CSE 574 Introduction to Machine Learning
Programming Assignment 3
Classification and Regression

Due Date: May 3rd 2017

Group No 67

Aniruddh Chaturvedi	**[5020 6958]**
Arnav Ahire	**[5020 8006]**
Ashwin Nikam	**[5020 7368]**

Logistic Regression

In this part we were supposed to perform logistic regression to classify hand-written digit images into correct corresponding labels. Since here the possible values that a label can take was 10 i.e multiple classes were involved and since logistic regression traditionally works on binary classification, we had to employ *one vs all* strategy where-in we built 10 binary classifiers (1 for each class) in order to distinguish each class from another.
So we implemented two methods:

1. blrObjFunction() - We performed the training of our classifier here.
2. blrPredict() - The actual classification of data was done.

The following results were obtained after executing the above two functions:

Results:

Training set Accuracy:84.918%

Validation set Accuracy:83.68%

Testing set Accuracy:84.15%

Inference:

The inference is quite straightforward and is as follows:

1. The training set has the highest accuracy of 84.918 %.

2. The validation set has the lowest accuracy of 83.68 %.

3. The testing set accuracy is 84.15 % and is more like that of the training set but slightly lower.

Support Vector Machines

In this part we were asked to use the Support Vector Machine tool in sklearn.svm.SVM to perform classification on our data set. We needed to learn the SVM model and compute the accuracy of prediction with respect to training data, test data and validation data by varying certain parameters.

1) **<u>Using linear kernel (all other parameters are kept default)</u>**
   - Accuracy of training data : 0.97286
   - Accuracy of test data : 0.9378
   - Accuracy of validation data : 0.9364

The linear kernel are best used when we have linearly separable data. The linear kernel is used to get a linearly separable hyperplane which correctly classifies the data. Using the linear kernel shows a very good accuracy for training data. The accuracies for test data and validation data are almost similar. This shows that the data provided to us is well separated by a linear hyperplane.

2) **<u>Using radial basis function with value of gamma setting to 1</u>**
   - Accuracy of training data : 1.0
   - Accuracy of test data : 0.1714
   - Accuracy of validation data : 0.1548

When we use radial basis function as a kernel two parameters have to be considered which are C and gamma. Here we are keeping the C parameter to default which is 1 and setting the value of gamma to 1. Gamma defines the influence of a single training example. Default value for gamma is 1/number of features. However, we have changed this value to 1 which is much greater than the default value. If the gamma is small then the similarity between two points slowly decreases as the distance between them increases. This leads to a smoother decision surface and acts as a regularization parameter. However, for a large gamma value, the kernel function or the similarity between two points decreases sharply as the distance between them increases. This results in complex decision surfaces which lead to overfitting issue. As we can see from the accuracies above this is a classic overfitting problem as gamma has been set to 1 which is pretty high. This doesn’t yield good results for test and validation data as can be seen above.

3) **<u>Using radial basis function with value of gamma setting to default</u>**
   - Accuracy of training data : 0.94294
   - Accuracy of test data : 0.9442
   - Accuracy of validation data : 0.9402

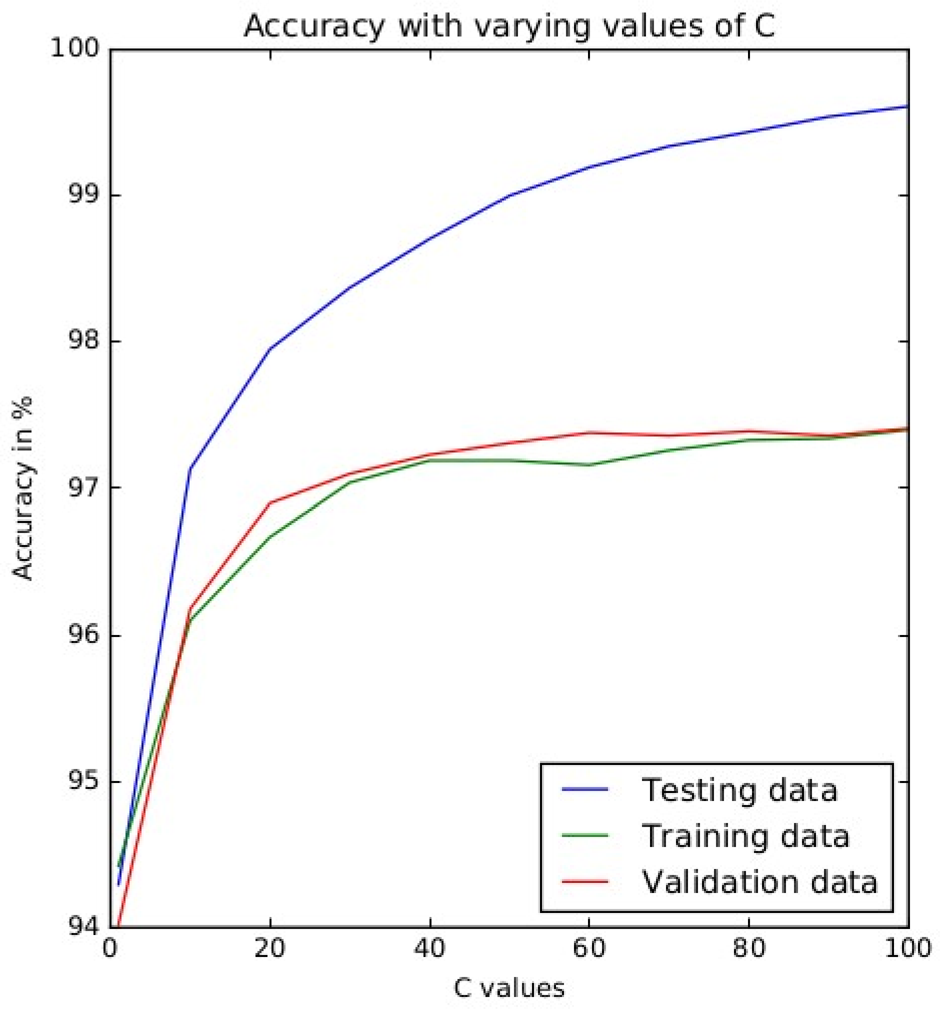
Radial basis function maps the d dimensional data to infinite dimensional space and finds a line which can correctly classify the data. Then brings that line back to the original d dimensional space so that we can learn non linear boundaries as the line becomes non linear. When training an SVM with radial basis function kernel two parameters have to be considered which are C and gamma. Both these parameters have got the default values in this case. The default value of C is 1.0 and the default value for gamma is 1/number of features.

