2021 IEEE Pune Section International Conference (PuneCon)
MIT ADT University, Pune, India. Dec 16-19, 2021

Effect of Hyperparameters on Backpropagation

Aaditree Jaisswal
Computer Engineering
M.K.S.S.S's Cummins College of Engineering for Women
Pune, India
aaditreejaisswal06@gmail.com

Anjali Naik
Computer Engineering
M.K.S.S.S's Cummins College of Engineering for Women
Pune, India
anjali.naik@cumminscollege.in

***Abstract*—In machine learning algorithms, parameters and hy-perparameters are important properties in the training process. Parameters are modified through machine learning algorithms while hyperparameters are the parameters that are adjusted manually to achieve the desired accuracy and increase efficiency. In neural networks, weights are parameters while hyper-parameters include layer size, momentum, learning rate, acti-vation function family, weight initialization and normalization scheme for input data.**

Multi-layer feed-forward Backpropagation neural network (BPN) has been used to solve a variety of classification problems as it can map "any" input-output mapping and classify linearly inseparable data. The research work presented in this paper gives a detailed description of BPN and explains various hyper-mutates for BPN. The paper describes the implementation of BPN, experimentation and analysis with hyperparameters like weight initialization and learning.

***Index Terms*—backpropagation, feed forward, neural network, parameters, hyperparameters, weights, learning rate, algorithm**

I. INTRODUCTION

Backpropagation (BPN) is a multi-layer feed-forward neural network which can theoretically perform "any" input-output mapping and can learn to solve linearly inseparable problems. The learning in BPN is the same as that for a perceptron and based on gradient descent where the function is differentiated. Error at a higher layer of a multi-layer network is propagated back to the nodes at all the layers. There are many use cases for backpropagation involving a combination of algorithms and parameter tuning for optimized results. The following few papers focus on the application of Backpropagation (BPN) in various domains.

The paper has been divided into two parts. In the first part, the papers written by different researchers have been reviewed, who have made use of backpropagation in different applications. In the second part, the different hyperparameters of backpropagation have been discussed.

There hasn't been a lot of focus on the effect of multiple hyperparameters being applied to neural networks and its advantages and this papers works to showcase the same.

II. LITERATURE REVIEW

The paper written by Ryan Hsiao et al.[5] proposed a hybrid approach to initialize the weights and hidden nodes in the backpropagation network. This approach used a combination of the original backpropagation network algorithm and the partial least squares algorithm and was implemented through software written in LabVIEW. A comparison was also made between the original algorithms, Backpropagation (BPN) and Partial Least Squares (PLS), and the hybrid algorithm to show the effectiveness of the latter in the approach. This was done by proving that the latter has the smallest root mean square error (RMSE) in comparison to the rest. This proved that it performed better due to the influence of the number of hidden nodes on the RMSE value. This algorithm also solved some conventional problems of the BPN method concerning the initial weights, the calibration time and the number of hidden nodes, while delivering an optimal solution.

The paper written by Nadir Murru et al. [10] proposed a solution for a better convergence rate in backpropagation. The algorithm used for the initialization of weights in this was created by using a customised Kalman filter along with the original backpropagation algorithm. The training dataset used was the MNIST dataset, with 60,000 handwritten digits. The results in the simulation trials showed that the algorithm had an accuracy of 99% when used in character recognition.

The paper written by T. Kathirvalavakumar et al. [6] proposed a new efficient procedure for training a single hidden layer feed-forward network. In this method, the output layers and hidden layers were trained separately, rather than at the same time. The layers were trained using the standard backpropagation algorithm and an optimization criterion respectively. The results, after training, showed that the accuracy level was high once the convergence level was reached in terms of the time and hidden nodes in the network. This method also proved efficient in terms of classification problems and nonlinear function approximation problems, if used in smaller networks.

The paper written by Sarfaraz Masood et al. [9] compared nine different weight initialization methods by using six different function approximation problems which could be used for their analysis over gradient descent approach using momentum algorithm, to find the best-suited method. Four of these problems are a part of the function approximation in [3]. The other two are MATLAB sample files, peaks.m and humps.m which were also used in [14] and [8]. The work used a multi-layer feed forward network with a tangent hyperbolic activation function and a Backpropagation algorithm for training it with momentum considered as a factor while updating the network weights. Thirty separate sets of weights

2021 IEEE Pune Section International Conference (PuneCon) | 978-1-6654-4479-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/PUNECON52575.2021.9686489

were considered for each method for each problem for statistical purposes. Tests such as the one-sided tailed t-test, the standard deviation of simulation error and their mean were considered as decision making parameters for comparison of the six methods. The final results showed that the best suited method was the one proposed by Nguyen and Widrow, which used the method of initialization of adaptive weights, due to their least value in simulation error. Some of the other methods which were included in this experiment were Random Initialization, Partially Deterministic weight initialization and Interval based weight initialization.

