MACHINE LEARNING

PROGRAMMING ASSIGNMENT - 1

BY:

ASHIMA GARG

PhD19003

Folder Description:

- Solution 1
  - Solution i)
    - DATASET
      - Dataset.data/Dataset.csv
    - RESULT
    - SOURCE
      - config : directory and hyperparameter information
      - main : driver functions
      - data : data reader and pre-processor class
      - model : class for model operations
      - utils : utility functions
  - Solution ii)
    - DATASET
      - Train.csv : obtained from Lowest RMSE Value in Solution i)
      - Test.csv : obtained from Lowest RMSE Value in Solution i)
    - RESULT
    - SOURCE
      - config : directory and hyperparameter information
      - main : driver functions
      - data : data reader and pre-processor class
      - model : class for model operations
      - utils : utility functions
      - model_L1 : class for model using L1 Regularization
      - model_L2 : class for model using L1 Regularization
      - model_sklearn : class for model using sklearn library
  - Solution iii)
    - DATASET
      - data.csv
    - RESULT
    - SOURCE
      - config : directory and hyperparameter information
      - main : driver functions
      - model : class for model operations
      - data : data reader and pre-processor class

- ❖ model_L1 : class for model using L1 Regularization
- ❖ model_L2 : class for model using L2 Regularization

- Solution 2
  - ➢ Solution a
    - DATASET
      - ❖ train.csv
      - ❖ test.csv
      - ❖ dataset_description : contains the description of the features in the dataset.
    - RESULT
    - SOURCE
      - ❖ config : directory and hyperparameter information
      - ❖ main : driver functions
      - ❖ model : class for model operations
      - ❖ data : data reader and pre-processor class
      - ❖ model_L1 : class for model using L1 Regularization
      - ❖ model_L2 : class for model using L2 Regularization
      - ❖ utils : general purpose utility functions
  - ➢ Solution b
    - DATASET
      - ❖ train-images-idx3-ubyte
      - ❖ train-labels-idx1-ubyte
      - ❖ t10k-images-idx3-ubyte
      - ❖ t10k-labels-idx1-ubyte
    - RESULT
    - SOURCE
      - ❖ config : directory and hyperparameter information
      - ❖ data : data reader and pre-processor class
      - ❖ main : driver functions
      - ❖ model : class for model operations

Solution 1) Linear Regression

Part i) Linear Regression on Abalone Dataset

Hyperparameters Used for Gradient Descent Plots are:

- Learning Rate: 0.001
- Number Of Epochs: 30

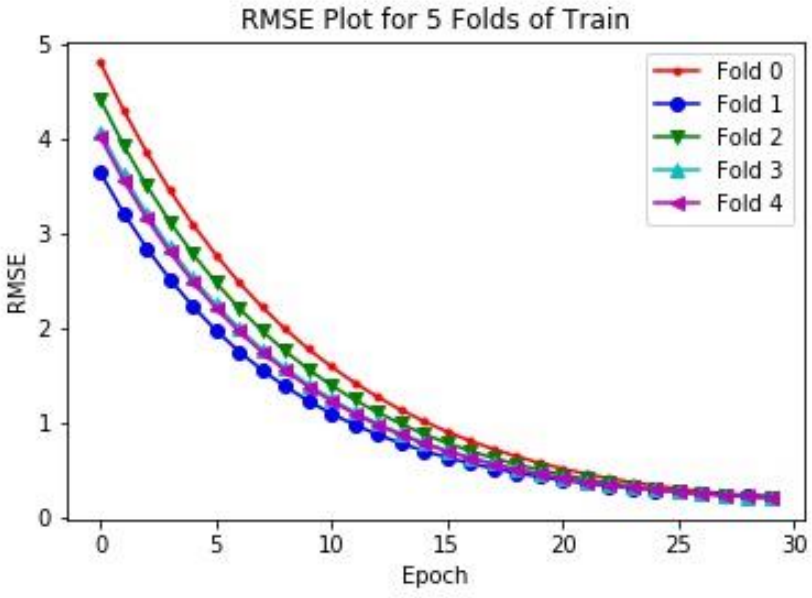
a) RMSE Using Gradient Descent Algorithm

- **Training Set**

- **RMSE Values For 5 Folds**

[0.4524278223932419, 0.3747874178412159, 0.4227582545441477, 0.40051982103426376, 0.39681849919133877]

