MACHINE LEARNING BOOTCAMP REPORT

WINTER OF CODE 4.0

SHRESHTH SHARMA
(21JE0895)
IIT (ISM) DHANBAD

<u>Contents</u>

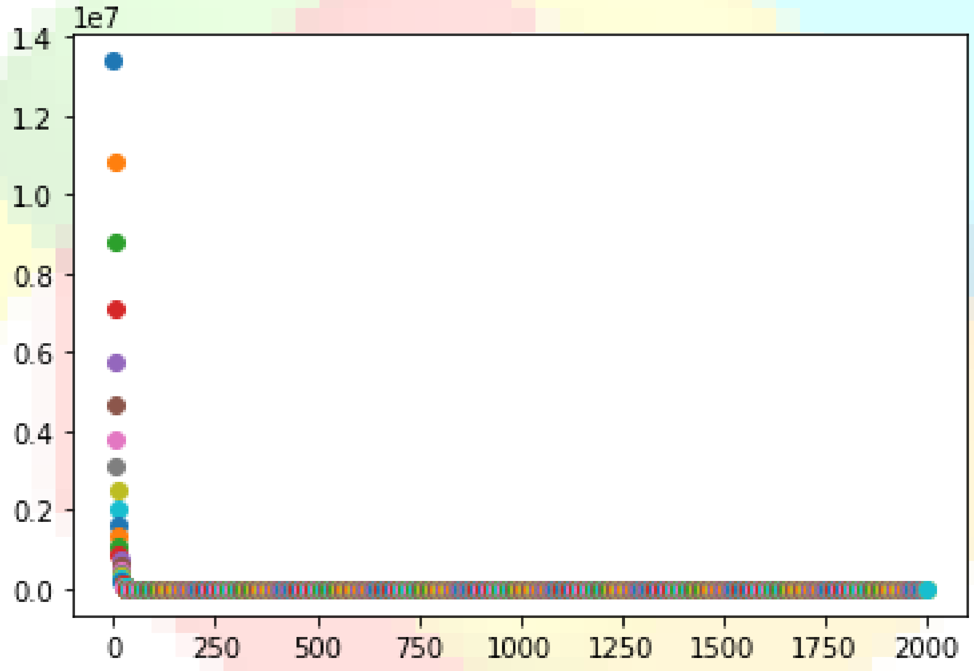
Cyber Labs

Linear Regression

This algorithm involves the use of a linear function as hypothesis and the mean squared error as the cost function. Modifications are made in the cost function if mean normalization and regularization are required for the given data sets. This algorithm is used in regression problems where a continuous output is required from the given dataset. It is an example of Supervised Machine Learning.

A dataset comprising of 50000 examples for training and 10000 examples for testing with 20 features was provided to us.

The following plot for gradient descent was observed. The x-axis represents number of iterations and the y-axis represents the value of cost function.



The mean squared error (MSE) on the training data (**cost function**) was found to be **1907.0986734578898** and on testing data, the **MSE** error was found to be **2790.8495110921144**.

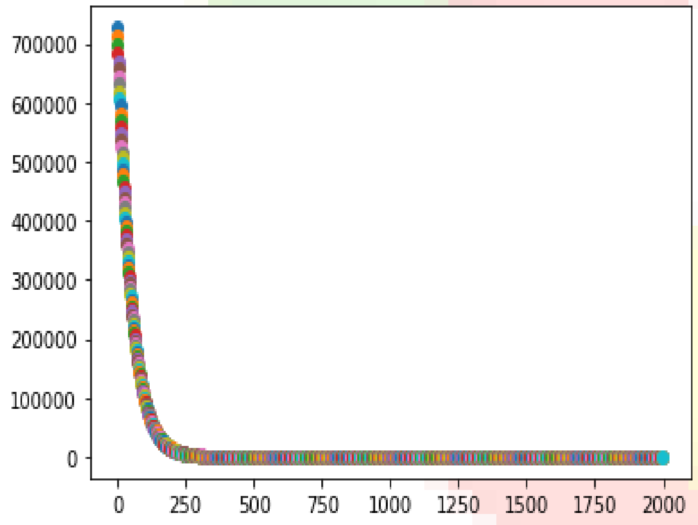
Initially, I normalized the output values for the training and testing data, which resulted in the involvement of data leakage. Thus, normalization or standardization of only the input data and not the output values must be done.

