Analysis of Learning Rate in Gradient Descent Predictions using Multi-Layer Perceptron Network

Sumeet Thakur
Research Scholar M.Tech student
Department of Computer Science,
Himachal Pradesh University,
Shimla, India

Abstract: In current scenarios with increasing advancements in Information Technology (IT) and data science. The importance and need of Machine Learning has also increased. Machine Learning algorithms are being used to analyse data for further use. Multi-Layer Perceptron Network is an artificial neural Network which uses machine learning techniques to Learn and Predict results. This research focuses on analysing Learning Rate of Gradient Descent Algorithm for Prediction Accuracy using Multi-Layer Perceptron Network.

Key points:(IT), Artificial Intelligence (AI), Multilayer Perceptrons(MLP), Multi-Layer Perceptron Network (MLPN)

I. INTRODUCTION

In today's world, a lot of data is flowing through the web. The data is being used for various types of analysis based tasks using Artificial Intelligence (AI). Machine Learning plays a key role in training of the model on particular data. It is an application of AI which gives the system, the ability to learn and evolve from experience. Learning can range from trivial to profound. An Agent is said to be learning if its performance is improved after making observations about the world [7].

Multilayer Perceptrons(MLP) are the classical type of artificial neural network. These have been comprised of one or more layers of neurons. Data is supplied to the input layer, there can be more hidden layers providing levels of abstraction and predictions, are made on the visible layer or output layer. This Research has conducted an analysis of Learning Rate for Gradient Descent Algorithm for prediction accuracy in Multi-Layer Perceptron Network (MLPN)[1] .

There are some properties of architecture of MLPN: (I) no connections within a layer, (II) no direct connections between input and output layers, (III) Fully connected between layers often more than 3 layers, (IV) Number of output units need not equal number of input unit and (V) Number of hidden units per layer can be more or less than input or output units.

1.1 Gradient Descent Optimization

Gradient descent is a popular optimization strategy, used in machine learning and deep learning. It is used for training models and can be combined with every algorithm and is also easy to understand and implement.

Gradient Descent optimization algorithm is used for training a machine learning model. It is an optimization algorithm**.** It is simply used to find the values of a functions parameters that minimize a cost function as far as possible. It simply measures the change in all weights with regard to the change in error. The model can learn faster with higher gradient descent. It can be thought of climbing down to the bottom of a hill, instead of climbing up. It is a minimization algorithm that reduces a given function.

The equation below describes the working of Gradient Descent: *"b"* describes the next position of climber, while *"a"* represents the current position of climber. The minus sign points to the minimization of gradient descent. The "gamma" is a waiting factor and the gradient term (Δf(a)) is the direction in which the steepest descent will move. [2]

$$b = a - \gamma \nabla f(a)$$

So the above formula basically tells the next position where you need to go, which is the direction of the steepest descent.

1.1.1 Learning Rate

The size of steps that gradient descent takes into the direction of the local minimum is determined by the learning rate. It determines how fast or slowly it will move towards the optimal weights. In order for Gradient Descent to reach the local minimum, appropriate value of learning rate has to be set, which is neither too low nor too high.

If the steps it takes are too big, It will not reach the local minimum because it just bounces back and forth between the convex function of gradient descent. If learning rate is set to a very small value, gradient descent will eventually reach the local minimum but it will maybe take too much time.

If gradient descent is working properly, the cost function should decrease after every iteration.When Gradient Descent can't decrease the cost-function anymore and remains more or less on the same level. The number of iterations that Gradient Descent needs to take can vary. It can take 50 iterations, 1000 or maybe even 10,000.Therefore the number of iterations is hard to estimate in advance.[2]

1.2 Choosing the learning rate

Generally, the value of the learning rate has been chosen manually by hit and trial. Here ,it usually start with a small value such as 0.1, 0.01 or 0.001 and adapt it based on whether the cost function is reducing very slowly (increase learning rate) or is increasing (decrease learning rate). Manually choosing a learning rate is still the most common practice.[3]

Libraries used

The analysis has been performed in Spyder for windows with Anaconda 2.

