IDEA GROUP PUBLISHING

701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com

ITP4801

Artificial Neural Networks: Enhanced Back Propagation in Character Recognition

Evon M. Abu-Taieh
Computer Information Systems Department
The Arab Academy for Banking and Financial Science
P.O. Box13190-11942
Amman – Jordan
Phone: 962-6-5332151, Fax 962-6533593
E-mail: evon2k@yahoo.com, a.elsheikh@aabfs.org

ABSTRACT

The initial weights in Back Propagation has a strong influence in the learning speed and the quality of the solution obtained after convergence. The main objective of the research is to improve the performance of BP through three concepts: First, propagating the error *back to the weights without squaring; second, limiting the initial randomly-generated weights to a certain range; and third, starting the matrix of weights with an ID matrix.*

INTRODUCTION

This research is based on the Back Propagation (BP) Method in Character Recognition, which is a famous approach in Artificial Neural Network (ANN). The main objective of the research is to improve the performance of BP through three concepts: First, propagating the *error* back to the weights without squaring; second, limiting the initial randomly-generated weights to a certain range; and third, starting the matrix of weights with an ID matrix.

OVERVIEW OF ANN

A Neural Network is composed of Processing Elements (PE). PE, or nodes, which represent the Soma, are also called artificial neurons, similar to those in the human brain. Each PE receives an input and produces an output. Each ANN is composed of a collection of PEs grouped in layers. Layers can be input, output, and hidden (intermediate) [Turban, 2000]. The hidden layers can be located between the input and output layers.

ANN network can be organized in several different ways (topologies or architecture); that is, a PE can be interconnected to different PEs in different configurations, called *architecture*. While processing information, many PEs perform their computations simultaneously [Turban, 2000]. This parallel processing resembles the brain activity [Turban, 2000], [Wlodzishaw,1997].

Each input corresponds to a single attribute or variable. In the character recognition program each pixel represents an input. The output of the network represents the solution to the problem being solved. Weights express the relative strength between PEs. It is a mathematical value. Weights are stored when a pattern is learnt and through the readjustment of weights learnt by the network. The Summation Function computes the weighted sum of all inputs entering a PE. The formula for ***n*** inputs to a PE ***j*** is as follows: $Y_j=SX_iW_{ij}$ where *i* ranges from 1 to ***n***, and ***n*** is the size of the input vector, X is the input, W is the weight. The summation calculates the internal stimulation or Activation Level. The relationship between the internal Activation Function and the output can be linear or nonlinear. The relationship can be expressed by several types of transfer functions. The Sigmoid Function is a popular and useful nonlinear transfer function: $Y_T=1/(1+e^{-y})$ where Y_T is the transformed (normalized) value of Y.

Back Propagation is an abbreviation for *Back Error Propagation*. The word PROPAGATION is synonymous with broadcast, dissemination, promulgation, and circulation; if we look at the entire name it means "disseminate the error" of a result back to the input in order to rectify the result. BP method is the Generalized Delta Rule[Carling, 1992] and it belongs to the Supervised Learning Family. There are two major steps in the BP: Forward Pass and Backward Pass. In the Forward Pass, the network functions as usual with each input multiplied by its own random weight, then a transfer function is applied to the result (depending on the number of layers). Subsequently, armed with actual, required, and error values the Backward Pass starts. In the Backward Pass, the process only enhances the weights so that new weights will be calculated, to be reused in the Forward Pass. The network normally has one or more hidden layers.

In general, the BP works as follows:

- Initialize weights (Wij) with random values and set parameters.
- Read the inputs vector (Xi) and the desired output.
- Compute the actual output (Y) via the calculation, working forward through the layers.
- Compute the error =Desired output-Actual output or (delta)=Zj-Yj.
- Change the weights by working backwards from the output layer through the hidden layers [Turban, 2000].

IMPLEMENTATION

For this research, each letter of the English alphabet (26 letters) is divided into a matrix of size 7 X 9. Only capital letters are tested. The network used is made up of three layers (input, hidden, output). The number of PEs is 63 nodes in each layer. The network is fully connected between the input and hidden layers, and fully connected between the hidden and output layers.