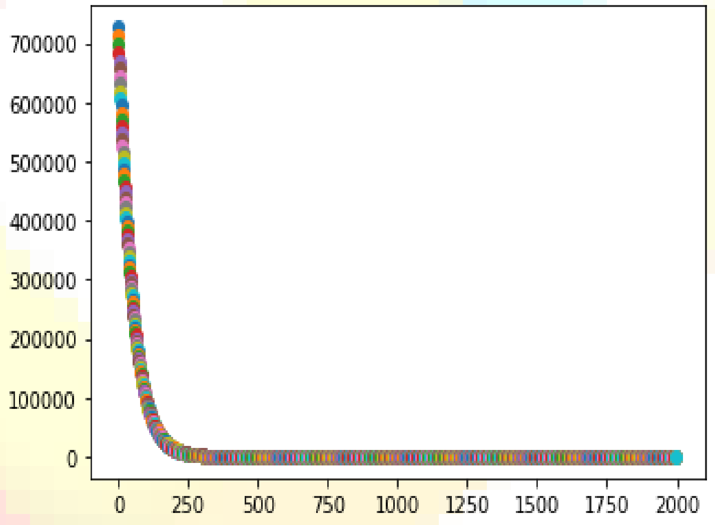
Polynomial Regression

This algorithm involves the use of a polynomial function as hypothesis and the mean squared error as the cost function. Modifications are made in the cost function if mean normalization and regularization are required for the given data sets. This algorithm is used in regression problems where a continuous output is required from the given dataset. It is an example of Supervised Machine Learning.

A dataset comprising of 50000 examples for training and 10000 examples for testing with 3 features (namely X, Y, and Z) was provided to us. We were supposed to form a quadratic and a cubic polynomial with the following features with every possible multiplication and check the functioning of the algorithm

The following plot for gradient descent was observed. The x-axis represents number of iterations and the y-axis represents the value of cost function. The first plot is for the degree 3 polynomial and the second plot is for degree 2 polynomial.





The MSE obtained on training datasets (**cost function**) were **62.340740826528695** and **71.69530953845312** for **degree 3** and **degree 2** polynomials respectively. The **MSE** obtained on testing data were **301675.1939167583** and **301882.2807949841** for degree 3 and degree 2 polynomials respectively.

The insanely high values of MSE on testing data indicate **over fitting of training data** in the model. Hence, regularization might be able to overcome this shortcoming. Low cost can never assure that our model will perform well on our testing data. The reason being over-fitting of the hypothesis on our training data and our testing data not being similar to our training data.

Initially I was using mean normalization but after replacing it with **standardization**, **the cost function** for degree 3 and degree 2 polynomials were **36.85437846888681 and 65.4327304780402**. The **MSE** obtained on testing data were **60.77484745657237** and **48.604307588836214** for degree 3 and degree 2 polynomials respectively.

Standardization is hence a better option as compared to Min-Max Normalization and Mean Normalization since it makes the data follow Gaussian distribution.

Logistic Regression

This algorithm involves the use of the sigmoid function as hypothesis and the cost function is defined by the equation:-

