Image Classification using Logistic Regression with Stochastic Gradient Descent

Aaron Briel

7/23/2018

1. Implementation

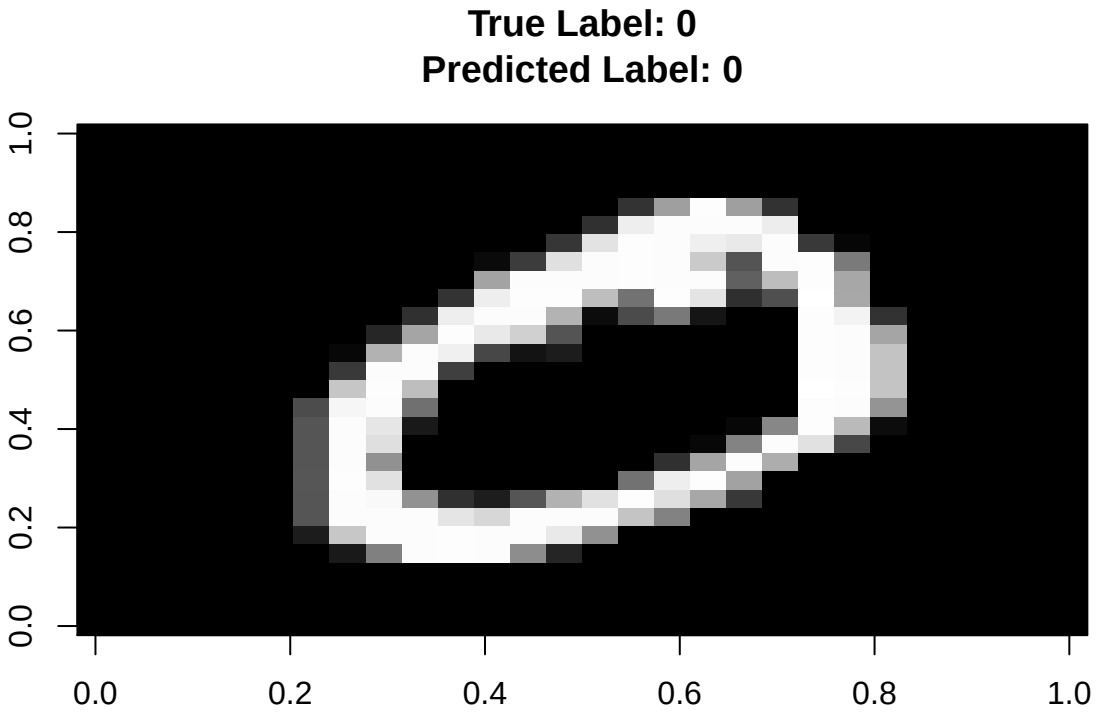
The train and test data are extracted from mnist_train.csv and mnist_test.csv and then partitioned for classification of 0/1 and 3/5 classes. The true class labels are then pulled from these partitions and stored in separate label vectors for test 0/1, test 3/5, train 0/1 and train 3/5, where 0 and 3 values are assigned the -1 label and 1 and 5 values are assigned to 1. The label row is removed from the datasets, thus separating the true class label from all the partitions created. It is assumed that bias is intrinsic in the training data and labels, so a value for this is not explicitly set. Although accuracies were generally consistent with observations across multiple seeded and non-seeded test runs, a seed was set to ensure reproducible results.