The Activation Function is: $O_j = \sum_{j=1}^{63}\sum_{i=1}^{63} w_{ji}o_i + \mathrm{Tj}$, T_j is the threshold of Neuron j , Wji is the weight between neurons j and i , O_j is the output neuron j, where j and i are from 1 to 63. The Transformation Function used in the implementation is the following: $O_j=1/(1+e^{-Oj})$, O_j is the Activation Function of neuron j. The *errors* are calculated as follows $E_j=Desired_j-O_j$. The *error* is propagated as is.

The idea behind this is to simulate the human memorization process: when humans memorize a letter and later find out they are wrong, they will discard the memorized information.

Forward Pass is typical, as mentioned previously :
1. For every layer (input, hidden, output).
2. For every node in that layer.
a. Activation $_j$=sum(W_{ji}*output $_i$).
b. O_j=1/(1+e$^{-\text{Activation }j}$).

The Backward Pass used in this implementation is typical of BP with some modification. The most important modification is the *error* calculation. *Error* is calculated as follows Error = Desired Value- Actual Output. The *error* is not squared, nor is it absolute. The philosophy behind this, as mentioned previously, is similar to the human memorization process. When the *error* is less than zero, it represents trying to *forget the mistake* and the ANN has to readjust the weights accordingly, unlike Magoulas relaying on weight adjustment rather than weights and second derivative [Magoulas, 1999].

Results of Implementation

After the input of a partial letter with input vector size 3 for one vector, the following is noted in the weights matrix:

- When the input is 0 the weights are all negative ranging from -1.4 to -2.9.
- When the input is 1 the weights are all positive ranging from 1.4 to 2.9.
- The bigger the input vector the smaller the number of iterations needed to reach the right weights, as the following table of experiments suggests:

Having observed these results, the next phase of the experiment is to attempt to feed more than one vector. The results are interesting: the number of iterations increase to more than 5000.

Finally, the initial matrix of weights is devised as an ID matrix with weights randomly generated in the range of 0.5 to 1.4 for the hidden layer weights. In other words, the initial ANN network is directly connected: each node in the input layer is connected to one node in the hidden layer, and each node in the hidden layer is connected to one node in the output layer. Then after the initial phase, the network starts building up its structure much like the cascade-correlation architecture describe by [Fahlman, 1991]. The results are also interesting in that the network learnt all 26 letters with an error margin of 0.03 in 1000 iterations. The only letter that is not perfectly learnt is the letter Z, which has one pixel missing.

The final solution is to have matrices of weights: one for the input weights and the other for the hidden weights. The values in the input weight matrix range from 3 to -4.6. The hidden layer weights range from 5.56 to -7.4.

Analyzing the numbers in the weights matrix, as the following figure may suggest, reveals that most weights are concentrated between 0.74 to -0.66, representing 3586 out of 3969 weights, which total to almost 90%.

Table 1: Number of iterations vs. the size of input vector

	Error = 0.03						
	Number of Iterations						
Size of input vector	Experiment #1	Experiment #2	Experiment #3	Experiment #4	Experiment #5	Experiment #6	Experiment #7
2	703	684	357	350	355	354	363
3	240	233	221	237	250	233	242
4	378	175	170	176	184	186	178
5	380	130	142	141	141	142	155
6	252	131	146	147	136	43	157
7	161	158	97	84	1	91	160
8	148	120	18	58	59	101	132
9	122	66	31	34	1	1	1
10	34	37	10	110	1	1	139
11-18	1	1	1	1	1	1	1

