Report for Assignment 3

Vishesh Singh Thakur
`50322513, vthakur@buffalo.edu`

Ashwin Nair
`50321006, anair3@buffalo.edu`

Divya Gawande
`50340326, divyagaw@buffalo.edu`

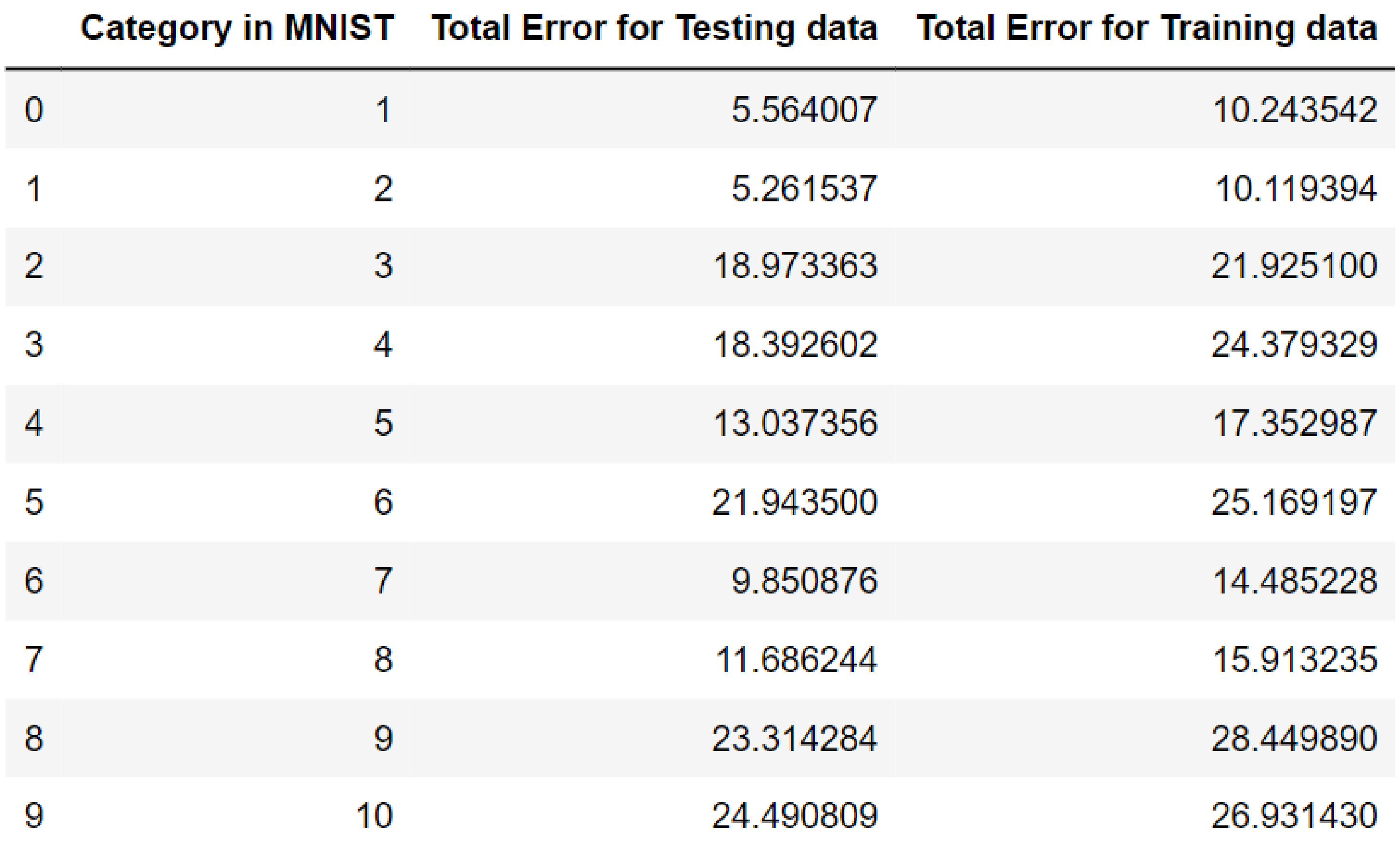
Group 2
CSE 574 Introduction to Machine Learning
December 2020

Abstract

In this assignment, we are using the MNIST dataset, which consists of handwritten digit images. We want to implement binary logistic regression for 10 different categories of numbers and classify them into corresponding labels. We also need to build a multi-class logistic classifier, which, unlike the binary logistic classifier, builds just 1 classifier instead of 10, to classify all 10 classes at the same time. Lastly, we need to implement Support Vector Machines for a fraction of randomly selected training examples for different parameters, and select the best choice of parameters to train the entire dataset