4) **<u>Using radial basis function with value of gamma setting to default and varying value of C (1, 10, 20, 30, … , 100)</u>**

   - For C = 1
     - Accuracy of train data: 0.94294
     - Accuracy of test data: 0.9442
     - Accuracy of validation data: 0.9402
   - For C = 10
     - Accuracy of train data: 0.97132
     - Accuracy of test data: 0.961
     - Accuracy of validation data: 0.9618
   - For C = 20
     - Accuracy of train data: 0.97952
     - Accuracy of test data: 0.9667
     - Accuracy of validation data: 0.969

- For C = 30
  - Accuracy of train data: 0.98372
  - Accuracy of test data: 0.9704
  - Accuracy of validation data: 0.971
- For C = 40
  - Accuracy of train data: 0.98706
  - Accuracy of test data: 0.9719
  - Accuracy of validation data: 0.9723
- For C = 50
  - Accuracy of train data: 0.99002
  - Accuracy of test data: 0.9719
  - Accuracy of validation data: 0.9731
- For C = 60
  - Accuracy of train data: 0.99196
  - Accuracy of test data: 0.9716
  - Accuracy of validation data: 0.9738
- For C = 70
  - Accuracy of train data: 0.9934
  - Accuracy of test data: 0.9726
  - Accuracy of validation data: 0.9736
- For C = 80
  - Accuracy of train data: 0.99438
  - Accuracy of test data: 0.9733
  - Accuracy of validation data: 0.9739
- For C = 90
  - Accuracy of train data: 0.99542
  - Accuracy of test data: 0.9734
  - Accuracy of validation data: 0.9736
- For C = 100
  - Accuracy of train data: 0.99612
  - Accuracy of test data: 0.974
  - Accuracy of validation data: 0.9741

When training an SVM with radial basis function kernel two parameters have to be considered which are C and gamma. The C value trades off some slack that is misclassification against the simplicity of decision boundary surface. So, for lower value of C the misclassification could be a bit high while the decision boundary would be smooth. However, as we increase the value of C, it aims at classifying all the training examples correctly thus increasing the accuracy.



In the above plot, X axis refers to the C values which are ranging from 1, 10, 20, …., 100. The Y axis shows accuracy of the SVM model in %. Observing the graph we can clearly see that for all three types of data (testing data, training data, validation data) the accuracy increases as we increase the C value.

Comparison between linear kernel and radial basis function kernel

The linear kernel are best used when we have linearly separable data. Linear kernel is faster to train as compared to non linear kernels like RBF. If the data is not linearly separable then RBF has to be applied. For the given mnist data, time taken for training using linear kernel was roughly 10-15 minutes while RBF (with gamma set to 1) took about 4 hours to complete.
The linear kernel is a degenerate version of the RBF. Quoting an abstract from the below link:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.880&rep=rep1&type=pdf

"The analysis also indicates that if complete model selection using the Gaussian kernel(rbf) has been conducted, there is no need to consider linear SVM."

RBF tries to map the data to an infinite dimensional space. We can see from the accuracies given below that a finely tuned RBF kernel is better and more accurate that the linear kernel although linear kernel is faster in terms of time taken.

	Linear Kernel	**Radial Basis Function Kernel (C = 100)**
Training data accuracy	0.97286	0.99612
Test data accuracy	0.9378	0.974
Validation data accuracy	0.9364	0.9741

The table above clearly shows that a properly tuned RBF kernel (in our case, when C = 100) provides much better accuracy than a linear kernel for train data, test data and validation data.

