CMPE 462 – PROJECT 1

Binary Classification with Logistic Regression

- **Student ID1:** 2014400051
- **Student ID2:** 2018400291
- **Student ID3:** 2019705072

Requirements and How to Run:

- The project uses some python libraries that need to be installed before starting the project. Namely, numpy, random, prettytable, matplotlib
- Also, the data needs to be in the folder data under data which is in the same workspace as the .ipynb file.

If you would like to load data from another folder. Specify it in the data loading cell. Dot means current working directory.

Load the training/test data and labels as numpy arrays (Hint: `np.load`). Train and test data are 1561x256 and 424x256 dimensional matrices, respectively. Each row in the aforementioned matrices corresponds to an image of a digit. The 256 pixels correspond to a 16x16 image. Label 1 is assigned to digit 1 and label -1 is assigned to digit 5.

In [3]:
```
#Read the data by using numpy library
train_data = np.load('./data/data/train_data.npy')
train_labels = np.load('./data/data/train_labels.npy')
test_data = np.load('./data/data/test_data.npy')
test_labels = np.load('./data/data/test_labels.npy')
```

INTRODUCTION:

In this project, we implemented a solution to a handwritten digit classification problem. The dataset we used for this project is some portion of the MNIST dataset. This project only deals with the classification of digit-1 and digit-5. So, this is binary classification problem.

The algorithms we have implemented are Feature Extraction, Logistic Regression with Gradient Descent and Cross-Validation.

TASK 1: Feature Extraction

Since this is a binary classification problem, we need to assign labels to the digits.

If label is equal to 1, the image is digit-1.

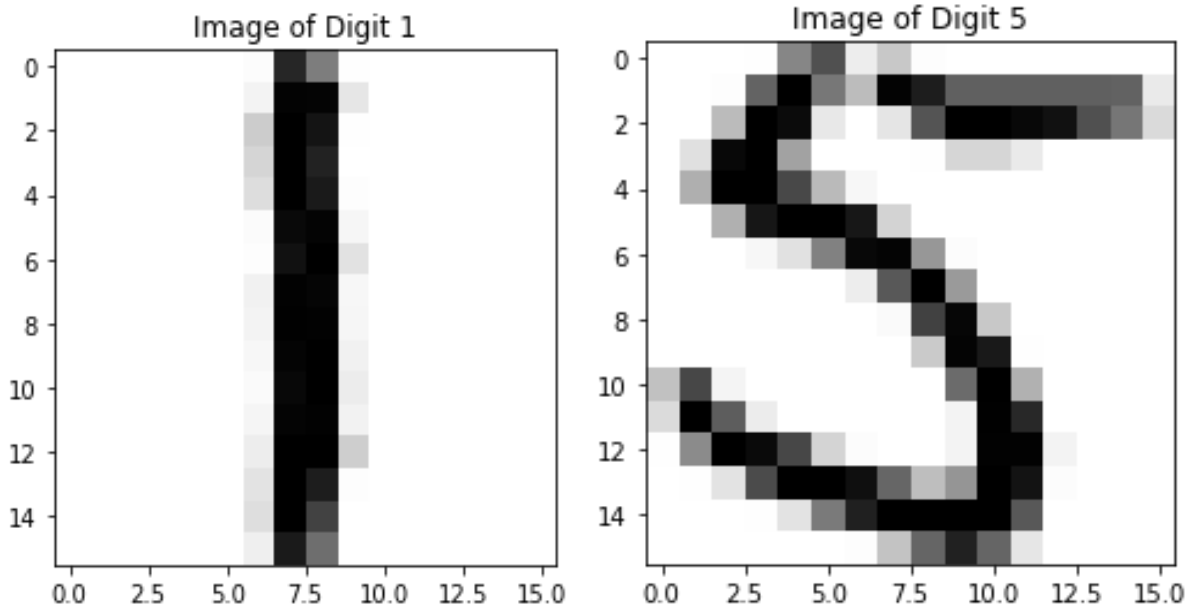
If label is equal to -1, the image is digit-5.

Reading the data:

