BOĞAZİÇİ UNIVERSITY

CMPE 462

Project 1

BetaZero

Baran Deniz KORKMAZ - 2015400183

Doğukan KALKAN - 2015400132

2020 SPRING

Contents

1 Introduction

In the project we are asked to classifying handwritten digits by implementing logistic regression from scratch. Our target classes are digit 1 and 5 labeled by +1, and -1 respectively. The flow of project has been designed in a way that step by step we are going to introduce logistic regression related concepts, and utilize these concepts properly in our project.

Our aim is to analyze the data comprehensively, to obtain a well-designed implementation that combines related concepts, and finally to evaluate our models. In the project, we use MNIST data set which provides a very beneficial data for handwritten digit recognition applications. Gathering the target classes (digit 1 and 5) from the MNIST, we obtain training data of 1521 images and test data of 424 images, where each image consists of 256 flattened pixel values. We later will converted the flattened pixels into 16x16 matrices for the purposes of visualization and feature extraction accordingly.

In this report, we are not just aiming to present our work into our instructors, but also to provide a more formally written text that will enlighten as possible those who are interested in handwritten digit recognition.

2 General Work

2.1 Task1: Feature Extraction

At the very center of machine learning, there lies the data. Any machine learning application obtains power by comprehension of the data. Anyone would probably say that the computers can think in terms of data. This derives from the fact that the data brings an insight about the subject we are interested in. This section describes the way our data presents its wisdom into our logistic regression algorithm.

Gaining the information that lies within the data requires a process that is called the feature extraction. This process comprises the artistic character of our work. For each image within the data set, we try to deduce an attribute that describes its characteristics as well as possible. The better we understand the data, the better we can find key attributes to split it into clusters, therefore an improve in the accuracy of logistic regression will follow.

2.1.1 Displaying Images

The very first key step of feature extraction is to understand the general structure of images in the data set. Analyzing the images by visualizing on grid and printing their content as a matrix will conduce our ability to understand structure of image examples in the provided data set.

Therefore displaying some image examples both in terms of visualization and printing occupies our first step. A comprehensive analysis of images will be constituting the center of our art to perceive a key attribute that will lead us into a satisfactory accuracy. Our aim was to obtain our best, to do so we

were aware that we should have discussed on many examples. Below, there is just one example for each digit is visualized for your observation. However, as stated, we have analyzed many digits within our data set.

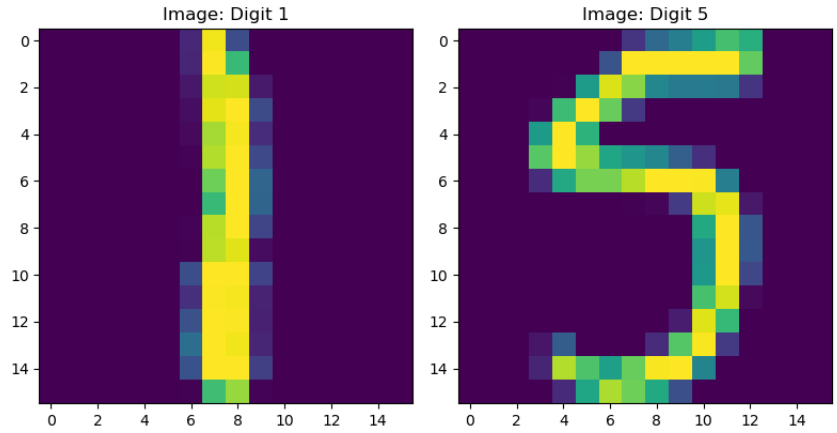


Figure 1: Digits

2.1.2 Implementing Representation 1

The project consists of exactly two feature extraction processes. In the first one,the features extracted are already determined by the instructor to implement. In the second process, the assignees are free to implement their own way of extraction to provide the training data that will later be used in logistic regression algorithm. These final forms are called Representation 1 and Representation 2, respectively.

Representation 1 will be formed based on two attributes of images in the data set. These attributes are defined and computed as follows:

1. Intensity: It is the average pixel value of images, which are 16x16 matrices actually. Intensity is computed simply by taking the average value of pixels which construct the 16x16 image matrix.

2. Symmetry about y-Axis: It is the negative of the norm of the difference between the image and its y-axis symmetry. Using NumPy package, we can easily convert a 2D matrix into its symmetrical about y-axis form **(See: numpy.flip)**. After taking difference of original and symmetrical form, we can calculate the norm of result by using "linalg" module inside NumPy package **(See: linalg.norm)**.

The general-purpose array-processing package NumPy, and the availability of functional programming approach for creation of lists are utilized for this task. In the end, we obtained a training data matrix of size 1561x2 and test data matrix of size 424x2 after feature extraction for Representation 1, whereby

the columns of matrices denote the intensity and the symmetry about y-axis values

2.1.3 Visualization of Representation 1

The final step of feature extraction is to visualize the clusters that you have obtained by the attributes used to form the training and test data. The visualization provides a structured perception about how accurately the attributes separate the data points according to the class labels. Thanks to Matplotlib package, we can obtain enhanced plots for visualization of clusters. We can analyze how well same classes of groups are separated in a way that they can form clusters separate from the rest. Scatter plots enable construction of a new feature extraction technique to obtain a better separation, given that the result is not satisfactory.

We should note that our scatter plot will be in a 2-dimensional region, since we have 2 attributes extracted for Representation 1. Below, you will be able to see the scatter plot for training and test data points for Representation 1, whereby the blue markers shaped 'o' denote the data points of label 1 (digit 1), and the red markers shaped 'x' denote the data points of label -1 (digit 5) respectively. You can check the legend provided as well.

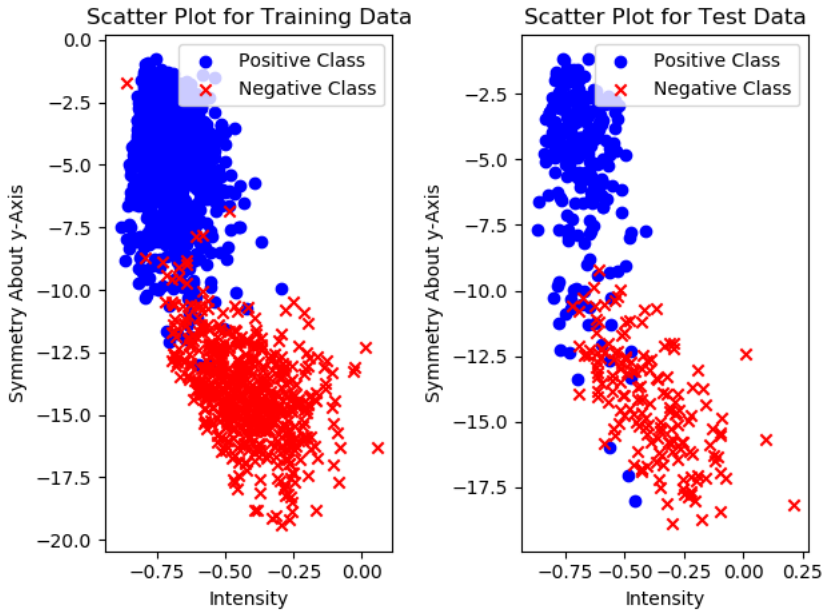


Figure 2: Scatter Plots for Representation 1

Regarding the scatter plots, we observe that there exists some outliers that are not linearly separable for both training and test data set in terms of inten-

sity and symmetry about y-axis features. This situation will give rise to some marginal error in accuracy, as we will be observing in **Section 2.3**.

Finally, we have analyzed the correlation between two attributes used to construct training and test data for Representation 1. The correlation is an important aspect of data points, since we do not want to train our logistic regression model with linearly dependent attributes for diversifying our classification attributes as possible. You can see our work related to correlation analysis of Representation 1 in **Appendix A.1**.

2.1.4 Implementing Representation 2 & Visualization

Implementation of Representation 2 is the stage where the creativity and endeavour of assignees play a key role at most. At this stage, we try to figure out new attributes that describes the images within the data as well as possible. As mentioned earlier, extraction of new features require a well analysis of data. Therefore we have observed some examples for the values within the image matrices, which will lead us into finding a way of perceiving some differentiating the images of digit 1 and 5 from each other.