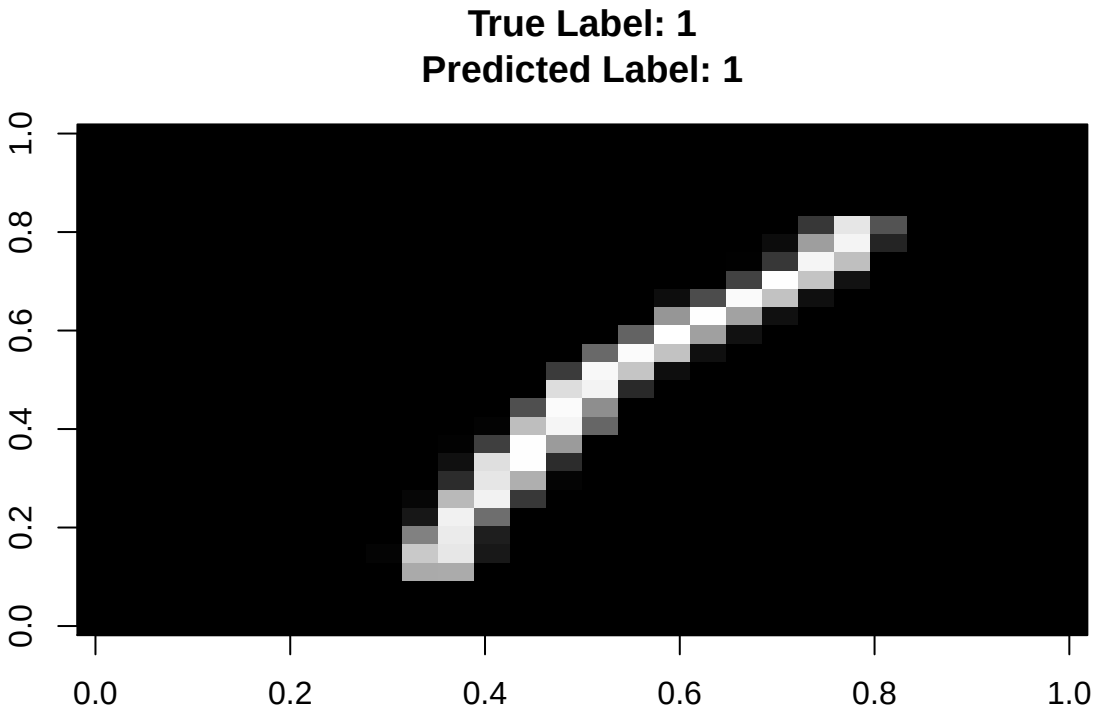
The train function implements Logistic Regression using Stochastic Gradient Descent. It accepts data, labels and alpha, and has a default convergence threshold of 0.0005. dataset_name is another optional parameter that can be set for displaying accuracy values during epochs. theta is initialized to random values between -1 and 1. Meanwhile, theta_old and theta_new, used later for the convergence criteria, are set to all 100s and 200s respectively.

