CSE574 Introduction to Machine Learning

Programming Assignment 3

Classification and Regression

Report

PROJECT GROUP 16

Rohit Balasayee

Vivek Nagaraju

Sriram Venkataramanan

Problem:

Our task in the third assignment is to solve the problem of MIMST data set classification i.e. the handwritten digit classification. The two important things to be done in this assignment is to implement Logistic regression and Support Vector Machine to classify the handwritten digits images and compare it's performance.

TASKS TO BE IMPLEMENTED

TASK 1
Implement Logistic Regression and give the prediction results.

TASK 2
Use the Support Vector Machine (SVM) toolbox sklearn.svm.SVM to perform classification.

TASK 3
Extra credit: Implement the gradient descent minimization of multi-class Logistic Regression (using softmax function)

DATSET

We are using the MNIST data set given to us. In the script provided to us, the function preprocess(), with pre-processing steps has already been implemented. This will apply feature selection, feature normalization, and divide the dataset into 3 parts: training set, validation set, and testing set.

SOLUTION:

TASK 1 : Logistic Regression

In the script given to us we have implemented both the methods blrObjFunction() and blrPredict() for implementing logistic regression.

Before we analyse the results of logistic regression, we need to know the basic concepts of

Logistic regression

- Logistic Regression is a classification algorithm that is used where the response variable is categorical.
- It is a statistical method/procedure for analysing a dataset in which there are more than one independent variable that can determine an outcome.
- It is used to explain the relationship between one dependent binary variable and one or more nominal independent variables.

Binomial logistic regression

A binomial logistic regression (often referred to simply as logistic regression), predicts the probability that an observation falls into one of two categories of a dichotomous dependent variable based on one or more independent variables that can be either continuous or categorical.

In our task one we need to implement this Binomial logistic regression and we implemented those in **blrObjFunction() and blrPredict()**.

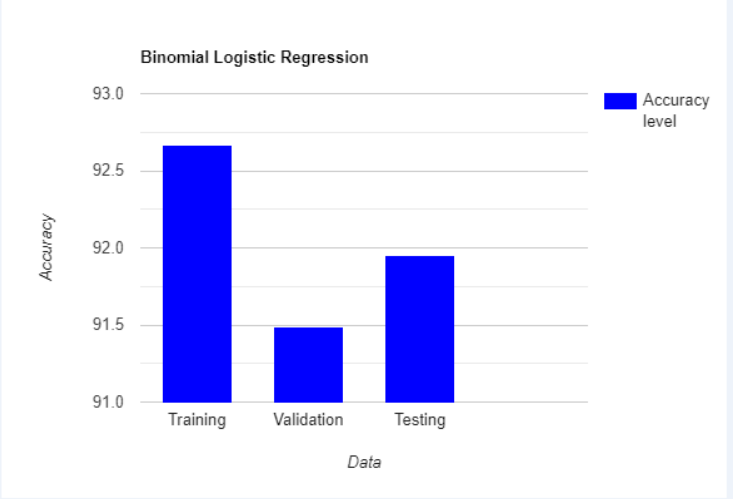
The accuracy obtained for this case are

Training Accuracy Obtained	Validation Accuracy Obtained	Test Accuracy Obtained
92.674%	91.490%	91.950%

Thus the error in each case be found out as

Training Error	Validation Error	Test Error
7.326%	8.51%	8.05%

GRAPH: BINOMIAL LOGISTIC REGRESSION



The error obtained in predicting the individual digits 1,2..10 in Training Testing and Validation in Binomial Logistic regression are tabulated as follow:

Digits	Training Error	Validation Error	Testing Error
1	3.93%	5.7%	5.62%
2	4.45%	7.46%	4.38%
3	7.47%	9.20%	6.71%
4	8.52%	11.2%	10.17%
5	7.48%	8.67%	8.69%
6	10.46%	9.38%	10.75%
7	4.8%	6.08%	6.11%
8	6.31%	6.29%	7.32%
9	10.91%	10.43%	11.68%
10	10.15%	11.12%	10.18%