Binary Logistic Regression

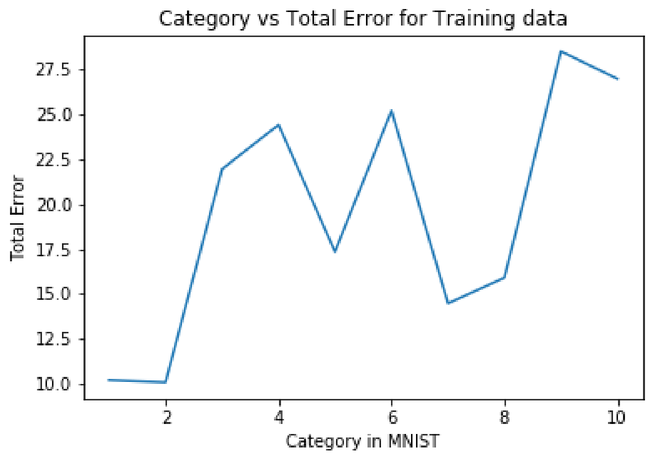
The following table gives a comparison between the total errors calculated by the blrObjFunction when using train data and test data

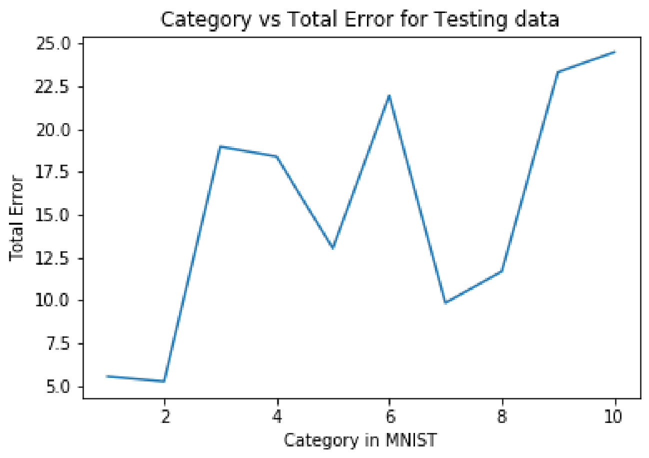


	Category in MNIST	Total Error for Testing data	Total Error for Training data
0	1	5.564007	10.243542
1	2	5.261537	10.119394
2	3	18.973363	21.925100
3	4	18.392602	24.379329
4	5	13.037356	17.352987
5	6	21.943500	25.169197
6	7	9.850876	14.485228
7	8	11.686244	15.913235
8	9	23.314284	28.449890
9	10	24.490809	26.931430

We can see from the table that the total errors calculated for train data is always greater than the total errors calculated for test data for every category in the dataset. This can be attributed to the difference in number of samples taken in both train and test data. Train data has 50000 samples while test data has 10000 samples. Therefore, while test data has a lower total error for each category, train data has a better total error per sample for every category

The following graph shows the variation in total error for both train and test data for each category





The following table shows the accuracies calculated by the blrObjFunction when using train data and test data

DATA USED	**ACCURACY CALCULATED**
TRAINING DATA	**92.696%**
VALIDATION DATA	**91.53%**
TEST DATA	**91.97%**

Multiclass Logistic Regression

The following table shows the total errors calculated using the mlrObjFunction

TOTAL ERROR FOR TRAIN DATA	TOTAL ERROR FOR TEST DATA
62.23434073389485	45.41213243506712

From the table above, it can be observed that the total error for test data is less than the total error for train data. The difference arises because the number of sample examples used in both sets are different. The train data has 50000 training examples, and hence has a much different learned model as compared to the model learned using test data which has 10000 examples to learn on. While train data has more total error as compared to the total error from the test data, it has less amount of error per sample data collected.

The following table shows the accuracies of training set, test set, and validation set calculated

DATA USED	ACCURACY CALCULATED
TRAINING DATA	93.448%
VALIDATION DATA	92.480%
TEST DATA	92.55%

Support Vector Machines

To begin, we randomly selected 10000 samples from train data and train label. We then used that to train our Support Vector Machine models for different settings. The following table gives the training, testing, and validation accuracies for each setting

For the default value of C = 1, we have the following table that shows the accuracies for different types of kernels

Kernel Method	Gamma Value	Training Accuracy	Testing Accuracy	Validation Accuracy
Linear	N/A	93.082	91.81	91.25
Radial Basis Function	1	34.414	17.99	15.98
Radial Basis Function	default	92.092	92.50	92.07

We notice that for the default values, Radial kernel performs better than the linear kernel, as Radial kernel maps the input features into an infinite dimensional space. Generally, non-linear kernel models perform better than linear kernels.