- **RMSE vs Epochs Plot using Gradient Descent**

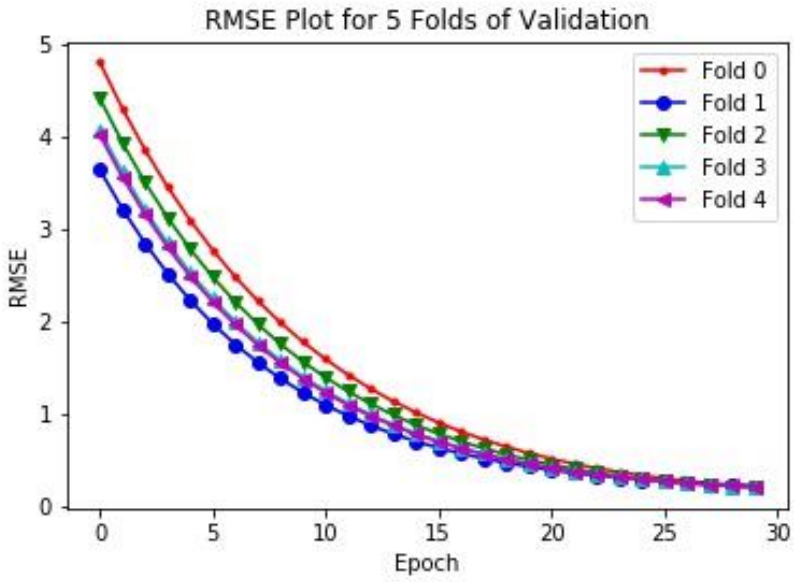


- **Validation Set**

- **RMSE Values for 5 Folds**

[0.19819966178479356, 0.2023279098498869, 0.1859415452252507, 0.1887365799335897, 0.19144745475521174]

- **RMSE vs Epochs Plot using Gradient Descent**



- While tuning hyperparameters for the model, it is observed that when learning rate was kept low as 0.00001 then number of epochs used for convergence was high as 5000.
- When learning rate was increased to 0.001, convergence took only 30 epochs.

b) RMSE using Normal Equation for 5 folds:

- Training Set for 5 Folds

[0.015754244029368328, 0.012777735692760998, 0.013003133295745366, 0.01523872977219804, 0.015108727240727563]

- Validation Set for 5 Folds

[0.00980025074660583, 0.0060083006427809225, 0.007159877140639773, 0.00676164065545948, 0.0070316088463255174]

c) Comparison

- RMSE Values obtained from Normal Equation are lower than those obtained from Gradient Descent Algorithm.

Part ii) Regularization

- **Model Using Scikit Learn Library**

Using Ridge, Lasso, GridSearchCV routines from sklearn library to perform 5-fold Cross Validation on train + validation test obtained from lowest RMSE using Normal Equation in Part i) which is **0.0060083006427809225**

Possible Values of Regularization Parameter (alpha) used are params = {'alpha':[0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10]}

a) Best Regularization Parameter for L2: **1e-06**

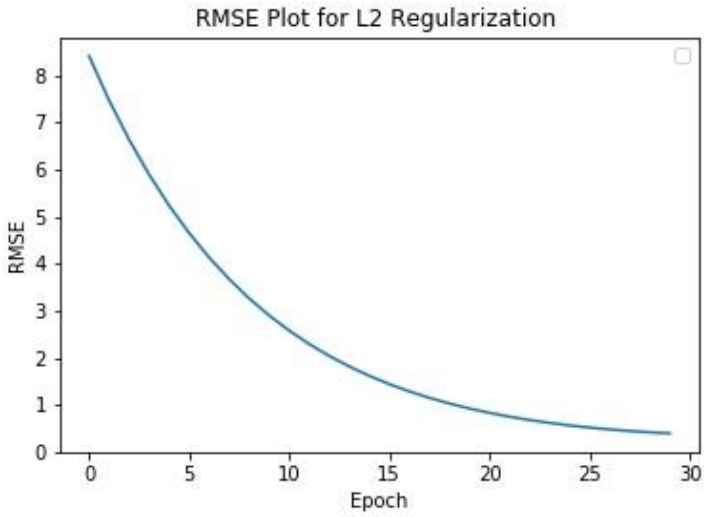
b) Best Regularization Parameter for L1: **1e-05**

- **Model Using Gradient Descent Algorithm**

Hyperparameters Used for Gradient Descent Plots are:

- Learning rate: 0.001
- Number of Epochs: 30
- ***L2 Regularization*** *with* Regularization Parameter : ***1e-06***

