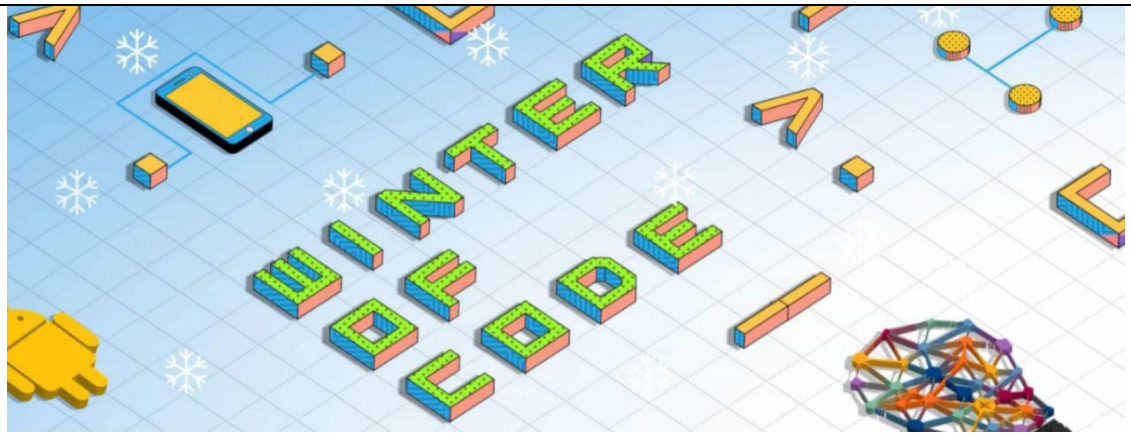


WINTER OF CODE 4.0

2022

PROJECT REPORT

VISHAK K BHAT

Project: ML BOOTCAMP

DATE: 17/04/2022

ML BOOTCAMP:

The project demanded the implementation of the machine learning algorithms with using only numpy, matplotlib and pandas and no other library. I have implemented the code in jupyter notebook and transferred it to git repository. In the ups and downs of emotions dealing with the errors and the logic to the algoritms, I have done the following algorithms.

I have implemented the algorithms of:

1. LINEAR REGRESSION RMSE=74.72
2. POLYNOMIAL REGRESSION RMSE=2.36
3. LOGISTIC REGRESSION ACCURACY=67.26%
4. KNN ACCURACY=90%
5. K MEANS CLUSTERING ----------
6. NEURAL NETWORK FOR REGRESSION AND CLASSIFICATION

   ACCURACY=35% (EMNIST)

   RMSE=28.84 (LINEAR DATA)

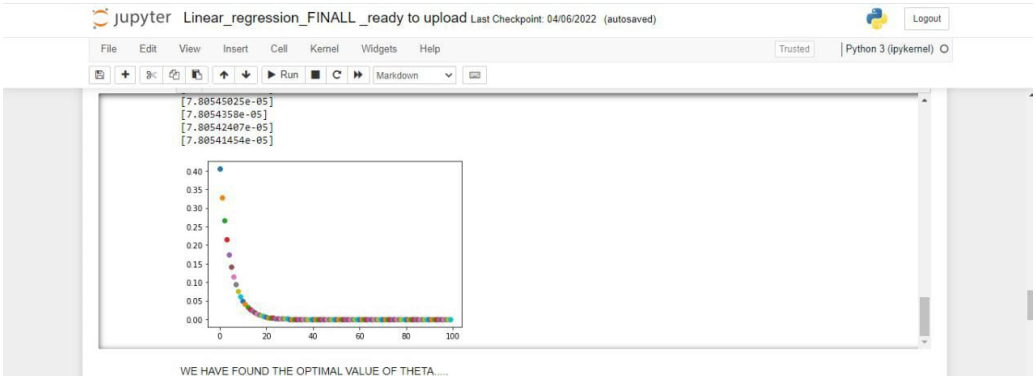
   RMSE=1107 (POLY DATA)

1) Linear regression:

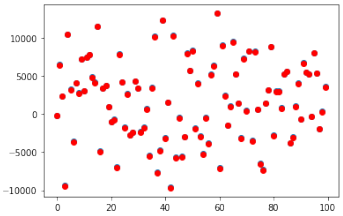
Regression basically means there is a continues data and we need to predict the value for the given input. The data set provided had 20 features and adding a biased feature it finally was 21 features.

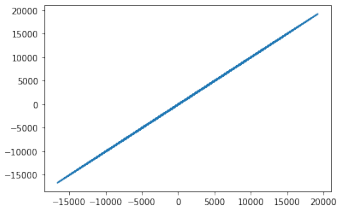
Initially when the code runs it asks to input the file name, there we are supposed to input the train data set and later on once the parameters are been found then it asks us to upload the test data set.

The below image is the graph of the cost vs iteration from my code.