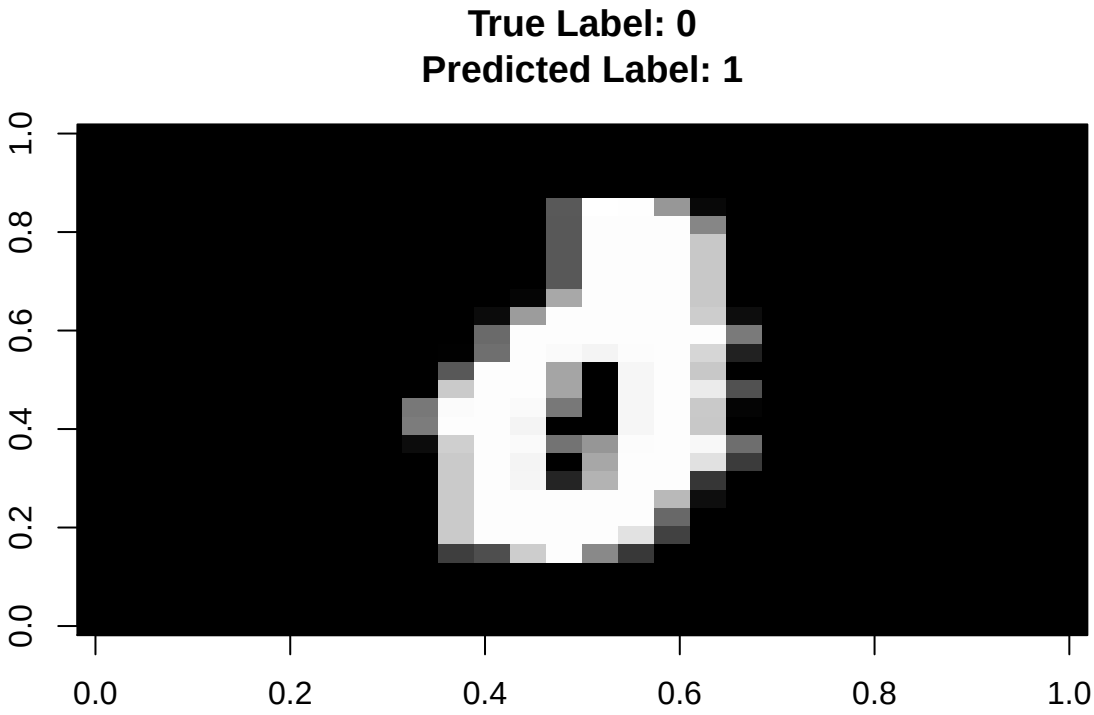
The convergence criteria specifies that iterations can proceed while the absolute value of any value in theta_new - theta_old is greater than the convergence threshold and the current epoch is less than the maximum number of epochs. Within this while loop an index vector is sampled randomly from 1 to the number of data columns (samples). A loop then iterates through all of the samples of data in the dataset passed in, pulling data and labels from the ith value of the index vector. This approach ensures that a random sample of data and its corresponding label are used for each iteration. The implementation of the derivative of the loss function is then executed within the loop to update theta, and a conditional is in place to break from from the loop if convergence occurs prior to a full iteration over the data samples. After the loop, theta_new is then set to the updated theta value. Once the convergence criteria is met, theta is returned.

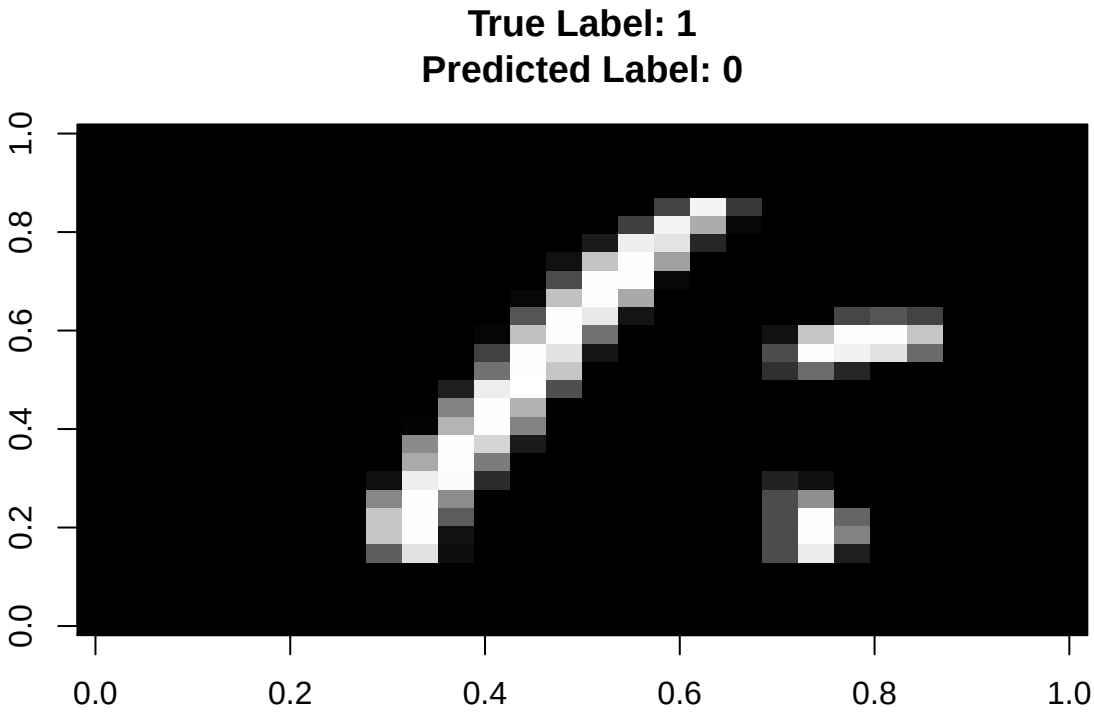
The predict function calculates a "sign" vector by computing the dot product of a dataset and theta. The values in the resulting vector are set to -1 for values less than 0, and to 1 for values above 0.