REASONING FOR TRAINING ERROR VS TESTING ERROR

- The features in training set might vary from those in testing set, which leads to variation in errors between training and testing set
- There are a number of possible combination of edges is there to represent a digit. For example when you take 7(It comprises of 2-3 edges). In training set certain combinations might not be covered, thus when you test the data, error is likely to happen while predicting and results won't be accurate

---

TASK 2 is written down after TASK 3

TASK 3: Multinomial Logistic Regression

- It is a classification method that is used to generalize a logistic regression to multiclass i.e. (more than 2 class) problems, i.e. it can have greater than two possible discrete outcomes.

- In simple words, multinomial logistic regression is a classification technique which is used to predict more than one target class. In our MNIST data set, the idea is to predict the different digits 0; 1; 2;.. 9.

We implemented this multinomial class distribution in the 2 functions **mlrObjFunction() and mlrPredict().** The accuracy obtained for this case are:

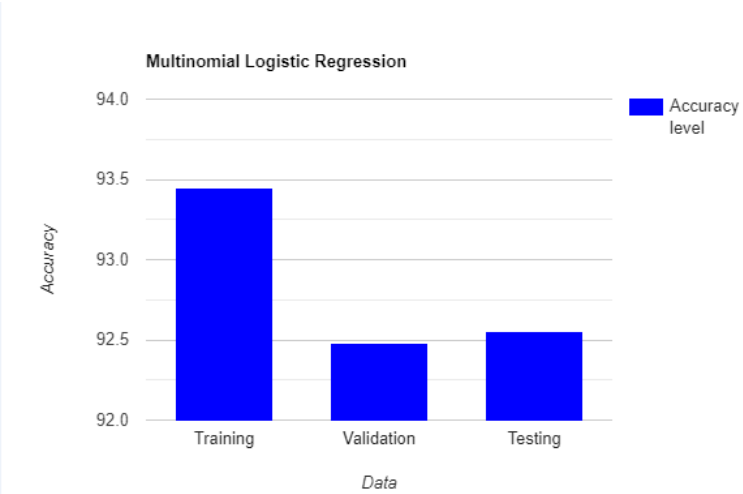
The accuracy obtained for this case are

Training Accuracy Obtained	Validation Accuracy Obtained	Test Accuracy Obtained
93.448%	92.479%	92.55%

Thus the error in each case be found out as

Training Error	Validation Error	Test Error
6.552%	7.521%	7.45%

GRAPH: MULTINOMIAL LOGISTIC REGRESSION



Hence in this case, multinomial logistic regression performed slightly better than Binomial logitstic regression, but it's not a guaranteed always this can happen.

The error obtained in predicting the individual digits 1,2..10 in Training Testing and Validation in Multinomial Logistic regression are tabulated as follow:

Digits	Training Error	Validation Error	Testing Error
1	3.14%	4.79%	5.05%
2	3.74%	5.18%	3.48%
3	6.79%	8.01%	6.1%
4	7.86%	10.79%	10.22%
5	6.41%	7.84%	7.07%
6	9.99%	8.49%	9.49%
7	4.9%	5.9%	6.26%
8	5.59%	5.68%	6.4%
9	10.25%	9.59%	12.86%

10	7.97%	9.17%	8.47%

REASONING FOR TRAINING ERROR VS TESTING ERROR

- The features in training set might vary from those in testing set, which leads to variation in errors between training and testing set
- There are a number of possible combination of edges is there to represent a digit. For example when you take 7(It comprises of 2-3 edges). In training set certain combinations might not be covered, thus when you test the data, error is likely to happen while predicting and results won't be accurate

TASK 2:

In-order to implement SVM, we need to know the basics of it.

SUPPORT VECTOR MACHINE

- A Support Vector Machine (SVM) is a classifier which works by finding the separating hyperplane between the classes.
- In other words, given labelled training data (supervised learning), the algorithm outputs a hyperplane which can used to categorize any new input.
- In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.