Regarding the logistic regression algorithm, the strategy for extraction of new features must satisfy the requirements for the description of training and test data. In other words, our aim was to extract as many features that will describe training data without causing any over-fitting trouble, that is, in compatible with the testing data as well. Therefore, we decided to extract 3 new features, since we thought that it will increase the probability of success, decrease the probability of over-fitting, considering much higher dimensional attributes, and also will enable a good visualization of 3D-scatter plotting for our observations. We should note that we have also applied a 2D model, however, then we realized that 3D model does not yield any over-fitting problem and explains data better.

After we have completed our strategy, the attributes for digit recognition was waiting for extraction. Observations in the images led us into using symmetry about x-axis, in contrary to symmetry about y-axis we used in Representation 1. We figured out some specific differences between the matrices for 1s and 5s, and tried to utilize them as possible as we can, by combining these differences into the statistical parameters, such as mean and standard deviation. Our approach was to fit the images as much as possible by deriving a pattern bright pixels and dark pixels follow. Our though was to force clusters into certain patterns by utilizing the differences between the locations of bright and dark pixels. This way we can also derive an implied implementation for the curvature aspect of digit 5, which lies in its nature in contrary to digit 1. Since we have introduced our approach for deriving new features, it is time to explicitly give the definitions. Below, you will see the definitions of new attributes that we have extracted for Representation 2.

1. Symmetry About x-Axis: It is the negative of the norm of the difference between the image and its x-axis symmetry. Using NumPy package, we

can easily convert a 2D matrix into its symmetrical about x-axis form **(See: numpy.flip)**. After taking difference of original and symmetrical form, we can calculate the norm of result by using "linalg" module inside NumPy package **(See: linalg.norm)**.

2. Brightness: The title is just the name we have used during project. Let me introduce the logic behind our attribute, and its power. After long observations for 16x16 data matrices for an image, we figured out that the indices that the bright pixels that lie within a row vary more for digit 5 rather than digit 1. This makes sense considering the nature of their shape. Our strategy was imagining as if we were drawing both the digits into a clean paper and to perceive the main differences in their characteristics. This very first attribute, our strategy has brought us, work as follows. We simply determine a threshold value of 0.8 for labeling a pixel as bright. Then in each row, we take the average value of bright indices. Finally, for the entire image, we compute the standard deviation of mean values for 16 rows. The reason why we have chosen this approach was to imitate a drawing pencil as possible as we could.

3. Darkness: Our approach for the last attribute is very similar to that of the second one which we called 'Brightness'. This final attribute is quiet much like the previous one, which could be considered as the other side of the coin. While the brightness attribute analyzes the points that the pencil moves towards, this feature aims at analyzing the empty spaces. Combined with the second feature, we aim at fitting our digits into a followed pattern. Computation of the final attribute is as follows. Just as we did for attribute 2, for each row we set a threshold value of -0.7 for labeling a pixel as dark. Then for each row, we analyze the consecutive dark pixels, simply take the first and last consecutive dark space, and compute the absolute value of their differences in lengths. Finally, for the entire image, we compute the standard deviation of values found for each row. The reason why we have chose this approach was to describe the curvature property of digit 5 as possible as we could. Plus, we can say that digit 1 does not vary a lot in terms of the differences between the dark spaces of left and right side of bright pixels.

The general-purpose array-processing package NumPy, and the availability of functional programming approach for creation of lists are utilized for this task as well. In the end, we obtained a training data matrix of size 1561x3 and test data matrix of size 424x3 after feature extraction for Representation 2, whereby the columns of matrices denote the symmetry about x-axis and the brightness, and the darkness values.

Regarding Representation 2, we must follow a 3D-visualization considering that this time we have 3 attributes to form training and test data. **Matplotlib** package provides an easy way of implementing a 3D scatter plot that you can observe below for Representation 2.

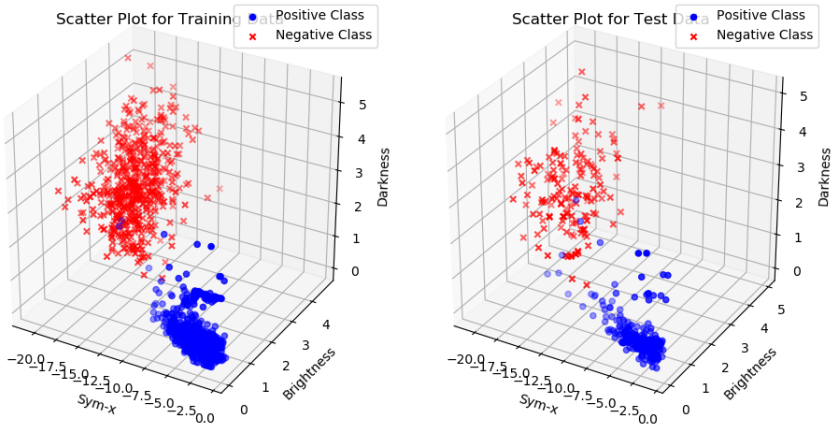


Figure 3: Scatter Plots for Representation 1

As we carried out for Representation 1, we have analyzed the correlation between the features that constitute Representation 2 respectively again. You can see our work related to correlation analysis of Representation 2 in **Appendix A.2**.

We must note that the thresholds values for 'Brightness' and 'Darkness' features are adjusted by subsequent experimental observations based on correlation analysis provided in **Appendix A.2**. Adjusting correlation values into larger absolute values is likely to yield better accuracy. We especially cared about increasing absolute correlation values between features for testing data as possible as we can. We sequentially tried to increase the correlation magnitude and evaluated the accuracy levels respectively. The final version is what we have found as optimal.

Finally, let us make a brief review for this section by summarizing the general procedure that we have followed for Feature Extraction:

1. Getting Familiar with Data: Observation of Images and Data Matrices
2. Extraction: Candidate Features
3. Analysis: Correlation and Scatter Plots
4. Acceptance

2.2 Task2: Logistic Regression

Logistic regression is a widely used machine learning application that is used to model the probability of a certain class. Binary classifier can be extended to a multivariate version by subsequently occupying binary clusters.

2.2.1 Logistic Regression

In the project, we are asked to build a model for a binary class of value +1 and -1 where labels denote digits 1 and 5 respectively.

Modularity and reusability are among key characteristics of any written code in modern era. Therefore, the assignees are decided to carry out implementation by dividing main computation into sub-problems. We have defined functions that carry out auxiliary calculations for overall logistic regression algorithm.

Let us present the logistic regression algorithm utilized in the project. Then we can introduce auxiliary calculations and how they are implemented. Then we will discuss more on our approach.

Logistic Regression Algorithm
Reference: Learning From Data, Yaser S. Abu-Mostafa, Malik Magdon-Ismail, Hsuan-Tien Lin

1. Initialize the weights at time step t=0 to w(0). (Random Initialization by Normal Distribution is applied.)
2. **for** t=0,1,2,... **do**
3. Compute the gradient $g_t = \nabla E_{in}(w(t))$. (The gradient computation will be discussed later.)
4. Set the direction to move $v_t = -g_t$
5. Update the weights: w(t+1) = w(t) + ηv_t
6. Iterate to the next step until it is time to stop. (Until the absolute difference between current loss value and previous loss value is less than $\epsilon = 10^4$)
7. Return the final weights.

The auxiliary computations handle two operations required for logistic regression algorithm.