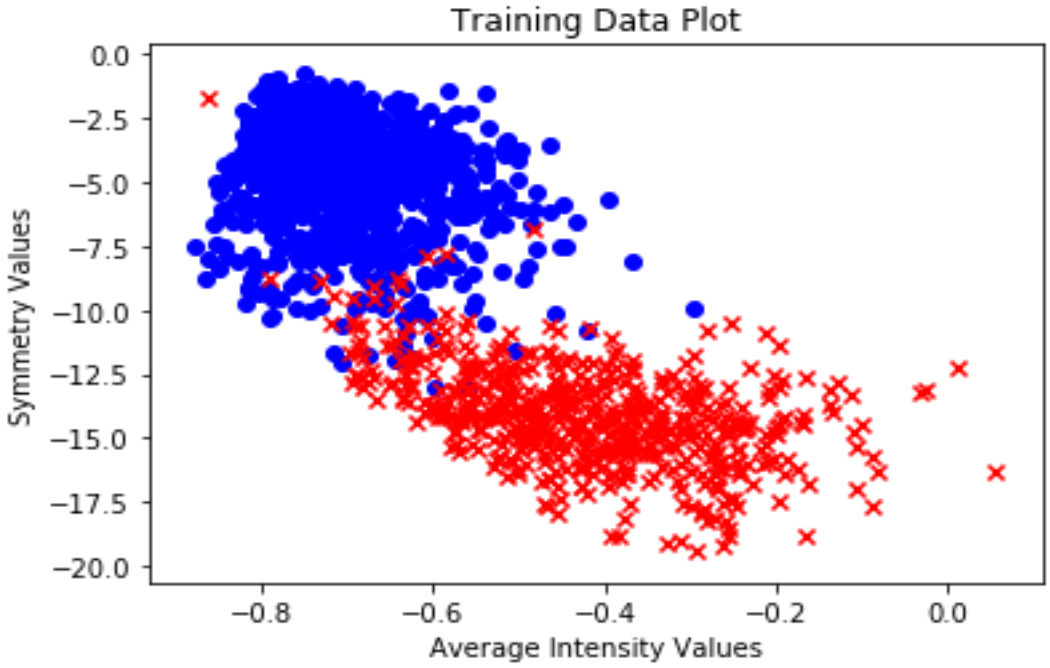
Load function of the Numpy (short as np) is used to load the data to memory.

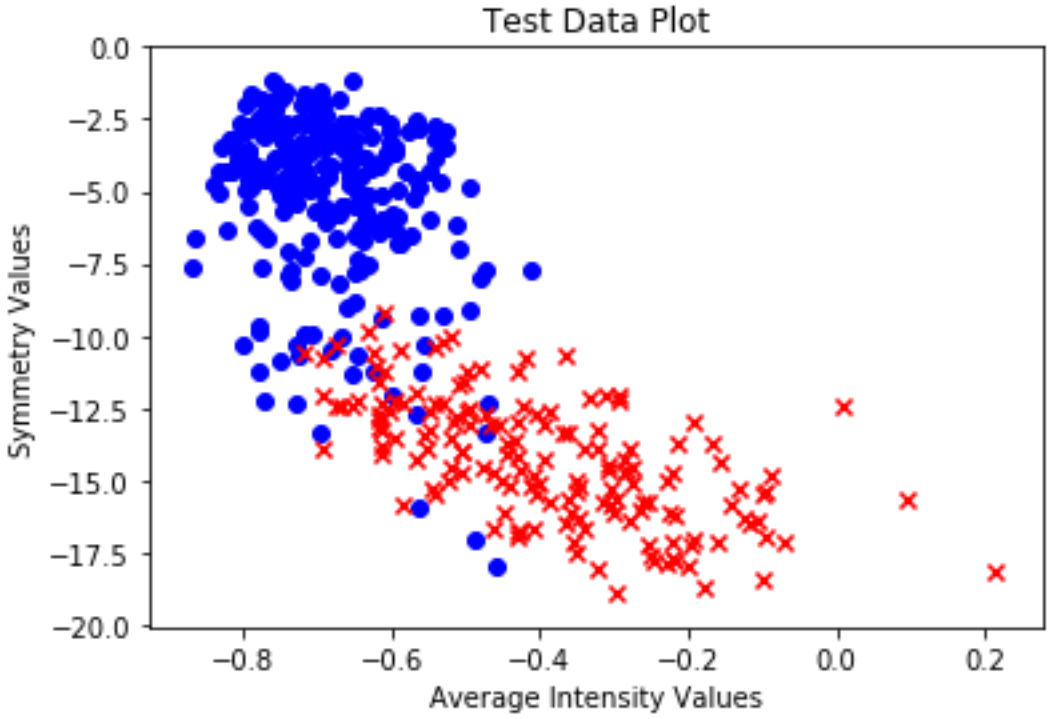
train_data, train_labels variables hold the train data and the labels for train data.

test_data, test_labels variables hold the test data and the labels for test data.



Representation 1: Features with symmetry and average intensity. By using these features, training and test data plots look like the following figure:

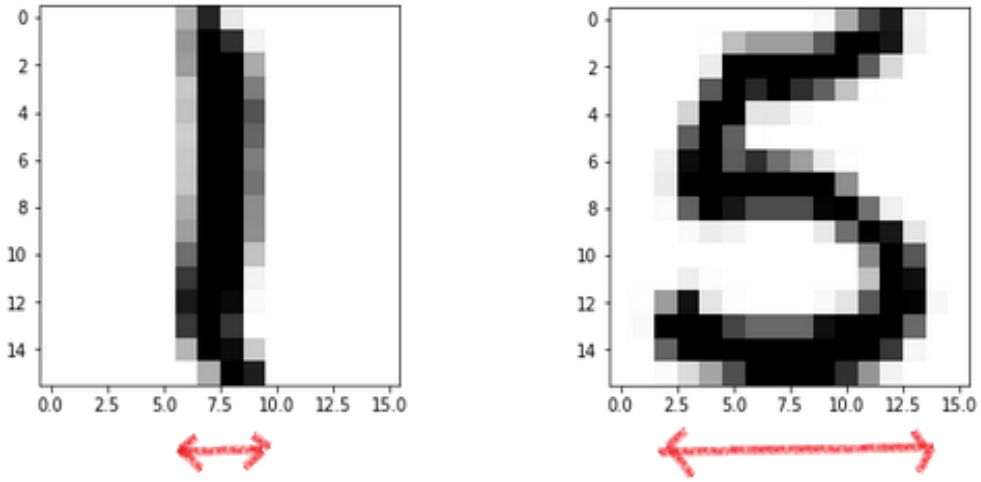




Representation 2: In this task, we used the width of the digit and number of color change for lines from diagonals and center horizontally.

Feature 1 - Width: This feature comes from the idea that digit-5 is fatter than digit-1. So, the algorithm looks for the starting and finishing index of pixels along columns. Then, takes the difference and puts the value to the features vector.

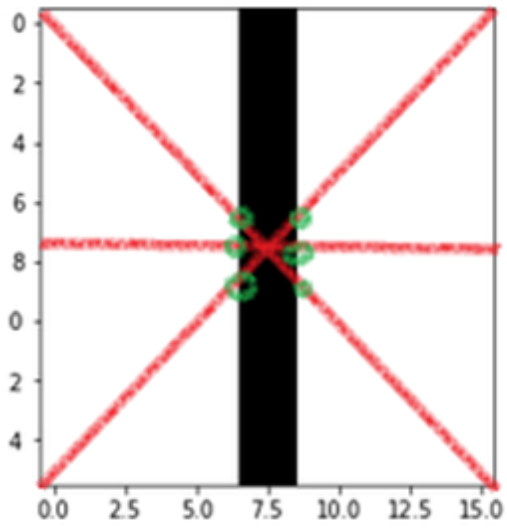
Below figure shows the difference between digit 1 and digit 5 in width feature.

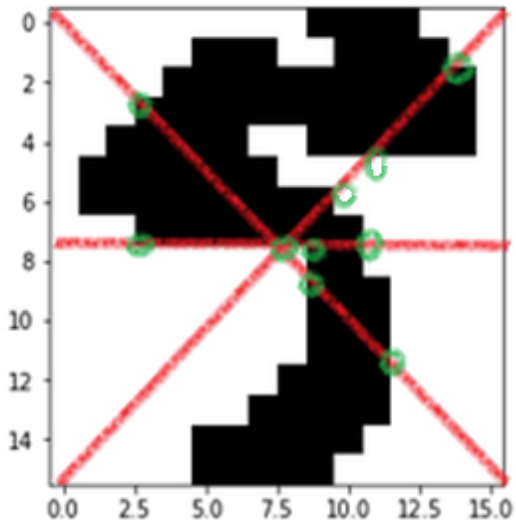


Feature 2 – Negative of Number of Color Changes: This feature comes from the idea that digit 5 has curvier shape. So, if you draw a line across certain angles, digit-5 needs to intersect the line, on average, more times than digit-1.

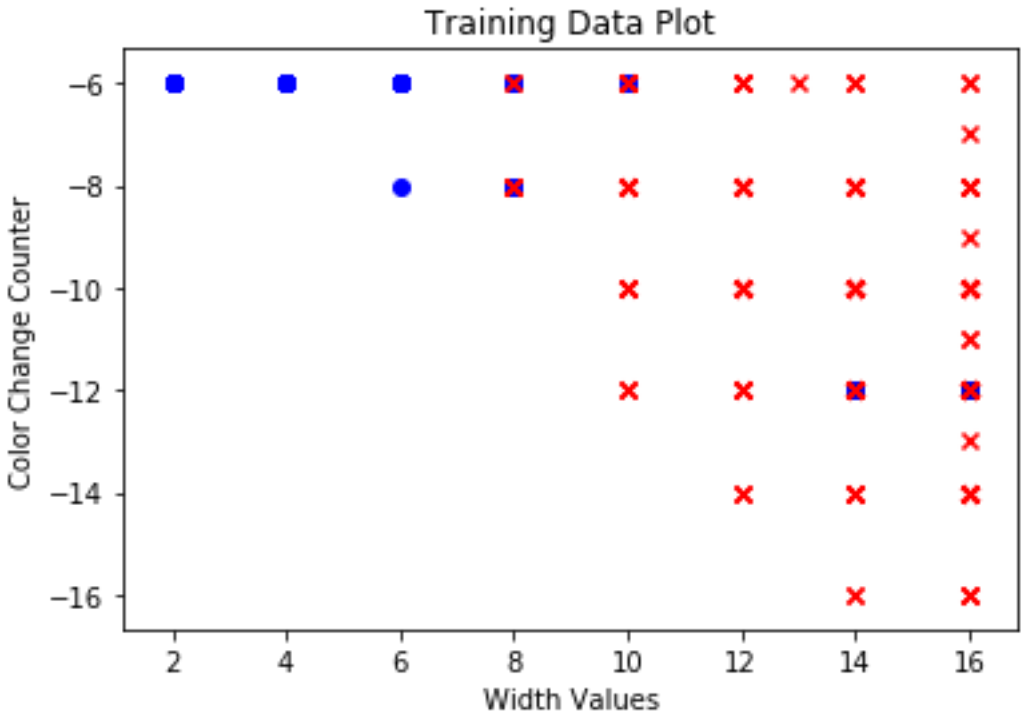
To count the intersection points, simply count the number of color changes along the line. The lines we used for this feature is left and right diagonals and a horizontal line from the center. To make calculations easier, pixel -1 values are made 0 and the rest is 1. This way, we can use the XOR operation to detect any change in value.