The SVM algorithm is implemented in practice using a kernel. A kernel transforms an input data space into the required form. SVM uses a technique called the kernel trick.

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role.

Linear Kernel:

- **Linear Kernel** is used when the data is Linearly separable, that is, it can be separated using a single Line.
- It is one of the most common kernels to be used. It is mostly used when there are a Large number of Features in a particular Data Set.
- One of the examples where there are a lot of features, is **Text Classification**, as each alphabet is a new feature. So we mostly use Linear Kernel in Text Classification.

Polynomial Kernel:

- The **polynomial kernel** is a **kernel** function commonly used with **support vector machines** (**SVMs**) and other kernelized models, that represents the similarity of vectors (training samples) in a feature space over **polynomials** of the original variables, allowing learning of non-linear models..
- The polynomial kernel can distinguish curved or nonlinear input space.

Radial Basis Function Kernel:

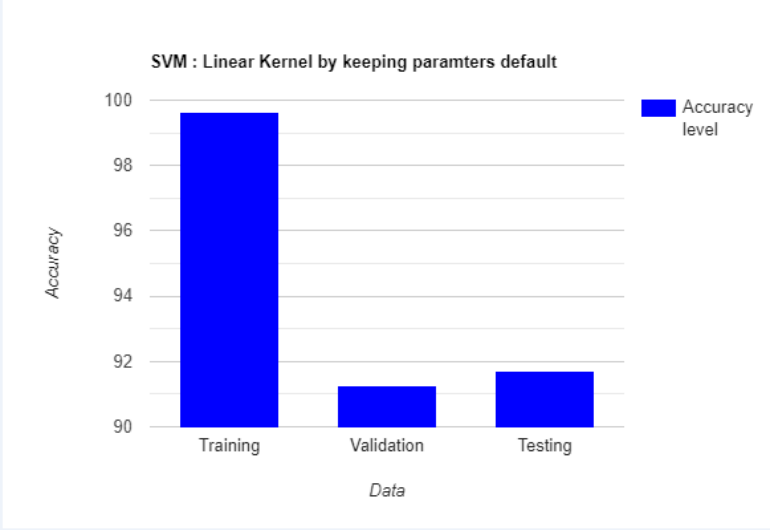
- In machine learning, the radial basis function **kernel**, or **RBF kernel**, is a popular **kernel** function used in various kernelized learning algorithms.
- In particular, it is commonly used in **support vector machine**classification. may be recognized as the squared Euclidean distance between the two feature vectors.
- RBF can map an input space in infinite dimensional space

All the above are implemented in code and the accuracy results are noted down.

CASE 1: **Used a Linear Kernel by keeping the other parameters default**.

The values obtained are:

Training Accuracy Obtained	Validation Accuracy Obtained	Test Accuracy Obtained
99.65%	91.28%	91.71%



PARAMETERS OF RBF KERNEL

Before we go into implementing the RBF kernel by changing the parameters, it is important to understand the parameters used in it.

Regularisation parameter:

- The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example
- For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly.
- Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

Gamma parameter:

- The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'.
- In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line.
- Whereas high gamma means the points close to plausible line are considered in calculation.

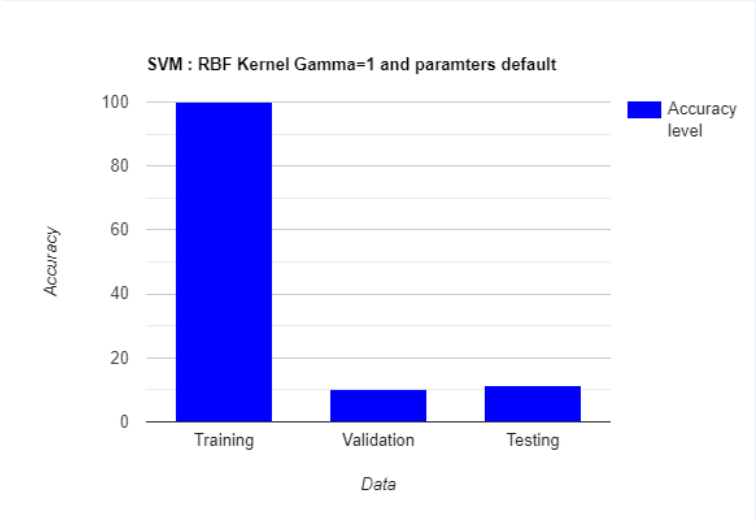
MARGIN:

- Finally last but very importrant characteristic of SVM classifier. SVM to core tries to achieve a good margin.
- ***A margin is a separation of line to the closest class points.***
- A ***good margin*** is one where this separation is larger for both the classes.
  A good margin allows the points to be in their respective classes without crossing to other class.

Now varying the above mentioned parameters, the code is run and the results obtained are noted down.

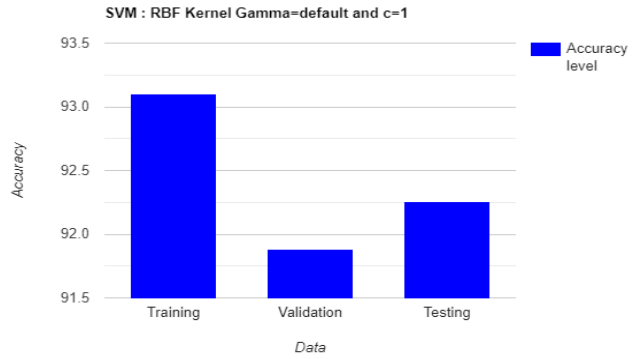
CASE 2: **Used a Radial basis function with value of gamma = 1 and all other parameters are kept default:**

Training Accuracy Obtained	Validation Accuracy Obtained	Test Accuracy Obtained
100.0%	10.0%	11.35%



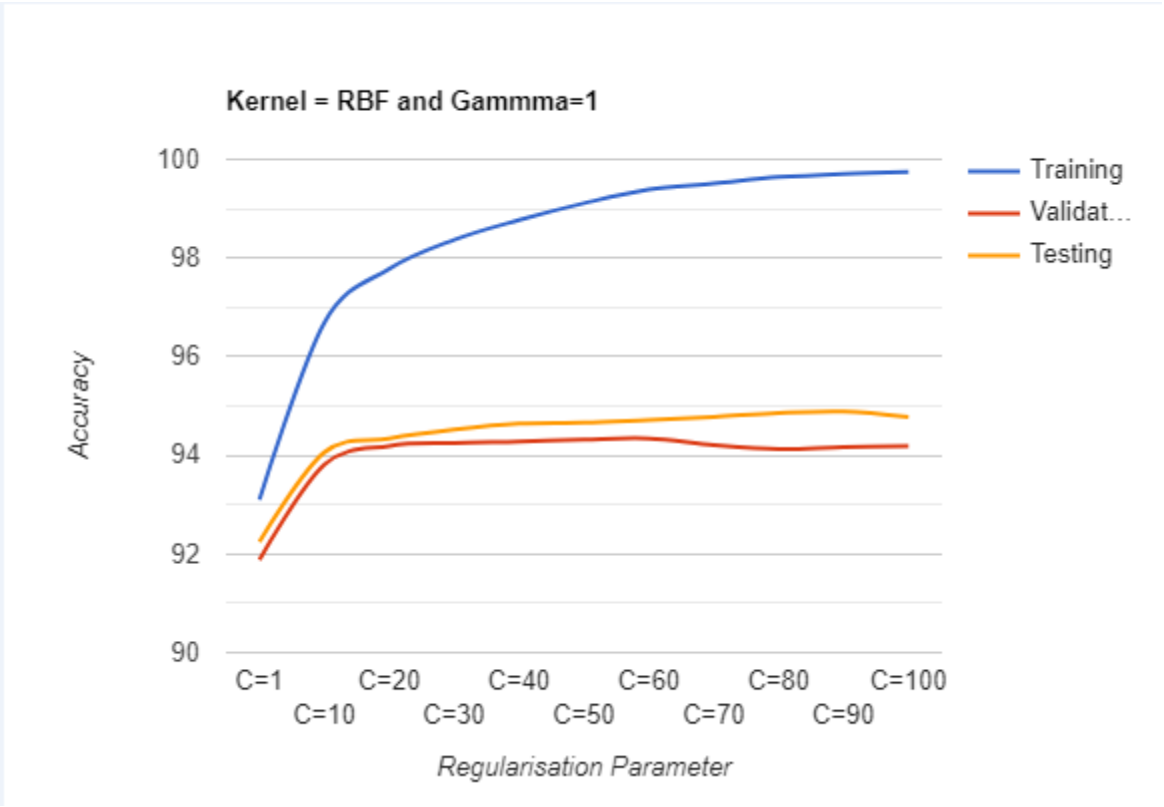
CASE3: **Used a Radial basis function with value of gamma=default and C=1**