A. NumPy in Python : NumPy is a general-purpose array-processing package. It provides a multidimensional array object with high performance, and also provides various tools for working with these arrays. It is a basic package for scientific computing using Python. It contains various features including the following:(I)a powerful N-dimensional array object (II) sophisticated (broadcasting) functions, (III) tools for integrating C/C++ and Fortran code and (IV) linear algebra, random number capabilities and fourier transform. Other than its obvious scientific uses, NumPy can be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.[4]

B. matplotlib.pyplot: It is a collection of command style functions that make matplotlib work like MATLAB. Every pyplot function makes a few change to the figure: e.g., creates a figure, creating a plotting area, plots some lines in a plotting area, decorates the plot with labels, etc.

In matplotlib.pyplot some states are kept preserved across function calls, so that it keeps track of things like the current figure and plotting area, and the plotting functions are directed to the current axes (please note that "axes" here and in most places in the documentation refers to the axes of the figure and not the strict mathematical term for more than one axis).[5]

C. NeuroLab: It is a library of basic neural network algorithms with flexible network configurations and learning algorithms for Python.[6]

II. RELATED WORK

H Ramchoun et al introduce a new approach for the optimization of the network architecture, for solving the obtained model the genetic algorithm is used and training of the network is handled with a back-propagation algorithm. The numerical results seem to assess the effectiveness of the theoretical results. Various advantages of the new modeling technique are compared to the previous models. This method is tested to determine the optimal number of hidden layers and connection weights in the Multilayer Perceptron.[8]

L Grippo et al define a wide class of batch learning algorithms for Multilayer Perceptrons (MLP), based on the use of block decomposition techniques in the minimization of the error function. The problem of learning is decomposed into sequential smaller and structured minimization problems in order to exploit the structure of the objective function. A specific algorithm is constructed and evaluated through an extensive numerical experimentation. The comparisons with state-of-the-art learning algorithms show the effectiveness of the proposed method. The biggest feature of these techniques is that in providing, for most of the cases, good generalization performances, with much smaller training times. [9]

III. RESULTS AND ANALYSIS

A range of values from minimum to maximum and calculate the results for a mathematical equation ($y = 5*_{(x)}^2 +11$) where y will be the results and x is the input value range. Learning rate will vary for various observations from 0.1, 0.01 and 0.001.While the training of the data for learning also varies from 1000 to 10,000 with various combinations of learning rate.