The paper written by Timothy P. Lillicrap et al. [7] worked towards understanding the effect of changes occurring in synapses, while learning, and their effect on an individual's behaviour. It implemented the standard backpropagation algorithm to understand the effects of this change by learning about it through every update made in the synapses. This was done by using feedback connections to continuously send error signals, where it gave good input representations through learning. The training was done through the raw sensory input where it generates an error, and then, this was fed back as input to the system. The results showed the learning process in the cortex.

The paper written by Shyla Afroge et al. [1] designed an ANN-based approach for the recognition of English characters. It used the backpropagation algorithm for training the model with nonlinear thresholding function as its learning method. The training dataset consisted of images of 62 different characters. The accuracy for individual classification of character sets, like A-Z, a-z and 0-9, was high but this went down when the different character sets were combined.

The paper written by Shrinivas R. Zanwar [15] evaluated the performance of the character recognition system. The backpropagation algorithm was used in both training and testing the model, using independent component analysis for feature extraction, and swarm intelligence for feature selection. The training dataset used for this was the Chars74K dataset containing 64 classes having characters (0-9, A-Z, a-z). The results showed that this algorithm provided a high rate of classification and simulated the network during character recognition and could be used for various languages.

The paper written by N. Venkata Rao et al. [12] proposed a new method for optical and handwritten character recognition. It compared a set of existing algorithms and then proposed a modified backpropagation method using moment invariant as the feature to recognize the characters. The training dataset used was designed by the authors, in MS Paint 6.1 with Arial font, size 28 with a dimension of image kept as 31X39. The results showed a 100% accuracy in character recognition.

The paper written by Shrinivas R. Zanwar et al. [16] proposed a hybrid optimization approach to automate character recognition. It used component analysis, hybrid PSO and firefly optimization for optimization of the network. Backpropagation algorithm was used after optimization for classification of the characters. The training dataset used was the MNIST dataset. The researchers achieved high precision, low error rates and increased sensitivity for high rates of classification.

The paper written by Swapnil Desai et al. [13] delivered an algorithm for error-free recognition of text from images. It used a template matching along with the backpropagation algorithm to achieve this. The training dataset consisted of a set of blurred, smeared and broken character images captured via a camera. The results showcased an accuracy of 91.82% after the implementation of the algorithm.

The paper written by Mamata Nayak [11] proposed a complete model to recognize the characters of Odia language. It used a binary feature extraction technique followed by a modified backpropagation algorithm to achieve optimal classification. The training dataset used was a plain script consisting of 335 symbols of the Odia script. The results showed successful recognition of printed Odia characters and stated that the proposed method could be applied to handwritten characters as well.

The paper written by Shilpa Gite et al. [4] observes the use of backpropagation in the paper written by Peipei Zhang [17} and studies the effect of different number of hidden layers on the performance of Artificial Neural Networks, for the prediction of driver activity using Spatio-temporal data.

Hence we can say that through these papers, we have discussed various implementations involving the use of the backpropagation algorithm. Now, we will be talking about the different hyperparameters that lead to the optimization of the backpropagation algorithm.

III. HYPERPARAMETERS

A model has two kinds of coefficients; parameters and hyper parameters. Parameters are those coefficients whose values change according to the algorithm via learning as it gets optimized, to reduce errors. On the other hand, hyper parameters are the coefficients of the model that are not updated by the algorithm and manual intervention is needed depending on the optimization strategy. [2]

Initialization and decision of optimal hyper-parameter values is crucial for getting higher accuracy in intelligent models like BPN. Usually, the values are continuously changed till we find the best ones through trial and error.

There are six hyper parameters used in the backpropagation algorithm:

1) Initial Weights
2) Learning Rate
3) Momentum Factor
4) Generalization
5) Number of Training Data

A. Number of Hidden Layer Nodes Initial Weights

The ultimate solution and network convergence is affected by the initial weights of a multi-layer feed-forward network. If the initial weights are very high then sigmoidal function gets saturated in the beginning and the system may stick at local minima or flat plateau at the starting point itself.

B. Learning Rate

Learning Rate, with values ranging from 0 to 1, refers to that step of backpropagation which affects the updating of weight during the training process.

A small Learning Rate will allow slow progress in training and the neural network convergence will be very slow. This causes the problem of 'Vanishing Gradient'.