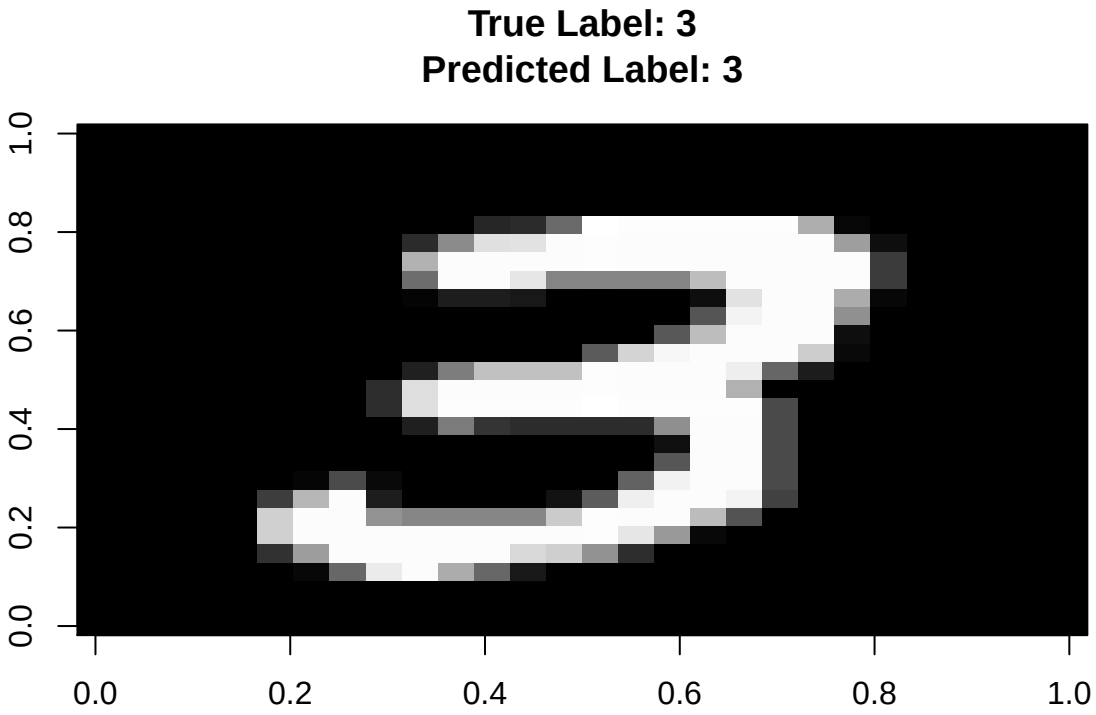
Below are visualizations of two correct and two incorrect predictions for the 0/1 and 3/5 training sets respectively, including the true labels and predicted labels.

