Back Propagation Algorithm: The Best Algorithm Among the Multi-layer Perceptron Algorithm

[1]Mutasem khalil Sari Alsmadi, [2]Khairuddin Bin Omar and [2]Shahrul Azman Noah
[1]Department of Science and Information Technology, University Kebangsaan Malaysia, Kuala Lumpur, Malaysia
[2]Department of System Science and Management, University Kebangsaan Malaysia, Kuala Lumpur, Malaysia

Abstract:
A multilayer perceptron is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate output. It is a modification of the standard linear perceptron in that it uses three or more layers of neurons (nodes) with nonlinear activation functions and is more powerful than the perceptron in that it can distinguish data that is not linearly separable, or separable by a hyper plane. MLP networks are general-purpose, flexible, nonlinear models consisting of a number of units organized into multiple layers. The complexity of the MLP network can be changed by varying the number of layers and the number of units in each layer. Given enough hidden units and enough data, it has been shown that MLPs can approximate virtually any function to any desired accuracy. This study presents the performance comparison between multi-layer perceptron (back propagation, delta rule and perceptron). Perceptron is a steepest descent type algorithm that normally has slow convergence rate and the search for the global minimum often becomes trapped at poor local minima. The current study investigates the performance of three algorithms to train MLP networks. It was found that the back propagation algorithm are much better than others algorithms.

Key words:
Back propagation, perceptron, delta rule learning, classification

INTRODUCTION

Recognition and cataloging are the vital facets in this up-to-the-minute era of research and development, hence exploiting the accessible techniques in Artificial Intelligence (AI) and Data Mining (DM) to achieve optimal production levels, examination procedures and enhancing methodologies in most fields principally in the agricultural domain.

Artificial neural networks are defined as computational models of nervous system. Significantly natural organisms do not only possess nervous system; in fact they also evolve genetic information stored in the nucleus of their cells (genotype). Furthermore, the nervous system as a whole is part of the phenotype which is derived from the genotype through a specific development process. The information specified in the genotype determines assorted aspects of the nervous system which are expressed as innate behavioral tendencies and predispositions to learn[7], acknowledges that when neural networks are viewed in the broader biological context of Artificial Life, they tend to be accompanied by genotypes and to become members of budding populations of networks in which genotypes are inherited from parents to offspring. Many researchers such as Holland, Schwefel and Koza, have stated that artificial neural networks are evolved by the utilization of evolutionary algorithms.

The perceptron is a type of artificial neural network invented in 1957 at the Cornell Aeronautical Laboratory by Frank Rosenblatt. It can be seen as the simplest kind of feed forward neural network: A linear classifier [3]. The learning algorithm is the same across all neurons; therefore a pattern that follows is applied to a single neuron in isolation.

Feed forward back propagation: The feedforward, back-propagation architecture was developed in the early 1970's by several independent sources (Werbor; Parker; Rumelhart, Hinton and Williams) [7, 8]. This independent co-development was the result of a proliferation of articles and talks at various conferences which stimulated the entire industry. Currently, this synergistically developed back-propagation architecture is the most popular, effective, and easy to earn model for complex, multi-layered networks. This network is used more than all other combined. It is used in many different types of applications. This architecture has spawned a large class of network types with many different topologies and training methods. Its greatest strength is in non-linear solutions to ill-defined problems [4, 9, 10, 11, 12, and 15].

The typical back-propagation network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there is just one or two [1, 2]. Some work has been done which indicates that a minimum of four layers (three hidden layers plus an output layer) are required to solve problems of any complexity. Each layer is fully connected to the succeeding layer, as shown in Figure 1.

The in and out layers indicate the flow of information during recall. Recall is the process of putting input data into a trained network and receiving the answer. Back-propagation is not used during recall, but only when the network is learning a training set.

Manuscript received April 5, 2009
Manuscript revised April 20, 2009

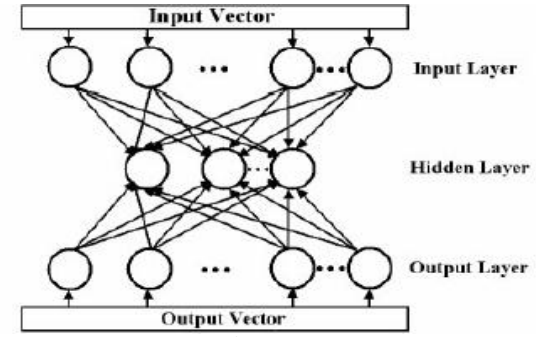

Figure1: An Example Feedforward Back-propagation Network

The number of layers and the number of processing element per layer are important decisions. These parameters to a feedforward, back-propagation topology are also the most ethereal. They are the ³art² of the network designer. There is no quantifiable, best answer to the layout of the network for any particular application. There are only general rules picked up over time and followed by most researchers and engineers applying this architecture of their problems.