In Gradient based learning, weight adjustment is proportional to partial derivative of error function. Vanishing Gradient is a problem when this gradient (weight change) is vanishingly small or completely stopped.

A large Learning Rate will make the learning jump over minima and let the neural network undergo the problem of 'Exploding Gradient'.

In the Exploding gradients problem, large error gradients result in very large updates to neural network model weights during training and make the model unstable and unable to learn from training data.

C. Momentum Factor

Momentum Factor is added in the weight updating process for faster convergence. This factor helps in large weight adjustments and hence used in the backpropagation network. It should be noted that for using this momentum factor, the weights of the previous training patterns must be saved. The momentum factor is added with the normal gradient descent method in order to have a larger learning rate without oscillations.

The Momentum factor is denoted by, ,ranging from 0 to 1, and the value of 0.9 is often used for the momentum factor.

$$w_{jk}(t+1) = w_{jk}(t) + \underbrace{\alpha\delta_k z_j + \eta\,[w_{jk}(t) - w_{jk}(t-1)]}_{\Delta w_{jk}(t+1)}$$

$$v_{ij}(t+1) = v_{ij}(t) + \underbrace{\alpha\delta_j x_i + \eta\,[v_{ij}(t) - v_{ij}(t-1)]}_{\Delta v_{ij}(t+1)}$$

Fig. 1. Weight Updation Formula

Here, w is the current weight, is the learning rate, is the error portion between input layer and hidden layer, t and z are the layers, and w and v are the changes in weights between two layers.

D. Generalization

When trainable parameters are more for the given amount of training data, training is well done but the network shows overtraining or overfitting. One solution to this problem is to monitor the error on the test set and terminate the training when the error increases.

On the other hand, for a small number of trainable parameters, the network fails to learn the training data and performs poorly. For improving the generalization of the network, small changes are made to the input space of the without altering the output components. A net with a large number of nodes is capable of memorizing the training set at the cost of generalization. Hence the back propagation network is the best network for generalization.

E. Number of Training Data

The training data should be sufficient and proper and should cover the entire expected input space.

The training-vector pairs should be selected randomly from the set. Assume that the input space is linearly separable into "L" disjoint regions with their boundaries being part of hyper planes.

Let "T" be the lower bound on the training parameters, then choose T such that T/L ¡ ¡ 1:

Also in some cases, scaling or normalization has to be done to help learning.

F. Number of Hidden Layer Nodes

For multi-layer feed-forward neural networks, calculations are performed for every hidden layer, thus, size of hidden layer is very important. The size of the hidden layer is determined experimentally. If the network does not converge to a solution, then increase hidden layer size.

When the network converges to a solution, try with a lesser number of nodes and decide the size of the hidden layer based on overall system performance.

IV. METHODOLOGY

In machine learning, intelligence is induced in models by making that model learn various parameters for specific application. This process is called training the model for that application with available data set. Data set is divided into training, testing and validation.

The methodology is divided into two parts. In the first part, the standard backpropagation algorithm has been explained with respect to the Forward and Backward Phase. In the second phase, the backpropagation algorithm that has been used in the research work, has been explained, with the algorithm being coded in MATLAB.

Lets start with the first part of the methodology.

In Backpropagation, training is performed in two phases:

Phase I [Forward Phase] and Phase II [Backward Phase]

A. Phase I [Forward Phase]

Firstly, a training input pattern is presented to the network input layer.

The network will propagate the input pattern from layer to layer until the output layer generates the output pattern.

B. Phase II [Backward Phase]

After that, if this pattern is different from the desired output, an error is calculated. The error is then propagated backwards through the network from the output layer to the input layer. The weights are

modified as the error gets propagated. This process repeats till the optimum values are found. Thus the terminating condition for training is achieved. This condition could be the number of epochs, error reaching to some negligible value or permissible value.

After training is over, the backpropagation model is validated by a validation data set. The model is tested using a test data set when the required accuracy is achieved in validation.

Now, in the second part of the methodology, the backpropagation implementation used in the research work will be explained.

