International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 02 | Feb - 2022 ISSN: 2582-3930

Comparison of Effect of learning rate of Neural Network Performance in Deep learning neural networks using the stochastic gradient descent algorithm

Madhav Sharma[1], Vijay Mohan Shrimal [2]
Pushpendra Kumar Sikarwal[3] Siddharth Singh [4]

[1] Assistant Professor, Computer science & Engineering Jagannath University, Jaipur, India
[2] Assistant Professor, Computer science & Engineering Jagannath University, Jaipur, India
[3]Assistant Professor & Ex. HoD Computer science & Engineering JNIT, Jaipur, India

[4] Assistant Professor, Computer science & Engineering Rajiv academy For Technology and Management, Mathura, India

Abstract: The stochastic gradient descent optimization approach is used to train deep learning neural networks in this paper. Artificial neural networks are a subfield of deep learning that uses algorithms inspired by the structure and function of the brain. Deep learning systems are designed to learn feature hierarchies based on the composition of lower level characteristics at the top of the hierarchy. The various sorts of learning models are also discussed. We build a train and dataset with different samples (500, 2000, and 4000) and change the Epochs value in this research (100, 300 and 400).We also change the learning rate for different result. We Test the accuracy of learning rates. Classification accuracy on the training dataset is marked in blue, whereas accuracy on the test dataset is marked in orange. In This paper we find the best learning rate for good performance on the train and test sets.

Keywords: Deep Learning, Epochs, Learning rate, Optimization

Introduction:

Artificial neural networks are a subfield of machine learning that is concerned with algorithms inspired by the formation and purpose of the brain. Deep learning algorithms are used to learn feature hierarchies, which are made up of features from higher levels of the hierarchy that are composed of lower level features. The ability to learn complicated functions mapping the input to the output directly from data by automatically learning features at several levels of abstraction allows a system to learn complex functions mapping the input to the output without relying entirely on human-crafted features..

Deep learning, as opposed to task-specific algorithms, is a broader family of mechanism learning methods based on learning data demonstrations. There are two types of learning: supervised and unsupervised. It is a set of machine learning methods for learning several levels of representation, which correspond to various layers of abstraction and aid in data interpretation. To calculate nonlinear functions with highly complicated data, many layers are used. Each layer receives data from the previous layer, calculates and alters it, and then delivers it to the subsequent layersDepending on the type of network, each layer consists of neurons with various modes of connections to other neurons in the same layer as well as those in other levels. Deep learning is based on the use of brain simulations to aid in the development of more efficient learning algorithms and significant improvements in machine learning and artificial intelligence. With the advancement of modern technology and the ease with which it may be implemented, deep learning is gaining more attention these days.

Stochastic Gradient Descent:

On really big datasets, gradient descent can be slow. When there are several millions of examples in the training dataset, one iteration of the gradient descent technique requires a prediction for each occurrence in the training dataset. When you have a lot of data, you can utilise stochastic gradient descent, which is a version of gradient descent. Batch algorithms that employ the entire training set to compute the next update to parameters at each iteration, such as limited memory BFGS, converge very effectively to local optima. They're also simple to set up if you have a competent off-the-shelf solution because there are only a few hyper-parameters to adjust. However, if the dataset is too large to fit in main memory, computing the cost and gradient for the full training set can be very difficult and intractable on a single machine. Another disadvantage of batch optimization approaches is that they do

not provide a convenient way to incorporate fresh data in a 'online' environment. After witnessing only a single or a few training instances, Stochastic Gradient Descent (SGD) tackles both of these concerns by following the negative gradient of the objective. The application of SGD The high cost of executing back propagation over the entire training set motivates it in the neural network context. SGD can overcome this barrier while still achieving rapid convergence.

Stochastic Gradient Descent

The standard gradient descent algorithm updates the parameters θ Of the objective $J(\theta)$

as, $\theta=\theta-\alpha\nabla\theta E[J(\theta)]$

The above equation's expectation is estimated by considering the cost and gradient across the entire training set. Stochastic Gradient Descent (SGD) essentially removes the expectation from the update and computes the parameter gradient using only a few or a few training samples. The new information is provided by, $\theta=\theta-\alpha\nabla\theta J(\theta;x(i),y(i))$

with a pair $(x(i),y(i))$

from the training set.

