Fast Learning Neural Network using modified Corners Algorithm

Rahul Kala[1] *rahulkalaiiitm@yahoo.co.in*

Anupam Shukla[2] *dranupamshukla@gmail.com*

Ritu Tiwari[3] *rt_twr@yahoo.co.in*

[1, 2, 3]*Department of Information Technology*
Indian Institute of Information Technology and Management, Gwalior, INDIA

Abstract

In the past we have seen various developments in the philosophy and application of neural networks. We today have backpropagation algorithm, Hopfield networks, perceptrons, etc All these are very precise tools which model the data very well. But unfortunately, the problem being faced these days is of training the neural network in short span of time, over the test data. All the above mentioned tools may not be useful in various situations where the neural network needs to be trained rapidly. Hence the solutions offered to the same were the Corners rule and the associated CC1 to CC4 algorithms. All these had various pros and cons. This paper uses a different type of modeling to represent data and hence solve the problem of fast learning. Here we have taken the help of distance separation of training data and an unknown input to calculate the most probable output in the neural network. This algorithm is better than the others as it does not place any special restrictions on the inputs, which was the case with CC3. Also the algorithm uses an input model very similar to the traditional model, in terms of inputs and outputs. Hence the users may find it very easy to switch between the traditional neural network style and the network proposed in this paper.

The algorithm sets up a neural network. The weights are assigned by looking at the inputs. In testing, the inputs are provided and the most probable output is calculated. The neural network uses a single hidden layer. The best neurons of the hidden layer are invoked at every input. This algorithm was trained on some points of a 2 color picture. When we tried to reproduce it, the results showed the algorithm was efficient and accurate

Key Words: Neural Network, Fast learning, Instantaneous learning, Corners Algorithm

1. Introduction

Neural Networks are one of the best choices these days to solve any problem of predictions using unstructured or structured data. They have found a huge variety of uses in various places. These are being extensively used in biomedical, robotics, etc. The data used to train these networks is often quite large. One of the main disadvantages of using these is the time taken for training. Usually the training takes a big span of time that makes the neural networks useless to be put on a set of situations. Fast learning is a good tool in these kinds of scenarios.

Numerous means were suggested for fast training of the neural network [15, 16]. In this paper we mainly study the Corners Algorithm [1, 2] that solves the problem. Based on this algorithm, several methods to train the new feedforward network were presented. These included CC1, CC2, CC3 and then CC4. [12,13,14]

This paper proposes a new algorithm which would solve the problem of instantaneous learning or fast learning. All the inputs are given to the neural network. The neural network has one hidden layer. This layer is directly connected to the output neuron. Hence this algorithm has to select the best neurons out of the hidden layer neurons and activate them. This algorithm finds out the best neurons by examining the similarity between the training input and the given input. Since this is an instantaneous learning algorithm, the number of neurons in hidden layer is equal to the number of inputs given in training data, with each neuron representing one input.

Like the other algorithms we also have something similar to the radius of generalization. This factor is set at start and is applied to all outputs. The higher this factor, the more 0s would be there in output and vice versa.

Section 2 of the paper talks about the motivation of the algorithm and the fast learning network. Section 3 covers all the aspects of the algorithm. This includes the assumptions, general description, weights and activation functions, training and testing. We take one example to clarify the points. We also discuss the advantages of this algorithm. Section 4 gives the results of the algorithm. Section 5 is for conclusion.

2. Motivation and Prerequisites

He we discuss the motivation and the present works in this area, which are the guiding factors behind the development. We also present some details which should be known before hand.

2.1 Motivation

As discussed above, various algorithms have been proposed to solve the problem of fast leaning. The basic motivation of all these is the time we save in the training, even though compared to the cost of accuracy. These methods are a fast way of instantaneously training the artificial neural network for short memory [11].

This problem was first solved by Corners Algorithm [1, 2]. This provided a breakthrough in the fast learning. Now the weights were assigned by considering the inputs. Experiments showed than this approach was much faster than the conventional approach, with the loss of some accuracy.