BACKPROPAGATION ALGORITHM IN MATLAB

The standard algorithm for backpropagation is as follows:

1. Initialize the number of hidden neurons and the maximum number of epochs for which the algorithm can run before displaying the optimal values for initial weights.
2. Set the training input and output values.
3. Standardise (Normalize) the training input and output values by using the following formula:
data = (data (:,:) - mean (:, 1)) / std (:, 1)
, where mean and std are the mean and standard deviation of the data respectively. data' here will be replaced by training inputs and training outputs.
4. Add a bias to all the training inputs. Set its value to 1.
5. Use the random function to set a random value for learning rate in the range 0.01 and 0.1. Different values for learning rate will give different results.
6. Use the random function to initialize the weights for the algorithm. This will keep changing according to the error generated in each loop.
7. Run a loop for the number of epochs.
7.1. For each epoch, run a loop for the total number of patterns and choose a random pattern each time.
7.2. For each pattern, calculate the error generated and adjust the input weights.
7.3. Continue this process till the error generated becomes too small i.e. less than 0.001.
8.Once this condition becomes true, the loop will stop running and display the output weights.
9. End.

In this backpropagation algorithm, the neurons have been initialized with random weights, their 'x' values and the bias for the algorithm. The learning rate for this algorithm have also been randomly initialized for the weight updation. Using the initial weights, the error generated is found by the algorithm by comparing the output values to the target values and backpropagating it to update the values of the initial weights. This backpropagation continues till the error becomes zero or becomes negligible in comparison. At this point, it is found that the output values are equal or almost equal to the target values and the final weights are found.

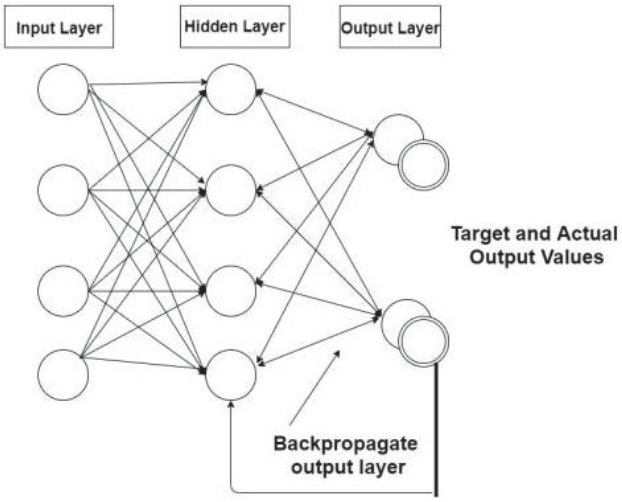

Fig. 2. Backpropagation

V. RESULT AND DISCUSSION

We have performed standard backpropagation on simple training inputs and outputs, on MATLAB, as shown below:

Hidden neurons	Training inputs	Training outputs	Learning Rate
3	[1 1; 1 0; 0 1; 0 0]	[1; 0; 0; 0]	0.01
1	[0.05;0.1]	[0.01;0.99]	0.069602

Fig. 3. Training inputs and target outputs

The backpropagation algorithm chooses a random value for the learning rates for the training input and outputs in the range 0.01 to 0.1, and it finds the optimal weights to generate the outputs while minimising the error generated. The number of epochs, here, have been specified at 10000 and as soon as the optimal values for weights are found, the algorithm stops running and displays the output. For the values given above, the following optimal weights are generated and the epoch number at which these values were found.

In the first example, the optimal results are as follows:

In the second example, the optimal results are as follows:

The algorithm starts with random weights and accordingly changes with every epoch. Once an optimal value with minimum error has been reached, it displays the results.

The below diagram shows the physical working of the algorithm and the blue line depicts the error between the actual and targeted values.

converged at epoch Number: 645

input hidden weight : -1.719761
input hidden weight : -1.786416
input hidden weight : 1.459730
input hidden weight : -0.436778
input hidden weight : 0.480512
input hidden weight : -1.010747
input hidden weight : 0.281471
input hidden weight : -0.355126
input hidden weight : -0.714947
output hidden weight : -1.033381
output hidden weight : -0.396899
output hidden weight : -0.388452

Fig. 4. Training results for (1)

converged at epoch Number: 127

input hidden weight : 1.052168
input hidden weight : 0.000971
output hidden weight : 1.119222

Fig. 5. Training results for (2)

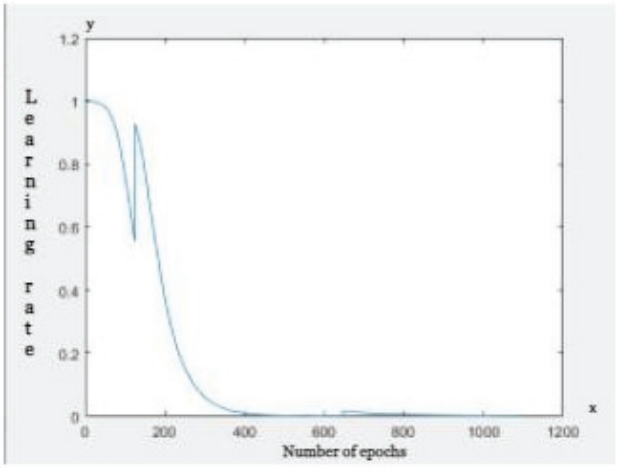

Fig. 6. Implementation of the algorithm

VI. CONCLUSION

The results show that the combination of applying multi-ple hyperparameters, in this case, the learning rate and the initialization of weights, have a great impact on the speed and efficiency of the backpropagation algorithm. Furthermore, finding the optimal values of the hyperparameters can result in optimal solutions.