CASE 1. Training rate 0.01 epochs 1000

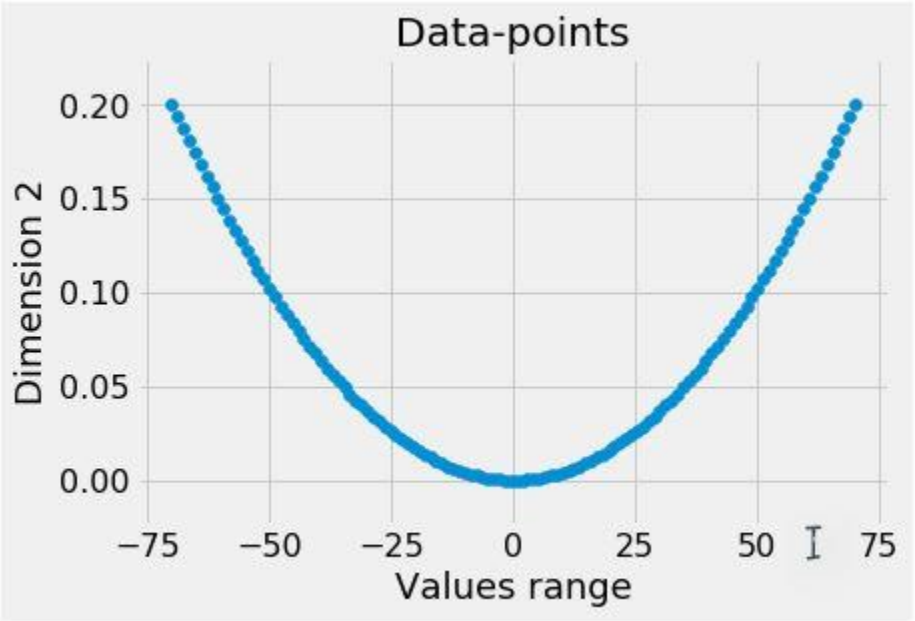

Figure 1.1 : Input value range

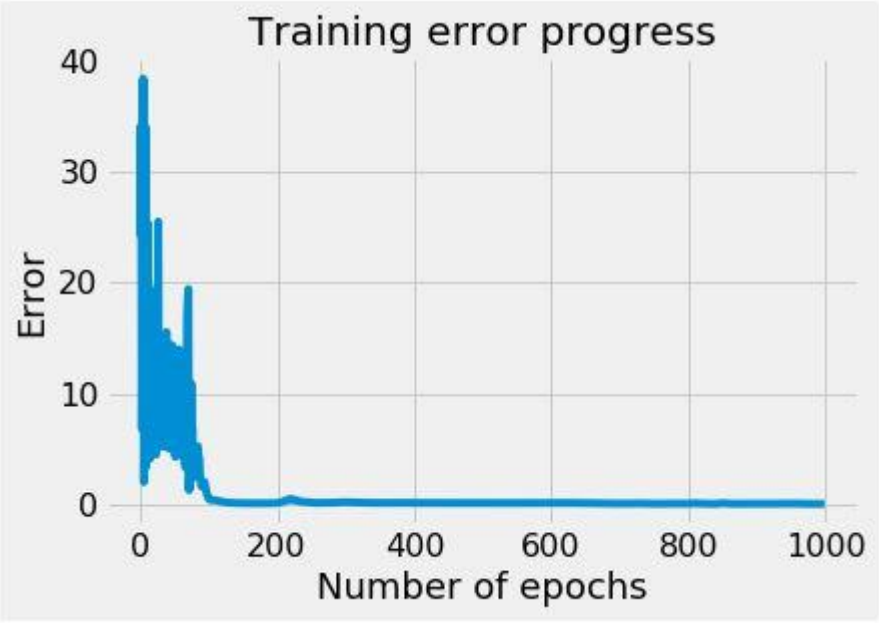

Figure 1.2 : Training error

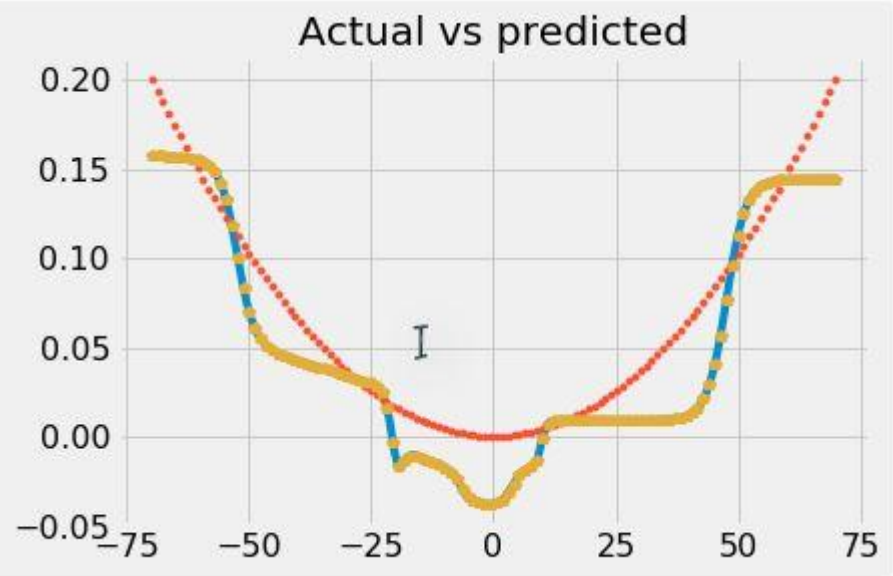

Figure 1.3: Actual vs predicted output graph

Here normal training rate for 1000 training iterations is taken. As it is clear from the above diagrams, there is a certain amount of fluctuation in the predicted results to the actual results.