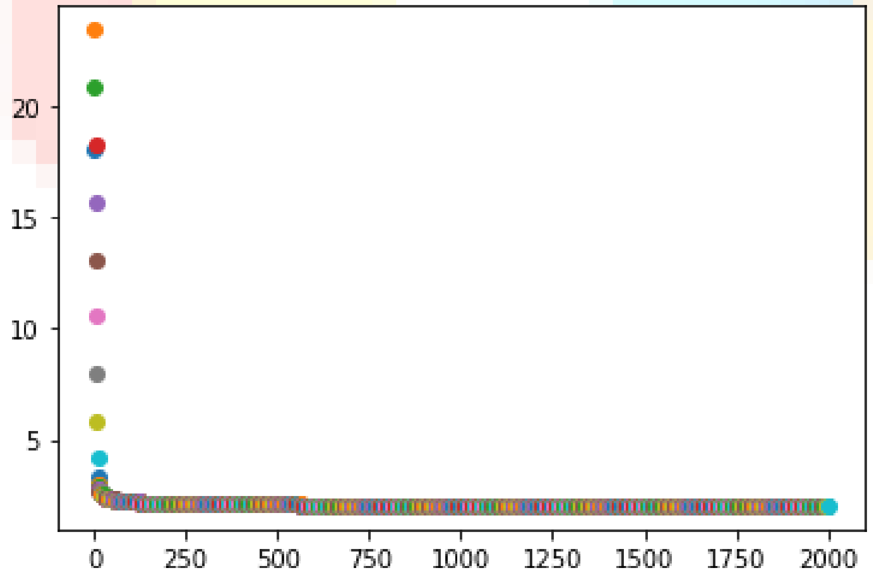
$$h(x) = 1/1+e^{-z}$$

$$y * \log(h(x)) – (1-y) * \log(1-h(x))$$

Here y is the output (one hot encoding) and h(x) represents our hypothesis. The cost function of logistic regression is called ‘Binary Cross Entropy’ or ‘Log Loss’. Modifications are made in the cost function if regularization is required for the given data sets. This algorithm is used in classification problems where a discrete output is required from the given dataset. It is an example of Supervised Machine Learning.

EMNIST Dataset was provided to us. It is a dataset which represents the pixel values of handwritten letter images (28*28 pixel matrixes). The training EMNIST dataset has 88800 datasets with 784 (28*28) features and the testing EMNIST dataset has 14800 datasets with 784 features.

The following plot for gradient descent was observed. The x-axis represents number of iterations and the y-axis represents the value of cost function.



The **cost function** was found to be equal to **2.056131135722447**. Out of 14800 testing images, the algorithm was able to predict 10142 images correctly leading to an

accuracy of **68.52702702702703 %**. The algorithm works well for 2000 iterations with a reasonable computation time.

One v/s all classifier plays a major rule in this algorithm and must be taken care of. One must not confuse it with a regression algorithm because of its name. It is a Classification Algorithm only.

K-Nearest Neighbours

This algorithm works on the concept that similar things tend to exist in close proximity or similar data points are closer to each other. Hence it works on the idea of similarity and it calculates the distance between the testing example and the current examples. Lesser the distance, more is the similarity. This algorithm is used in both regression and classification problems and returns the output according to the type of the problem. It is an example of Supervised Machine Learning.

The same EMNIST data was provided to us for the algorithm. Hence, it was used as a Classification algorithm.

With help of repeated list comprehensions and ‘list.append’, the algorithm was implemented. Due to the slow nature of this algorithm, I chose to consider at max 100 Testing examples. Also due to a high number of calculations, my Jupyter Notebook crashed in the middle of computation. Hence use of 100 Testing Examples is justified. An **accuracy** of **89%** was obtained for the **first 100 testing examples**.

Due to the slow nature of this algorithm, its use must be restricted to small testing and training datasets. It is also called as ‘The Lazy Algorithm’.

K-Means Clustering

This algorithm works on the idea of grouping things into various clusters based on the patterns in the existing data. It tries to minimize the sum of distances between points and their respective cluster centroid. This algorithm is used in unsupervised learning problems since it tries to identify a pattern in the given data sets. For the good functioning of this algorithm the clusters must be concise and as different as possible.

The same EMNIST data was provided to us for the algorithm. Hence, it was used as a Classification algorithm. For this task I used 19 Clusters since in the testing data, a total of 19 distinct outputs were present.

I faced difficulties in the understanding and implementation of this algorithm; therefore I wrote this algorithm for one iteration and then with the help of a function, I used it for all the iterations. After clustering, I evaluated the **Dunn Index** for the clusters and it was equal to **0.17288231039992585**.