Rather than a single example, every stricture update in SGD is usually computed based on a few training examples or a micro batch. The purpose for this is twofold: first, it reduces parameter update variance, which can lead to more stable convergence; second, it allows the computation to take advantage of highly efficient matrix operations, which should be used in a well vectorized cost and gradient computation. Although the best potential size of the mini batch can vary for different applications and architectures, a typical mini batch size is 256.

Because there is substantially more variance in the update, the learning rate in SGD is often much lower than an equivalent learning rate in batch gradient descent. Choosing the right learning rate and schedule might be difficult. One common approach that works well in practise is to start with a small enough stable learning rate that gives steady convergence in the first epoch or two of training, then halve the learning rate when convergence slows. Even better, examine a held out set after each epoch and anneal the learning rate when the difference in intent between epochs is less than a tiny threshold. This tends to provide a local optima good convergence. Another typical method is to anneal the learning rate at each iteration t as ab+t, where a and b are the learning rates. Set the initial learning rate and the start of the annealing process. To discover the most beneficial update, more advanced methods include employing a backtracking line search.

The sequence in which we deliver the data to the algorithm is a final but crucial detail about SGD. If the data is presented in a relevant manner, the incline may be skewed, resulting in poor convergence. A good way to avoid this is to shuffle the data at random before each epoch of training.

Learning Rate:

The step size, often known as the "learning rate," is the amount by which the weights are updated through training. The in sequence of the lowercase Greek letter eta is frequently used to symbolize the learning rate (n). The back-propagation of fault estimates the amount of error for which the weights of a node in the classification are responsible during training. Instead of updating the weight with the full quantity, the learning rate is used to scale it.

Effect of Learning Rate

From examples in the training dataset, a neural system learns or approximates a function to optimally map inputs to outputs. The learning rate hyper constraint regulates how quickly the representation learns. It regulates the total

apportioned error with which the replica's weights are updated each time they are updated, such as at the end of each batch of training instances. In a certain amount of training epochs, given a well configured learning rate, the model will learn to best approximate the function given available resources (passes through the training data).

A high learning rate allows the model to learn more quickly, but at the cost of a sub-optimal final set of weights. A slower learning rate may allow the model to learn a more optimal or even globally optimal set of weights, but training will take much longer. A learning rate that is excessively high will result in too big weight updates, and the model's performance (such as its loss on the training dataset) will swing throughout training epochs. Weights that diverge are thought to be the reason of oscillating performance (are divergent). A slow learning rate may never converge or become trapped on an inferior solution.

Algorithm for Learning Rate:

Step 1: Create a function to create the problem's samples and divide them into train and test datasets.

Step 2: Encode the target variable so that we can build a model that predicts the likelihood of each class being represented by an example.

Step 3: Create a function to implement this behavior, which will return train and test sets separated into input and output parts.

Step 4: Create a function that can be used to fit and assess an MLP model.

Step 5: We'll utilize the stochastic gradient descent optimizer now, and we'll need to specify the learning rate so that we may compare different rates. Cross entropy will be minimized by training the model.

Step 6: Use a little trial and error to find training epochs, and the test set will be used as the validation dataset so we can gain an understanding of the model's generalization error during training.

Step 7: Over the training epochs, plot the model's accuracy on the train and test sets.

Step 8: Using the train and test datasets as well as a specific learning rate to evaluate, fit a model and visualize its performance.

Step 9: Investigate the dynamics of various learning rates on the train and test the model's correctness.

Result and Parameters: First Experiment

S.No	Elements	Value
1	Samples	500
2	Centers	3
3	features	2
4	Cluster	2
5	Random state	2
6	Train	250
7	Epochs	100
8	Learning Rate	1.0 to 1E-7