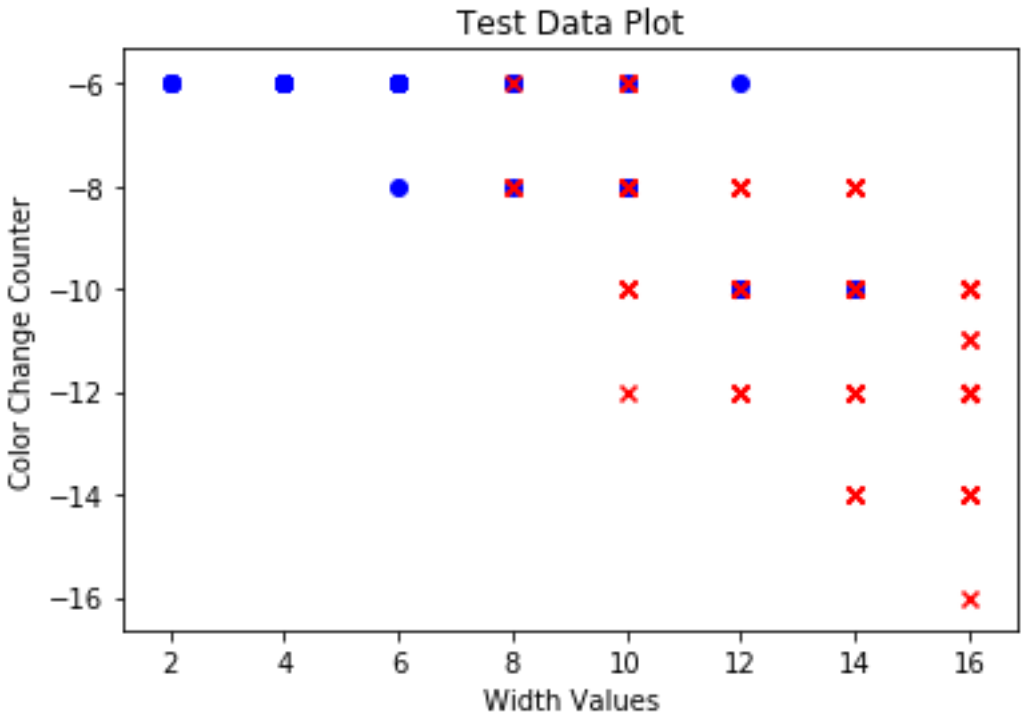
Below figure shows the difference between digit 1 and digit 5 in color change feature. Green circles represent the intersection points. Digit-1 has 6 color changes and digit-5 has 10 color changes. We expect more values from the digit5.





By using these features, train and test data plots look like the following figure:





In the rest of the document, representation 1 is used for the features recommended by the description. Representation 2 is used for the models that have features width and color change count.

TASK 2: Logistic Regression

We have a linear classification problem. To solve this problem, logistic regression will be used. Since logistic regression loss function has no analytical solution, numerical approach will be used to minimize the loss function.

Gradient of loss function has the following formula:

$$\nabla E(w) = \frac{\partial E(w)}{\partial w} = \frac{1}{N}\sum_{n=1}^{N}\frac{\partial(\ln(1+e^{-ynw^T xn}))}{\partial w} = \frac{1}{N}\sum_{n=1}^{N}\frac{\frac{\partial(1+e^{-ynw^T xn})}{\partial w}}{(1+e^{-ynw^T xn})}$$

