INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 05 ISSUE: 04 | APRIL - 2021 ISSN: 2582-3930

Neural Network: The Backpropagation Algorithm

Miss. Sneha Ramchandra Sawale
Department of Computer Science and Engg
Padm. Dr. V. B. Kolate College Of Engg,
Malkapur- Dist. Buldana

Prof. Miss. Manjiri Karande
Department of Computer Science and Engg
Padm. Dr. V. B. Kolate College Of Engg
Malkapur- Dist. Buldana

Abstract- *In this chapter, we discuss a learning method that handles large learning problems the backpropagation algorithm. This numerical method was used by different research communities in different contexts. It was discovered and rediscovered in 1985 it found its way into connectionist AI. It has been one of the most studied and used algorithms for neural network learning. We see the gradient algorithm in this paper which is based on the backpropagation of neural networks.*

Index Terms- *Backpropagation in Neural Network, Gradient Descent method*

I. INTRODUCTION

For many years hundreds of Neural Network types have been proposed. In fact, because Neural Nets are so widely studied, they are given many different names. You can say as Artificial Neural Networks (ANNs), Connectionism or Connectionist Models, Parallel Distributed Processing (PDP), and Multi-layer Perceptrons (MLPs) [1]. However, despite all the different terms and different types, there are a small group of "classic" networks which are widely used and on which many others are based. These are Back Propagation, Hopfield Networks, Competitive Networks, and networks using Spiky Neurons.

Backpropagation is an abbreviation for "backward propagation of errors"[2], it is a common method of training artificial neural networks. The network learns from many inputs for desired output. It is useful for feed-forward networks. Backpropagation requires the activation function which used artificial neurons (or "nodes"). It is a supervised learning method and a generalization of the delta rule. It requires a dataset of the output for many inputs, to make the training set.

Backpropagation is an iterative process that can often take more time to complete. When multicore computers are used multithreaded techniques can decrease the amount of time that backpropagation takes to converge. If batching is being used, it is very simple to adapt the backpropagation algorithm to operate in a multithreaded manner. The training data is broken up into equally large batches for every thread. Each thread executes the forward and backward propagations. The weight and threshold deltas are together for each of thread. At the end of each iteration, all threads must pause briefly for the weight and threshold deltas to be summed and applied to the neural network. For each iteration, this process gets continued.

Backpropagation is a generalization of the LMS algorithm [3]. We define an error function and would like to minimize it using the gradient descent mean squared error as the performance index. This is achieved by adjusting the weights and the generalized delta rule. It calculates the error for the current input example and error from layer to layer by using Backpropagation.

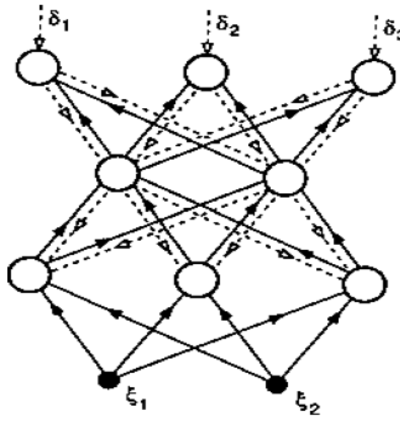


Figure1.1: Simple Backpropagation Network

II.THE ALGORITHM

Many people would consider the Back Propagation network to be the quintessential Neural Net. BackPropagation is the training or learning algorithm other than the network itself. The network used is generally of a simple type and in the examples until now. These are called Feed-Forward Networks or occasionally Multilayer Perceptrons (MLPs) [4].

This network operates in the same way as the others networks worked. Now, let's consider what is the Back Propagation and how it used. A Back Propagation network learns by example. You give the algorithm examples of what you want the network to do and it changes the network's weights so when training gets finished, it gives you the required output for a particular input. BackPropagation networks are ideal for simple Pattern Recognition and Mapping Tasks. As just mentioned, to train the network you need to give it examples of what you want the output you want for a particular input as shown in Figure 2.1.