For the Radial basis function kernel, we see that the accuracies of all training, testing, and validation are drastically low at gamma value of 1, that is because the Gamma value controls the influence of each training example on the learned hyperplane, so, for a value too high, we have overfitting with comparatively much lower testing and validation accuracies as compared to training accuracy. We have the training accuracy of 34.414% instead of 100% because we have set the model to take random 10000 values from the entire training set, instead of the whole set.

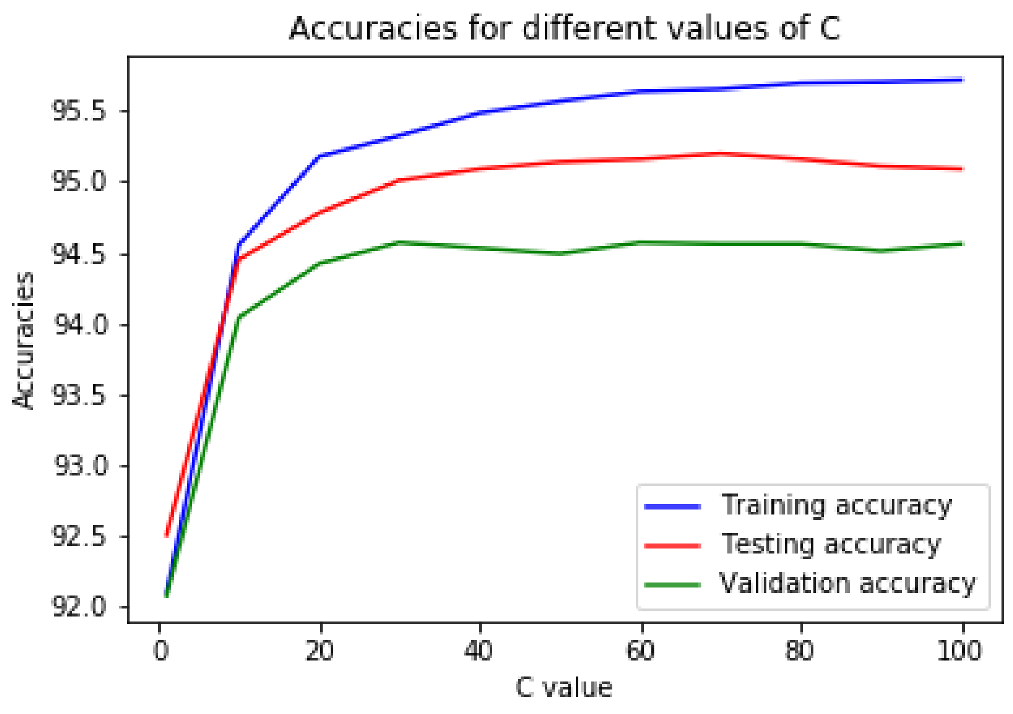
Naturally, the model performs much better for the same kernel but with gamma value of 0 (or close to 0 since SVC sets it to 1 / Number of features)

The above models are performing for the default value of C, which is set to 1. The parameter C determines the impact of errors on the training samples, and hence, controls the complexity of the learned hyperplane. A lower value of C gives a larger marginal hyperplane, whereas a higher value of C gives a smaller marginal hyperplane.

For the Radial kernel, with default value of gamma, we see how the model performs for the varying values of C in the following table

	C Value	Training Accuracy	Testing Accuracy	Validation Accuracy
0	1	92.092	92.50	92.07
1	10	94.554	94.45	94.04
2	20	95.178	94.78	94.42
3	30	95.326	95.01	94.57
4	40	95.486	95.09	94.53
5	50	95.570	95.14	94.49
6	60	95.638	95.16	94.57
7	70	95.656	95.20	94.56
8	80	95.698	95.16	94.56
9	90	95.706	95.11	94.51
10	100	95.720	95.09	94.56

The following plot shows the variation of accuracies for different values of C



From the previous plot, it can be observed that we get higher accuracies for higher values of C. This is because C controls the penalty for error term for each training example.

When the value of C is low, the weight of each term is small, and consequently a larger value of error is accepted during training and as a result a higher marginal hyperplane is created. When we start increasing the values of C, the weight on error terms increases as well, thus accepting lower error values and as a result creating smaller marginal hyperplane. Less number of points are misclassified and hence the accuracy increases.

After comparing all the accuracies, the highest validation accuracy is observed for the Radial kernel, with default value of gamma, at C value of 60. Using those values, we train the entire training set and calculate the accuracies. The following table gives us the accuracies for the entire dataset using the best choice of parameters

SAMPLE DATA	ACCURACY ACHIEVED
TRAINING DATA	**99.196%**
VALIDATION DATA	**97.38%**
TESTING DATA	**97.16%**