CASE 2. Training rate 0.1 and epochs 1000

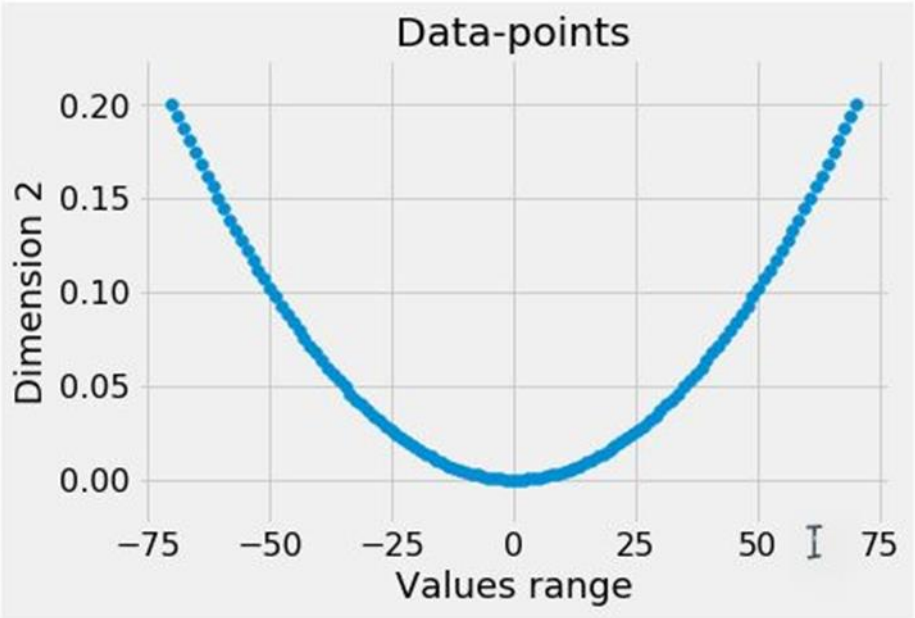

Figure 2.1: Input data range

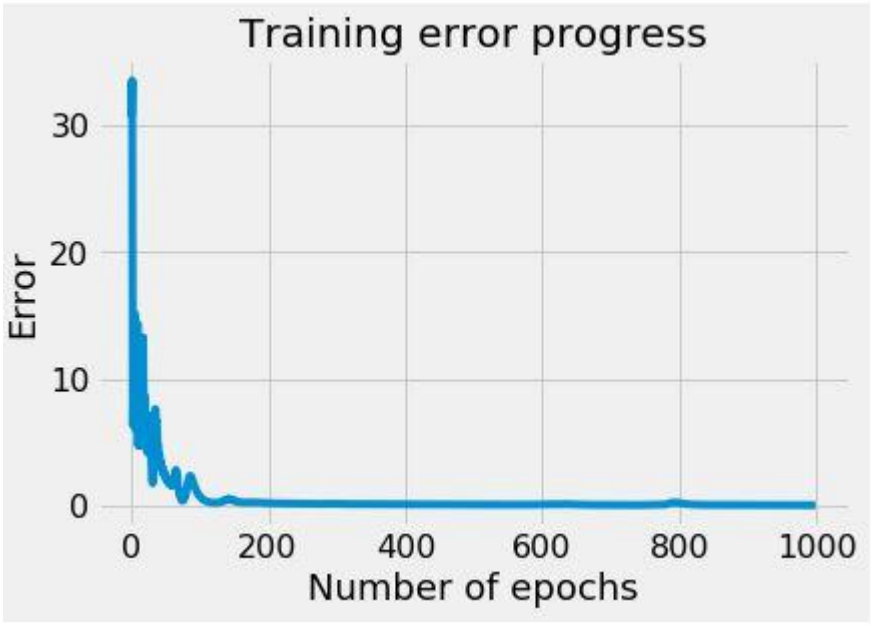

Figure 2.2: Training error

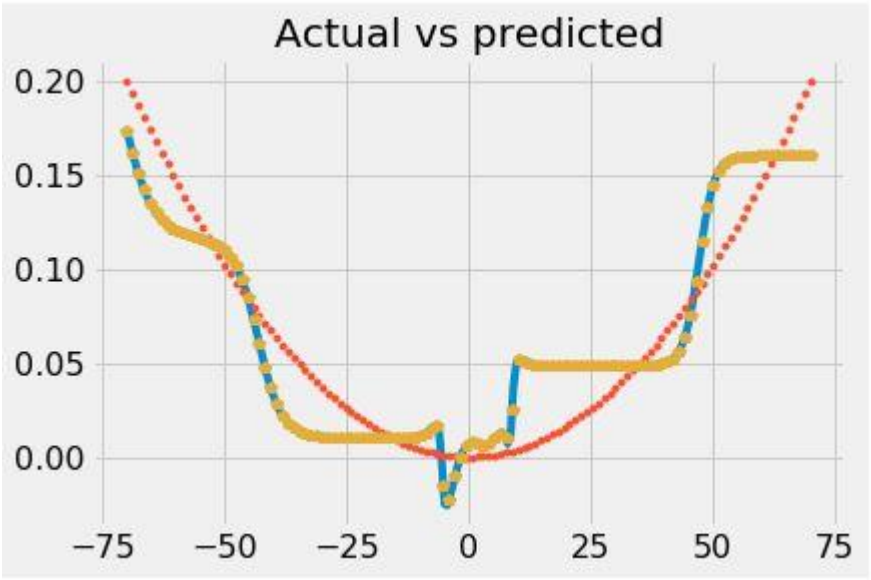

Figure 2.3: Actual vs predicted results

Here the learning rate is increased. The learning rate is high with no change in the training iterations. This algorithm is struggling a lot to predict the results as there is a greater gap in actual and predicted data results.