Several algorithms to train the new feedforward network were presented. These algorithms were of three kinds. In the first of these (CC1) the weights were obtained upon the use of the perceptron algorithm. In the second (CC2), the weights were obtained by inspection from the data, but this did not provide generalization. In the third (CC3), the weights obtained by the second method were modified in a variety of ways that amounted to randomization and which now provided generalization. During such randomization some of the learnt patterns could be misclassified; further checking and adjustment of the weights was, therefore, necessitated. Various comparisons were reported in [3, 4, 5]. The comparisons showed that the new technique could be 200 times faster than the fastest version of the backpropagation algorithm with excellent generalization performance. CC4 algorithm was also introduced later on. This algorithm placed some restrictions on the style on input, and hence was good in a limited sense. This algorithm was further improved and the generalizations were removed [6, 7, 8, 9, 10].

2.2 Fast Learning Network

In this section we discuss, in brief, various the fast learning networks. A fast learning network is a network in which there is only one hidden layer, and the number of neurons in the hidden layer is always equal to the number of input cases for the training. This is shown in Fig.1

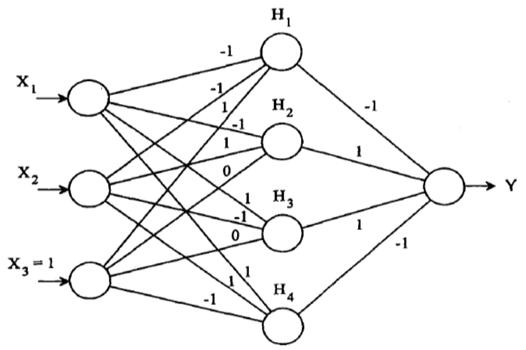

Figure 1: A Fast Learning Model

As we can see in this model, there are 4 hidden input layers (H1 to H4). This means that there would be only 4 input/output cases for the training of this network. This phenomenon has a special significance in fast learning algorithms. We apply the weights to all the neurons of the hidden layers only by seeing the inputs. Hence the extra overload of the backpropagation and constant learning is reduced. We just need to scan the inputs once.

2.2.1 Hidden Layer Weights and Activation Function

The weights of the hidden layer can be assigned by just seeing the inputs. We choose the weights in such a way, such that at every sequence of test cases, only one neuron is activated. Rest all the other neurons fail to activate. This means that the combination of weight is in such a way that when the 1st input is applied, the H1 is activated, H2, H3 and H4 are dead. Similarly in second input sequence only H2 is activated

2.2.2 Output weights and Activation Function

As discussed, on the application of any input, only one neuron is activated. The output weights are taken such that the summation gives the correct results. Hence if the output required is high, the weight is high and vice versa.

3 New Fast Learning Algorithm

In this section we describe the algorithm. We look at the inputs, outputs, assumptions and the implementation details

3.1 Assumptions

The following are the key assumptions for the algorithm, as compared to the normal artificial neural network. These assumptions must be addressed, before applying the algorithm

- Every output is always either 0 or 1.
- The training data consists of enough points for the algorithm to perform. The higher the number of these points, the better the accuracy.
- We also assume that very high accuracy is not needed. As the fast learning algorithms are never very accurate as compared to neural networks with backpropagation training.
- The number of training inputs are enough to fit on the system's primary memory

3.2 General Description

This algorithm is motivated from the traditional Corners Algorithm. The main aspect of the algorithm is the use of modified weight and activation functions which fire the most appropriate neurons for predicting the output. For the training data, only one neuron is fired.