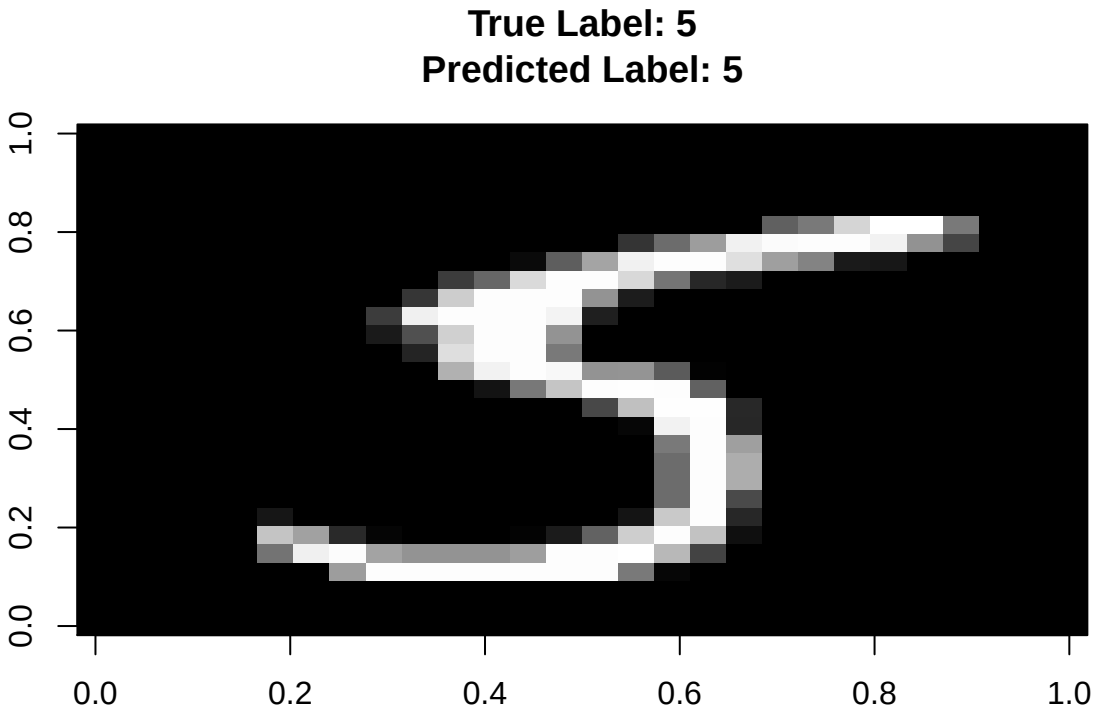

True Label: 0
Predicted Label: 0
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