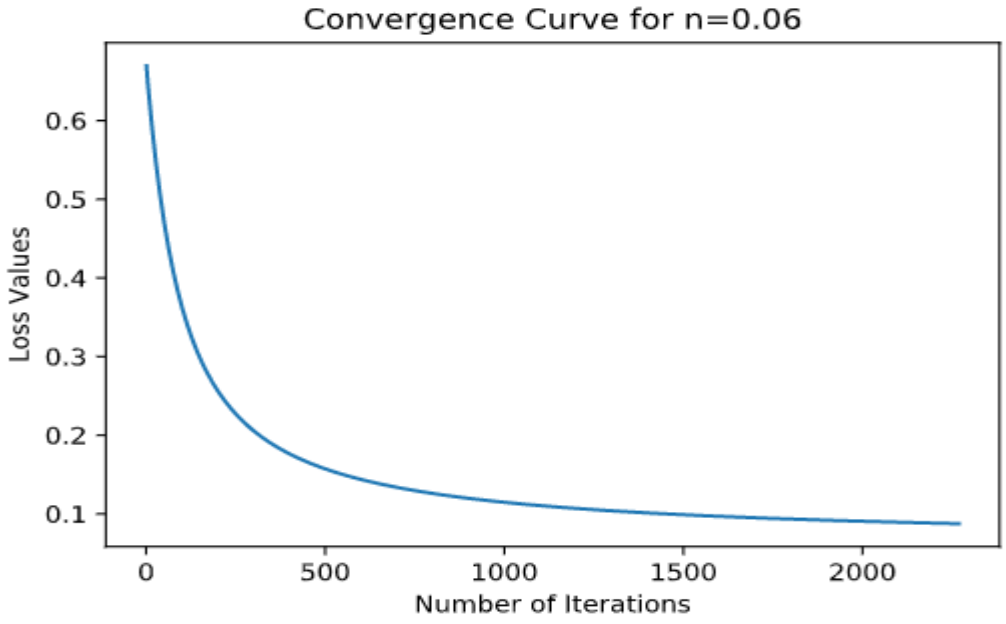
$$\nabla E(w) = \frac{1}{N}\sum_{n=1}^{N}\frac{-ynxne^{-ynw^T xn}}{(1+e^{-ynw^T xn})} \qquad , \nabla(w^T xn) = xn$$

$$\nabla E(w) = \frac{1}{N}\sum_{n=1}^{N}\frac{-ynxne^{-ynw^T xn}e^{ynw^T xn}}{(1+e^{ynw^T xn})}$$

$$\nabla E(w) = -\frac{1}{N}\sum_{n=1}^{N}\frac{ynxn}{1+e^{ynw^T xn}}$$

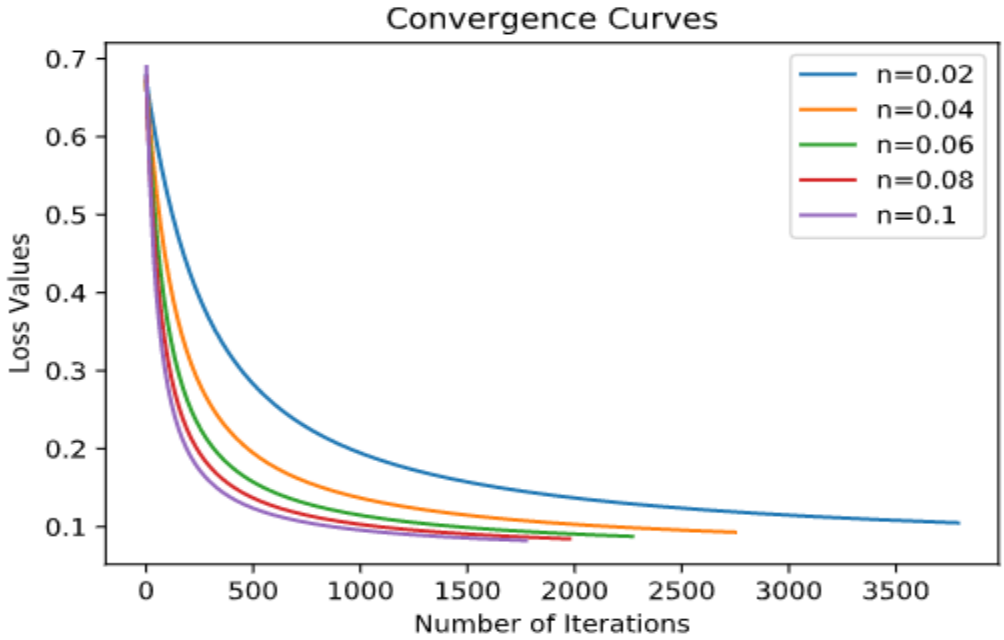
Representation 1:

Implementation is converging and the plot of the loss values with respect to iteration count is like the following plots.

