Project-4

Machine Learning

Spring 2018

Submitted to

Dr. Hamed Sari-Sarraf

Submitted by

Sabbir Hassan

Taslim Anupom

Shamsul Arefeen

April 3, 2018

**Simple Linear Logistic Regression model on 2D synthetic dataset:**

As the first step in the project implementation, a synthetic data-set was created with two classes and the logistic regression approach was implemented in MATLAB. The logistic regression approach is initially implemented with the linear model of logistic regression using the sigmoid function which could validate the practical implementation of the fundamental logistic regression theory by providing the classification at an apparently acceptable accuracy.

Train Accuracy: 97.000000

Elapsed time is 0.800431 seconds.

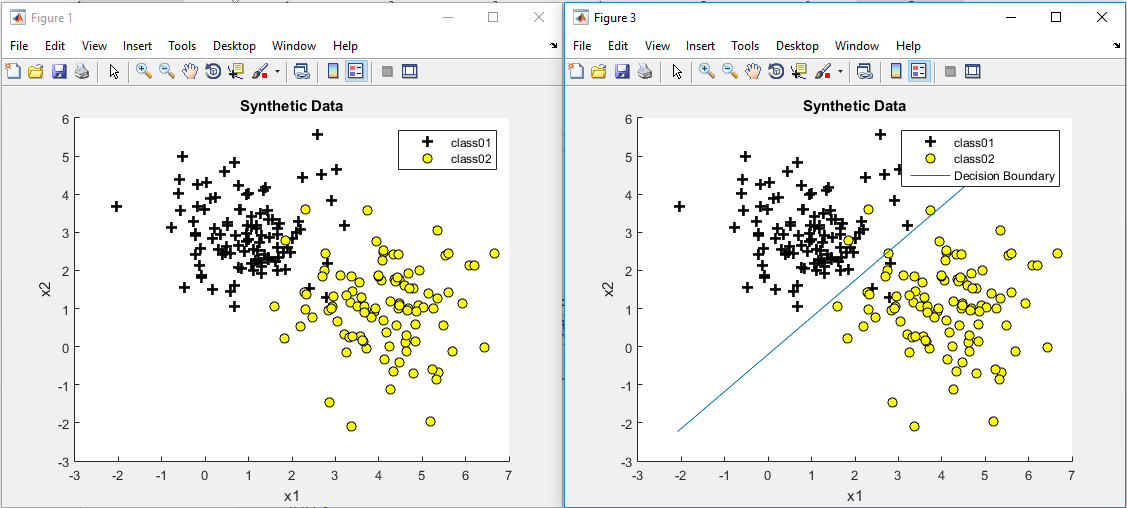


Figure 1: Logistic Regression using Linear Model with Synthetic Data

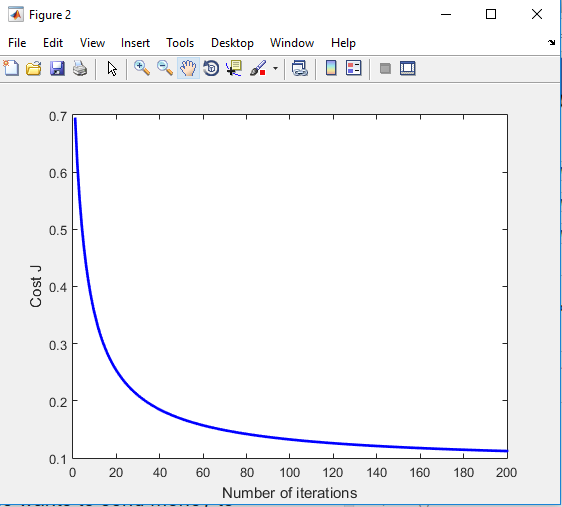
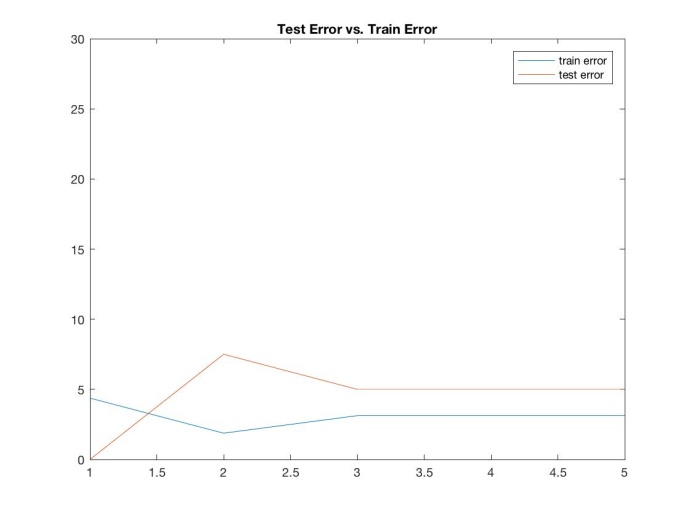