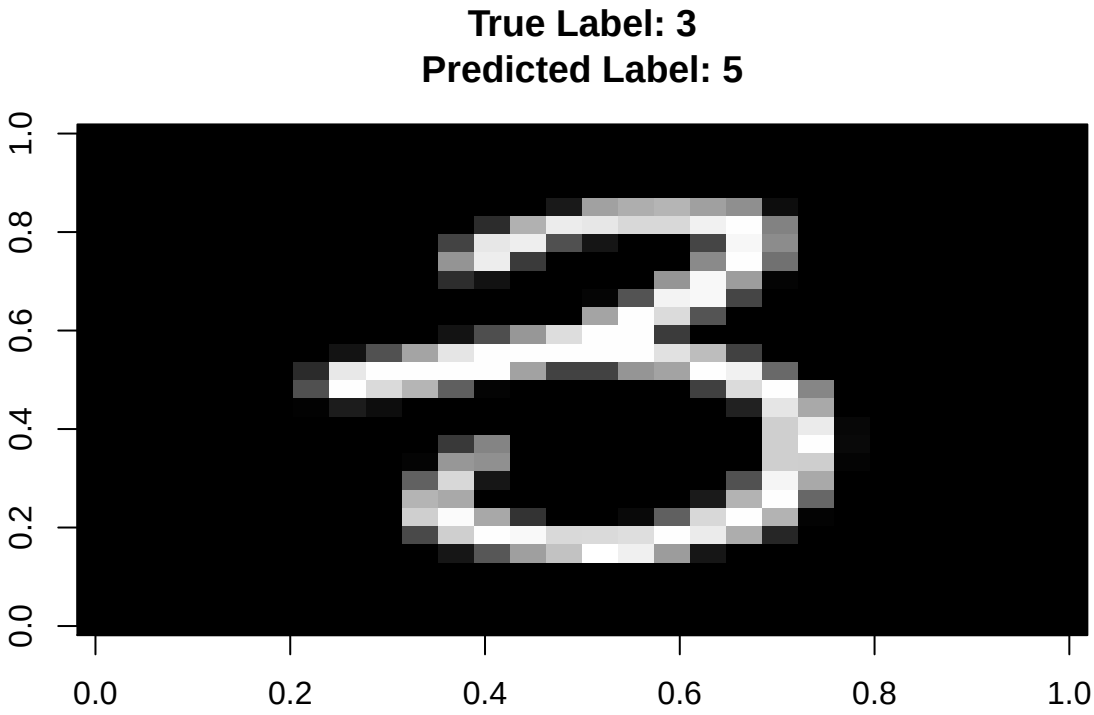

True Label: 1
Predicted Label: 1
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

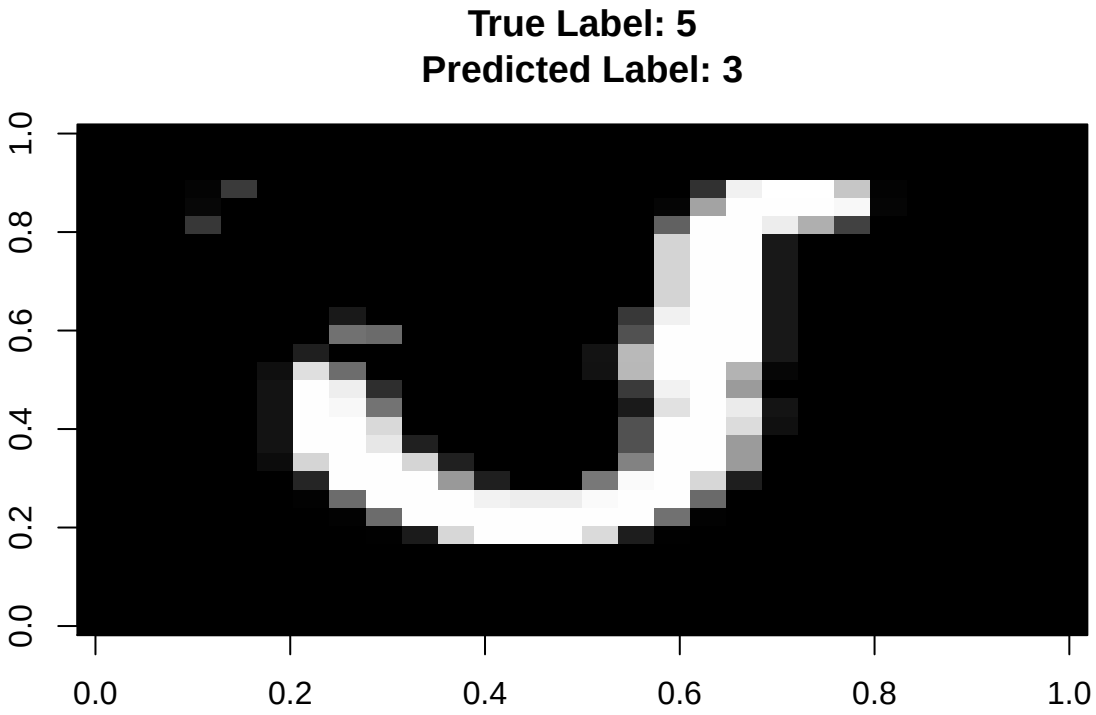

True Label: 0
Predicted Label: 1
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


True Label: 1
Predicted Label: 0
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


True Label: 3
Predicted Label: 3
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


True Label: 5
Predicted Label: 5
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0


True Label: 3
Predicted Label: 5
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