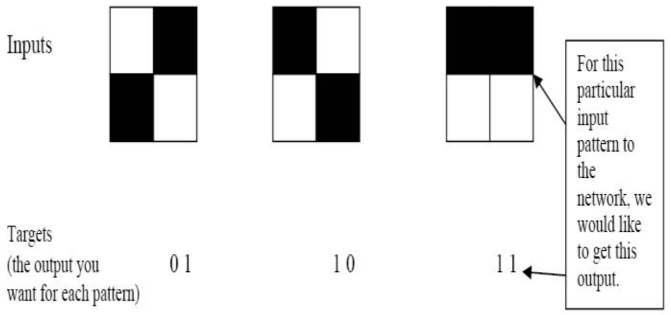

Inputs
Targets
(the output you want for each pattern)
0 1
1 0
1 1
For this particular input pattern to the network, we would like to get this output.

Figure 2.1: A BackPropagation training set

So, if we put in the first pattern to the network, we would like the output to be 0 1 as shown in figure 2.2. The input and its corresponding target are called a Training Pair.

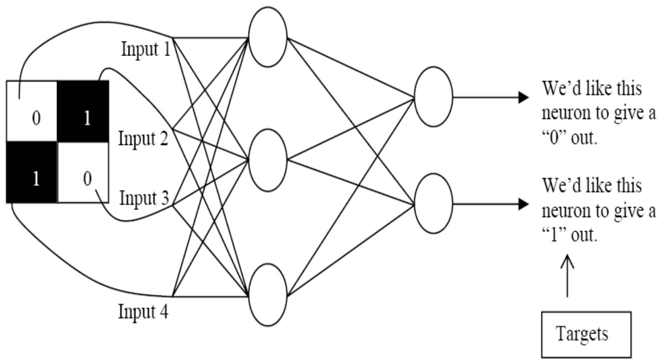

Input 1
Input 2
Input 3
Input 4
0
1
1
0
We'd like this neuron to give a "0" out.
We'd like this neuron to give a "1" out.
Targets

Figure 2.2: Applying a training pair to a network

Once the network is trained, it will provide the desired output for any input patterns. Now, look at how the training works. The network is first initialized by setting up all its weights to be small random numbers between –1 and +1 [5]. Next, the input pattern is applied and the output is calculated. The calculation gives an output that is completely different what you want (Target) since all the weights are random.

Then we calculate the Error of each neuron, which is essentially Target i.e. actual output. This error is then used mathematically to change the weights in such a way that the error will get smaller. In other words, the output of each neuron will get closer to its Target. The process is repeated again and again until the error gets minimal. Let's see an example with an actual network to show how the process works. We will now look at one connection initially, between a neuron in the output layer and one in the hidden layer, figure 2.3.

.

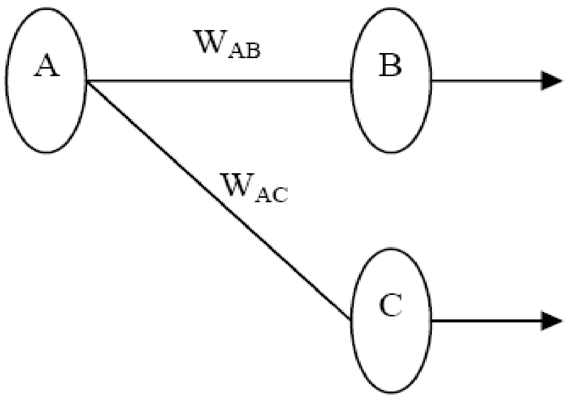


Figure 2.3: A single connection learning in a BackPropagation network

The connection we are interested in is between neuron A (input neuron) and neuron B (output neuron) and has the weight WAB[6]. The diagram also shows another connection, between neuron A and C.