Figure 2: Convergence Curve of Gradient Descent

Here, batch gradient descent was used for cost function optimization.

**Linear Logistic Regression model with RBF and K-Fold Cross Validation with Regularization on 2D synthetic dataset:**

In most cases, if not all, it is not justified to assume a linear model could appropriately represent the data. Therefore, it is essentially important to integrate non-linearity of the data by transforming it into the feature space through some basis function. In our case, we implemented the logistic regression algorithm by incorporating radial basis function (RBF) in the synthetic data. The optimization was done using batch gradient descent.



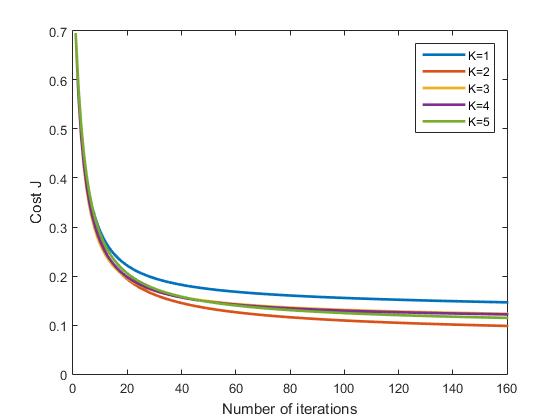


Figure 1: Convergence Curve Figure 2: Test and Train Error

While incorporating nonlinearity in the model, we also applied K-fold cross validation and regularization in this model. After experimenting with the hyper parameters, we came up with the training and test error as below:

Average Train Error : 3.125000

Average Test Error : 4.500000

The K-value for the K-fold cross validation was taken as 5 for our dataset of 200 samples divided in two classes. So, in each of the iterations, 40 test data was played through the logistic regression while rest 160 data samples acted as the training set.

**Real Image Dataset From "Kaggle":**

In terms of choosing a real-world dataset we resorted to the Kaggle Data Repository of following URL:

<https://www.kaggle.com/rhammell/ships-in-satellite-imagery>

The data description as obtained from the website read as below:

*"The dataset consists of image chips extracted from Planet satellite imagery collected over the San Franciso Bay area. It includes 2800 80x80 RGB images labeled with either a "ship" or "no-ship" classification.*

*Provided is a zipped directory shipsnet.7z that contains the entire dataset as .png image chips. Each individual image filename follows a specific format: {label} \_\_ {scene id} \_\_ {longitude} \_ {latitude}.png*

*label: Valued 1 or 0, representing the "ship" class and "no-ship" class, respectively.*

*scene id: The unique identifier of the PlanetScope visual scene the image chip was extracted from. The scene id can be used with the Planet API to discover and download the entire scene.*

*longitude\_latitude: The longitude and latitude coordinates of the image center point, with values separated by a single underscore.*

*The dataset is also distributed as a JSON formatted text file shipsnet.json. The loaded object contains data, label, scene\_ids, and location lists.*

*The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue. The image is stored in row-major order, so that the first 80 entries of the array are the red channel values of the first row of the image.*