1. **In-Sample Error**

   Loss value constitutes the core of logistic regression algorithm. Its characteristic that is defined by sigmoid ($\theta(s)$) function enables providing binary classification models. In logistic regression, in-sample error measure is defined as

$$E_{in}(w) = \frac{1}{N}\sum_{i=1}^{N} ln(1 + e^{-y_i w^T x_i})) \tag{1}$$

This equation becomes a smart choice when you realize that this error measure is small when $y_i w^T x_i$ is large and positive, which implies $sign(w^T x_i) = y_i$

2. **Gradient**

   Now that we have defined out in-sample error function, let us introduce the computation of gradient with respect to w. The computation is formulated step by step as follows:

$$\nabla E(w) = \frac{1}{N}\sum_{i=1}^{N} \frac{-y_i x_i \cdot e^{-y_i w^T x_i}}{1 + e^{-y_i w^T x_i}}, \tag{2}$$

$$\nabla E(w) = -\frac{1}{N}\sum_{i=1}^{N} \frac{y_i x_i}{1 + e^{y_i w^T x_i}}, \tag{3}$$

$$\nabla E(w) = \frac{1}{N}\sum_{i=1}^{N} -y_i x_i \cdot \theta(-y_i w^T x_i) \tag{4}$$

   * Multiply and divide the division by $e^{y_i w^T x_i}$ to obtain (2) from (1).

   The gradient computation will yield a column matrix of same dimensionality with w, which will be multiplied by learning rate to update the model weights.

Considering that the general procedure and required computations are explained above in detail, we can move further our discussion into our approach which will determine how we will observe the logistic regression algorithm, and its outcomes. In the project, we are already asked to plot the loss values with respect to iteration count **See Matplotlib: plt.plot**. By observing their behavior about convergence, we anticipate to be led into consistent learning rate parameters. For this purpose, we have carried out experiments with 5 different learning rates between 0 and 1. Below you can see the convergence curves for each learning rate in the same figure for Representation 1 and Representation 2 accordingly. Note that the learning rates used are 0.001, 0.005, 0.01, 0.05, and 0.12 respectively.

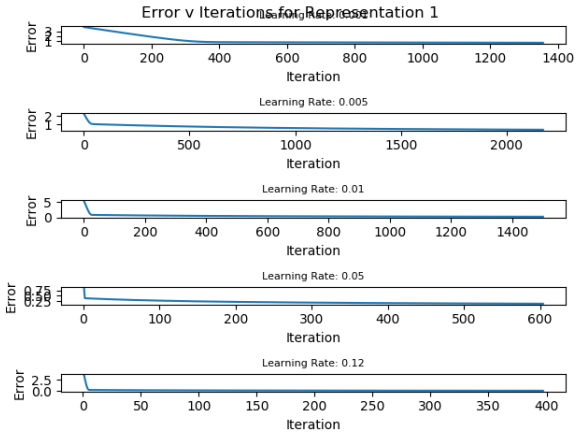


Figure 4: In-Sample Error v Iterations: Representation 1

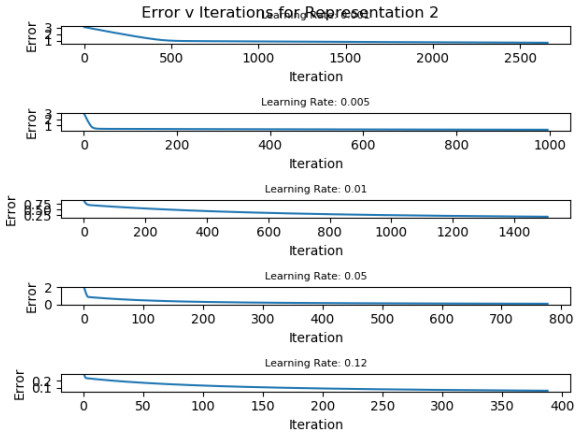


Figure 5: In-Sample Error v Iterations: Representation 2

As obviously seen by the figures, the convergence is satisfied by the given learning rates. However, conducting further experiments and observations will yield a better insight into the logistic regression algorithm. Therefore, we have aimed at trying other applications that we might follow to improve our knowledge about the algorithm. The more the analysis, the more knowledge we obtain. For this purpose, we have analyzed the in-sample error value versus weight coefficients and accuracy evaluations. The first one aims at observing the behavior of loss value as the weight coefficients change. We aim at investigating whether it oscillates wildly and observing the final values of weights to gain a first impression about whether over-fitting might become an issue. Two instances of both representations are left for providing a first impression. You

can see the complete version of related work in **Appendix B.1**.

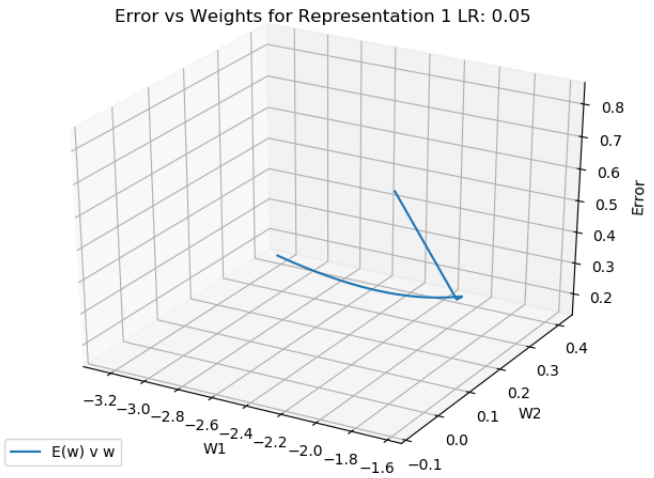


Figure 6: Error vs Weights: Representation 1 (Learning Rate = 0.05)

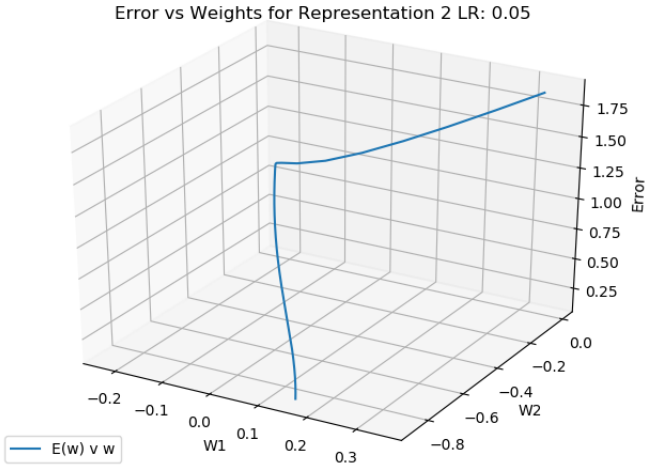


Figure 7: Error vs Weights: Representation 2 (Learning Rate = 0.05)

The accuracy analysis is actually related to the evaluation section of project. Therefore after the required implementations for evaluation (**(See 2.3)**) have been completed, we have thought that we may carry out an accuracy analysis for each learning rate to make sure that they provide satisfactory accuracy results. This way, we can conclude that we can continue using a reliable subset

of learning rates that will give rise to satisfactory accuracy results. You can see the related work in **Appendix B.2**.

2.2.2 Regularization

In machine learning, regularization is concept of adding a penalty-term into the in-sample error function in order to prevent large magnitude of model weights. Generally, over-fitting occurs whenever the magnitudes become larger, implying a well-formed weights for training data, but not a generalized solution for test data.

The regularized in-sample error function becomes as below, to discourage larger magnitude of coefficients in model weights:

$$E_{in}(w) = \frac{1}{N}\sum_{i=1}^{N} ln(1 + e^{-y_i w^T x_i})) + \frac{\lambda}{2} \cdot ||w||_2^2 \tag{5}$$

The gradient of error function becomes:

$$\nabla E(w) = \frac{1}{N}\sum_{i=1}^{N} -y_i x_i \cdot \theta(-y_i w^T x_i) + \lambda \cdot w \tag{6}$$

The implementation of regularization can easily be done by adding additional terms into the error and gradient computations. Notice that the standard logistic regression in-sample error function and gradient function are simply the regularized forms where $\lambda = 0$.

After the implementation is provided, just as we did in **2.2.1 Logistic Regression**, again it is time to analyze the behavior of implementation. The tuned parameter was the learning rate before. This time by analyzing the behavior of algorithm, we try to find a subset of regularization coefficients that we can move forward. Plus, we must ensure that our algorithm converges for certain lambda values as expected, since we are going to implement **Cross Validation** in next section to determine the optimal lambda coefficient.