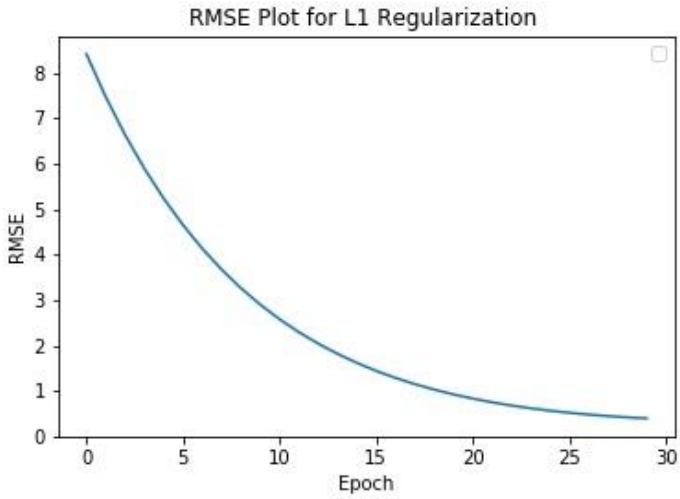
RMSE Plot for L2 Regularization



RMSE for Test Set using L2 Regularization: **0.3791853038179016**

- ***L1 Regularization*** *with* Regularization Parameter : ***1e-05***

RMSE Plot for L1 Regularization



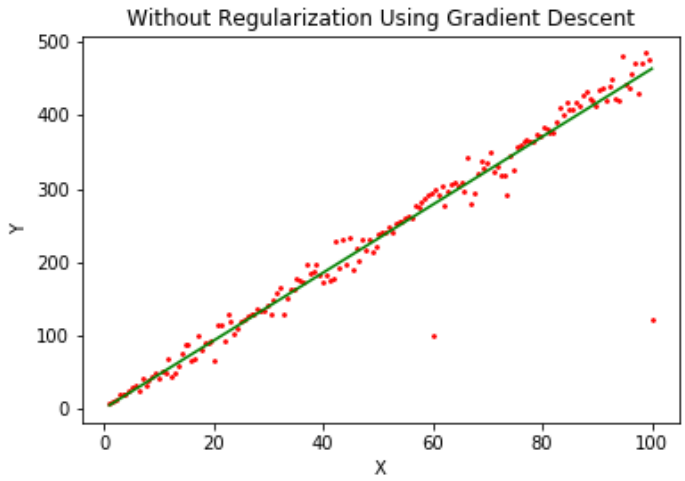
RMSE for Test Set using L1 Regularization: **0.3791853062673536**

Part iii) Best Fit Line Plots Using Gradient Descent Algorithm

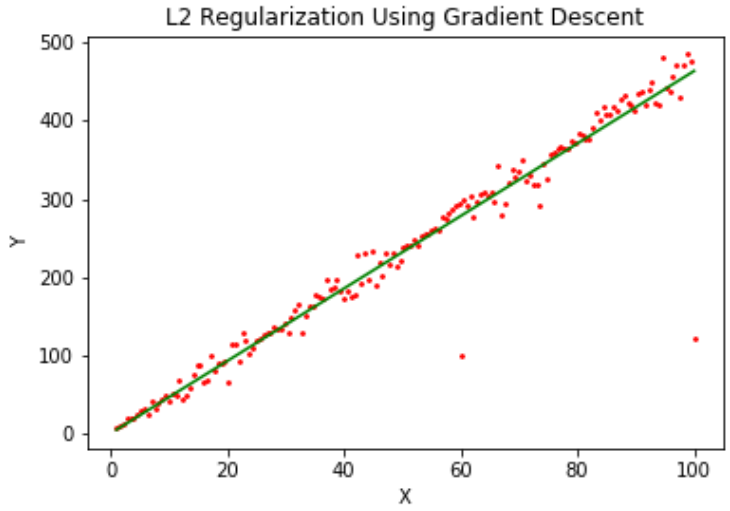
Hyperparameters Used for Gradient Descent Plots are:

- Learning rate = 0.0001
- Number of epochs = 10
- Regularization parameter(Lambda) = 0.01

