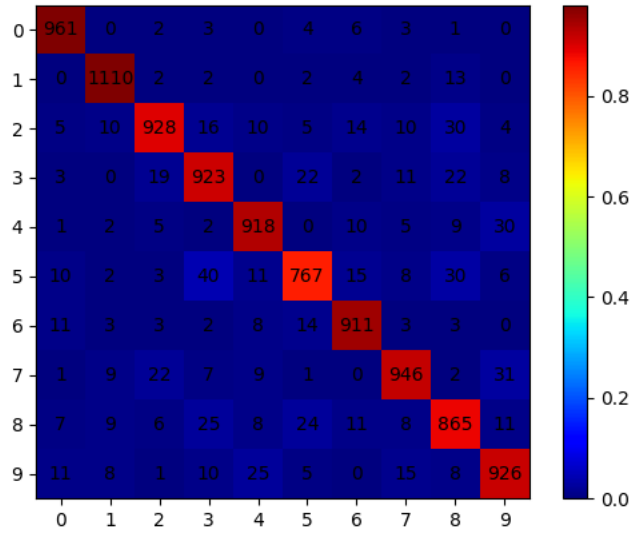
Accuracy Reported:

Training Set Accuracy	**93.11%**
Validation Set Accuracy	**92.54%**
Testing Set Accuracy	**92.55%**

The Confusion Matrix for the training set is as follows:



The Confusion Matrix for the testing set is as follows:



The performance of Multi Class Logistic regression is better than that of the previous method because in this case we are using only training one classifier instead of training 10 classifiers. The previous method took around 20 minutes to run on a local machine where as the new method ran in 5 minutes approximately. Although the values of the accuracy do not vary a lot.

3. Support Vector Machines

We use the scikit-learn library to perform classification in this case. First we make a model, then we fit the model according to our training data and then we validate the model and then finally if we are happy with the training we run the model on a testing data. We changed the SVM in the following ways that were mentioned in the assignment:

*Note: As the dataset was taking a long time to run on the entire data we chose 10,000 random points for training and 2000 points for validation and testing each. This gave a runtime of around 20 minutes on our local computer and the

This is the accuracy we got by running our script

1. Using a linear kernel

Training Set Accuracy	**99.82%**
Validation Set Accuracy	**92.15%**
Testing Set Accuracy	**90.45%**

A linear kernel performs well when the data is linearly separable as we can see here the SVM is able to work extremely well with the training set but fails to maintain the same amount of accuracy with the validation and testing data set.

2. Using a radial basis function with default gamma

Training Set Accuracy	**93.64%**
Validation Set Accuracy	**92.55%**
Testing Set Accuracy	**91.65%**

RBF maps a d dimensional data to infinite dimensional space and tries to fit a line which can correctly classify our data. Then transforms the line to original d dimension to learn the non linear boundaries. The default value of C = 1 and gamma = 1/number of features in the SVM.

3. Using a radial basis function with gamma = 1

Training Set Accuracy	**100%**
Validation Set Accuracy	**16.35%**
Testing Set Accuracy	**18%**

As gamma defines the importance of the feature and in this case it is 1 the SVM takes a lot of time to compute and then ends up overfitting the data as we can see by the accuracies respectively. The training set is completely fit but the model fails at correctly classifying of the labels of the validation and the testing data.

4. Using different values of C

When C = 1

Training Set Accuracy	**93.64%**
Validation Set Accuracy	**92.55%**
Testing Set Accuracy	**91.65%**

When C = 10

Training Set Accuracy	**97.15%**
Validation Set Accuracy	**94.4%**
Testing Set Accuracy	**92.95%**

When C = 20

Training Set Accuracy	**98.19%**
Validation Set Accuracy	**94.5%**
Testing Set Accuracy	**93.3%**

When C = 30

Training Set Accuracy	**98.75%**
Validation Set Accuracy	**95.15%**

Testing Set Accuracy	**92.9%**

When C = 40

Training Set Accuracy	**99.12%**
Validation Set Accuracy	**95.2%**
Testing Set Accuracy	**93.05%**

When C = 50

Training Set Accuracy	**99.41%**
Validation Set Accuracy	**95.2%**
Testing Set Accuracy	**93.1%**

When C = 60

Training Set Accuracy	**99.61%**
Validation Set Accuracy	**95.1%**
Testing Set Accuracy	**93.25%**

When C = 70

Training Set Accuracy	**99.73%**
Validation Set Accuracy	**95.51%**
Testing Set Accuracy	**93.2%**

When C = 80

Training Set Accuracy	**99.76%**
Validation Set Accuracy	**95.0%**
Testing Set Accuracy	**93.1%**