By observing their behavior about convergence, we anticipate to be led into consistent lambda parameters and the observation of properly implemented regularization. For this purpose, we have carried out experiments with 5 different regularization coefficients by a constant learning rate of 0.05 which we observed to be most consistent in the previous section and in our subsequent observations in the following sections. Below you can see the convergence curves for lambda values in the same figure for Representation 1 and Representation 2 accordingly. Note that the lambda values used are 0 (no regularization implied), 0.5, 0.05, 0.005, and 0.0005 respectively.

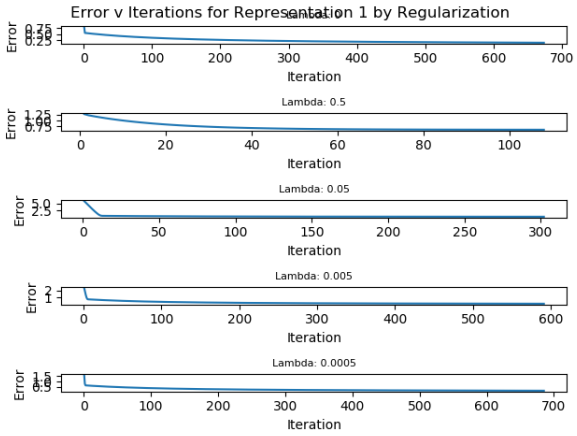


Figure 8: In-Sample Error v Iterations: Representation 1 by Regularization

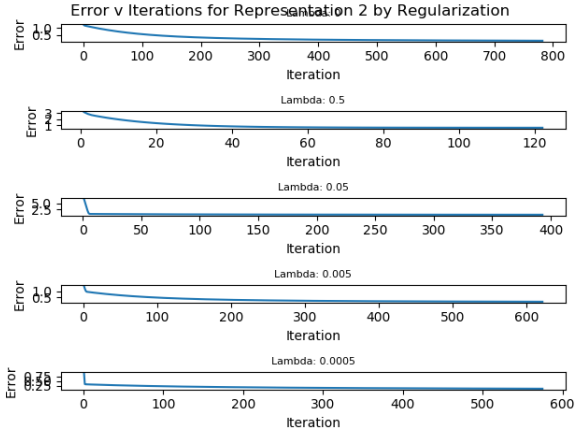


Figure 9: In-Sample Error v Iterations: Representation 2 by Regularization

As obviously seen by the figures, the convergence is satisfied by the given lambda coefficients. However, conducting further experiments and observations will yield a better insight into the regularization algorithm. Therefore, we have aimed at trying other applications that we might follow to improve our knowledge about the algorithm.

For this purpose, we have analyzed the in-sample error value versus weight coefficients and accuracy evaluations. The first one aims at observing the behavior of loss value as the weight coefficients change, considering that we aim at obtaining smaller magnitudes in vector of weights. Two instances of both representations are left for providing a first impression. You can see the complete version of related work in **Appendix B.3**.

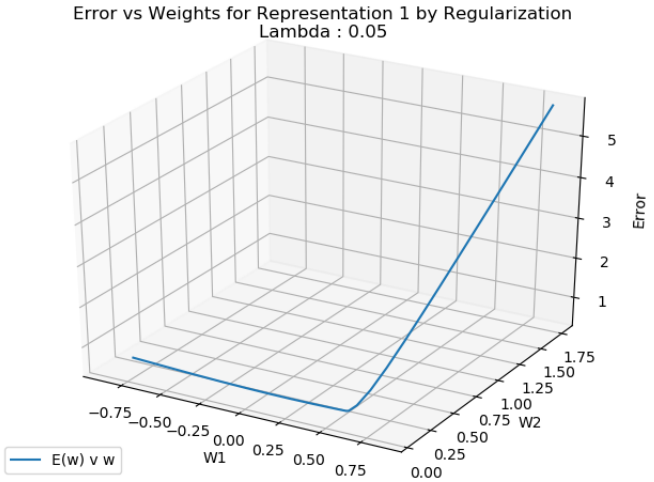


Figure 10: Error vs Weights: Representation 1 by Regularization (Lambda = 0.05)

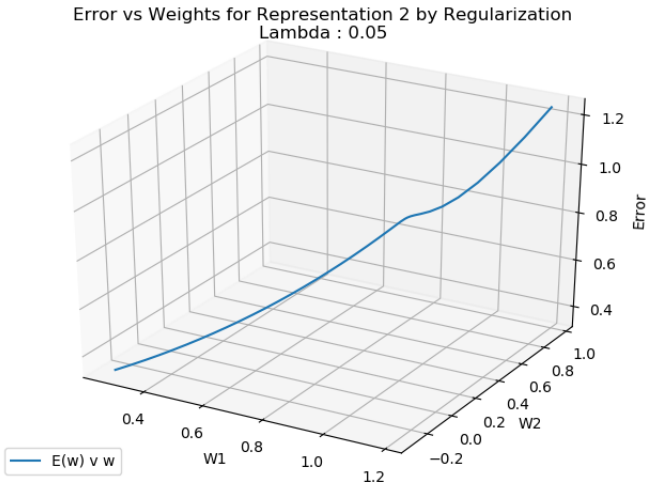


Figure 11: Error vs Weights: Representation 2 by Regularization (Lambda = 0.05)

The latter is actually related to the evaluation section of project. Therefore after the required implementations for evaluation (**(See 2.3)**) have been completed, we have thought that we may carry out an accuracy analysis for each lambda coefficient to make sure that they provide satisfactory accuracy results. This way, we wanted to make our first observations about the effects of

regularization on our learning algorithm, before moving forward into the **Cross Validation**. You can see the related work in **Appendix B.4**.

2.2.3 Cross Validation

As there is never enough data at our disposal to train a model, we always have to remove a considerable amount of the data for validation, in other words testing.

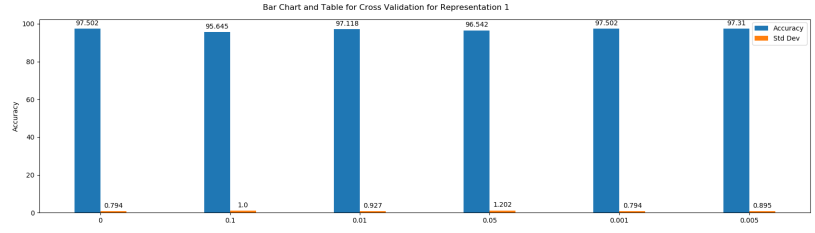
In k-fold cross validation, the training data set is divided into k subsets. At each iteration, one of them is selected for validation purpose, while the rest is used to train the model. The goal of this process is to inspect the average accuracy and the average standard deviation calculated from k different models. By performing this, we are able to gain detailed information about our model and possible outcomes.

In this project, we are asked to implement a 5-fold cross validation procedure to find the optimal λ value for our model considering the average accuracy and the average standard deviation corresponding to each λ value. Below, there will be an explanation of our implementation of 5-fold cross validation.

Our implementation of cross validation consists of two modes. In one mode, one can call the function with a given list of lambda values, and in the other mode, one can call the function in an autonomous way in which the function finds the optimal lambda value as accurately as possible. You can see the complete explanation in **Appendix C.1**.

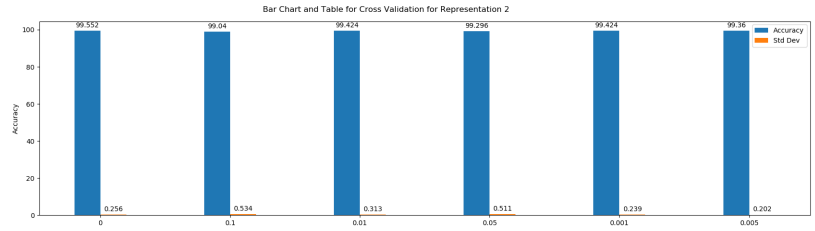
The first method to call the function, as stated above, takes a list of lambda values and returns the optimal lambda value results in the highest accuracy.