Thus it is a poor algorithm for the given dataset as indicated by the Dunn Index.

Dunn Index is the ratio of the minimum inter nuclear distance to the maximum intra nuclear distance. It indicates how distinct and concise the clusters are. Higher the Dunn Index better is the clustering. For Dunn Index to be high, the minimum distance between the clusters is high implying the clusters are well different and the maximum intra nuclear distance must be low implying the clusters are concise.

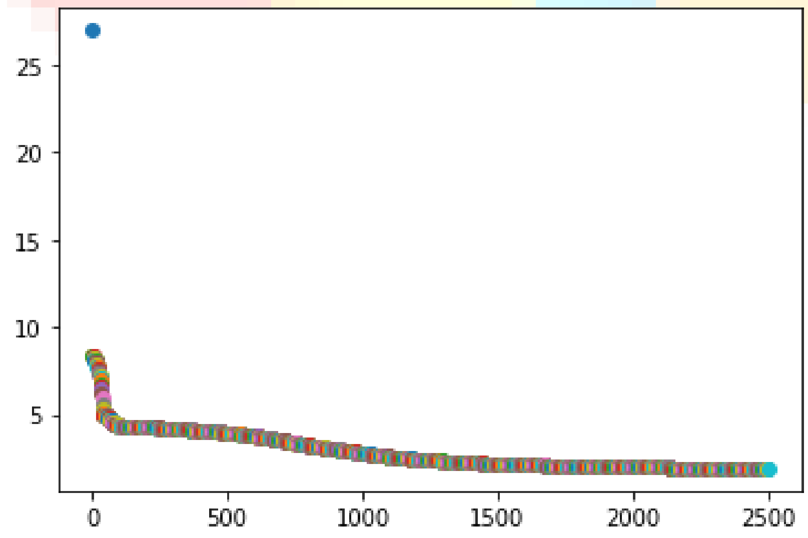
Neural Network

This algorithm tries to mimic the functioning of a human brain. It consists of an input layer, output layer and hidden layers (not necessarily). Each layer acts like a neuron, taking in input, processing the output and sending it to the other layer (other neuron). It is generally used in the case where the decision boundary in case of logistic regression is a complex function to deal with. The cost function used is the same as the one used in logistic regression i.e. Binary Cross Entropy.

The same EMNIST data was provided to us for the algorithm. Hence, it was used as a Classification algorithm. For this task, I used a Two Layer Neural Network: 1 Input Layer, 1 Hidden Layer and 1 Output Layer.

With the help of back propagation, the algorithm was implemented. Initially, the accuracy of the algorithm was observed to be 0 % and was predicting the same output within a row. But with the help of **Xavier Initialization**, the **accuracy improved to 5.236486486486487 %**. After rechecking the back propagation implementation, it was found that the derivative of the sigmoid function was not taken into account by me. Thus, due to the violation of the chain rule of derivatives, the implementation was faulty.

The final plot of the cost function v/s number of iterations is as shown.



After making the necessary changes, **the accuracy boomed to 50.57432432432433 %**. After tuning the hyper-parameters, an **accuracy** of **67.89864864864865 %** was observed.

Due to the error prone nature of the back propagation, caution must be taken while implementing it and computing the gradients and one must check that the chain rule of derivatives is followed.

Xavier initialization is an attempt to improve the initialization of neural network weighted inputs, in order to avoid some traditional problems in machine learning like exploding gradient and vanishing gradient. It basically helps in choosing the right quantity of weights in the network.

Tech Stack

- Jupyter Notebook for Python Environment
- Python Library : NumPy
- Python Library : Pandas
- Python Library : Matplotlib

About Me

Name :	Shreshth Sharma
Place :	Jaipur, Rajasthan
Degree :	Integrated Master of Technology
Branch :	Mathematics and Computing
Year :	First
E-Mail :	shresthrw@gmail.com , 21je0895@iitism.ac.in
Mobile Number :	+918005724831
LinkedIn :	linkedin.com/in/shreshth-sharma-799b9b226
GitHub :	github.com/ShrenzyPanda