The general steps of the algorithm have been given in Figure 2. The general model of the system is given in Figure 3. We discuss each one of these steps in the coming sub sections

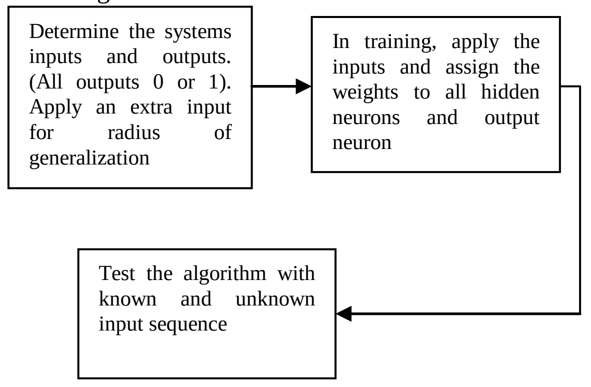
Determine the systems inputs and outputs. (All outputs 0 or 1). Apply an extra input for radius of generalization
In training, apply the inputs and assign the weights to all hidden neurons and output neuron
Test the algorithm with known and unknown input sequence

Figure 2: The general procedure of the algorithm

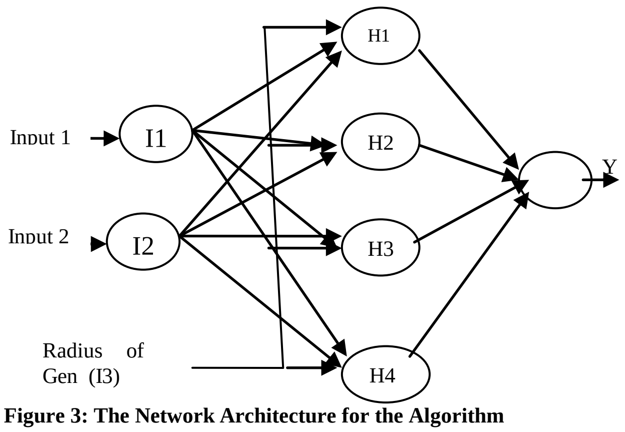

Figure 3: The Network Architecture for the Algorithm

We have added an extra input. This is called the radius of generalization. This input is used for the controlling of the output. In this algorithm, we use this input to control the output. The more positive this number, the more number of 1s would be present in the final output, when used in test mode. Similarly, he more negative this number, the more number of 0s would be present in the final output, when used in test mode.

3.2.1 Weights and Activation Functions

The task here is to assign weights to all neurons and to find out their activations functions. We know that for the training data, only one neuron has to be activated corresponding to the input.

Weights of hidden layer: We know that the network has x number of hidden layers, where x is the number of input cases in the training data. Our job is to assign weights to all the hidden neurons for each of the inputs. As this is a fast training algorithm, the weights are assigned just by looking at the input sequences.

Let the inputs given to the Neural Network for x test cases be (I_{11},I_{21}), (I_{12},I_{22}), (I_{13},I_{23})……(I_{x1},I_{x2}). Here any I_{ij} represents a particular input. The weights of an input sequence i for a corresponding hidden layer j is denoted by H_{ij}. For this algorithm we take H_{ij} equal to the jth input of the ith input sequence I_{ij}. Hence the weights of the hidden layer would be (I_{11},I_{21}), (I_{12},I_{22}), (I_{13},I_{23}) ……(I_{x1},I_{x2}).

Hence the weight vector of the hidden layer is the same as the input sequence vector of the training data. This is shown in Figure 4.

The last input I_3 is the radius of generalization which is artificially applied to each of the inputs of the hidden layer.

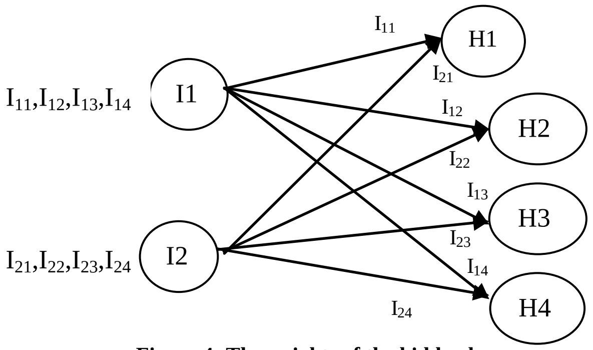

Figure 4: The weights of the hidden layer

Activation Functions of hidden layer: A hidden layer takes all the inputs. For every input it has an associated weight. Conventional neural networks used to multiply the weights to the input sequence and add the results. In this algorithm we take a different approach. Here we take the square of distance between the input and weight in place of multiplication.