*The list values at index i in labels, scene\_ids, and locations each correspond to the i-th image in the data list.”*

***Class Labels***

*The "ship" class includes 700 images. Images in this class are near-centered on the body of a single ship. Ships of different ship sizes, orientations, and atmospheric collection conditions are included. Example images from this class are shown below.*

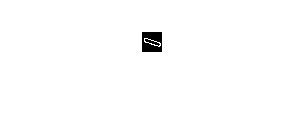
**

*The "no-ship" class includes 2100 images. A third of these are a random sampling of different landcover features - water, vegetion, bare earth, buildings, etc. - that do not include any portion of an ship. The next third are "partial ships" that contain only a portion of a ship, but not enough to meet the full definition of the "ship" class. The last third are images that have previously been mislabeled by machine learning models, typically caused by bright pixels or strong linear features. Example images from this class are shown below.*



**Data Pre-Processing:**

The original RGB images had a Design matrix of dimensions 2800 x 19200. To reduce the dimensionality we transformed the RGB image into (i) grayscale images and (ii) B&W edge image. Even after that the Design matrix was 2800 x 6400. To obtain further reduction of dimension while keeping the important features intact, the image was resized by 25%. The Design matrix dimensions became 2800 x 400 which worked well. The resized images were as below:

**Systematic evaluation of performance of logistic regression on real data:**

As we started working with the chosen dataset, due to the very nature of image dataset of 2800 colored images (RGB) with 80X80 resolution, we faced the Curse of Dimensionality problem while trying to implement logistic regression with RBF and regularization. The implementation took more than 13 minutes in a 4-core parallel processing computing operation as the Design Matrix had the dimension 2800 x 19200. After reduction of the dimension using grayscale image and edge image we also had to resize the images so that the design matrix dimension reduced to 2800 x 400.

**Linear Logistic Regression model with K-Fold Cross Validation with Regularization on Real Grayscale Image dataset:**

We implemented the linear model of Logistic Regression on the pre-processed grayscale dataset. The gradient descent converged expectedly as in the following Figure.