Rule One: As the complexity in the relationship between the input data and the desired output increases, then the number of the processing elements in the hidden layer should also increase.

Rule Two: If the process being modeled is separable into multiple stages, then additional hidden layer(s) may be required. If the process is not separable into stages, then additional layers may simply enable memorization and not a true general solution.

Rule Three: The amount of training data available sets an upper bound for the number of processing elements in the hidden layers. To calculate this upper bound, use the number of input output pair examples in the training set and divide that number by the total number of input and output processing elements in the network. Then divide that result again by a scaling factor between five and ten. Larger scaling factors are used for relatively noisy data [4, 10, 11, 12, 13, 14, and 15].Extremely noisy data may require a factor of twenty or even fifty, while very clean input data with an exact relationship to the output might drop the factor to around two. It is important that the hidden layers have few processing elements. Too many artificial neurons and the training set will be memorized. If that happens then no generalization of the data trends will occur, making the network useless on new data sets. Once the above rules have been used to create a network, the process of teaching begins.

This teaching process for a feedforward network normally uses some variant of the Delta Rule, which starts with the calculated difference between the actual outputs and the desired outputs. Using this error, connection weights are increased in proportion to the error times a scaling factor for global accuracy. Doing this for an individual node means that the inputs, the output, and the desired output all have to be present at the same processing element. The complex part of this learning mechanism is for the system to determine which input contributed the most to an incorrect output and how does that element get changed to correct the error [4, 9, 10, 11, 12, and 15] .An inactive node would not contribute to the error and would have no need to change its weights.

To solve this problem, training inputs are applied to the input layer of the network, and desired outputs are compared at the output layer. During the learning process, a forward sweep is made through the network, and the output of each element is computed layer by layer. The difference between the output of the final layer and the desired output is back-propagated to the previous layer(s), usually modified by the derivative of the transfer function, and the connection weights are normally adjusted using the Delta Rule [4, 9, 10, 11, 12, and 15] .This process proceeds for the previous layer(s) until the input layer is reached.

There are many variations to the learning rules for back-propagation network. Different error functions, transfer functions, and even the modifying method of the derivative of the transfer function can be used. The concept of ³momentum error² was introduced to allow for more prompt learning while minimizing unstable behavior. Here, the error function, or delta weight equation, is modified so that a portion of the previous delta weight is fed through to the current delta weight. This acts, in engineering terms, as a low-pass filter on the delta weight terms since general trends are reinforced whereas oscillatory behavior is canceled out. This allows a low, normally slower, learning coefficient to be used, but creates faster learning.

Another technique that has an effect on convergence speed is to only update the weights after many pairs of inputs and their desired outputs are presented to the network, rather than after every presentation. This is referred to as cumulative back-propagation because the delta weights are not accumulated until the complete set of pairs is presented. The number of input-output pairs that are presented during the accumulation is referred to as an ³epoch². This epoch may correspond either to the complete set of training pairs or to a subset.
There are limitations to the feed forward, back-propagation architecture. Back-propagation requires lots of supervised training, with lots of input-output examples. Additionally, the internal mapping procedures are not well understood, and there is no guarantee that the system will converge to an acceptable solution. At times, the learning gets stuck in local minima, limiting the best solution. This occurs when the network systems finds an error that is lower than the surrounding possibilities but does not finally get to the smallest possible error. Many learning applications add a term to the computations to bump or jog the weights past shallow barriers and find the actual minimum rather than a temporary error pocket [4, 9, 10, 11, 12, and 15].

Typical feedforward, back-propagation applications include speech synthesis from text, robot arms, evaluation of bank loans, image processing, knowledge representation, forecasting and

prediction, and multi-target tracking. Each month more back-propagation solutions are announced in the trade journals.

Delta rule: The delta rule is a further variation of Hebb's rule and it is one of the most commonly applied ways available to modify the strengths of the input connections in order to reduce the difference between the desired output value and the actual output of neuron . This rule changes the connection weights in the way that minimizes the mean squared error of the network . The error is back propagated into previous layers one layer at a time. The process of back-propagating the network errors continues until the first layer is reached. The network type called feed forward , back-propagation derives its name from this method of computing the error team. This rule is also referred to as the windrow-hoff learning rule and the least mean square learning rule.