III. AN ANALOGY FOR UNDERSTANDING GRADIENT DESCENT

The basic intuition behind gradient descent can be described by a hypothetical scenario. A person is stuck in mountains and trying to get down (i.e. trying to find the minima). There is heavy fog whose visibility is extremely low. Therefore, the path down from the mountain is not visible, so he must use local information to find the minima. Now he can use method of gradient descent, which involves looking at the steepness of the hill from his current position, then proceeding in the direction with the most negative steepness (i.e. downhill). If he was trying to find the top of the mountain (i.e. the maxima), [7]then he would proceed in the direction with positive steepness (i.e. uphill). Using this method, eventually, he would find his way down the mountain. Now assume that the steepness of the hill is not immediately obvious with simple observation, but rather it requires a sophisticated instrument to measure, which the person happens to have at the moment. It takes some time to measure the steepness of the hill with that instrument, and so he should minimize his use of the instrument if he wanted to get down from the mountain before sunset. The difficulty then is to choose the frequency at which he should measure the steepness of that hill and not go off track.

In this example, the person represents the backpropagation algorithm [8], and the path down the mountain represents the set of weights that will minimize the error. The steepness of the hill represents the slope of the error surface at that point. The direction he must travel in corresponds to the gradient of the error surface at that point. The distance he travels in between measurements is the learning rate of the algorithm. The instrument used to measure steepness is differentiation.

IV. LEARNING OF GRADIENT DESCENT

Differentiable activation functions- The backpropagation algorithm looks for the minimum of the error function in weight space using the method of gradient descent. The combination of weights that minimizes the error function is considered to be a solution to the learning problem. So this method requires the computation of the gradient of the error function at each iteration step, we must guarantee the continuity and differentiability of the error function. We have to use a kind of activation function rather than the step function used in perceptrons, because the composite function produced due to interconnected perceptrons is discontinuous, and therefore the error function too. One of the more popular activation functions for backpropagation networks is the sigmoid [9], a real function Sc: IR ∟(0, 1) defined by the expression

$$s_c(x) = \frac{1}{1 + e^{-cx}}$$

The constant c can be selected arbitrarily and its reciprocal 1/c is called the temperature parameter in stochastic neural networks.

Regions in input space- The sigmoid output range contains all numbers strictly between 0 and 1. Both the extreme values can be reached asymptotically. The computing units considered in this chapter can evaluate the sigmoid using a net amount of excitation as its argument. Given weights w1, . . . ,wn a sigmoidal unit computes for the input x1, . . . , xn the output

$$\frac{1}{1 + \exp\left(\sum_{i=1}^{n} w_i x_i - \theta\right)}$$

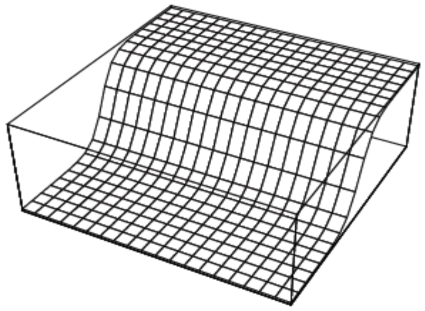
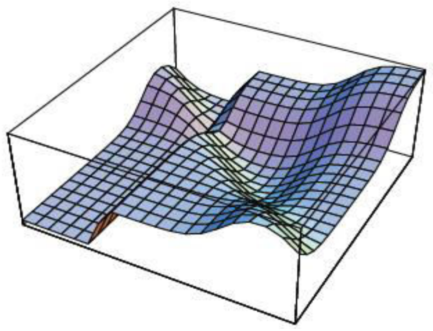


Figure 4.1: A step of the error function

A higher net amount of excitation reaches the unit's output nearer to 1. The continuum of output values can be compared to a division of the input space in the continuum of classes. A higher value of c makes it separate from the input space sharper.

Local minima of the error function- A specific price has to be paid for all positive features of the sigmoid as an activation function. The most important problem is that, under some circumstances, local minima appear in the error function which would not be there if the step functions had been used. Figure 4.2 shows an example of a local minimum with a higher error level than in other regions [10]. The function has to be computed for a single unit with two weights, a constant threshold, and four input-output patterns in the training set. There is a gap in the error function. If gradient descent is started there the algorithm will not converge to the global minimum



.

Figure 4.2: A local minimum of the error function

In many cases, local minima appear because the targets of the outputs of the computing units are values other than 0 or 1 [11] [12]. If a network for the computation of XOR is trained to produce 0.9 at the inputs (0,1) and (1,0) then the surface of the error function develops some protuberances, where local minima can appear. In the case of binary target values, some local minima are also present, shown by Lisbon and Peranton that analytically found all local minima of the XOR function.