For an input $I_i(=I_{i1},I_{i2})$ and Hidden Layer H_j whose weights for the two inputs are I_{j1}, I_{j2}, we have the input to the hidden neuron as:

Input to $H_j = (I_{i1}-I_{j1})^2+(I_{i2}-I_{j2})^2$

Also we have an artificial input I_3. This is added to the result directly (assume corresponding weight to be 0 and I_3 as input).

The output of this neuron will be 1 or 0 always. The procedure is slightly modified. The main aim is to find the closest match from the historical trained network (in test mode). Hence the neuron output is 1 just for the set of neurons which are having minimum input (obtained from sum of distances of weights and inputs + I_3). Hence the algorithm calculates the input of each neuron in the hidden layers, selects the minimum distances. For the set of neurons closest to the input, the output is 1 and for all others, the output is 0.

It may be seen that when we train the network, the product of any input with its corresponding hidden neuron will be zero (distance of point with itself) and for the others will be non-zero (distance of 2 different points). When we multiply all the corresponding weights of a particular neuron with the inputs and add their results, we get a zero answer when an input is same as the corresponding weight else answer is non-zero (provided input sequences have no input in common). It may be observed that any hidden layer neuron i has weights (I_{i1},I_{i2}) with corresponding input sequence (I_{i1},I_{i2}). If we simulate the network, we get the input of the hidden layer as $(I_{i1}-I_{i1})^2+(I_{i2}-I_{i2})^2$, which is always zero. Hence in training, only one neuron is selected for every input.

Weights of the Output Layer: The hidden layer selects a set of neurons in all cases and gives its results. Hence the weight of an output layer for an input from hidden layer neuron i is high (or 1), if the result of input sequence i is high (or 1) and the weight is low (or -1), if the corresponding output is low (or 0). Hence the output vector weight is equal to the output vector given in the training data sequence with 0 replaced by -1.

Activation Function of Output Layer: This is exactly as done in any neural network. The inputs are multiplied with the corresponding weights and the results are added. The final answer is the output.

The output of the neuron is high (or 1) if the input is greater than or equal to 0, and low (or 0) if the input is less than 0.

Table 1: Working in Test Mode

						Input to				Output to				Output
Input			Output	Weights		H1	H2	H3	H4	H1	H2	H3	H4	Weights
3	5	0	1	3	5	0	2	4	2	1	0	0	0	1
2	6	0	0	2	6	2	0	2	8	0	1	0	0	-1
1	5	0	0	1	5	4	2	0	10	0	0	1	0	-1
4	4	0	1	4	4	2	8	10	0	0	0	0	1	1

Figure 5: The network architecture of the example

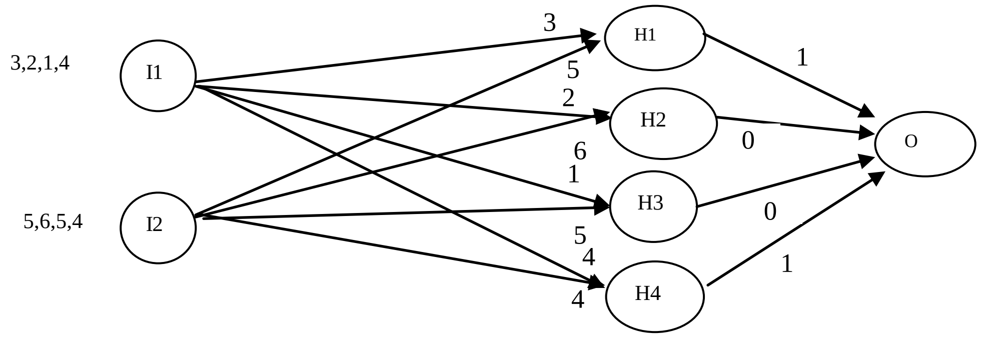

Table 2: The performance of the algorithm

Picture Original	**Training Points (marked with I and O)**	**Picture generated**	**Statistics**