CASE 3. Training rate 0.001 and epochs 1000

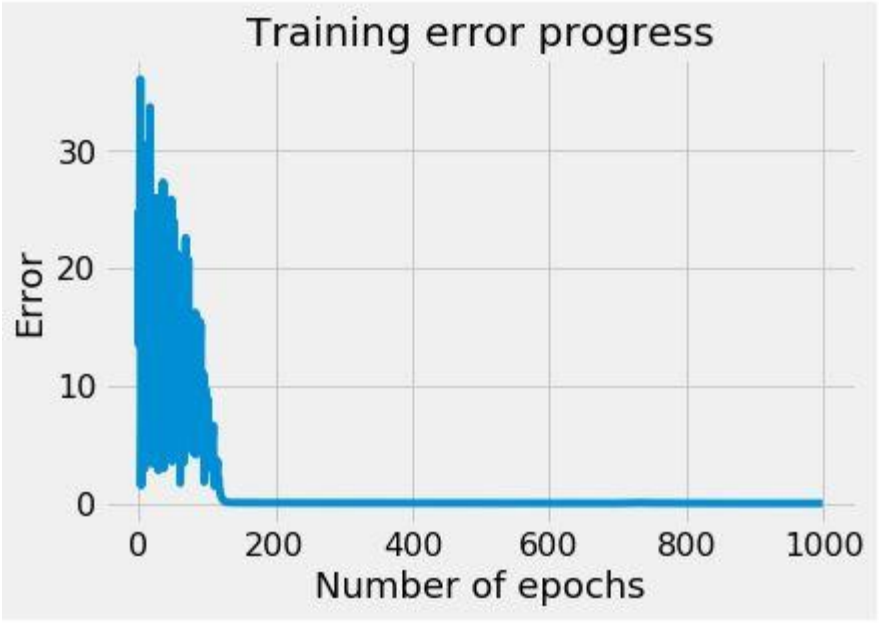

Figure 3.1 : Training error

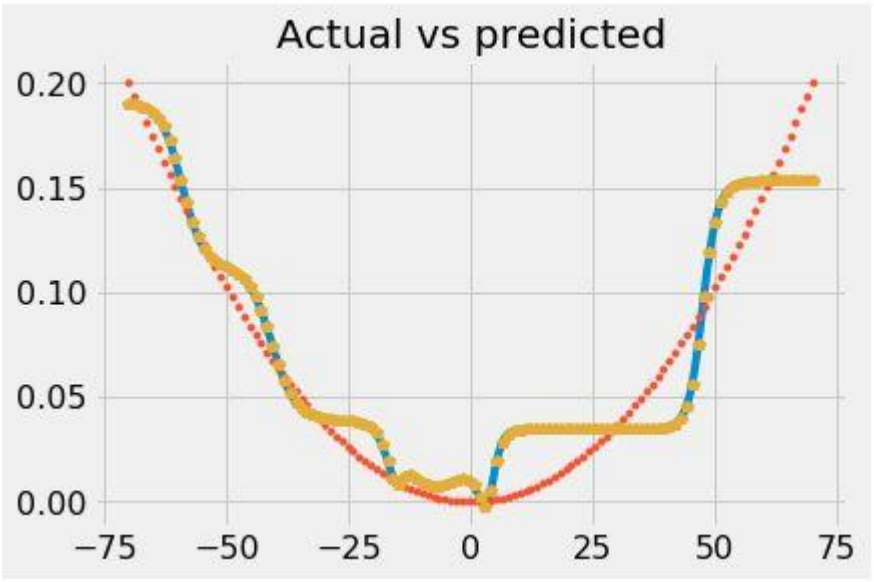

Figure 3.2: Actual vs predicted results

Here the learning rate is decreased. With this rate of learning our predictions are much better than when learning rate is high. From above analysis of actual and predicted results, it is clearly conclude that this algorithm will work better for prediction of results if our training rate is low and our training iterations are high.