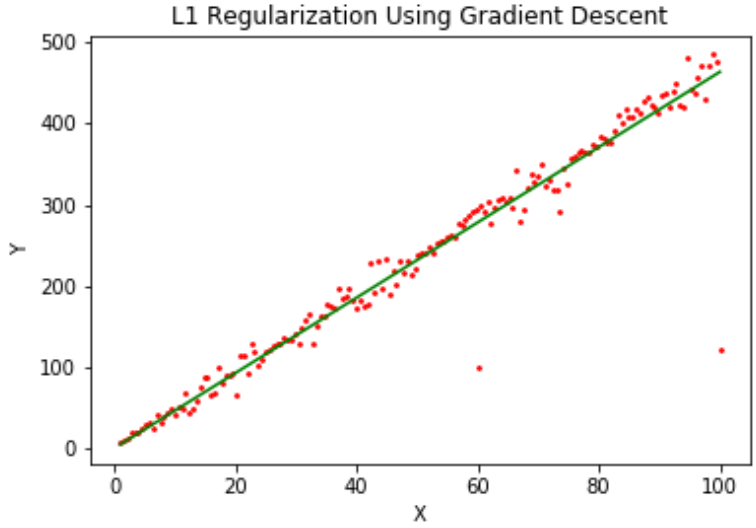
a) Without Regularization Plot



b) L2 Regularization Plot



c) L1 Regularization Plot



Observations Using Actual and Predicted Values

Case 1: Without Regularization

Actual	Predicted
8.4293375	4.63086793
10.51622485	7.36773623
12.33974404	10.10460453

Case 2: L2 Regularization

Actual	Predicted
8.4293375	4.63086383
10.51622485	7.36773217
12.33974404	10.1046005

Case 3: L1 Regularization

Actual	Predicted
8.4293375	4.63086793
10.51622485	7.36773623
12.33974404	10.10460453

Solution 2) Logistic Regression

Part i) Logistic Regression with L1 and L2 Regularization

All the features are not used to train the model. Only age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week are used as the features to model the data.

Data from Train.csv is divided into train and validation set with validation set having 30% of the randomly shuffled data. Data for Test Set is taken from Test.csv. Labels are encoded using sklearn library. Features are normalized before feeding into the model using:

$Z = (x - \min(x))/(\max(x) - \min(x))$

Hyperparameters used:

- Number of Epochs: 250
- Learning Rate: 1.0
- Regularization Parameter: 1

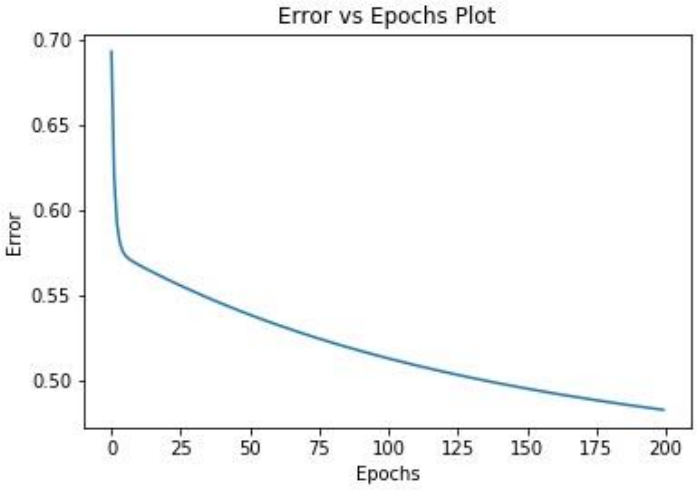
- **No Regularization**

Train Set accuracy: [76.45637965] without Regularization

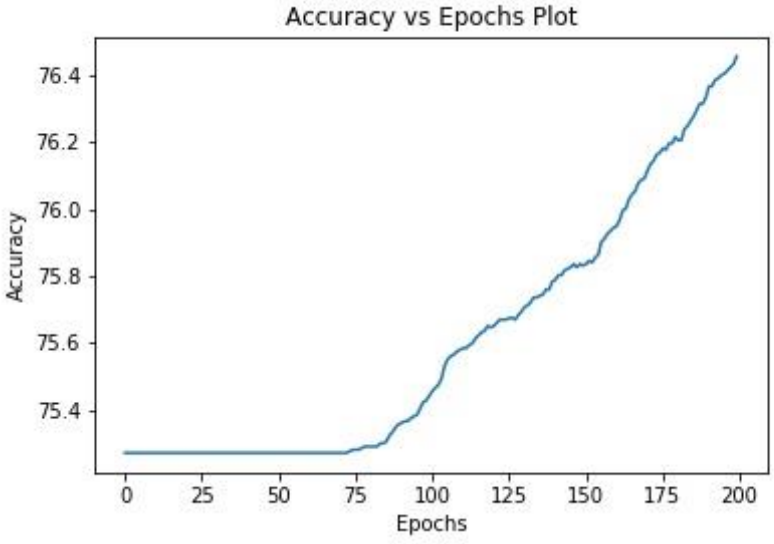
Validation Set Accuracy: [75.7847038] without Regularization