Direct Multi-Class Logistic Regression

In this part we performed Multi Class Logistic Regression to classify hand-written digit images into correct corresponding labels. Traditional Logistic Regression could also be extended to solve multi class classification as done in the previous part. In this part, we implemented only 1 classifier that can classify 10 classes at the same time.
This part involved implementing two methods:

1. mlrObjFunction() - We performed the training of our classifier here.
2. mlrPredict() - The actual classification of data was done.

The following results were obtained after executing the above two functions:

Results:

Training set Accuracy:93.47%

Validation set Accuracy:92.34%

Testing set Accuracy:92.67%

Inference:

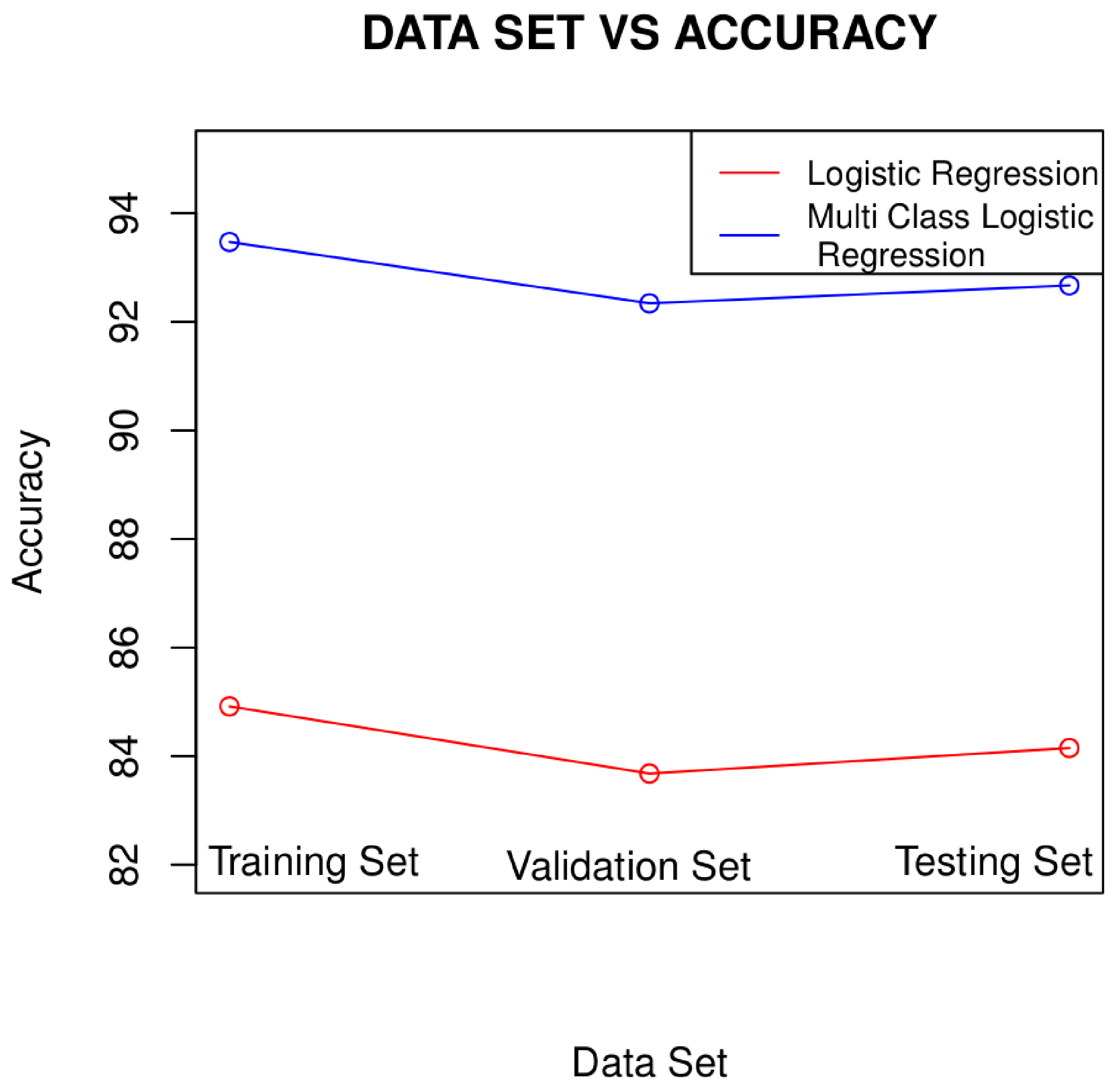
1. The training set has the highest accuracy of 93.47 %.

2. The validation set has the lowest accuracy of 92.34 %.

3. The testing set accuracy is 92.67 %. and is more like that of the training set but slightly lower.

Comparison of *one vs all* Logistic Regression with Multi Class Logistic Regression:

From the graph shown below, it can be clearly seen that Multi-Class Logistic Regression is better in Accuracy as compared to traditional Logistic Regression for Multi-Class Classification task.


DATA SET VS ACCURACY
Logistic Regression
Multi Class Logistic Regression
Accuracy
Training Set
Validation Set
Testing Set
Data Set