The optimal λ value and corresponding accuracy and standard deviation for representation 1 and representation 2 are shown below.



Lambda	0	0.1	0.01	0.05	0.001	0.005
Accuracy	97.502	95.645	97.118	96.542	97.502	97.31
Std Dev	0.794	1.0	0.927	1.202	0.794	0.895

Figure 12: 5-fold Cross Validation: Representation 1



Lambda	0	0.1	0.01	0.05	0.001	0.005
Accuracy	99.552	99.04	99.424	99.296	99.424	99.36
Std Dev	0.256	0.534	0.313	0.511	0.239	0.202

Figure 13: 5-fold Cross Validation: Representation 2

2.3 Task3: Evaluation

The ultimate goal of each model in machine learning is to predict the possible outcomes depending on the given data. Since we want the models to predict the outcomes with a high accuracy, it is very crucial to compare the results obtained through our model and the real data that is given to us. In rather more important models in which the accuracy of results is important, the evaluation becomes much more important because certain problems cannot tolerate high amounts of error.

In our project, it is also important since we want our model to distinguish the digits, 1 and 5, and this part of our project determines the level of success of our model.

2.3.1 Accuracy Analysis

As stated above, the accuracy of our model is very crucial in order to decide whether our model is powerful enough to give the correct result with provided data. The procedure is quite simple. We already have the correct labels of each digit and by performing logistic regression on our model, we obtain our model's predictions for each digit. Thus, the number of correctly classified digits determines our model's accuracy. The provided formula is also shown below.

$$\frac{\text{number of correctly classified samples}}{\text{total number of samples}} \times 100$$

It is necessary to point out that it is quite useful to train our model with different learning rates because the learning rate has an immediate impact on the accuracy and the number of iterations for the accuracy to converge and for

this reason, there are several accuracy result performed with different learning rates. Note that, when the learning rate is relatively small, the accuracy is low since it is more difficult to converge.

Below, there are bar charts visualizing accuracy of training and test data for both representation 1 and representation 2, where the optimal lambda derived from the cross validation process is used.

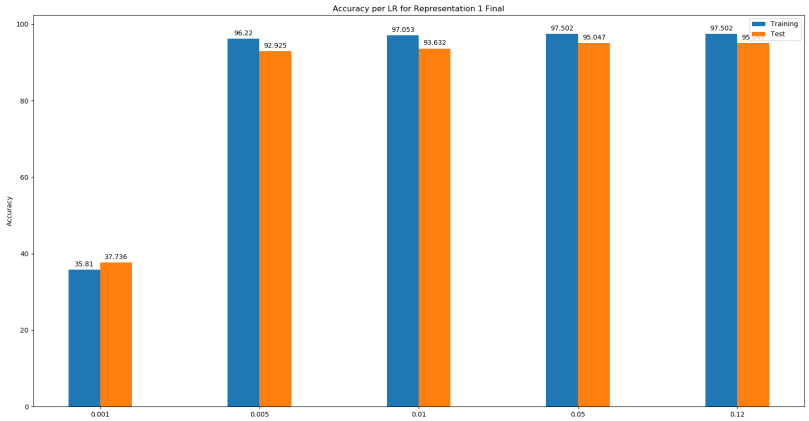


Figure 14: Accuracy Results with different learning rates: Representation 1 ($\lambda = 0$)

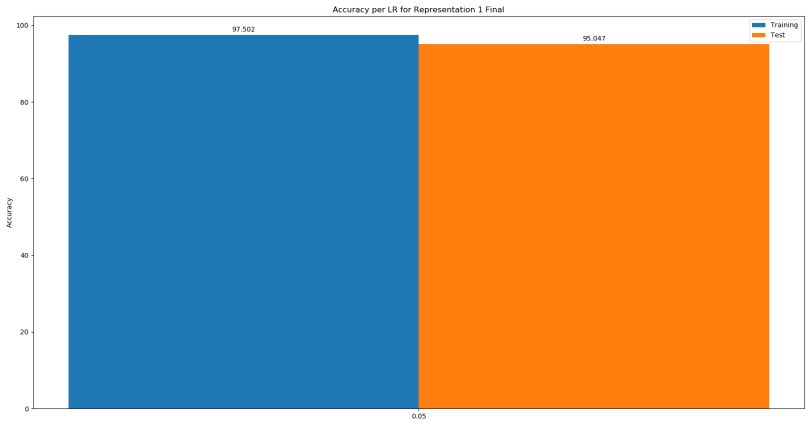


Figure 15: Accuracy Results with the optimal learning rate: Representation 1 ($\lambda = 0$)

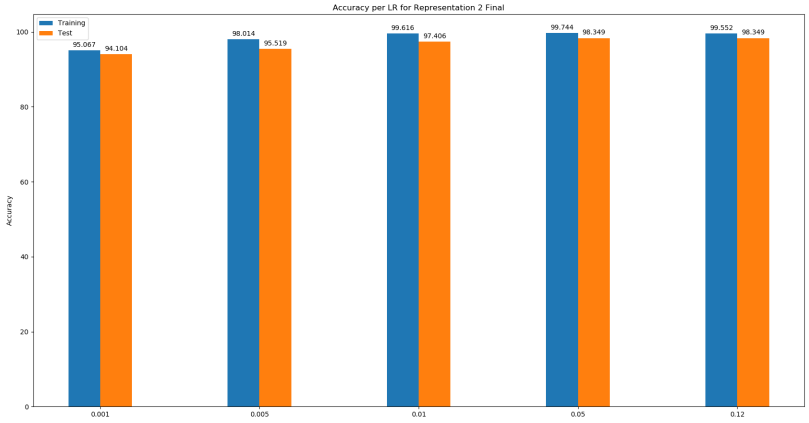


Figure 16: Accuracy Results with different learning rates: Representation 2 ($\lambda = 0$)

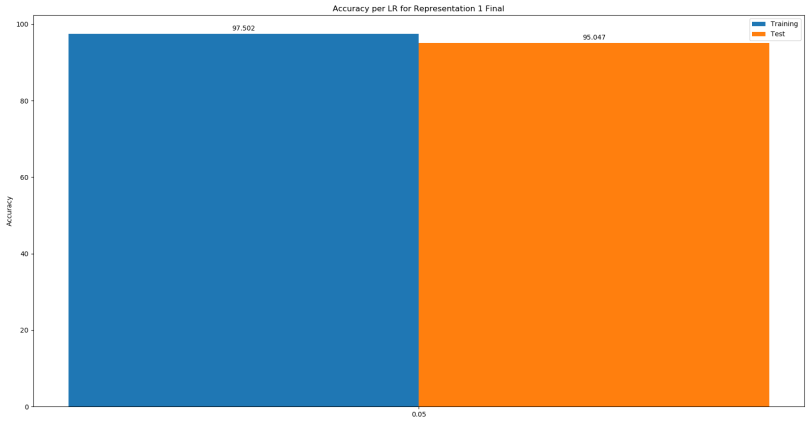


Figure 17: Accuracy Results with the optimal learning rate: Representation 2 ($\lambda = 0$)

As can be seen above in the charts, our training accuracy results are between 96.22% and 97.502% and test accuracy results are between 92.925% and 95.047% for Representation 1. If we select the learning rate as 0.12 which is not too high, we can clearly see that our training and test accuracies are 97.502% and 95.047%, respectively.

Below, there are four different bar charts that we have observed in our observations as final accuracy for Representation 2.