Resultant Graphs

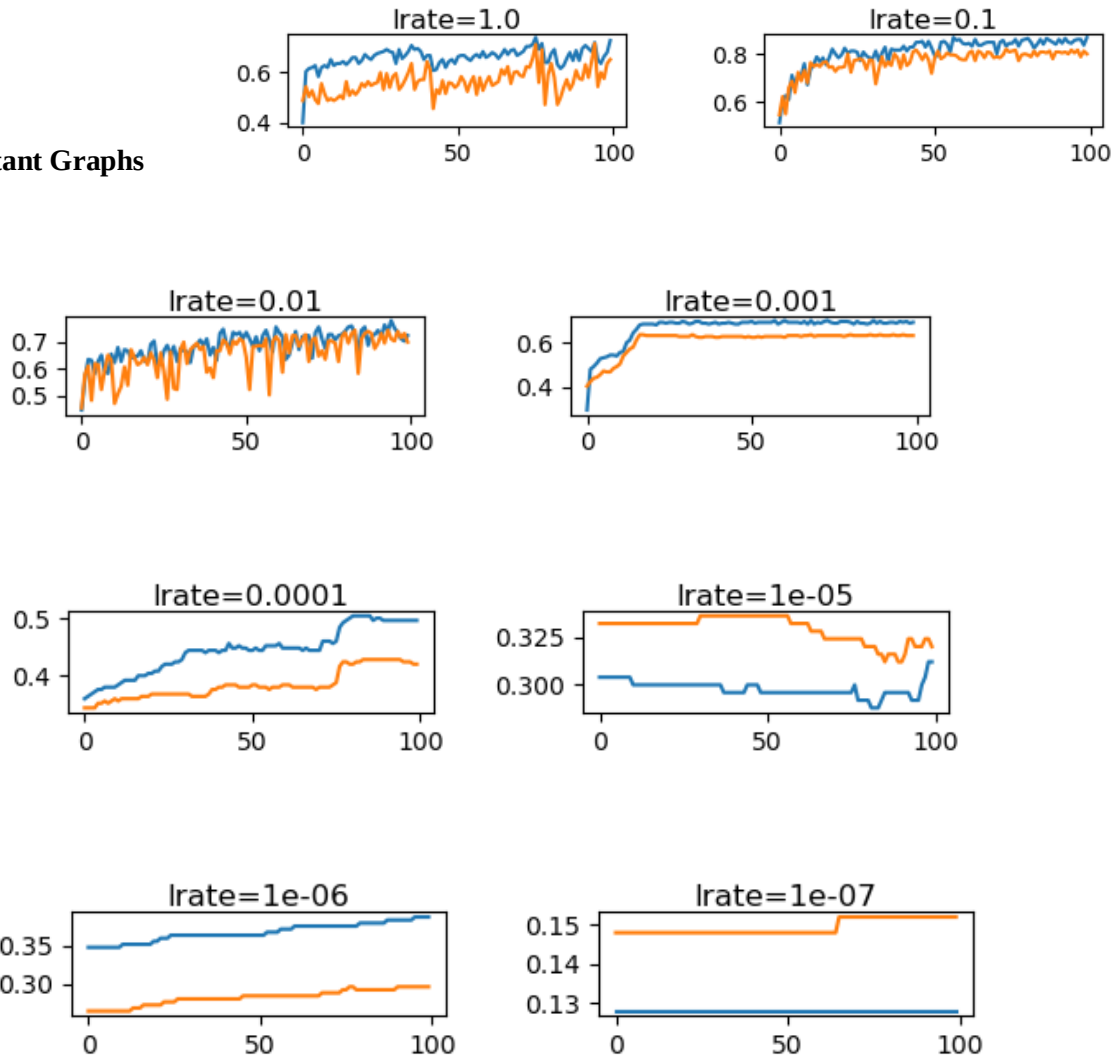

Fig 1: Line Plots of Train and Test Accuracy for Learning Rates

figures that contains eight line plots for the eight different evaluated learning rates. Classification accuracy on the training dataset is marked in blue, whereas accuracy on the test dataset is marked in orange.

The plots show oscillations in behavior for the too-large learning rate of 1.0 and the inability of the model to learn anything with the too-small learning rates of 1E-6 and 1E-7.

We can see that the model was able to learn the problem well with the learning rates 1E-1, 1E-2 and 1E-3, although successively slower as the learning rate was decreased. With the chosen model configuration, the results suggest a moderate learning rate of 0.1 results in good model performance on the train and test sets.