Training Accuracy Obtained	Validation Accuracy Obtained	Test Accuracy Obtained
93.10%	91.88%	92.259%



CASE4: **Used Radial basis function with value of gamma setting to default and varying value of C (1; 10; 20; 30; …. ; 100). The values of Accuracy obtained are:**

Types of Accuracy	Kernel = 'RBF' and the analysis for varying values of c with gamma =1 and gamma =default											
	gamma=1	gamma= 'Auto'										
	C=1	C=1	C=10	C=20	C=30	C=40	C=50	C=60	C=70	C=80	C=90	C=100
Training Accuracy	100.0%	93.10	96.69%	97.78%	98.37%	98.77%	99.11%	99.39%	99.51%	99.64%	99.71%	99.75%
Validation Accuracy	10.0%	91.88	93.81%	94.19%	94.25%	94.28%	94.32%	94.34%	94.21%	94.13%	94.17%	94.19%
Testing Accuracy	11.35%	92.259	94.06%	94.34%	94.52%	94.64%	94.66%	94.72%	94.78%	94.86%	94.89%	94.78%

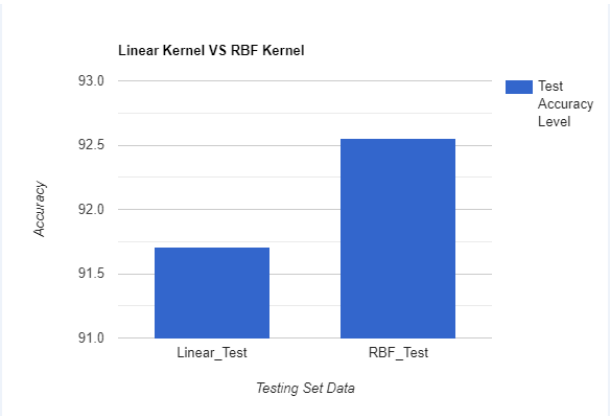


OBSERVATIONS

- In sum SVM is all about Margin Maximization based on the given parameters. In the above experiment conducted, when the values of C and gamma gets high, it leads to overfitting and when the values of C and gamma become low, it leads to under fitting.
- Thus it is very important to adjust the parameter settings without leading to overfitting and under fitting.

COMPARISON OF A LINEAR KERNEL VS RBF KERNEL

- As you can see from the accuracies obtained, the RBF kernel tends to perform better compared to a linear Kernel.
- The Test accuracy obtained in Linear kernel is 91.71%
- The Test accuracy obtained in case of RBF kernel is above 92% and it reaches 94% for certain parameter Tuning.



Thus you can see RBF kernel performed slightly better than Liear kernel.

CONCLUSION:

- Binomial Logistic regression is implemented and the accuracy values are noted down. The two functions blrObjFunction() and blrPredict() are implemented in the given script.
- Multinomial Logistic regression has been implemented in the two functions **mlrObjFunction() and mlrPredict() for extra credits.**
- SVM has been implemented using the inbuilt functions. The parameters were changed and the accuracy values are noted down

REFERENCES

- https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72