Test Set Accuracy: [76.74634794] without Regularization

- ***Error vs Iteration Graph***



- ***Accuracy vs Iteration Graph***



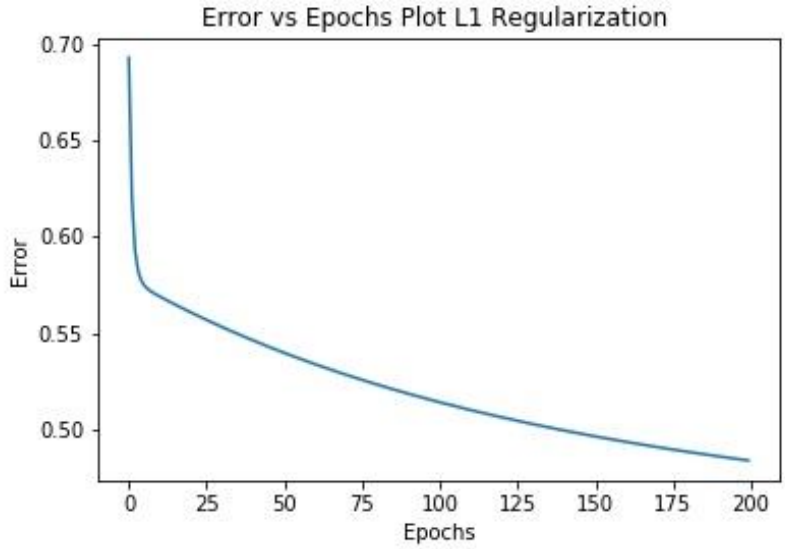
- **L1 Regularization:**

Train Set accuracy: [76.32850242] using L1 Regularization

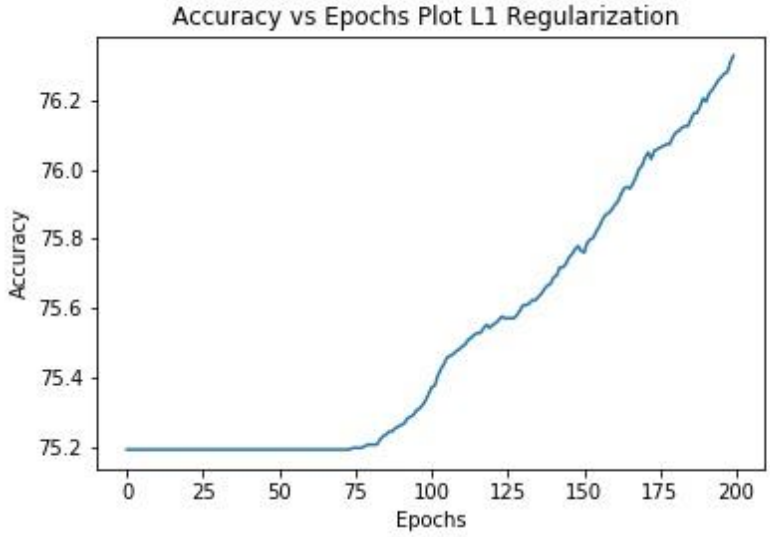
Validation Set Accuracy: [76.07206012] using L1 Regularization