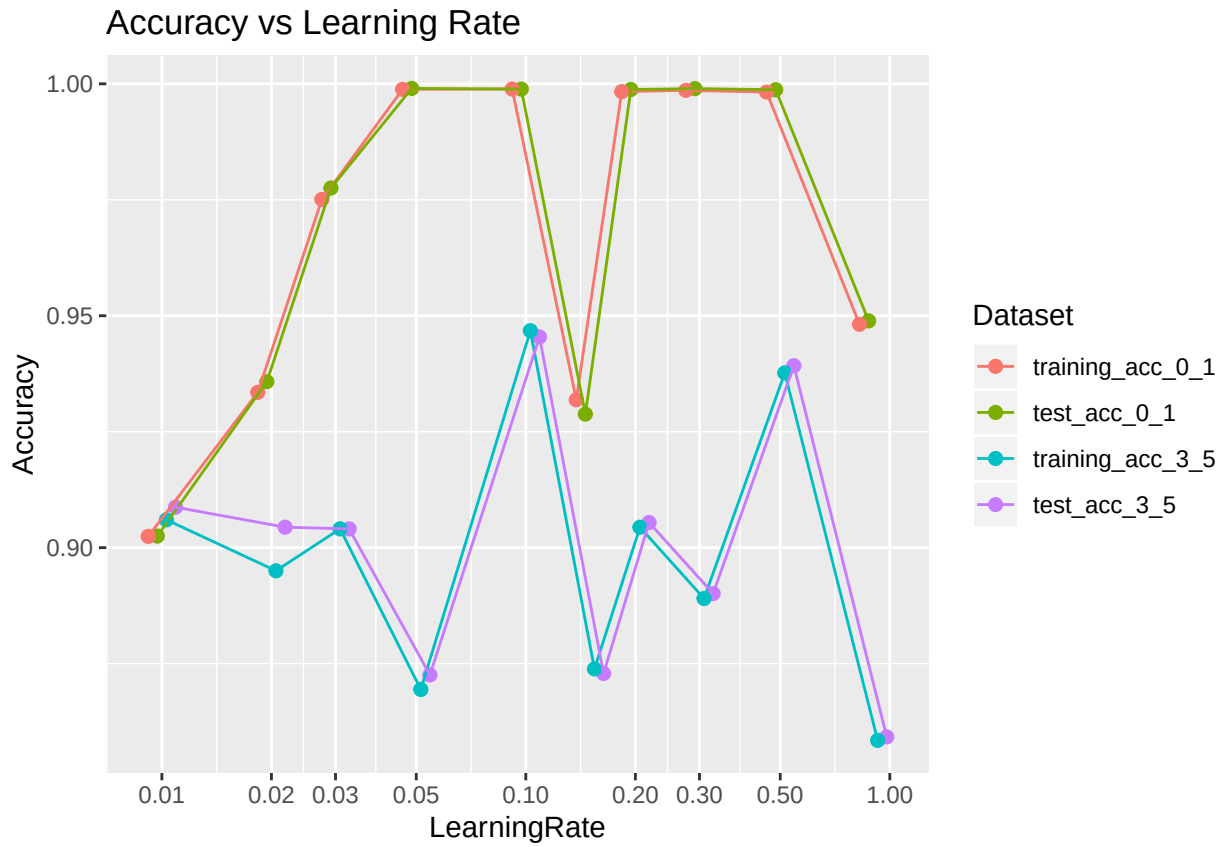


2. Modeling

Two models were trained, one on the 0/1 dataset and another on the 3/5 dataset, and their training and test accuracies were displayed.

```
## [1] "Running train with dataset: train_data_0_1"
## [1] "Epoch: 1"
## [1] "Accuracy: 0.998499802605606"
## [1] "Epoch: 2"
## [1] "Accuracy: 0.998499802605606"

## [1] "Model 0/1 training accuracy: 0.998499802605606"

## [1] "Model 0/1 test accuracy: 0.998581560283688"

## [1] "Running train with dataset: train_data_3_5"
## [1] "Epoch: 1"
## [1] "Accuracy: 0.941222299168975"
## [1] "Epoch: 2"
## [1] "Accuracy: 0.941308864265928"

## [1] "Model 3/5 training accuracy: 0.941308864265928"

## [1] "Model 3/5 test accuracy: 0.936908517350158"
```