- This being the first algorithm I took time adjusting to use the jupyter notebook and type the code. Took 4 days of time in which have made many attempts.
- **The cost during gradient descent fell to a range of 10^{-5} . On the test data the root mean square error(rmse) was 33.418 and on the test data the rmse was 74.72.**





The image above is a scatter plot of the predicted value (in red) of the data and the actual value (blue) taken from my code. This shows that the prediction and the actual value almost matches.

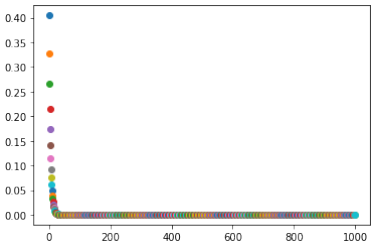
The second image on the right is a plot between prediction and the actual value. Since it should be equal, we are getting a graph of line with slope 45 degree (y=x).

2) Polynomial Regression:

Same like linear regression I have written the code of polynomial regression. Here I have defined the function that creates the higher degrees of the features and after that the code is similar to that of the linear regression. After adding the extra degrees, the features were increased to 20 including the biased feature.

Initially when the code runs it asks to input the file name, there we are supposed to input the train data set and later on once the parameters are been found then it asks us to upload the test data set.

The below image is the image of the graph of iteration vs cost.



We can see that the cost is decreasing.

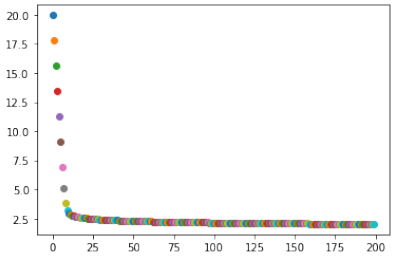
- **The rmse here was 2.36 in the test data set.**

3)Logistic regression:

This was the first code of the classification problem. In this code I have used complete vectorised implementation to perform the one vs all type of classification.

To think of the code to create the higher degrees of features spent me couple of days. There were 26 classes and the main task was to convert the prediction value to binary format. After many attempts and many fall backs, I could manage to accomplish this algorithm.

The below image is taken from my code and shows the graph between the cost and the iteration.



- **The accuracy of this code on the train data was 69%**
- **The accuracy of this code on the test data was 67.26%**

4) K Nearest Neighbour:

This code also solves the classification problem. Here had to see many videos from YouTube and to go through documents to get the feel of the code and the logic behind it. This code doesn't have the training time, Its direct testing time.

The main logic of this code is distance and finding distance using the coordinate geometry formula.

The problem I faced here was the time factor. It took almost 2 hours for the complete code to get implemented but the good thing is that the accuracy was mind blowing.

The accuracy of this algorithm was 90%.

5) K Means clustering:

This was the first unsupervised code that I performed. The main aim of this code was to form the clusters. Due to time constraint couldn't apply the elbow method to find the value of K.

Learnt a new tern called wcss which stands for within the cluster sum of squares, which acts as the cost of this algorithm.

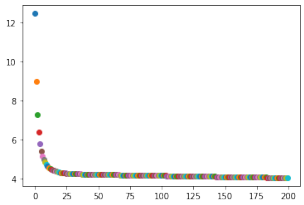
The error I faced was that the initialisation of the centroids. I had taken the centroids randomly and hence wasn't getting the desired result. Further on I rectified it by taking the random centroids from the given data set itself.

6)Neural networking:

In this code I have implemented 2 layer neural network algorithm. For the classification part did the forward propagation and did to get the gradient use the backward propagation. The units inside the hidden layer I took it as 2/3 of the input layer.

Did complete vectorised implementation.

The below image show the graph of the cost vs iteration



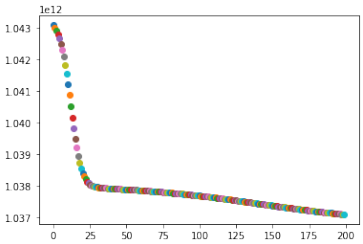
The accuracy of the code of neural network classification was giving around **33%** accuracy.

And an accuracy of **35%** in the test data set.

I tried a lot to increase the accuracy of the code. Rechecked 3 4 times but still couldn't catch my mistake. Had asked mentors also but they were busy with end semester exam. So I am still eager to find out where I am going wrong.

Later on due to time constraint dint go to increase the layers of the neural network but would like to try that and increase the accuracy.

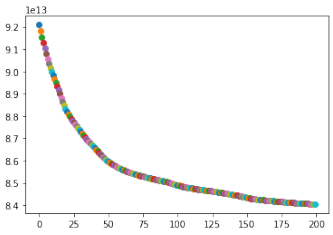
In the regression code using the neural network the below graph was obtained.



The rmse in this algorithm is **28.03** on the train data set. The rmse on the test data set was **28.84.**

The error terms are between -50 and +50. Being a huge data set this much error can be expected.

Finally, when applied the neural network on the polynomial data set got almost results like the linear data. The graph below shows the cost vs the iterations.



The rmse in this case is **1073.**