Perceptron: The perceptron is the simplest form of a neural network used for the classification of a special type of patterns, which are linearly separable. It consists of a single McCulloch-Pitts neuron with adjustable synaptic weights and bias (threshold), proved that if the patterns (vectors) used to train the perceptron are drawn from linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyper plane between the classes[5,6,12]. The proof of convergence of the algorithm is known as the perceptron convergence theorem.

The single-layer perceptron shown has a single neuron. Such a perceptron is limited to performing pattern classification with only two classes as shown in Figure 2.

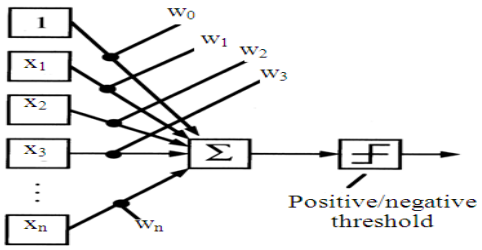

Figure 2: Perceptron.

$$v(x) = \sum_{i=0}^{n} w_i x_i = w^T x \qquad (1)$$

$$y = \begin{cases} 1, v \geq 0 \\ 0, v < 0 \end{cases} \qquad (2)$$

Equation v(x) = 0 defines a boundary between the region where the perceptron fires at and the region where it outputs zero. This boundary is a line (decision line, decision hyperplane), which must be appropriately located during the process of learning. The perceptron can distinguish between empty and full patterns if and only if they are linearly separable as shown in figure 3.

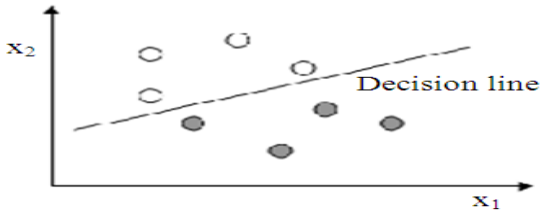

Figure 3: Linearly separable sets

Training set T is a set of pairs [x^i, d^i], where the desired value d^i equals either 1 or 0 and:

$$T = \begin{bmatrix} [x^1\ d^1 \\ [x^2\ d^2 \\ [x^N\ d^N \end{bmatrix}$$

$$x^i = [x_0^i\ x_1^i \ldots x_n^i]^T \qquad (3)$$

The training instance (sample) [x^m, d^m] is misclassified if the perceptron output y^m does not produce d^m.A^n appropriate measure of misclassification is this criterion:

$$J = \sum_{i=1}^{N} (y^i - d^i) v(x^i) = \sum_{i=1}^{N} (y^i - d^i) w^T x^i \qquad (4)$$

Let us discuss individual terms in the criterion J:

- If the training sample is correctly classified then (y^i-d^i) = 0
- If d^m = 1 and y^m = 0 then (y^m-d^m) = -1 and (y^m-d^m)v(x^m)>0. (It follows from (6) that the output y^m = 0 only if v (x^m) < 0.)
- If dm = 0 and y^m = 1 then (y^m-d^m) = 1 and (y^m-d^m) v (x^m)≥. (It follows from (6) that the output y^m = 1 only if v (x^m) 0.)

We can mine from the above, that the criterion J grows if training samples are misclassified and J = 0 if all samples are classified correctly.

A gradient descent method will be used to minimize the criterion J. Let w (k) be the k-th iteration of the weight vector w. The gradient descent method is based on the formula:

$$W(k+1) = w(k) - \text{grad}(J(w(k))) \qquad (5)$$

$$\text{grade}(J) = \frac{\partial J}{\partial J} = \sum_{i=1}^{N} (y^i - d^i) x^i \qquad (6)$$

The learning coefficient controls the size of a step against the direction of the gradient (because of a minus sign). If is too small learning is slow; if too large the process of the criterion minimization can be oscillatory. The optimum

value of the learning coefficient is usually found experimentally. If is kept constant we speak about a fixed-increment learning algorithm.

Experiment design: In this experiment we are testing the back propagation, delta rule and perceptron, to find out the best algorithm of learning in order to train our data. This will be achieved by providing the neural network structure by the learning algorithm and the training samples to learn. Our sample consists of distinct 600 fish images, 400 images used for trained neural network and 210 images used for tested. For futher information about extraction features from color texture measurements refer to [3].