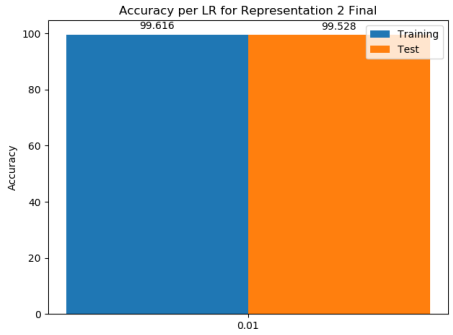


Figure 18: Accuracy Results with Learning Rate = 0.01: Representation 2 (λ = 0)

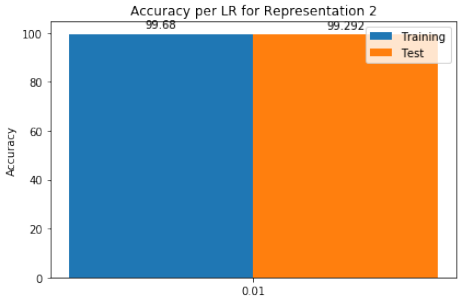


Figure 19: Accuracy Results with Learning Rate = 0.01: Representation 2 (λ = 0)

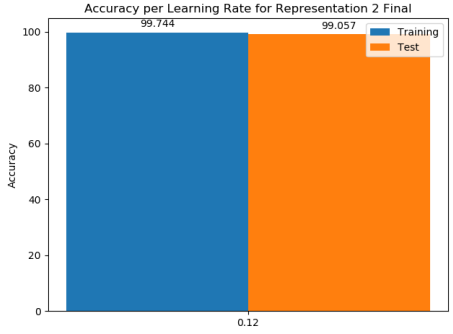


Figure 20: Accuracy Results with Learning Rate = 0.12: Representation 2 (λ = 0)

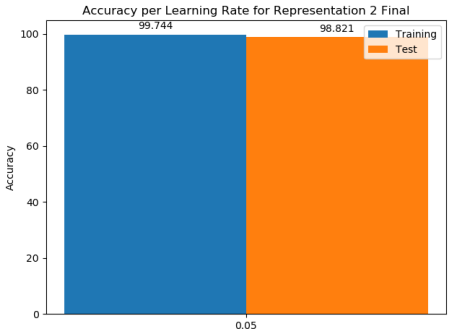


Figure 21: Accuracy Results with Learning Rate = 0.05: Representation 2 (λ = 0)

In Figure 16, it can be observed that our training accuracy results are between 95.067% and 99.552% and test accuracy results are between 94.104% and 98.349%.

However, if we look at Figures 18,19,20,21 we can see that our training and test accuracy are much higher than the statistics above. This difference is more significant in the test accuracy results.

During our analysis, we observed that our training and test accuracy results are often fluctuated and there is a plausible reason behind it. Our algorithm starts the logistic regression procedure with randomly determined weights and in different executions, this randomness affects the initial weights as well as the optimal weights. In other words, in each execution of the program, we end up with different optimal weights, hence different accuracy results.

Specifically, if we decide to pick the best learning rate in **Task 2.1**, in some rare cases, some learning rates giving the best accuracy first might result in worse accuracy values in the evaluation part. Therefore, we again carry out training by 5 different learning rates and optimal regularization coefficient in order to prevent emergence of such situations. We conclude that

We conclude that it can be seen in the Figures 18,19,20. our training accuracy goes up to 99.744% and our test accuracy goes up to 99.528%. In overall, we have observed that the test accuracy falls into the range of 98.3%-99.5%. The most frequent accuracy we have observed is 98.82%.

2.3.2 Visualization of Decision Boundary

In the final part of the project, we create a decision boundary, which is essentially a line that separates the two class. This line is given by the formula $w^T x = 0$. The line is acquired by multiplying each component of the weight matrix, w with components of x and then equating it to zero.

$$w^T x = \begin{bmatrix} w_0 & w_1 & w_2 \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} = w_0 + w_1 x_1 + w_2 x_2 = 0$$

And when we leave solve it for x_2, we get the following equation.

$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{w_0}{w_2}$$

Below, scatter plots of training data and test data for representation one are shown.

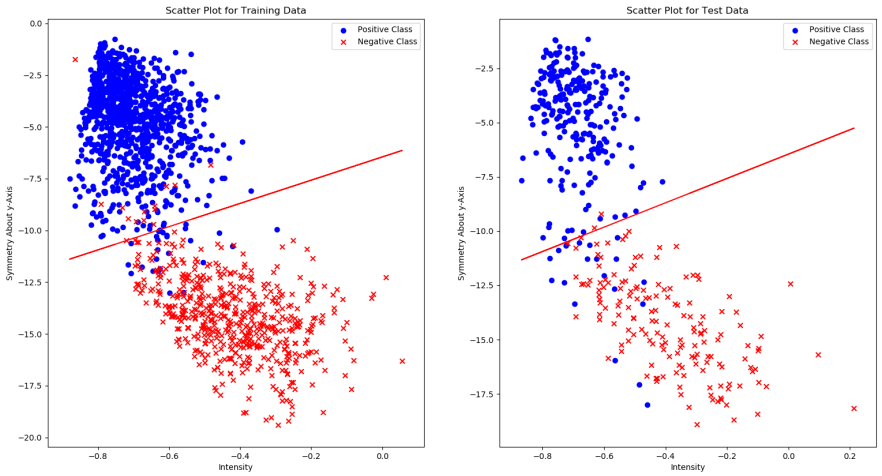


Figure 22: Decision boundary for Representation 1

2.3.3 Comments

The team BetaZero formed by enthusiastic computer engineering students who want to enhance their skills in machine learning as possible. This project was our first curricular assignment in CMPE462-Machine Learning. We aimed at discovering the conceptual related topics and applying as well as possible. We tried to not restrict us into the content of the project, but also to apply as many possible related conducive techniques, and analysis.

Considering our work, we believe that we tried to pursue the best as we have previously decided. We are glad that we present our best efforts.

The progress is always possible, so enhancement is. Below you can see our remarkable points on the projects:

1. Regularization has not resulted in a better accuracy in the project. Considering the size of the feature set, the outcome seems possible, since we continue our implementation by a small set of attributes. In both representations, we observed that the case where lambda is zero resulted in best accuracy levels in overall.

2. The feature set that we have derived, that is Representation 2, resulted in a better test accuracy of 98.8-99.0 in overall of total observations compared to Representation 1, which resulted in an overall test accuracy of 94.5-94.7 of total observations. We can conclude that Representation 2 presents better attributes that describes the overall data.

3. We could obtain a better test accuracy by implementing advanced algorithms for finding optimal learning rates and lambda coefficients instead of using a small subset of possible range as we did in the project. Plus, the features used in training of model can also be improved, which is much more likely to enhance the test accuracy in overall by utilizing more advanced techniques.

3 Conclusion

In this project, our task is to write a program that recognizes and separates two specified digits with a high accuracy. In order to achieve the tasks given to us, we utilized several methods to write our program in a modular and efficient way. We came up with different ideas to further improve our model and its success rate. The project was a suitable one as the first project, and it was a good opportunity to gain information and practise in Machine Learning.

4 Appendix

4.1 Appendix A

4.1.1 Correlation Analysis: Representation 1

In statistics, correlation or dependence is any statistical relationship between two random variables. Correlation analysis plays a key role in machine learning applications considering that the data enlightens our way. Our objective is to produce linearly independent attributes for yielding much information on the data, which will increase the accuracy.

In our project, we have observed the correlation between attributes in both representations. Below, you will observe the correlation between two attributes of Representation 1, which is predetermined by the instructors.