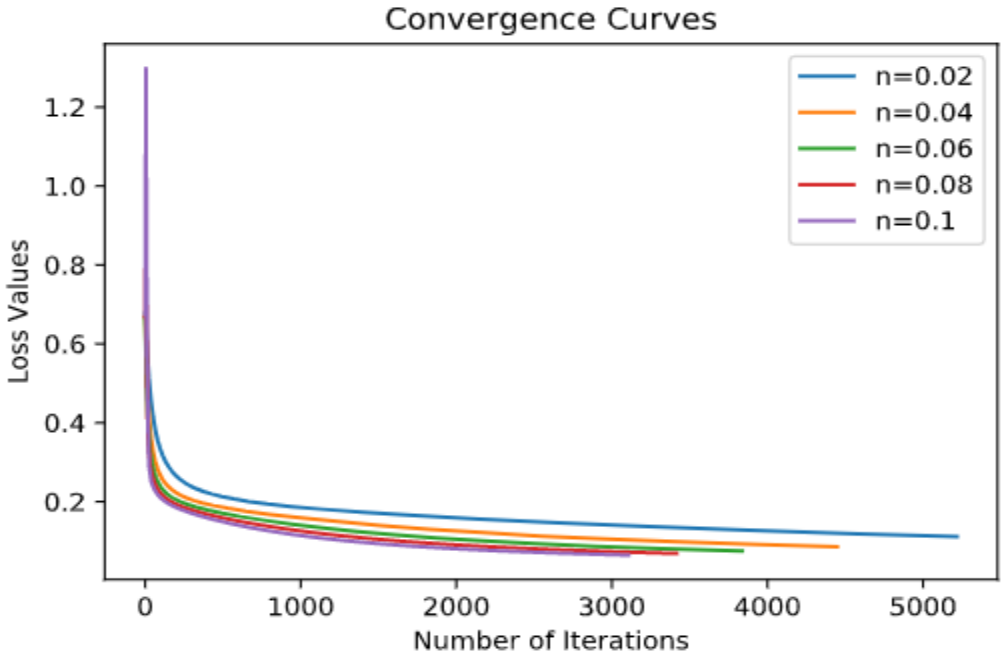


The following plot clearly shows that while learning rate is decreasing, number of iteration is increasing. In another perspective, one can say that while learning rate is increasing, the implementation converges faster. When we increased learning rate from 0.02 to 0.1, we got smaller loss values. When we run the learning algorithm with learning rate which is above the value 0.1, we got fluctuations on the curve. In other words, our aim is to reach minimum on the curve with gradient descent. When we use big step size which is learning rate, it

skips the minimum point and jumps points where loss values are big. It reaches suboptimal points. This is why curve fluctuates when we run the algorithm with learning rate is above the value 0.1. Thus fluctuations on the curve is undesired situation. All in all, learning rate with value 0.1 is just right step size for us.



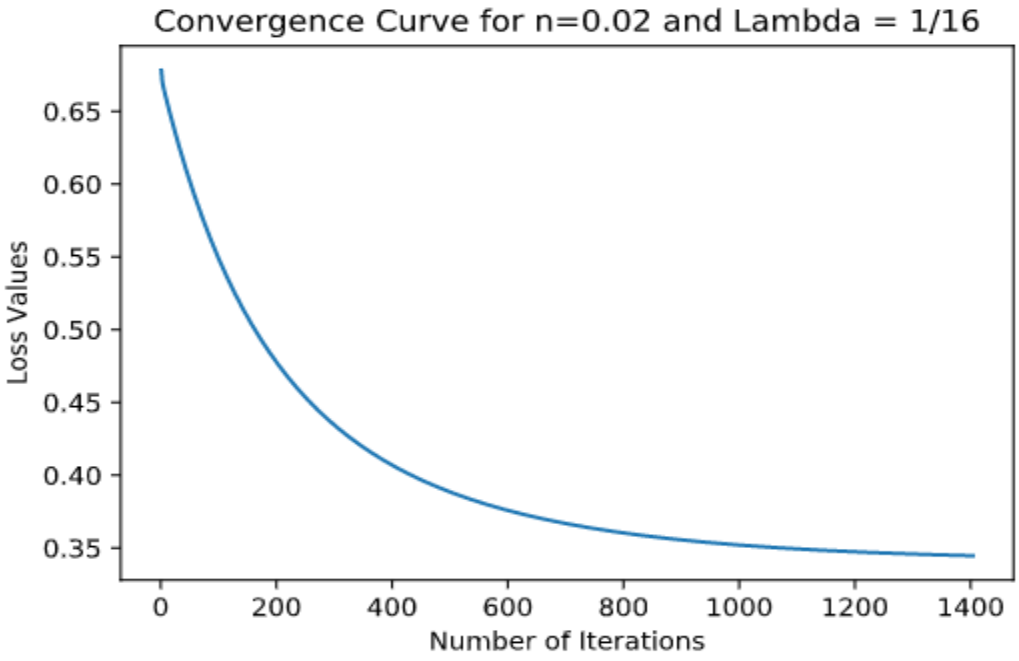
Representation 2: The same situation is valid with the above comment.