Test Set Accuracy: [76.71978752] using L1 Regularization

- ***Error vs Iteration Graph***



- ***Accuracy vs Iteration Graph***



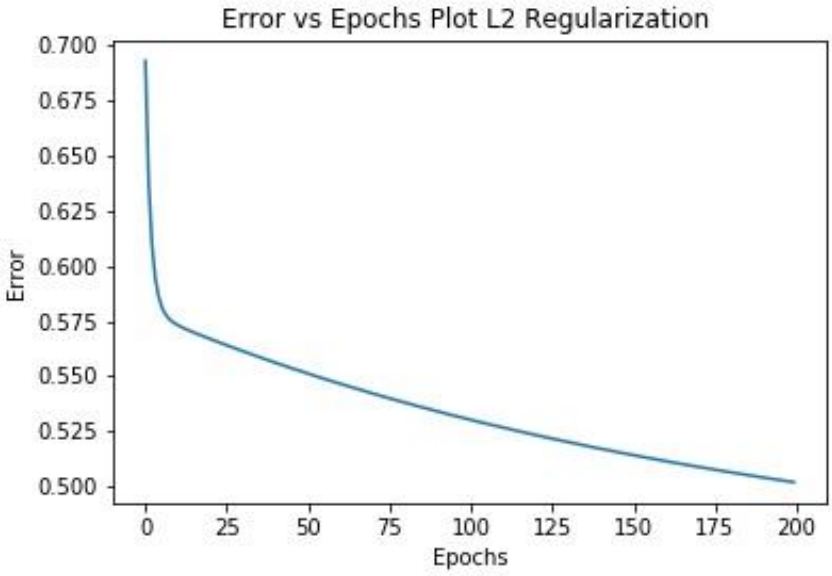
- **L2 Regularization:**

Train Set accuracy: [75.53282182] using L2 Regularization

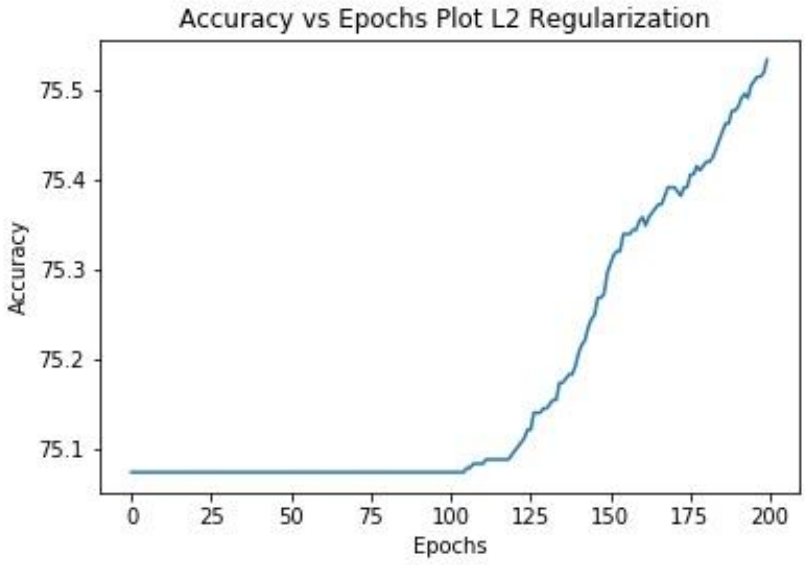
Validation Set Accuracy: [75.79575597] using L2 Regularization