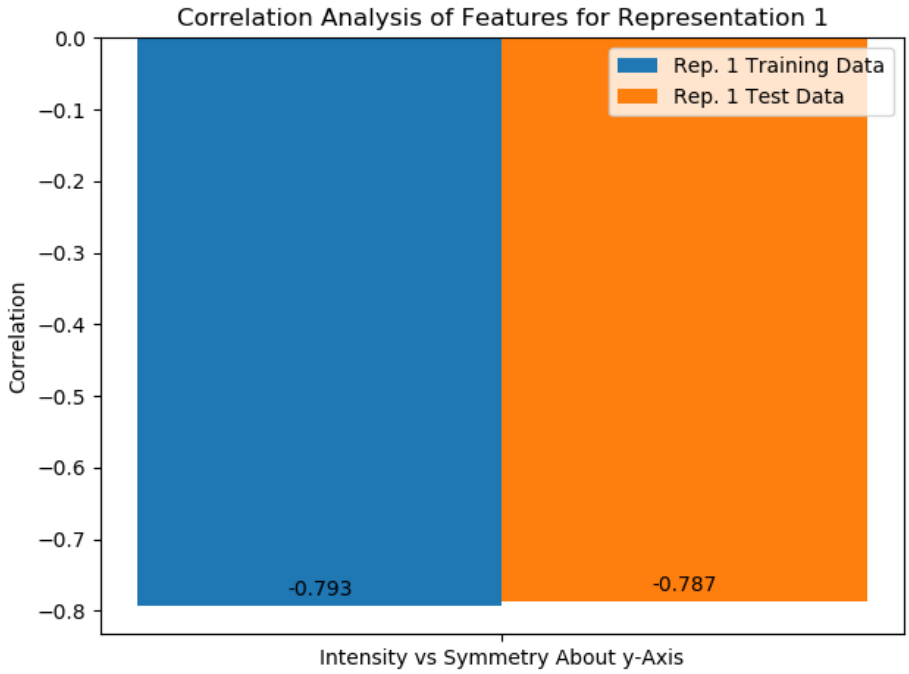


Figure 23: Correlation Analysis: Representation 1

4.1.2 Correlation Analysis: Representation 2

As mentioned earlier, feature extraction is the stage where we present our skills at most. The features we have obtained plays the key role for the accuracy. Therefore, in a machine learning application, the assignees must be aware that this procedure must be carried out carefully, since the real magic lies here.

In our project, we utilized correlation analysis for our objective. Attributes generated by predetermined threshold values are analyzed in terms of correlation which precedes the evaluation of model. We have conducted plenty of experimental subsequent observations by changing the threshold, then analyzing the correlation and evaluated accuracy. Finally, we have experimentally reached an optimal level by our observations as stated in **Section 2.1.4**.

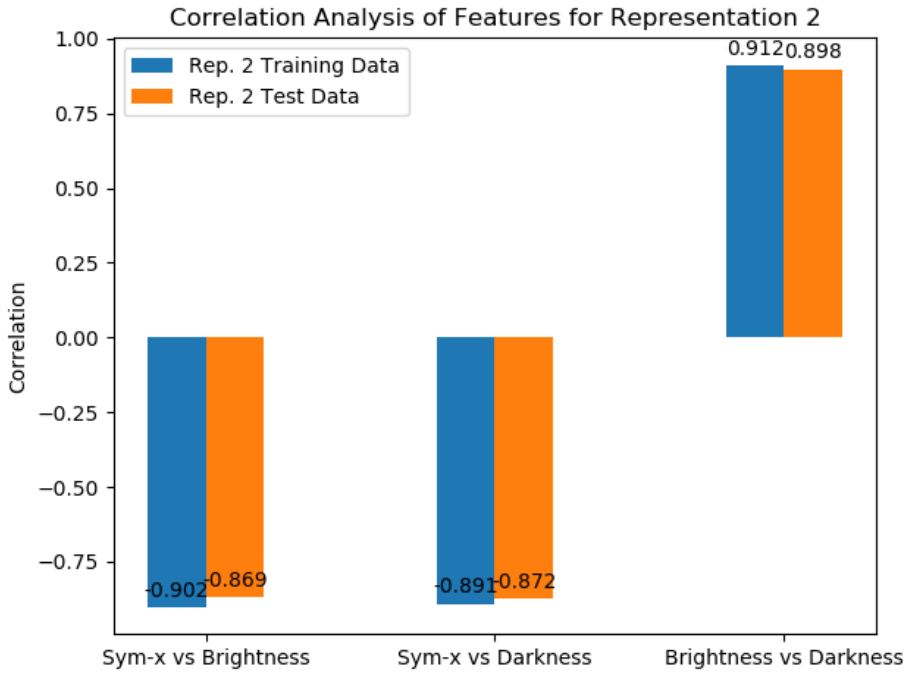


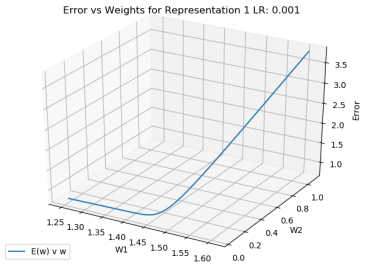
Figure 24: Correlation Analysis: Representation 2

4.2 Appendix B

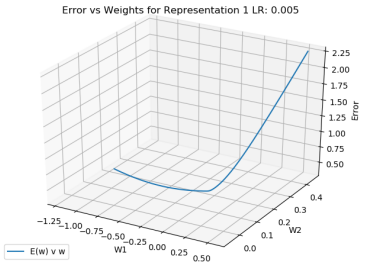
4.2.1 Visualization: Error vs Weights

The convergence of loss value during iterations was an aspect that we had to observe, since we would like to make sure that our algorithm does not diverge. However, another important issue lies within the behavior it presents while it converges. Our error value must not show a wild behavior while the model weights change. We must also consider the magnitude of vector of weights, because it is a good indicator for over-fitting.

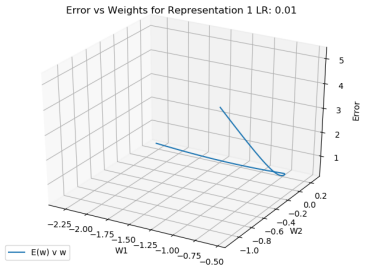
During the construction of model, the tuned parameters should be selected after a set of well-designed observations. finally, in our project, the predetermined learning rates will be tested by the observation of their effects in error vs weight plots. Below, you can see the plots for Representation 1 and 2 respectively.



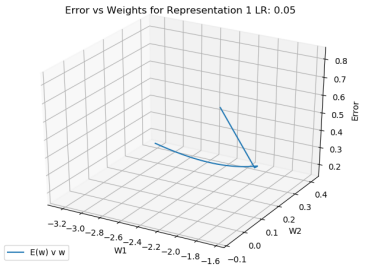
(a) Learning Rate = 0.001



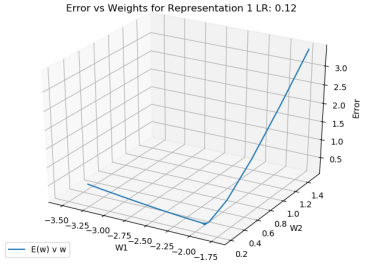
(b) Learning Rate = 0.005



(c) Learning Rate = 0.01

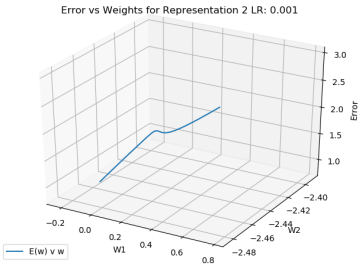


(d) Learning Rate = 0.05

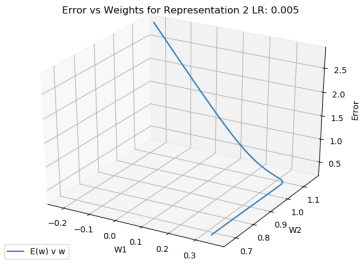


(e) Learning Rate = 0.12

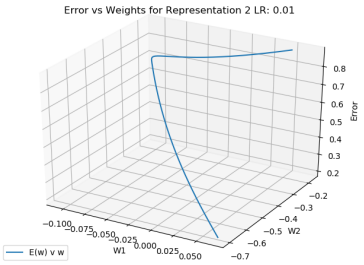
Figure 25: Error vs Weights: Representation 1