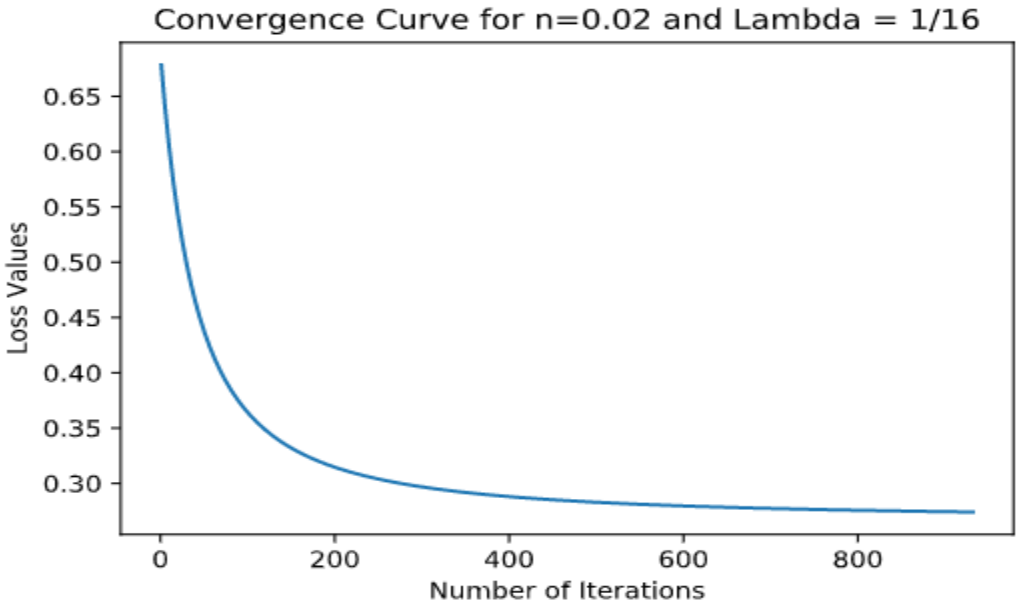
The following plots show that regularization doesn't work to decrease loss. Let's compare the above plot for representation 1 in 1400 iterations:

Loss value was less than 0.2. Adding the regularization constant made it about 0.35 and 0.3 for Representation 1 and Representation 2 respectively.

Representation 1 with regularization:



Representation 2 with regularization:



Cross Validation:

Lastly, 5-fold cross validation procedure is implemented for both representation 1 and representation 2 with a learning rate 0.1 and 4 different regularization parameters.

For representation 1, the table below shows that when regularization parameter (lambda) decreases, the accuracy mean is increasing and the standard deviation is decreasing. Therefore, one can say that the best lambda for this model is zero, which means regularization doesn't make the model better.

Representation 1 with n=0.1	Lambda = 1/16	Lambda = 1/32	Lambda = 1/64	Lambda = 1/128
Cross Validation Accuracy Mean	96.4102564102564	97.05128205128203	97.43589743589743	97.43589743589743
Cross Validation Std	1.3658510097860568	1.3507248401093266	1.3446267284232751	1.1992491624275459

One can also observe that the same situation with representation is still valid for representation 2.

Representation 2 with n=0.1	Lambda = 1/16	Lambda = 1/32	Lambda = 1/64	Lambda = 1/128
Cross Validation Accuracy Mean	94.23076923076923	94.23076923076923	94.93589743589743	96.21794871794873
Cross Validation Std	1.233037439850724	1.233037439850724	1.061086240849156	0.9377396691235761

TASK 3: Evaluation

Classification accuracy metric is defined as

$$\frac{\text{number of correctly classified samples}}{\text{total number of samples}} x100$$

The best learning rate is 0.1, based on the plots on the convergence curves for both representation 1 and representation 2.

By using this learning rate, training representation 1 and representation 2 gives the following accuracies:

Training classification accuracy of representation 1: **97.69%**

Test classification accuracy of representation 1: **95.05%**

Training classification accuracy of representation 2: **99.1%**

Test classification accuracy of representation 2: **96.93%**

As it is mentioned above regularization doesn't improve our models, so the best lambda is zero which means that we should build the model without using regularization. Accuracies of the model without regularization are given above.

The best regularization parameter (lambda) in the experiment list was 1/128 based on the above table for both representation 1 and representation 2.

By using this lambda, training the representation 1 and representation 2 gives the following accuracies:

Training classification accuracy of representation 1: **97.5%**

Test classification accuracy of representation 1: **95.05%**

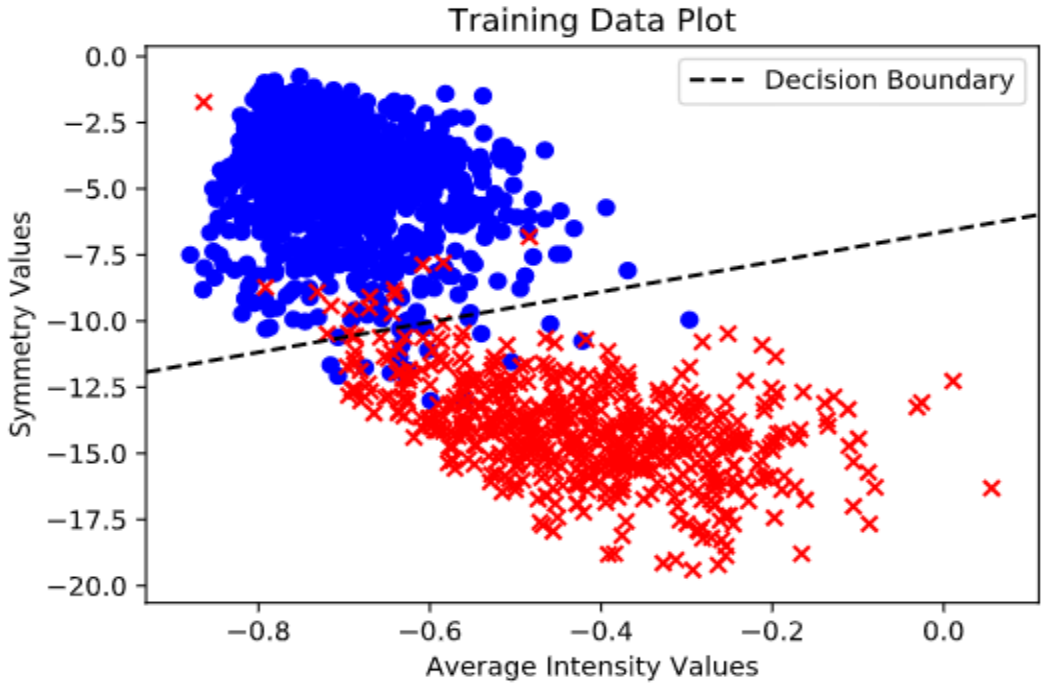
Training classification accuracy of representation 2: **96.22%**