CASE 4. Training rate 0.001 and epochs 10000

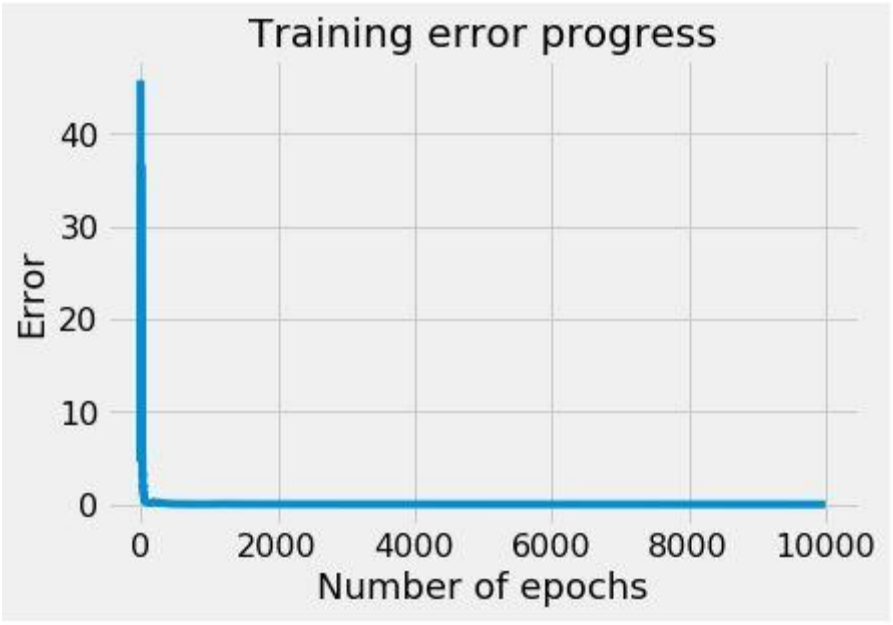

Figure 4.1: Training error progress

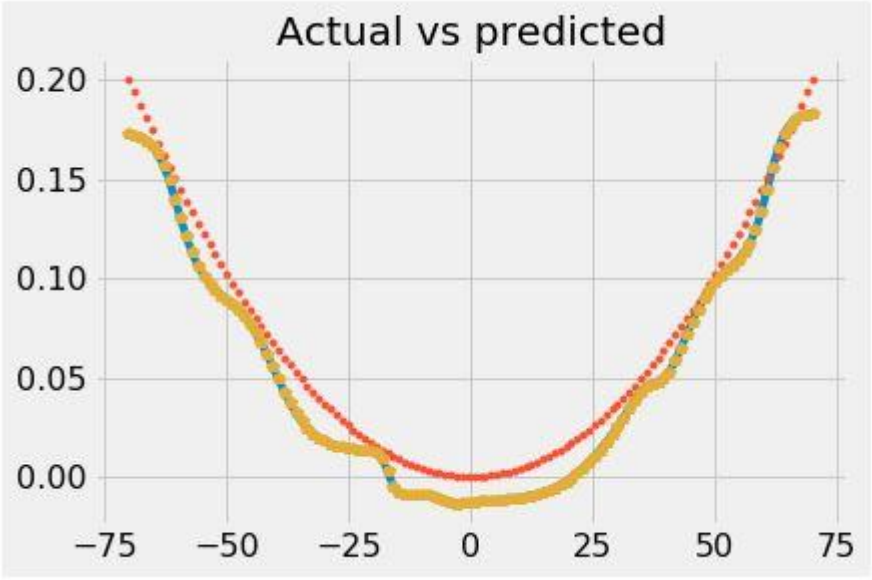

Figure 4.2: Actual vs predicted results

For 10,000 training iterations where the learning rate is low i.e 0.001.The algorithm can predict results almost accurately. If the learning rate is increased the number of training iterations should fall. More accurate results can be recorded with low learning rate and high training iterations, but also it might make the system slower. It is difficult to accurately predict results with high learning rate. As the input data size increases the learning rate can increase.

IVCONCLUSION AND FUTURE SCOPE

Multi-Layer perceptron network is an amazing AI technique for machine learning which can include big data to predict various types of results. The learning rate is very important for

accuracy. It can't be high but it can't be low as well. We can decrease the learning rate but it will take a lot of training time to reach a point where it can be termed as accurate. While if we increase the learning rate, It will take unexpected decisions very soon and the results will fluctuate a lot. With time this technique becomes more and more perfect as new inputs will keep on adding to the previous data thus making it more and more dependent on its own.

Further work is needs to be done for improving the learning rate such that the learning rate could be standardized and there should be confidence in selecting the learning rate for bigger problems. Multi-Layer Perceptron network is a neural network which imitates human brain to forecast results of real world data. As learning rate of human mind is not definite