EXPERIMENT OF FEATURES EXTRACTED FROM COLOR TEXTURE FEATURES

In the color texture measurements experiment we built the neural network with three layer and the number of neurons is varied from layer to another (except The output layer consist of 20 neurons since we need to classify 20 fish families [1, 2,…, 20], each of which correspond to one of the possible family's that might be considered) in order to determine the suitable number of neurons for both input and hidden layers, therefore, obtaining high accurate results. The following Table 1 shows the number of neurons for each layer that determined experimentally.

Table 1: Number of neurons for each layer

Training	NO. Neurons in Layers		
	Layer #1	Layer #2	Layer #3
BPC	22	33	20

Back propagation classifier were used with a total of 20 neurons for input layer, 30 hidden layer and 20 in the output layer, since this research need to classify 20 fish families [1, 2,…, 20], each of which correspond to one of the possible family's that might be considered. where the output represents twenty families of fish.

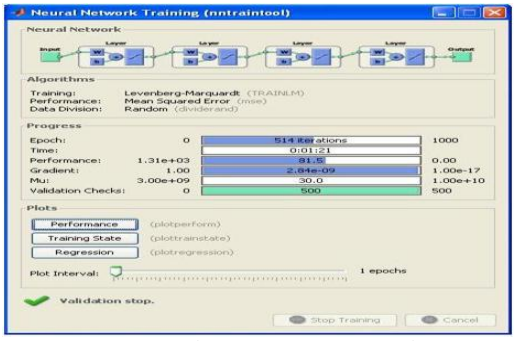

Figure 4: Results for training color texture features

The obtained results of the training part are shown in Figure 4. Where, number of maximum proposed iterations to get the results is 1000 while the neural network finishes training at iteration number 514. The time for accomplishing the training was 1 minute and 21 seconds. The Figure below shows the performance of the neural network including Gradient, Mu, and validation check.

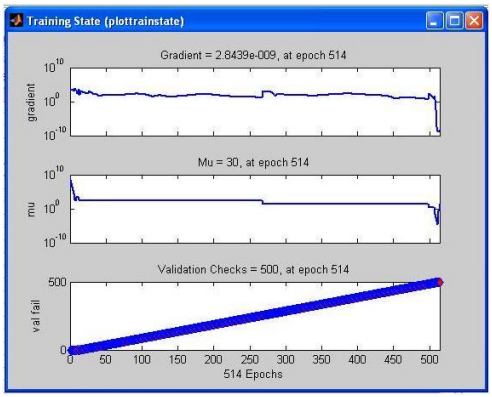

Figure 5: The training state results of color texture features.

In Figure 5, the performance parameters are discussed in detail. The gradient started at first epoch at 104 and slightly varied increase and decrease around the first epoch value until the last epoch when the gradient is decreased rapidly to (2.8439e-009) which is near to 10^{-8}. In the second diagram the Mu started from 108 and decreased rapidly until 108 and nearly remained stable until epoch 262. Then it has decreased a little to 101 and remained again stable until epoch 500. In the end it has rapidly decreased to 10^{-5} and increased again to 103. In the third diagram, the validation checks started at the value of zero and remained until epoch 42. Then it has started to increase the value gradually until epoch 514 where the validation checks are 500 during the training process.

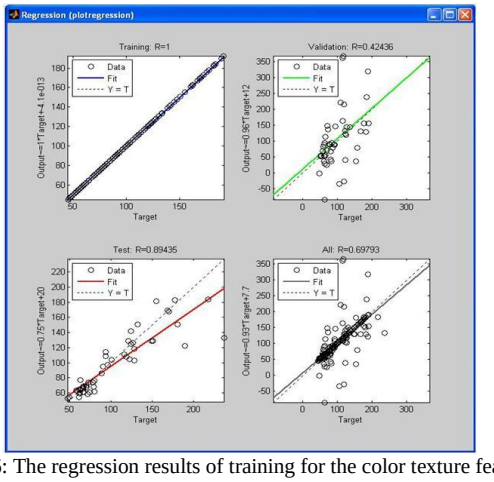

Figure 6: The regression results of training for the color texture features.

The regression results when using R =1, the target formulae (Output= 1* Target + 4.1e-013) for the training were near to the target (slope line) and the data fits in all iterations as shown in Figure 6.

The mean squared error for checking the stop conditions of training is shown in Figure 7. The Figure shows the diagram of MSE during all 514 training iterations. The best remained at epoch 10^3, although the MSE for the first iteration is near to 105, where it is gradually decreased to 10^{-2} in the 502nd iteration. From iteration 502 until iteration 514 it has been rapidly decreased until the MSE value is 10^{-20}.