`................`
`................`
`.......###......`
`....#########...`
`...###########..`
`..#############.`
`..#####...######`
`..####.....#####`
`..####.#...#####`
`...#####...#####`
`....###....#####`
`...........#####`
`...........#####`
`..........######`
`.........#######`
`........########`
	`...O....O....O..`
	`.....O.....O....`
	`.......#I#......`
	`...O##I###I##O..`
	`.O.#I#######I#..`
	`.OI###I##I##I#IO`
	`..#####...######`
	`.O#I##.O.O.##I##`
	`..I##IOI...I####`
	`...#I###...####I`
	`.O..#I#..O.##I##`
	`.........O.#####`
	`......O....#I###`
	`..O.....O.###I##`
	`.........I######`
	`..O....OI####I##`
		`................`
		`................`
		`......#####.....`
		`.....#######....`
		`....#########.#.`
		`..#############.`
		`..#####....####.`
		`..####.....#####`
		`..####.##.######`
		`..####.#...#####`
		`...#####...#####`
		`....##.....#####`
		`...........#####`
		`..........######`
		`........########`
		`........########`
			Correctly Matched: 238
			Incorrectly Matched: 18
			Total Points: 256
			Radius of generalization: 0
			Correctness:92%

For r=1, Correctness=90% (232 Correct)	**For r=2,** Correctness=84% (216 Correct)	**For r=-1,** Correctness=91% (234 Correct)	**For r=-2,** Correctness=89%(228 Correct)

`................`
`.......###......`
`......#####.....`
`....#########...`
`..#############.`
`..#############.`
`..#####.#######.`
`..####.....#####`
`.########.######`
`..#######.######`
`...#####...#####`
`...###.....#####`
`...........#####`
`.........#######`
`........########`
`........########`
	`................`
	`.......###......`
	`.....#######....`
	`....#########..#`
	`..##############`
	`.###############`
	`.###############`
	`..#######.######`
	`.########.######`
	`.########.######`
	`...#####...#####`
	`...####....#####`
	`....#......#####`
	`.........#######`
	`.......#########`
	`.......#########`
		`................`
		`................`
		`.......###......`
		`.....#######....`
		`....#########...`
		`...###########..`
		`...###.....###..`
		`...##......####.`
		`..###......#####`
		`...###.....#####`
		`...####.....####`
		`....#......#####`
		`...........#####`
		`..........######`
		`.........#######`
		`.........#######`
			`................`
			`................`
			`.......###......`
			`.....#######....`
			`.....#######....`
			`...###########..`
			`...###.....###..`
			`...##......####.`
			`...##......#####`
			`...##......#####`
			`....###.....####`
			`....#.......####`
			`...........#####`
			`...........#####`
			`..........######`
			`.........#######`

3.2.3 Training the neural network

This is procedure to supply the input sequence to the network and set the corresponding weights of the hidden layer and the output layer. As we know that for every input sequence, only one neuron should be activated. Hence the weights are modeled in such a way as explained above. The inputs are taken and examined one by one. For every input sequence, the weights are set as explained above. Finally, at the end of input sequence, the network is ready and set.

3.2.4 Testing the neural network

After the neural network has been set, we are ready to test. We supply the various inputs whose outputs are unknown. The neural network performs as explained above and gives the outputs.

3.2.5 Example

Following is an example to explain the points discussed in section 3. We suppose that a neural network takes 2 inputs. Both these inputs are integers which vary from 0 to 15. There is one output that can be either 0 or 1.

We have the procedure of setting up the network summarized in table 1. The network architecture is given in Figure 3 and 5.

3.3 Advantages of the algorithm

We have various algorithms that solve the problem of fast neural network learning. The algorithm is better than most of the others because of the following reasons:

- This architecture is very close to the actual neural network architecture and hence users may have very less problems in switching between the conventional and new approach
- This algorithm does not split the inputs into bits and take them as separate input. Hence the structure is very simple. This was a major problem in the preceding algorithms.
- The algorithm can independently take any number of inputs, without internal combining. This thing was not there in any of the earlier algorithms.
- The computation is less as the inputs are not in bits.