and cannot be recorded precisely, machine learning can be much more perfectly executed if the learning rate can dictate correctly the exact results for certain number of training iterations on data. Further if learning rate can be managed accordingly our training epochs can be decreased and the system can become faster.

References

[1] Jason Brownlee, "Multi-Layer Perceptron Neural Networks" https://machinelearningmastery.com/neural-networks-crash-course/
[accessed on:10 may 2019

[2] Niklas Donges, "Gradient Descent in a Nutshell" https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0, [accessed on:10 may 2019]

[3] *Keshav Dhandhania, et al* " How to understand Gradient Descent, the most popular ML algorithm" https://medium.freecodecamp.org/understanding-gradient-descent-the-most- popular-ml-algorithm-a66c0d97307
[accessed on:12 june 2019]

[4] Numpy in Python

https://www.geeksforgeeks.org/numpy-in-python-set-1-introduction/
[accessed on:21 june 2019]

[5]Pyplot Tutorial : An introduction to the pyplot interface
https://matplotlib.org/tutorials/introductory/pyplot.html

[accessed on:17 june 2019]

[6]Neurolab's documentation

https://pythonhosted.org/neurolab/

[accessed on:19 june 2019]

[7] Bottou, Léon, Frank E. Curtis, and Jorge Nocedal. "Optimizati on methods for large-scale machine learning."*SIAM Review*60.2 (2018): 223-311.
[accessed on:24 june 2019]

[8] Grippo, Luigi, Andrea Manno, and Marco Sciandrone. "Decomposition Techniques for Multilayer Perceptron Training."*IEEE Trans. Neural Netw. Learning Syst.*27.11 (2016): 2146-2159.

[accessed on:16 june 2019]

[9] Large-scale Machine Learning

https://las.inf.ethz.ch/research/large-scale-machine-learning

[accessed on:15 june 2019]