Test Set Accuracy: [76.02257636] using L2 Regularization

- ***Error vs Iteration Graph***

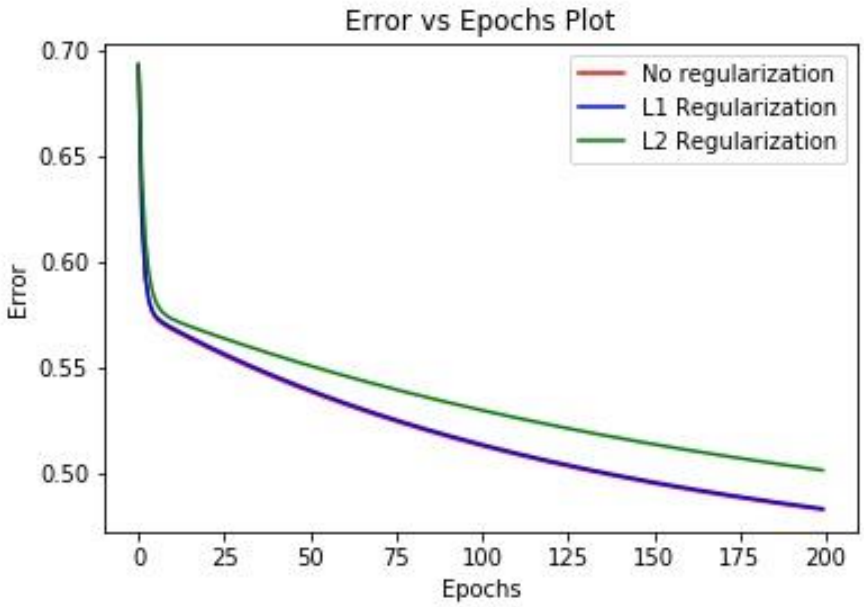


- ***Accuracy vs Iteration Graph***

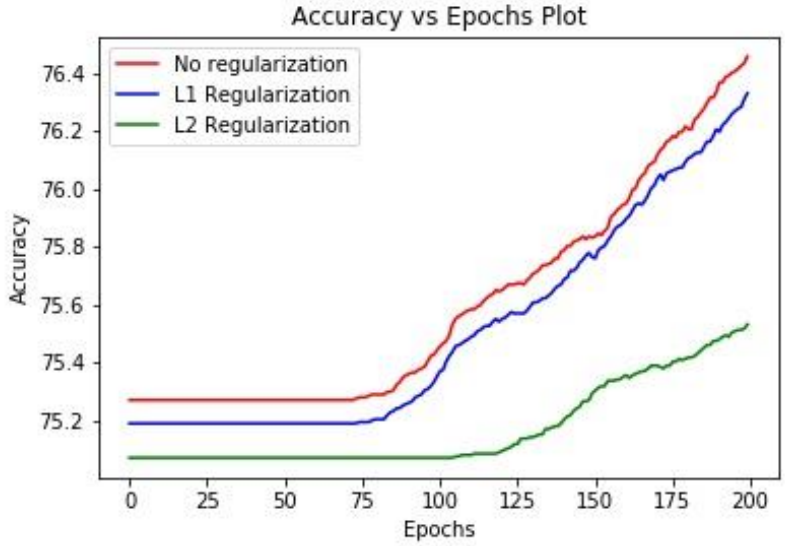


Comparison Plots of No Regularization, L1 Regularization, L2 Regularization

- ***Error vs Iteration Graph***



- ***Accuracy vs Iteration Graph***



- From the above graphs, it can be inferred that with current setting of learning rate, number of epochs and regularization parameter, the model performs best when not regularized.
- Also, L1 Regularization works better than L2 Regularization.

- It is observed when learning rate was changed from 0.01 to 1, the number of epochs required for convergence decreased from 2500 epochs to 250 epochs. Learning rate can be increased upto a certain value as very large learning rates can explode gradients. This was observed when cost value becomes *nan*.

Part ii) Regularization on MNIST Dataset with One V/S Rest Approach

- **L2 Regularization Accuracy of 10 Classes**
  - *Train Set Accuracy Score-L2 Regularized:* ***0.9311***
  - Accuracy of each class in Train Set in table listed below:

Class Label	Accuracy
Label 0	0.9803325223033252
Label 1	0.9786896794646002
Label 2	0.9160628019323671
Label 3	0.9000196039992159
Label 4	0.9423749742745421
Label 5	0.8961384820239681
Label 6	0.9634417289436478
Label 7	0.9435748792270532
Label 8	0.8816604708798017
Label 9	0. 8985565356856455

  - *Test Set Accuracy Score-L2 Regularized:* ***0.9177***
  - Accuracy of each class in Test Set in table listed below:

Class Label	Accuracy
Label 0	0.9775510204081632
Label 1	0.9797356828193833
Label 2	0.8914728682170543
Label 3	0.902970297029703
Label 4	0.9317718940936863
Label 5	0.8699551569506726
Label 6	0.9457202505219207
Label 7	0.919260700389105
Label 8	0.8624229979466119
Label 9	0.8850346878097126

- **L1 Regularization Accuracy of 10 Classes**
  - *Train Set Accuracy Score-L1 Regularized:* ***0.90502***
  - Accuracy of each class in Train Set

Class Label	Accuracy
Label 0	0.9724249797242498
Label 1	0.9682986967241987
Label 2	0.8743961352657005
Label 3	0.8737502450499902
Label 4	0.9220004116073266
Label 5	0.8666222814025744
Label 6	0.9495051504746516
Label 7	0.9153623188405797
Label 8	0.8323007021891781
Label 9	0.8626704089815558

  - *Test Set Accuracy Score-L1 Regularized:* ***0.9041***
  - Accuracy of each class in Test Set

Class Label	Accuracy
Label 0	0.9785714285714285
Label 1	0.9744493392070485
Label 2	0.872093023255814
Label 3	0.8960396039603961
Label 4	0.9287169042769857
Label 5	0.8497757847533632
Label 6	0.9405010438413361
Label 7	0.9036964980544747
Label 8	0.8305954825462012
Label 9	0.8543111992071358

Observations

- Since, the difference between Train and Test set accuracies is not very much, it is a good fit.
- It is observed when argument *'solver'* of Logistic Regression for L2 regularization was set to default '*liblinear'*, the convergence time was very high even for 10000 samples.