V. APPLICATIONS OF BACKPROPAGATION IN NEURAL NETWORK

Classification- In this classification problem, the aim is to identify whether a certain "data point" belongs to classes 1, 2, or 3 [13]. Random points are assigned to a specific class, and the neural network is trained to find the pattern. When training is completed, it will use to learned to accurately classify the new points.

Time series prediction- In this problem, the goal is to design a neural network to predict a value based on a given time series data. To approach this problem, the inputs to the neural network have to be refectories in chunks, and the resulting output will be the next data item directly following that chunk.

Function approximation- In this problem, the network tries to approximate the value of a specific function. It is fed with noisy data, and the aim is to find the true pattern. After training, the network successfully measures the value of the Gaussian function.

Character recognition - The idea behind character recognition has become very important as handheld devices like the palm pilot are becoming increasingly popular. Neural networks are used to recognize handwritten characters. The idea of using feed-forward networks to recognize handwritten characters is rather straightforward. In most supervised training algorithms, the bitmap pattern of the handwritten character is treated as an input, with the correct letter or digit as the desired output. Normally these programs require the user to train the network by providing the program with their handwritten patterns.

VI. CONCLUSION

In this paper, we learned about the gradient descent algorithm which is an important method in neural networks which is under backpropagation networks.

Backpropagation uses the approximate steepest descent algorithm for minimizing the mean square error. The standard gradient descent is slow because it requires a small learning rate for stable learning. Gradient descent with momentum is faster as it allows a higher learning rate while maintaining stability. There are several variations of the backpropagation algorithm.

.

REFERENCES

[1] Rumelhart, David e.; Hinton, Geoffrey e., Williams, Ronald j. (8 October 1986). "learning representations by backpropagating errors". Nature 323 (6088): 533–536.doi:10.1038/323533a0.

[2] Paul J. Werbos (1994). The Roots of Backpropagation. From Ordered Derivatives to Neural Networks and Political Forecasting. New York, NY: John Wiley & Sons, Inc.

[3] Stuart Russell and Peter Norvig. Artificial Intelligence A Modern Approach. p. 578. "The most popular method for learning in multilayer networks is called Back-propagation. It was first invented in 1969 by Bryson and Ho, but was largely ignored until the mid-1980s."

[4] D. A. White and D. A. Sofge, Eds., Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches. Reinhold, 1992.

[5] P. J. Werbos and X. Pang, “Generalized maze navigation: SRN critics solve what feedforward or Hebbian cannot,” in Proc. Conf. Syst. Man Cybern., 1996, pp. 1764–1769.

[6] G. K. Venayagamoorthy and G. Singhal, “Quantum-inspired evolutionary algorithms and binary particle swarm optimization for training MLP and SRN neural networks,” J. Comput. Theoretical Nanosci., vol. 2, pp. 1–8, 2005.

[7] A. R. Barron, “Approximation and estimation bounds for artificial neural networks,” Mach. Learn., vol. 14, no. 1, pp. 115–133, 1994.

[8] D. Prokhorov, R. Santiago, and D. Wunsch, “Adaptive critic designs,” IEEE Trans. Neural Netw., vol. 8, no. 5, pp. 997–1007, Sep. 1997.

[9] X. Pang and P. Werbos, “Neural network design for J function approximation in dynamic programming,” Math Model. Sci. Comp. vol. 2, 1996.

[10] G. V. Puskorius and L. A. Feldkamp, “Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks,” IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 279–297, Mar. 1994.

[11] L. Feldkamp, D. Prokhorov, C. Eagen, and F. Yuan, “Enhanced multistream Kalman filter training for re-current networks,” in Nonlinear Modeling: Advanced Black-Box Techniques. Norwell, MA: Kluwer, 1998, pp. 29–53.

[12] F. Rosenblatt, Principles of Neural Dynamic. New York: Spartan, 1962.

[13] M. L. Minsky and S. A. Papert, Perceptrons. Cambridge, MA: MIT Press, 1969.