When C = 90

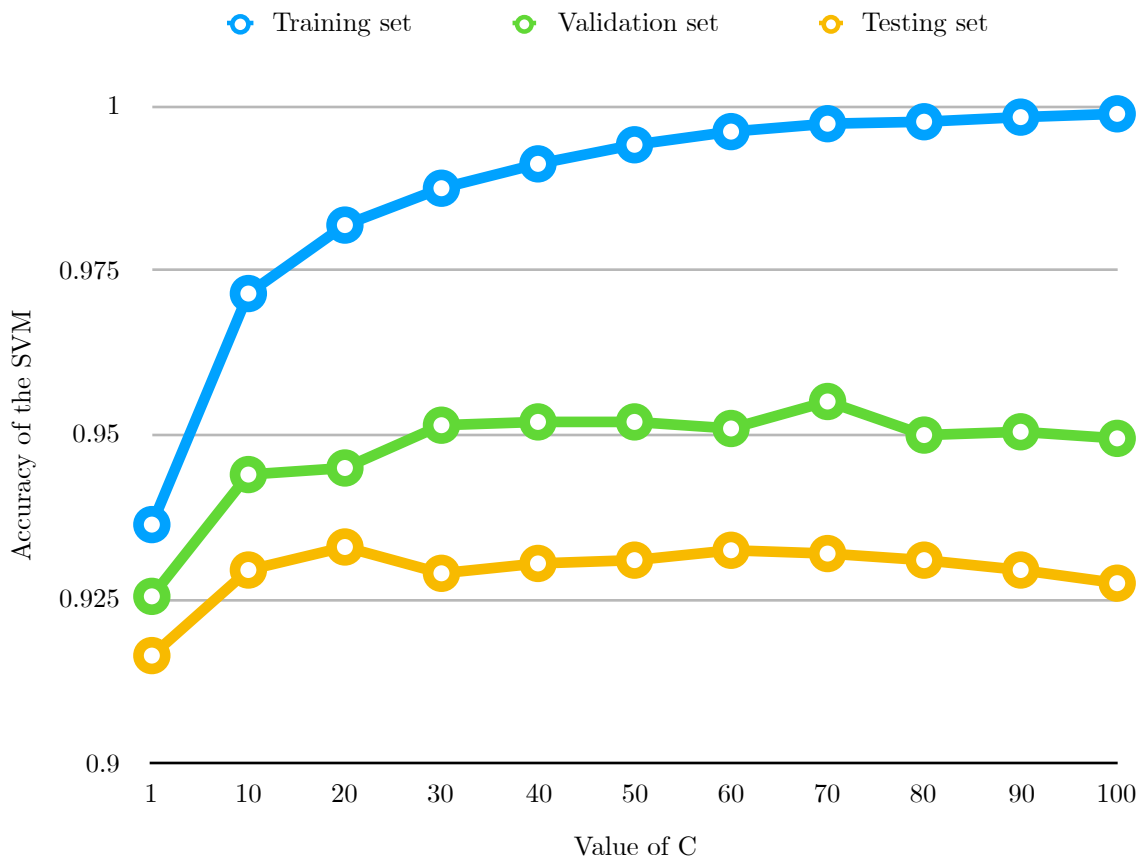
Training Set Accuracy	**99.83%**
Validation Set Accuracy	**95.05%**
Testing Set Accuracy	**92.95%**

When C = 100

Training Set Accuracy	**99.88%**
Validation Set Accuracy	**94.95%**
Testing Set Accuracy	**92.75%**

We see a peak in the value of accuracy of validation set at C = 70 thus we chose that to train the SVM on the entire data at C = 70.

As we can see from the following graph that the as the value of C increases the value of accuracy for training data increases, the accuracy of validation and testing set also stay around the same range after initial increase.



When we run the SVM with C = 70 on the entire data we get the accuracy as follows:

Training set Accuracy: 99.34 %

Validation set Accuracy: 97.36 %

Test set Accuracy: 97.26 %

Which is a very nice performance for a Support Vector Machine on the MNIST data set.

The linear kernel is much more suitable for a linearly separable data set making them faster in implementation and training, Radial basis functions are non linear and thus are used when the data is not linearly separable. The time that the linear kernel takes to compute is lesser than RBF. On our local machine when we ran the script for 10000 datapoints the Linear kernel took approximately a minute whereas the RBF took about 5 minutes for the same. The accuracies of the both are as follows:

	Linear Kernel	**RBF with C = 70 and default settings**
Training Set Accuracy	**99.82%**	**99.34%**
Validation Set Accuracy	**92.15%**	**97.36%**
Testing Set Accuracy	**90.45%**	**97.26%**

As RBF is non linear and tries to map the data into an infinite dimensional space thus better fin tunes the classifier and thus the accuracy is better over other techniques.