Figure 1: Weight count in weights matrix

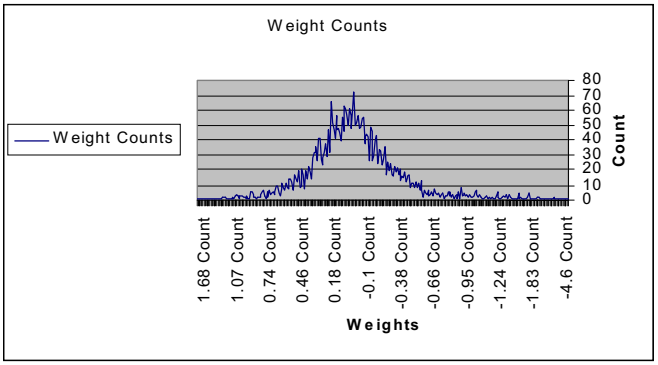
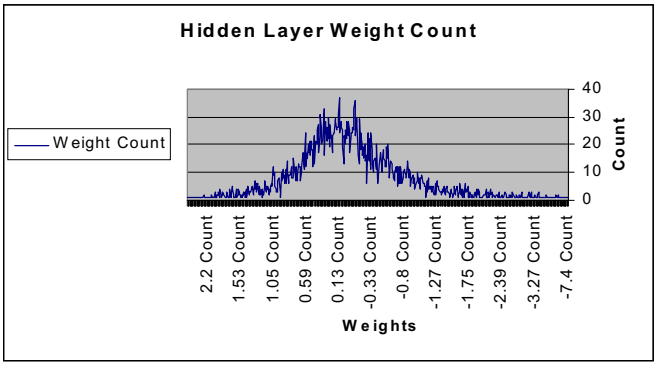


Figure 2: Weight count in the hidden-layer weight-matrix



Looking at the hidden layer weight count, as the following graph may suggest, reveals that most weights are concentrated around 1.53 to -1.75, representing 3648 weights out of 3969, which total to almost 92%

Such findings agrees with finding with [Kolen, 1990] in such that BP is extremely sensitive to initial weights and later was agreed by [Castro, 2001] and [Nolfi, 1999]. Unlike Plagianakos there was no need to limit the weights to integers since the interest there was in hardware implementation [Plagianakos, 2000].

CONCLUSION

The basis of this paper is to speed up the BP method. This can be accomplished by using three concepts: first, the error feedback must be left as is; second, using the weights in the ranges discussed in Figure 2, which are still random, but generated within certain ranges; and third, to use the ID matrix at the initial phase, as it will speed up the process.

The weights are a very important aspect of the BP. While randomly generated, the finding of this research suggests that concentrating weights in the ranges mentioned above will, logically, speed up the process of the ANN network.

ACKNOWLEDGMENTS

The author's research on ANN and Character Recognition has been gratefully supported by Dr. Walid Salameh and Dr. Assem Elsheikh. The authors would like to thank Senator Alia abu-Tayeh and Eng. Akef Abu-Tayeh for their financial support and encouragement.

REFERENCES

[Carling, 1992]. Carling, Alison. Introducing Neural Networks. Wilmslow: Sigma. pp.109

[Castro, 2001]. Castro, Leandro & Zuben, Fernando. An Immunological Approach to Initialized Feedward Neural Network Weights. 5th International Conference on Artificial Neural Networks and Genetic Algorithms. Prague, Czech

[Fahlman, 1991]. Fahlman, S.E. & Lebiere, C. The cascade-Correlation Learning Architecture. Technical Report CMU-CS-90-100, Carnegie Mellon University. Pittsburg, PA 15213, August 1991. http://citeseer.nj.nec.com/fahlman91cascadecorrelation.html

[Kolen,1990]. Kolen, J. F. & Pollack, J. B. Back Propagation is Sensitive to Initial Conditions, Technical Report TR 90-JK-BPSIC, 1990.

[Nolfi, 1999]. Nolfi, Stefano & Floreano, Dario. Learning and Evolution. Autonomous Robots, 7(1)):89—113. http://citeseer.nj.nec.com/nolfi99learning.html

[Magoulas, 1999]. Magoulas, G.D. Vrahatis, M.N. Androulakis, G.S. Improving the Convergence of the Backpropagation Algorithm Using Learning Rte Adaptation Methods, Neural Computation 11 (1999) 1769—1796. http://citeseer.nj.nec.com/magoulas99improving.html.

[Plagianakos, 2000] Plagianakos, V.P. & Vrahatis, M.N. (2000). Training Neural Networks with Threshold Activation Functions and Constrained Integer Weights, in Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN'2000), Como, Italy.http://citeseer.nj.nec.com/plagianakos00training.html

[Turban, 2000] Turban, Efraim. & Arnson, Jay. Decision Support Systems and Intelligent Systems. Prentice Hall, 6th edition. pp. 623.

[Wlodzishaw, 1997] Wlodzishaw, D & Jankowski, N. New neural transfer functions. Jour. of Applied Math. and Computer Science, 1997.citeseer.nj.nec.com/article/duch97new.html