Experiment 2:

S.No	Elements	Value
1	Samples	2000
2	Centers	3
3	features	2
4	Cluster	2
5	Random state	2
6	Train	1000
7	Epochs	300
8	Learning Rate	1.0 to 1E-7

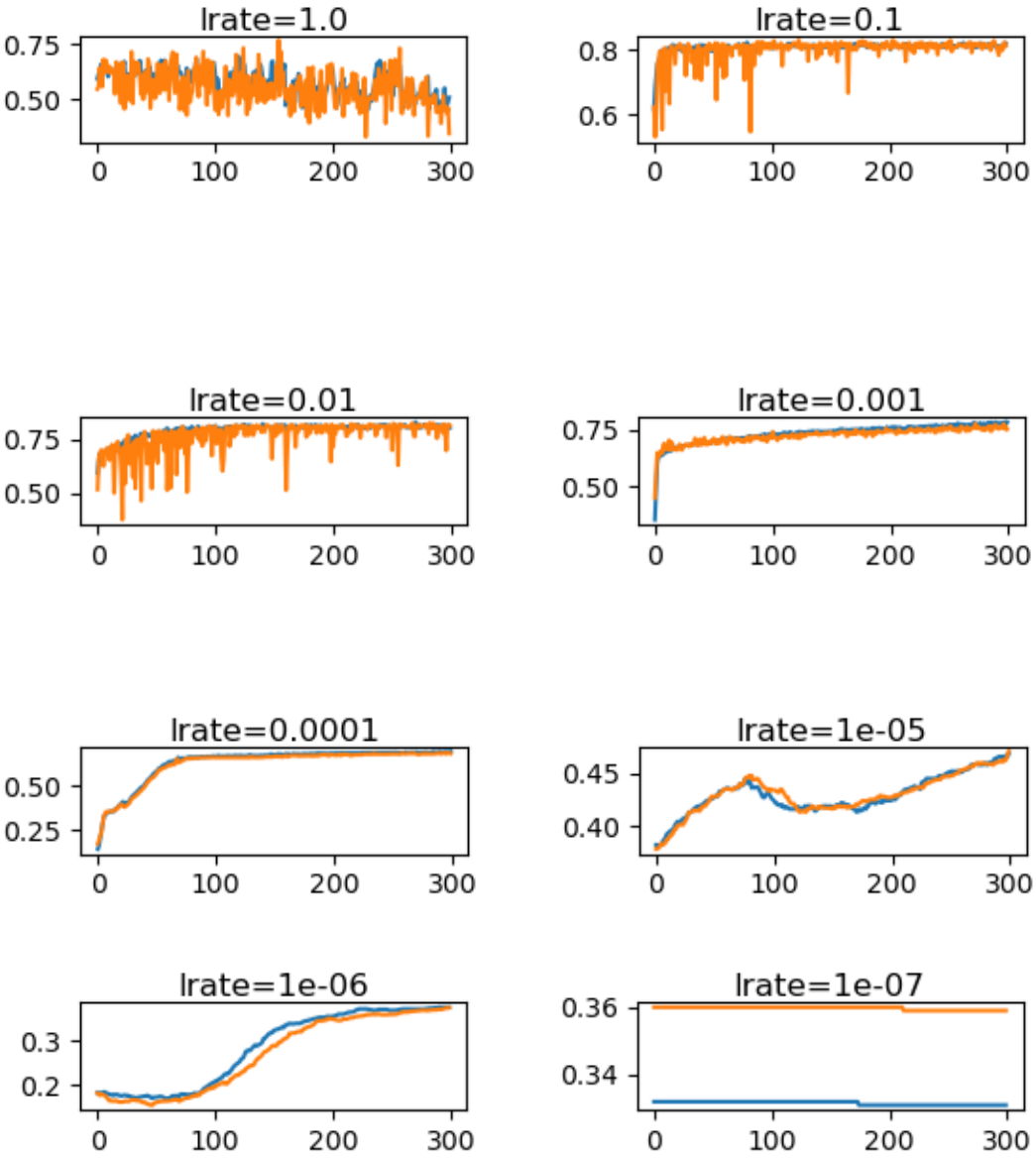

Fig 2: Line Plots of Train and Test Accuracy for Learning Rates

Experiment 3:

S.No	Elements	Value
1	Samples	4000
2	Centers	3
3	features	2
4	Cluster	2
5	Random state	2
6	Train	2000
7	Epochs	400
8	Learning Rate	1.0 to 1E-7