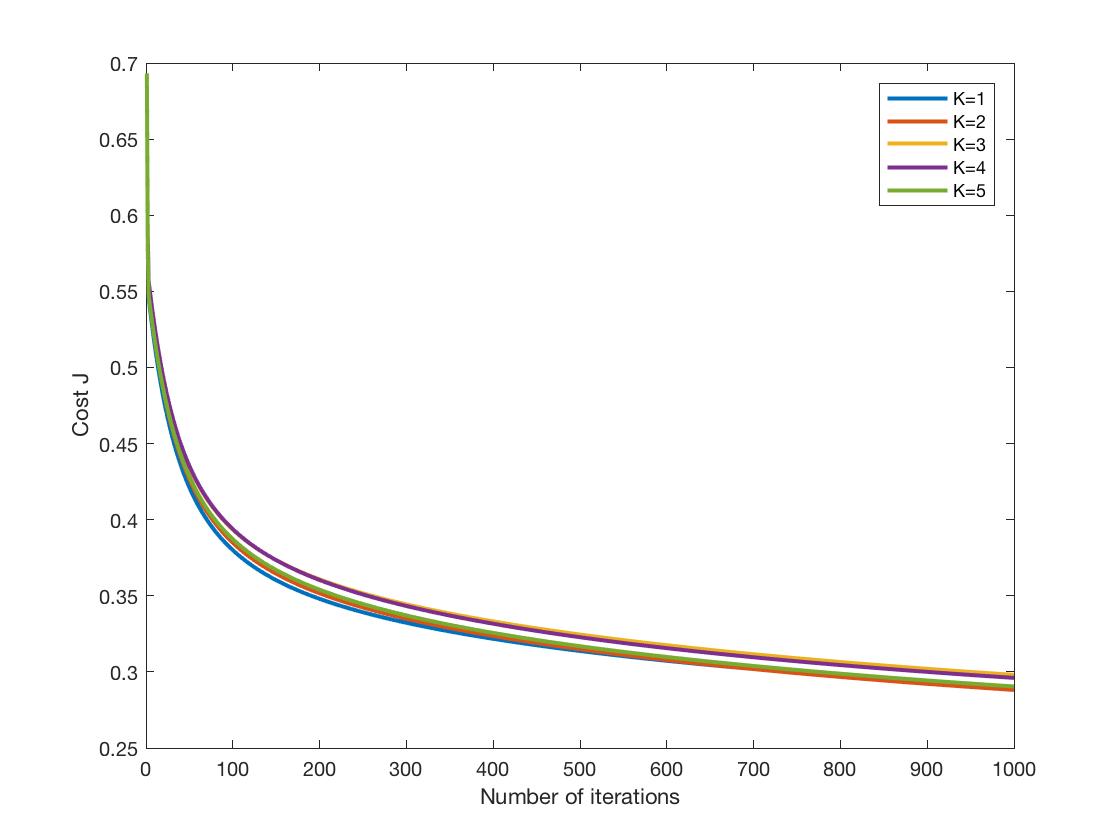


Figure: Gradient Descent Convergence of Linear LR with real data

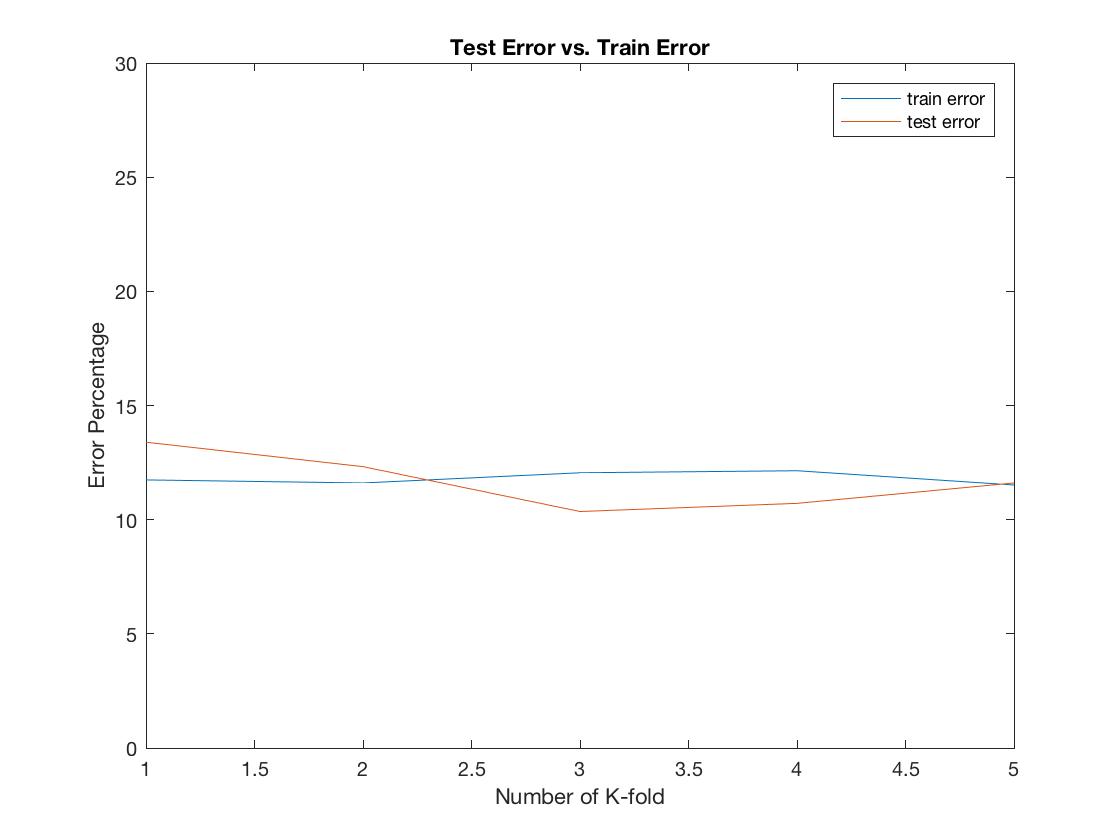


Figure: The Training and Test Error for different K-fold

The hyper parameters used are:

Learning rate, Rho = 0.1

Regularization parameter, lambda = 0.0

The implementation of linear model yielded the classification through logistic regression algorithm and we obtained the following results:

Average Number of misclassifications in train set (out of 2240) : 258.000000

Average Train Error (percent) : 11.812500

Average Number of misclassifications in test set (out of 560) : 65.000000

Average Test Error (percent) : 11.678571

Average Training time : 3.6546

Average Test time : 0.0037026

The lambda was taken as 0.0 because the above results indicate that there is no over-fitting situation even without regularization.

**Logistic Regression model with RBF and K-Fold Cross Validation with Regularization on Real Grayscale Image dataset:**

After the linear model, we implemented Logistic Regression of the real data with RBF (Radial Basis Function) and regularization element. Gradient descent was used for the learning optimization. Whereas the pre-processing of the data took care of the curse of dimensionality problem, the other parameters associated with regularization, learning rate, variance for RBF were optimized with various trial runs.

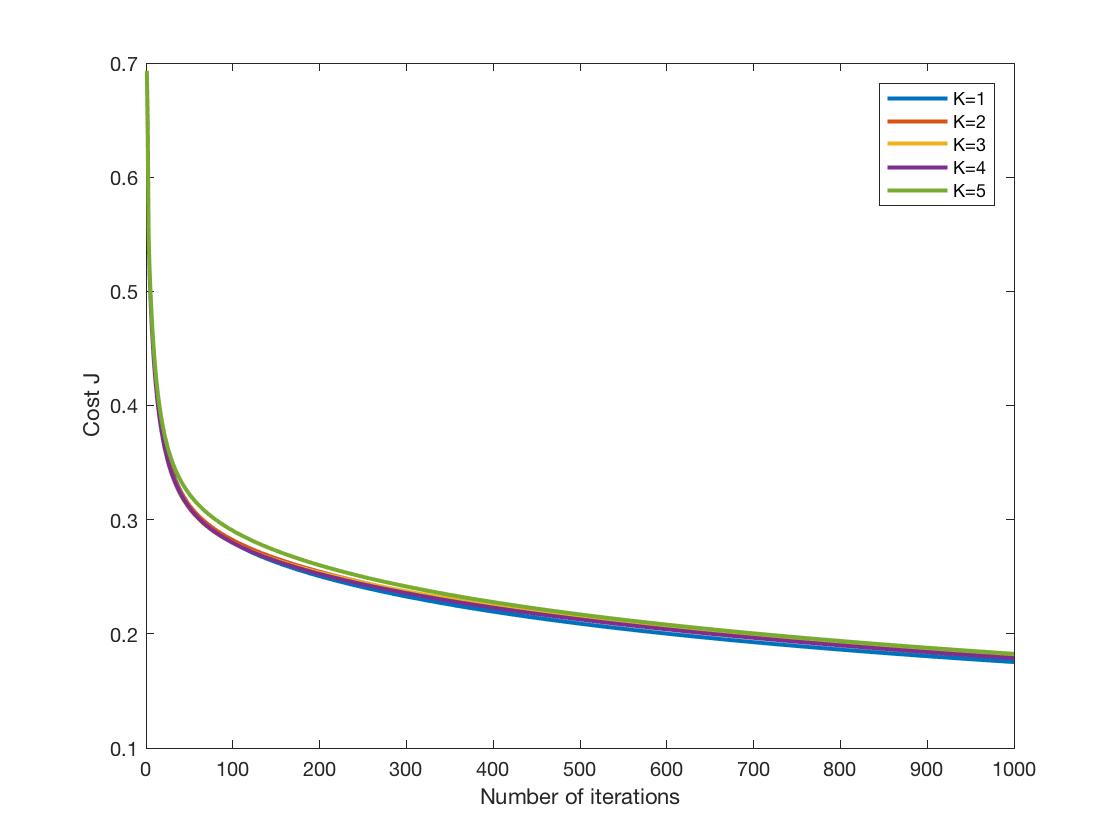


Figure: Gradient Descent Convergence of LR with real data

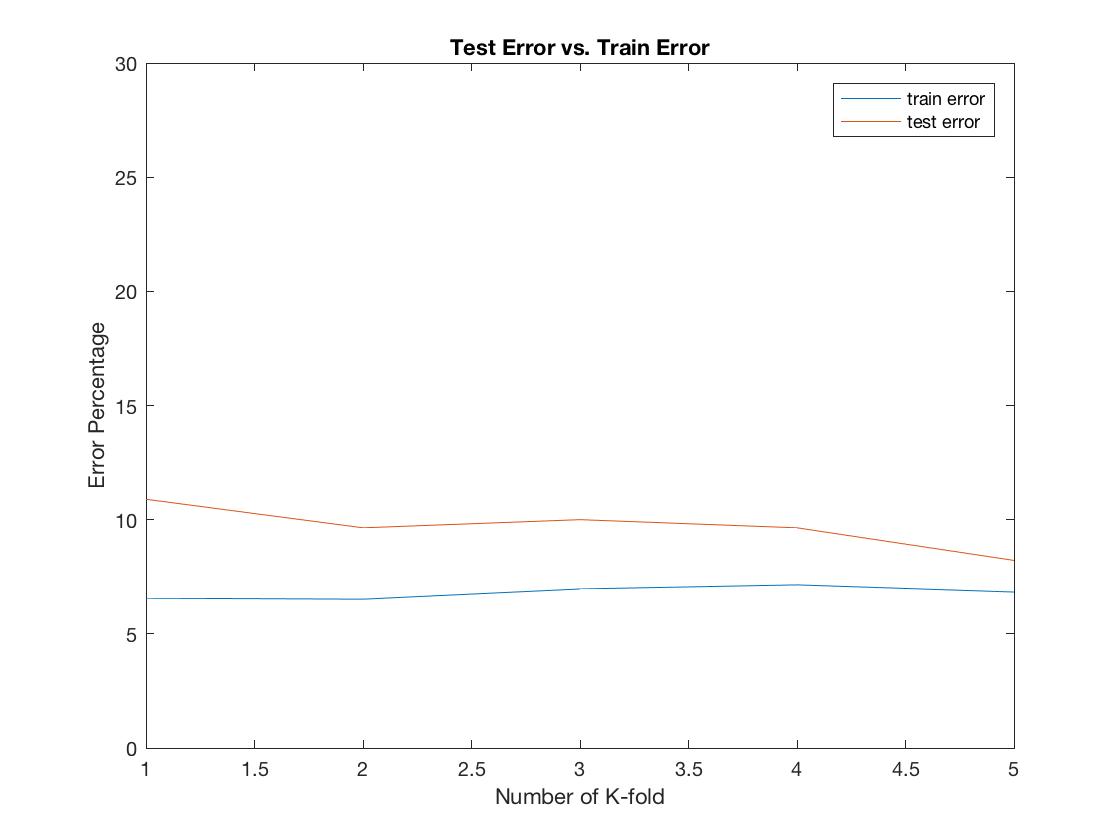


Figure: The Training and Test Error for different K-fold

The hyper parameters used are:

Learning rate, Rho = 1.0

Regularization parameter, lambda = 0.1

Standard Deviation for RBF, S = 1

The non-linear model of real data gave the following performance results:

Average Number of misclassifications in train set (out of 2240) : 152.400000

Average Train Error (percent) : 6.803571

Average Number of misclassifications in test set (out of 560) : 54.200000

Average Test Error (percent) : 9.678571

Average Training time : 41.3451

Average Test time : 5.1305

It is observed that the non-linear RBF model gives significantly better performance in terms of training and test error. The errors reduced by on an average 2%.