4 Results

The algorithm was tested using a simple picture learning problem. A picture was taken which consisted of black and white pixels. Some points of the picture were taken and used as training data. The network was trained using this data set. Then the algorithm was put into use. It was made to regenerate the picture. All the points were passed as the input. The neural network was made to predict whether the pixel is white or black. These were plotted.

The size of the picture was 16X16. Hence there were a total of 256 points. We used a neural network with 2 inputs. One with the x coordinate and the other with the y coordinate. We trained the neural network with 47 points from the picture. The results are given in table 2. We also varied the radius of generalization to see its effect on the working of the algorithm.

It can be easily seen that the efficiency of the algorithm is 92% with only 47 points trained out of 256. Hence using the algorithm we have been able to train a neural network very fast. Even though we have compromised with the efficiency, the performance improvement in speed is a big boom for various applications.

5 Conclusions

In this paper we have successfully proposed and tested a new algorithm that can be used for efficient fast training of the neural networks. We know the various problems because of which there is a need of fast learning in neural networks. Hence various algorithms had come up. This algorithm efficiently does the work with a better architecture.

We studied the theoretical foundations of the algorithm. Also the number of hidden layer neurons used were equal to the number of inputs given in training data. Only a set of neurons were made to activate in every input. The corresponding outputs were computed by the output neuron.

When we simulated the algorithm, we got an acceptable efficiency of 92%. The time required by the algorithm was very less as compared to the time taken by the neural networks. Hence this algorithm can be very easily used for the fast training of the neural networks.

6 References

[1] S.C. Kak, On training feedforward neural networks. *Pramana -J. of Physics*, 40, 35-42 (1993).
[2] S.C. Kak, New algorithms for training feedforward neural networks. *Pattern Recognition Letters*, 15, 295-298 (1994).
[3] S.C. Kak and J. Pastor, Neural networks and methods for training neural networks. *U.S. Patent No. 5,426,721*, June 20, 1995.
[4] P. Raina, Comparison of learning and generalization capabilities of the Kak and the backpropagation algorithms. *Information Sciences* 81, 261- 274 (1994).
[5] K.B. Madineni, Two corner classification algorithms for training the Kak feedforward neural network. *Information Sciences* 81, 229-234 (1994).
[6] Abhilash Ponnath, Instantaneously Trained Neural Networks, 1,2,3
[7] Pritam Rajagopal, The Basic Kak Neural Network with Complex Inputs, PhD Thesis
[8] Adityan Rishiyur, Neural Networks with Complex and Quaternion Inputs
[9] Subhash C. Kak, On generalization by neural networks, *ELSEVIER Information Sciences* 111 (1998) 293-302
[10] Anupam Shukla, Shivanshu Mittal, Fast Learning Neural Network using Advanced Algorithms, MTech Thesis, *IIITM Gwalior*
[11] M. T. Hagan, H. B. Demuth, M. Beale, Neural Network Design, *PWS Publishing Cp*, Massachusetts 1996
[12] S. C. Kak, New training algorithm in feedforward neural networks, *First International Conference on Fuzzy Theory and technology*, Durham N.C., October 1992. Also in Wang PP(Editor), Advance in fuzzy theory and technologies, Durham, N.C. Bookwright Press, 1993
[13] S. C. Kak and J Pastor, Neural Networks and methods for training neural networks, US Patent No 5,426,721, June 20, 1995
[14] S. C. Kak, A class of instantaneously trained neural networks, *Information Sciences*, 148,97-102,2002
[15] Rey-Chue Hwang, Yu-Ju Chen, Shang-Jen Chuang, Huang-Chu Huang, Wei-Der Chang, Fast Learning Neural Network with Modified Neurons, *Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05)*, IEEE Computer Society
[16] Enrique Castillo, A Very Fast Learning Method for Neural Networks Based on Sensitivity Analysis, *Journal of Machine Learning Research* 7 (2006) 1159–1182