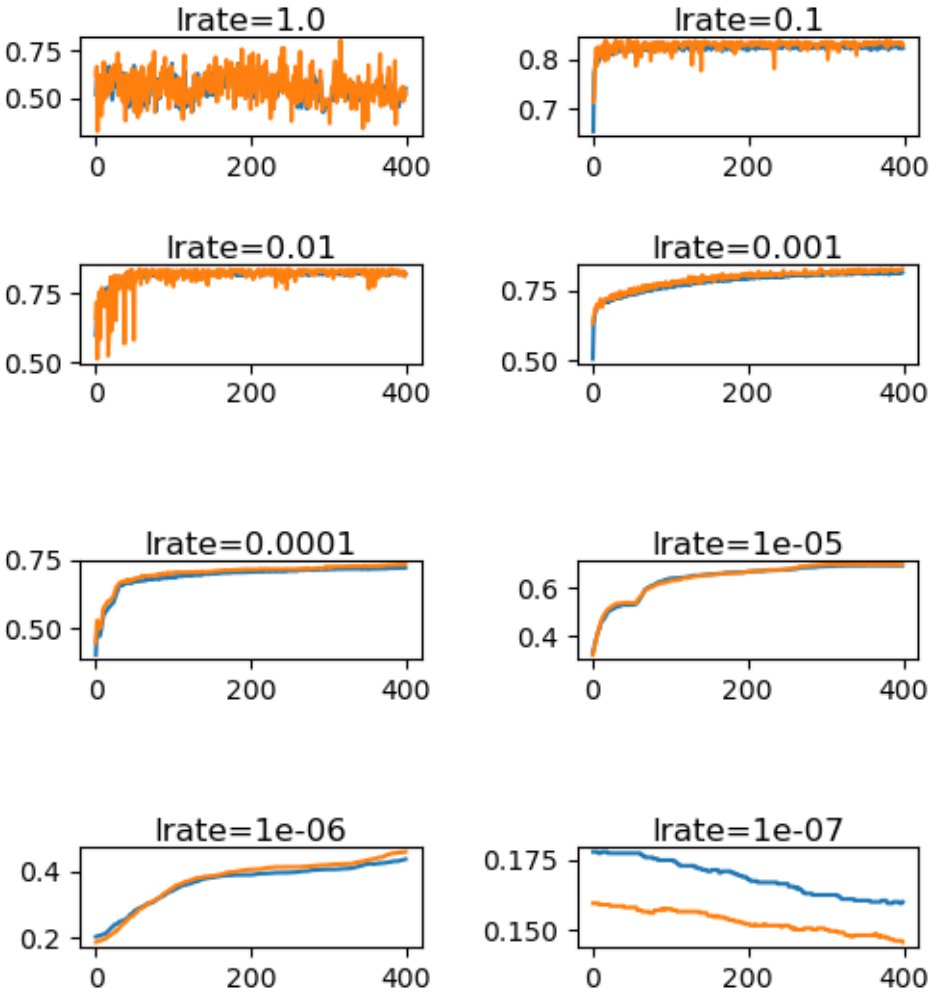

Fig 3: Line Plots of Train and Test Accuracy for Learning Rates

Conclusion:

In this paper we take the three comparisons and changes the values of Samples and Epochs. Result will be changes and learning rate varies according to Epochs. In comparison the results suggest a moderate learning rate of 0.1 results in good model performance on the train and test sets.

References:

[1] J. Mantas, \An overview of character recognition methodologies," Pattern Recognition, vol. 19, no. 6, pp. 425 { 430, 1986.
[2] T. S. El-Sheikh and R. M. Guindi, \Computer recognition of arabic cursive scripts," Pattern Recognition, vol. 21, no. 4, pp. 293 { 302,
[5] C. Tappert, C. Suen, and T. Wakahara, \The state of the art in online handwriting recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 12, pp. 787 {808, Aug. 1990.
[6] R. Bozinovic and S. Srihari, \O_-line cursive script word recognition," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 11, pp. 68 {83, Jan. 1989.
[7] V. Govindan and A. Shivaprasad, \Character recognition { a review," Pattern Recognition, vol. 23, no. 7, pp. 671 { 683, 1990.
[8] Q. Tian, P. Zhang, T. Alexander, and Y. Kim, \Survey: omnifont-printed character recognition," in Society of Photo-Optical Instru- mentation Engineers (SPIE) Conference Series (K.-H. Tzou & T. Koga, ed.), vol. 1606 of Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, pp. 260{268, Nov. 1991.