**Logistic Regression model with RBF and K-Fold Cross Validation with Regularization on Real Edge Image dataset:**

To check the robustness of our MATLAB implementation, we tried to apply the same model on the edge images and check if it could take care of the sparse matrix due to the binary image.

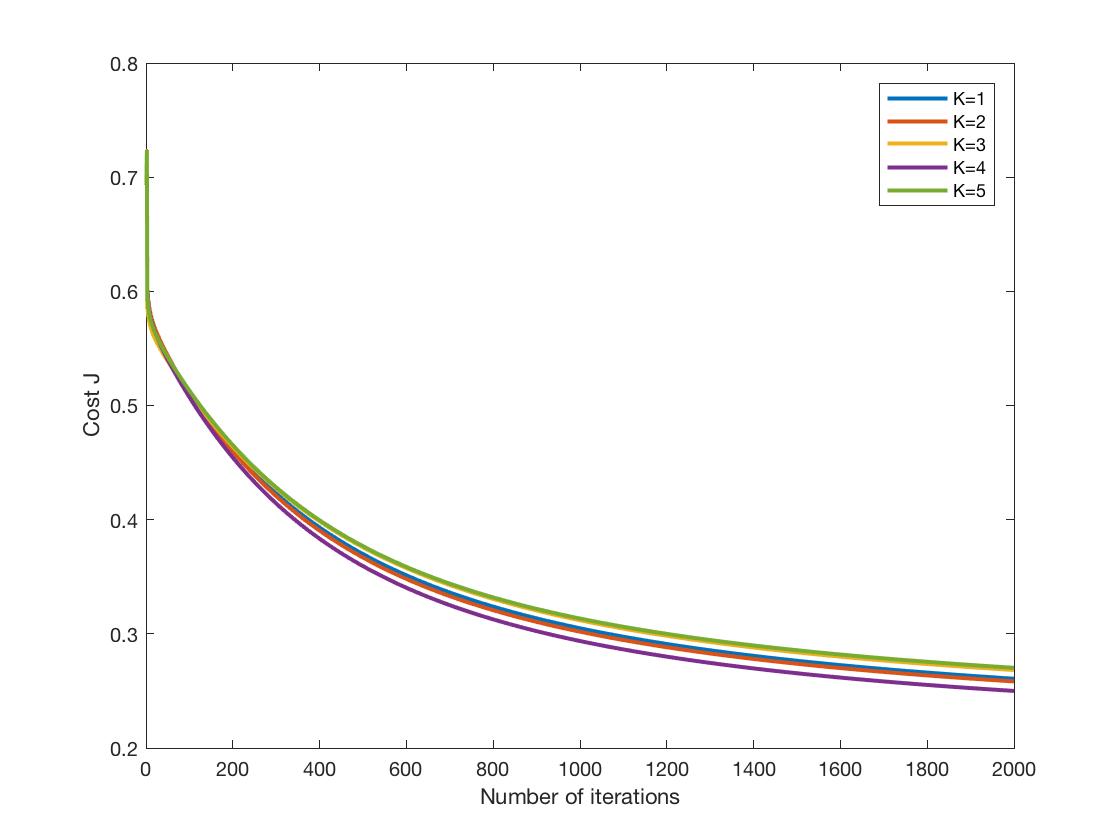


Figure: Gradient Descent Convergence of LR with real data

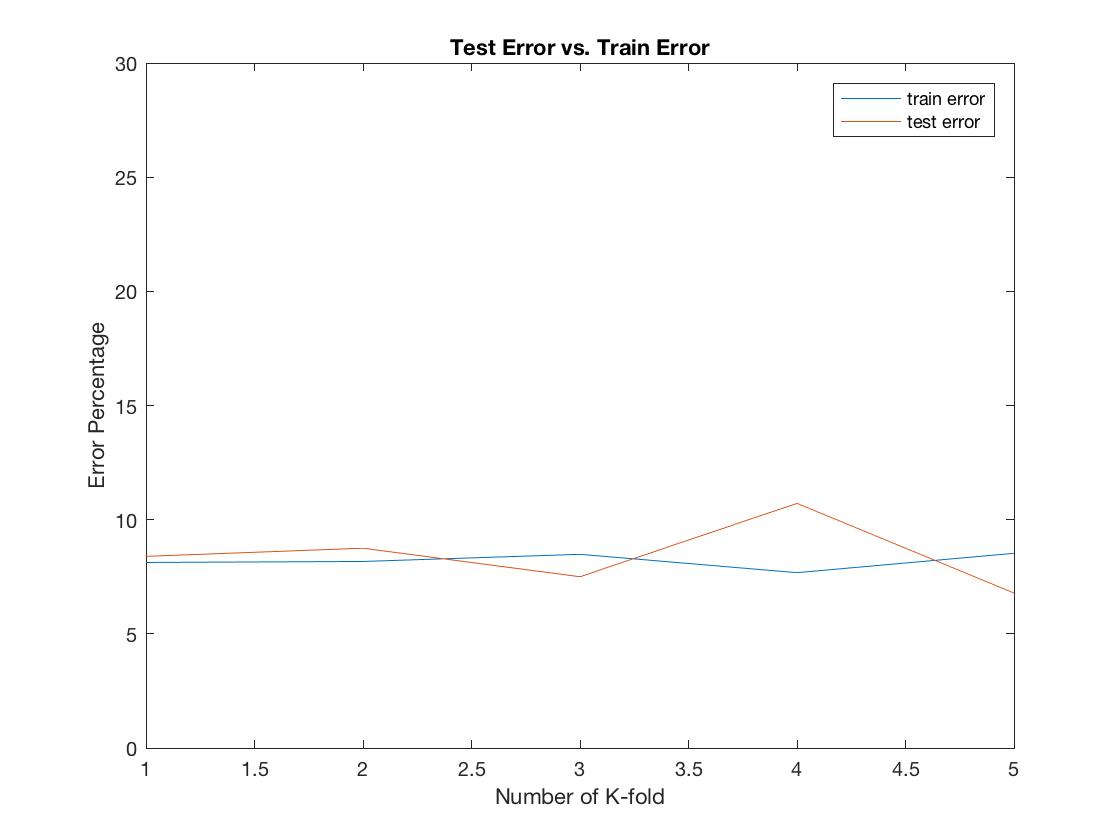


Figure: The Training and Test Error for different K-fold

The hyper parameters used are:

Learning rate, Rho = 0.01

Regularization parameter, lambda = 0.1

Standard Deviation for RBF, S = 0.1

The non-linear model of real data gave the following performance results:

Average Train Error: 8.500000

Average Test Error: 7.321429

Average Training time : 67.3896

Average Test time : 6.0903

After taking multiple run of the codes we get the timevalues such as:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Grayscale\_RBF | | Edge\_RBF | | Linear\_Grayscale | |
| Test Time | Train Time | test | train | test | train |
| 5.1305 | 41.3451 | 4.8292 | 59.8694 | 0.0037026 | 3.6546 |
| 4.9534 | 37.4069 | 4.706 | 57.5239 | 0.0035751 | 3.5087 |
| 4.7134 | 36.8821 | 4.6257 | 61.7746 | 0.0036228 | 3.5228 |
| 4.7283 | 36.8671 | 4.629 | 59.0651 | 0.0020667 | 3.6534 |
| 4.6867 | 38.645 | 4.9151 | 59.8167 | 0.0035557 | 3.5749 |

**To summarize here are the results of our implementations in tabular form:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Grey Linear | Grey RBF | Edge RBF |
| Training Error | 11.81 | 6.80 | 8.50 |
| Test Error | 11.68 | 9.68 | 7.32 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Grey Linear | Grey RBF | Edge RBF |
| Training Time avg | 3.58 | 38.23 | 59.61 |
| Test Time avg | 0.0033 | 4.84 | 4.74 |