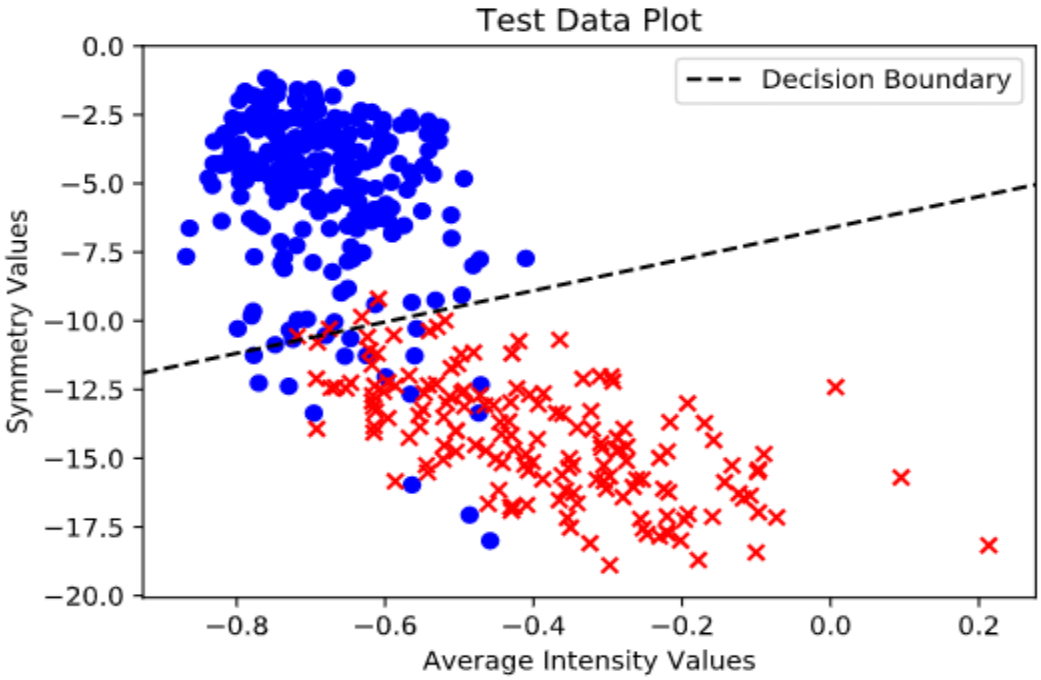
Test classification accuracy of representation 2: **94.34%**

Training and test classification accuracies are significantly high. As guessed, regularized models are giving lower accuracy than the one without regularized models.

Decision Boundary

The decision boundary visualization of logistic regression classifier learned without regularization for representation 1:





DISCUSSION

- Regularization did not improve generalization performance. It reduced generalization performance of both representation 1 and 2 slightly, When we trained with the best lambda which is 1/128 we obtained by 5-fold cross validation. When we increased value of lambda, both training and test accuracies gradually decreased. It did not help reducing the gap between training and test accuracies/errors. When we look at difference between training and test accuracies in both representation 1 and 2, It is small like 2% or 3% so We can say that there is no overfitting situation. Thus, We can expect increase in errors and decrease in accuracies while using regularization. In Representation 2, we got more training and test classification accuracy than Representation 1 but when we used regularization, Representation 1 has more training and test classification accuracy than Representation 2. We can expect this situation because elements of weight vector of Representation 2 have bigger value than elements of weight vector of Representation 1 so our best lambda value penalized Representation 2 more than Representation 1.

- Representation 2 gave the best results without regularization. According to accuracies and errors result, we can say that Representation 2 is more discriminative than Representation 1. It classified digits correctly more than Representation 1.

- To improve accuracy, we can add more features which is more discriminative than ours to Representation 2 to classify digits. In other words, we can extend our both feature vector and weight vector to construct more complex model than ours. However, while applying these steps, Overfitting and under fitting situations must be considered.