- Also, a large difference between train and test set accuracies as 0.9852 and 0.8606 respectively for L2 regularization, with 10000 samples, possible reason could be *overfitting* with less data samples.
- Therefore, it is changed to solver = '*lbfgs'* in Logistic regression L2 which could converge entire dataset of 60000 samples in small amount of time.
- Observations included above are when solver = 'lbfgs'. However, other observations are in the file located at RESULT/Accuracy_L2.txt.
- For L1 regularization, solver used is 'saga' as the documentation also says *for small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones.*
- Convergence of L2 regularization using 'lbfgs' is much faster than L1 regularization using 'saga'.
- Also, pre-processing step of standardizing the data helped in convergence. In the code, scikit-learn library function StandardScaler() is used to standardize the features. It standardizes the feature by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as: z = (x - u) / s
- C which is inverse of regularization strength default value of 1.0 is used in both L1 and L2.

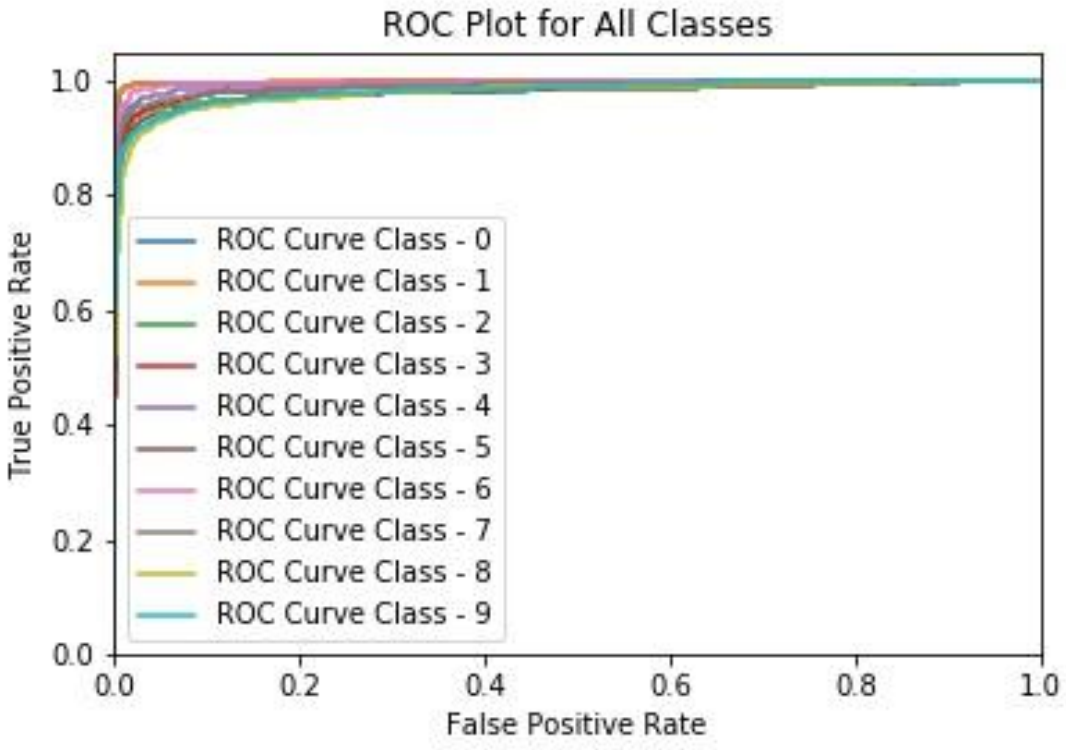
Part iii) ROC Plot for All 10 Classes.

Note: Sklearn library used to plot ROC curve.

True Positive rate: (TP)/(TP + FN)

False Positive rate: (FP)/(TN + FP)

- The lines get smoother as the number of training and test examples are increased. Plots were more distorted when trained using 100 samples. It gets smoother as the number of samples increased to 1000(As shown below).
- In Test Set, Accuracy of Class Label 1, i.e. digits that are 1 is the highest in both L1 and L2 Regularization. This can be verified in the ROC Curve also as the curve for Class Label 1 is the best as compared to other classes.



References:

1) Normal Equation: http://cs229.stanford.edu/notes/cs229-notes1.pdf
2) Ridge, Lasso, GridSearchCV, LogisticRegression: https://scikit-learn.org/stable/
3) Normalization of Features: https://medium.com/@rrfd/standardize-or-normalize-examples-in-python-e3f174b65dfc