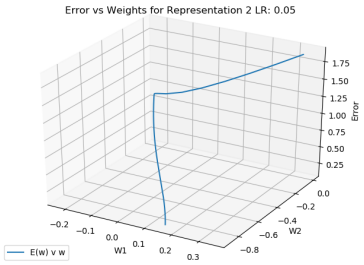
(a) Learning Rate = 0.001



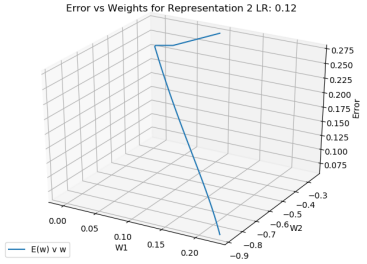
(b) Learning Rate = 0.005



(c) Learning Rate = 0.01



(d) Learning Rate = 0.05

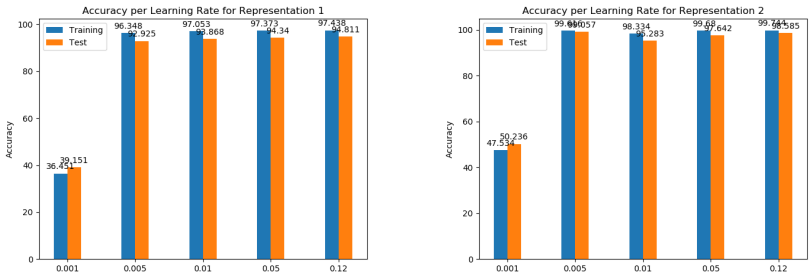


(e) Learning Rate = 0.12

Figure 26: Error vs Weights: Representation 2

4.2.2 Accuracy Analysis

As stated previously, after the implementation of evaluation **Section 2.3** is completed, we have decided to measure the accuracy of observed learning rates as a precaution. After the accuracy analysis has been conducted, there remains no reason that keeps us from not moving forward. Below, you will see the accuracy analysis as the final step for the decision of learning rates for Representation 1 and 2 respectively.

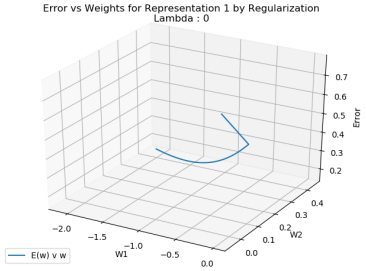


(a) Representation 1 (b) Representation 2

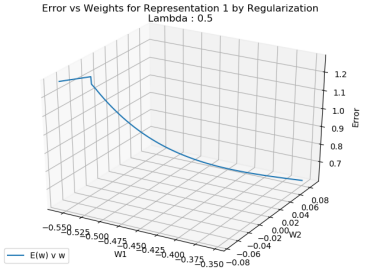
Figure 27: Accuracy Analysis for Representation 1 and Representation 2

4.2.3 Visualization: Error vs Weights by Regularization

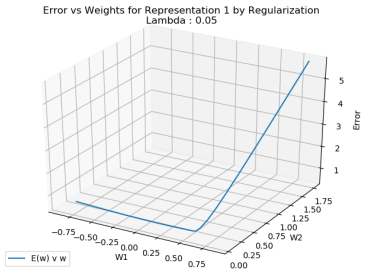
This section is the version for regularization what we did previously in **Appendix B.1**. Observe that the axis ranges become smaller, since we aim at obtaining a smaller magnitude of vector for model weights by regularization. Below you can see the plots for Representation 1 and 2 by regularization.



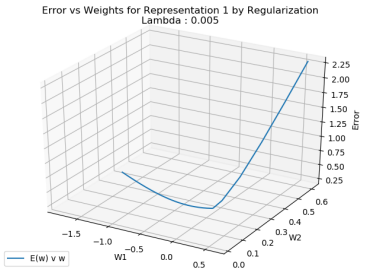
(a) Lambda = 0



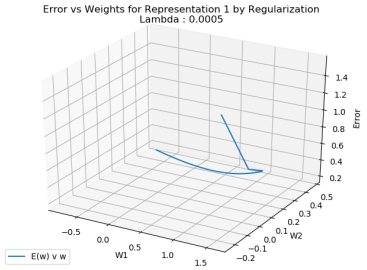
(b) Lambda = 0.5



(c) Lambda = 0.05

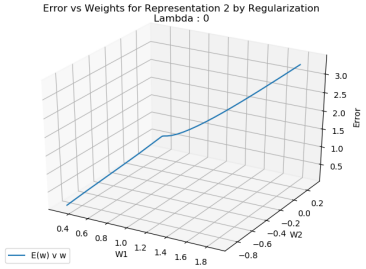


(d) Lambda = 0.005

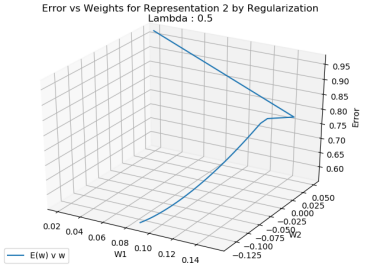


(e) Lambda = 0.0005

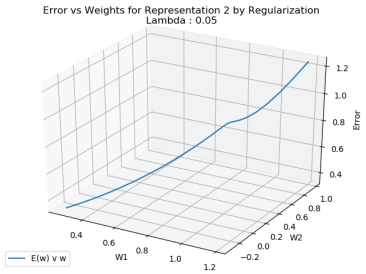
Figure 28: Error vs Weights: Representation 1 by Regularization



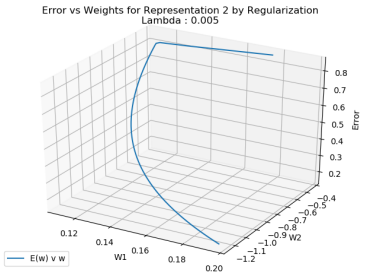
(a) Lambda = 0



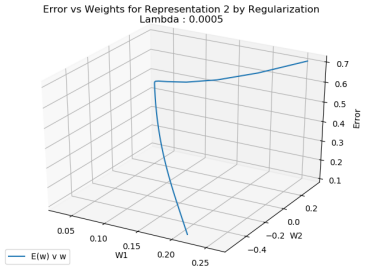
(b) Lambda = 0.5



(c) Lambda = 0.05



(d) Lambda = 0.005

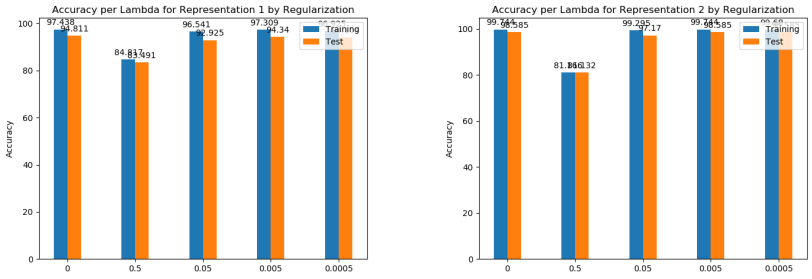


(e) Lambda = 0.0005

Figure 29: Error vs Weights: Representation 2 by Regularization

4.2.4 Accuracy Analysis by Regularization

As stated previously, after the implementation of evaluation **Section 2.3** is completed, we have decided to measure the accuracy of observed lambda coefficients as a precaution before moving forward into the **Cross Validation**. Gaining an early insight is always beneficial. Below, you will see the accuracy analysis as the final step for the decision of the stability of regularization implementation with different lambda coefficients for Representation 1 and 2 respectively.



(a) Representation 1 by Regularization (b) Representation 2 by Regularization

Figure 30: Accuracy Analysis for Representation 1 and Representation 2 by Regularization

4.2.5 Cross Validation

Since we aim to find the optimal λ value with which our model gives us the optimal accuracy, the general idea is to examine candidate λ values within a specified range at a time. The procedure works as follows:

1. Initialize the λ values within a predetermined range.
2. Execute the cross validation function for each *lambda* value
3. Select the λ value that gives the highest accuracy as the mean of the new range.
4. If the new accuracy is not higher than the currently optimal accuracy or there is not enough improvement on the accuracy, then terminate it
5. If not, create a new set of λ values within the new range, and continue from step 2.

In other words, the algorithm executes starting with a predetermined set of λ values, examining accuracy result corresponding to each λ value, creating a new set of λ values by narrowing down the range and repeating the procedure unless there is not enough improvement or there is no decrease in the accuracy.

5 References

- Learning from Data: A Short Course, Hsuan-Tien Lin, Malik Magdon-Ismail, and Yaser Abu-Mostafa
- https://en.wikipedia.org/wiki/Gradient_descent
- **Source code for a utilized function regards to plotting**