The above step was repeated ten times with learning rates among the following set: 0.01, 0.02, 0.03, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5, 0.9. Average accuracies were calculated by repeating the training runs ten times for each learning rate. Resulting mean accuracies were then plotted against those rates.




```
##  [1] "0/1 Test Dataset Accuracies: 0.902505910165485"
##  [2] "0/1 Test Dataset Accuracies: 0.935791962174941"
##  [3] "0/1 Test Dataset Accuracies: 0.977541371158392"
##  [4] "0/1 Test Dataset Accuracies: 0.999007092198582"
##  [5] "0/1 Test Dataset Accuracies: 0.99886524822695"
##  [6] "0/1 Test Dataset Accuracies: 0.928794326241135"
##  [7] "0/1 Test Dataset Accuracies: 0.998770685579196"
##  [8] "0/1 Test Dataset Accuracies: 0.998959810874704"
##  [9] "0/1 Test Dataset Accuracies: 0.998723404255319"
## [10] "0/1 Test Dataset Accuracies: 0.948888888888889"

##  [1] "3/5 Test Dataset Accuracies: 0.908727655099895"
##  [2] "3/5 Test Dataset Accuracies: 0.904416403785489"
##  [3] "3/5 Test Dataset Accuracies: 0.904048370136698"
##  [4] "3/5 Test Dataset Accuracies: 0.872555205047319"
##  [5] "3/5 Test Dataset Accuracies: 0.945425867507886"
##  [6] "3/5 Test Dataset Accuracies: 0.872870662460568"
##  [7] "3/5 Test Dataset Accuracies: 0.905415352260778"
##  [8] "3/5 Test Dataset Accuracies: 0.890115667718191"
##  [9] "3/5 Test Dataset Accuracies: 0.939221871713985"
## [10] "3/5 Test Dataset Accuracies: 0.859200841219769"
```

As indicated in the Accuracy vs Learning Rate graph above, accuracy did not seem to be linearly related to the learning rates in the range tested. The best accuracy for the 0/1 test dataset was found to be 99%, while the best for the 3/5 test dataset was 95%. Accuracies were generally higher for the 0/1 training and test datasets than for the 3/5 datasets. It was postulated that this was likely due to the numbers 3 and 5 being more visually similar to each other than the numbers 0 and 1, resulting in a higher number of misclassifications. It should be noted that the test and train results were very similar for the respective datasets involved. Specifically, the test results for 0/1 were very close to the 0/1 train results, and the test results for 3/5 were very close to its train results.

In the situation where there is a requirement to classify more than two classes using Logistic Regression, a two dimensional label matrix could be leveraged. For example, a combined 0/1/3/5 dataset could store the following values for each respective number, where the first label (-1 or 1) could be stored in the first dimension and the second label (-1 or 1) could be stored in the second dimension:
0 = -1, -1
1 = -1, 1
3 = 1, -1
5 = 1, 1

The predict function would then need to be appropriately modified to accommodate this two-dimensional label matrix.

3. Learning Curves

The 0/1 and 3/5 datasets were each trained ten times with sample sizes increasing by 10% in each iteration. A nested for loop repeated the training 5 times for each sample size value to calculate the mean accuracies. The mean accuracies were then plotted against sample size.

The graphs for both datasets of 0/1 and 3/5 indicate that the accuracy from training data did not appear to be strongly affected by sample size, with the exception of unexpected dips in the 60% and 70% sample size range which were not reliably reproducible in other seeded or non-seeded test runs. The test data, however, did appear to be influenced by sample size. In the example graphed, the 0/1 test set had accuracies in the 50% range until reaching a sample size of 50%, at which point the accuracy drastically increased and approached the training set measurement. Similar results were observed in the 3/5 dataset, where the said jump in accuracy occurred with a 60% sample size.