The future scope consists of handling the learning rate to find the optimal value by changing the range of the values that the learning rate can occur in, till it can no longer be reduced.

REFERENCES

[1] Shyla Afroge, Boshir Ahmed, and Firoz Mahmud. Opti-cal character recognition using back propagation neural network. In 2016 2nd International Conference on Electrical, Computer & Telecommunication Engineering (ICECTE), pages 1–4. IEEE, 2016.

[2] YY BAYDILLI and U ATILA. Understanding ef-fects of hyper-parameters on learning: A compara-tive analysis. In International Conference on Ad-vanced Technologies, Computer Engineering and Science (2018). http://indexive. com/uploads/papers/icatces2018-175. pdf, 2018.

[3] Vladimir Cherkassky, Don Gehring, and Filip Mulier. Comparison of adaptive methods for function estimation from samples. IEEE Transactions on Neural Networks, 7(4):969–984, 1996.

[4] Shilpa Gite and Ketan Kotecha. Evaluating the impact of ann architecture for driver activity anticipation in semi-autonomous vehicles. Engineering Letters, 29(3), 2021.

[5] Tzu-Chien Ryan Hsiao, Chii-Wann Lin, and Hui-hua Kenny Chiang. Partial least-squares algorithm for weights initialization of backpropagation network. Neu-rocomputing, 50:237–247, 2003.

[6] T Kathirvalavakumar and P Thangavel. A modified back-propagation training algorithm for feedforward neural networks. Neural Processing Letters, 23(2):111–119, 2006.

[7] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. Nature Reviews Neuroscience, 21(6):335–346, 2020.

[8] Sarfaraz Masood and Pravin Chandra. Training neural network with zero weight initialization. In Proceedings of the CUBE International Information Technology Con-ference, CUBE '12, page 235–239, New York, NY, USA, 2012. Association for Computing Machinery.

[9] Sarfaraz Masood, MN Doja, and Pravin Chandra. Anal-ysis of weight initialization methods for gradient descent with momentum. In 2015 International Conference on Soft Computing Techniques and Implementations (IC-SCTI), pages 131–136. IEEE, 2015.

[10] Nadir Murru and Rosaria Rossini. A bayesian approach for initialization of weights in backpropagation neural net with application to character recognition. Neurocomput-ing, 193:92–105, 2016.

[11] Mamata Nayak and Ajit Kumar Nayak. Odia character recognition using backpropagation network with binary features. International Journal of Computational Vision and Robotics, 7(5):588–604, 2017.

[12] N Venkata Rao, ASCS Sastry, ASN Chakravarthy, and P Kalyanchakravarthi. Optical character recognition technique algorithms. Journal of Theoretical & Applied Information Technology, 83(2), 2016.

[13] Ashima Singh and Swapnil Desai. Optical character recognition using template matching and back propa-gation algorithm. In 2016 International Conference on Inventive Computation Technologies (ICICT), volume 3, pages 1–6. IEEE, 2016.

[14] S. S. Sodhi and P. Chandra. A partially deterministic weight initialization method for sffanns. In 2014 IEEE International Advance Computing Conference (IACC), pages 1275–1280, 2014.

[15] Shrinivas R Zanwar, Abbhilasha S Narote, and Sandi-pan P Narote. English character recognition using ro-bust back propagation neural network. In International Conference on Recent Trends in Image Processing and Pattern Recognition, pages 216–227. Springer, 2018.

[16] Shrinivas R Zanwar, Ulhas B Shinde, Abhilasha S Narote, and Sandipann P Narote. Handwritten english character recognition using swarm intelligence and neu-ral network. In Intelligent Systems, Technologies and Applications, pages 93–102. Springer, 2020.

[17] Peipei Zhang and Chuanhe Shen. Choice of the number of hidden layers for back propagation neural network driven by stock price data and application to price prediction. In Journal of Physics: Conference Series, volume 1302, page 022017. IOP Publishing, 2019.