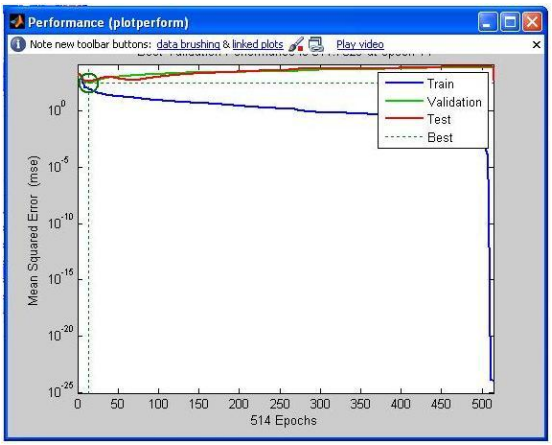

Figure 7: The performance by the mean squared error for training the color texture features.

The following Table 7 describes the overall training and testing accuracy. The results shown in the Table are the overall accuracy outcome for both training and testing accuracy obtained from the trained neural network using back-propagation classifier. The overall training and testing accuracy was 85% and 84% respectively.

Table 2: Description of the overall accuracy of training and testing for the color texture features

Description	Results
Overall training accuracy	85%
Overall testing accuracy	84%

CONCLUSION

In short, this study demonstrated the study of multi-layer perceptron (back propagation, delta rule and perceptron) algorithms in neural networks in the previous study and explained the performance of the BP algorithm based on extracted features from color texture measurements of fish images. Eventually experimental findings revealed that the back propagation algorithm is the best algorithm to be used in the multi-layer perceptron in a neural network due to back propagation algorithm Complex logical operations pattern classification analysis and Back-Propagation (BP) learning algorithm is designed to reduce an error between the actual output and the desired output of the network in a gradient descent manner.

REFERENCES

[1] Anderson, J.A., 1995. An Introduction to Neural Networks. MIT Press, Cambridge, MA., ISBN: 10: 0262011441, pp: 672.

[2] Chauvin, Y. and D.E. Rumelhart, 1995. Backpropagation: Theory, Architectures and Applications. Erlbaum, Mahwah, NJ., ISBN: 080581258X, pp: 561.

[3] Alsmadi, M. K., Omar, K. B., Noah, S. A. And Almarashdeh, I. Fish Recognition Based On Robust Features Extraction From Color Texture Measurements Using Back-Propagation Classifier. Journal of Theoretical and Applied Information Technology 11-18.

[4] Gupta, L., M.R Sayeh. and R. Tammana, 1990. A neural network approach to robust shape classification. Patt. Recog., 23: 563-568. http://cat.inist.fr/?aModele=afficheN&cpsidt=19457517

[5] Qiyao Yu, C. Moloney and F.M. Williams, 2002. SAR seaice texture classification using discrete wavelet transform based methods. Proceeding of the IEEE International Geoscience and Remote Sensing Symposium, (IGRSS'02), IEEE Xplore Press, USA., pp: 3041-3043. DOI: 10.1109/IGARSS.2002.1026863

[6] Rumelhart, D.E. and J.L. McClelland, 1986. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. 8th Edn., I and II, MIT Press, Cambridge, MA., ISBN: 0262631105.

[7] Veerendra Singh and S. Mohan Rao, 2005. Application of image processing and radial basis neural network techniques for ore sorting and ore classification. Mineral. Eng., 18: 1412-1420. http://cat.inist.fr/?aModele=afficheN&cpsidt=17289210

[8] Werbos, P.J., 1974. Beyond regression: New tools for prediction and analysis in the behavioral sciences. Ph.D. Thesis, Harvard University. http://www.citeulike.org/user/yannael/article/1055600.

[9] Philip, M., D. Merikle, Smilek and D. John, 2001. Perception without awareness: Perspectives from cognitive psychology. Cognition, 79: 115-134. http://www.ncbi.nlm.nih.gov/pubmed/11164025

[10] Hand, D., H. Mannila and P. Smyth, 2001. Principles of Data Mining. MIT Press, Cambridge, ISBN: 026208290X, pp: 546.

[11] Hastie, T. R. Tibshirani and J. Friedman, 2001. The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, New York, ISBN: 0387952845, pp: 533.

[12] Leo Breiman, H. Jerome Friedman, A. Richard Olshen and J. Charles Stone, 1998. Classification and Regression Trees. Chapman and Hall/CRC. Paperback: 368 pages. ISBN 978-0412048418.

[13] Jiawei Han and Micheline Kamber, 2000. Data Mining: Concepts and Techniques. 1st Edn., Morgan Kaufmann Publishers, ISBN: 10: 1558604898, pp: 500.
[14] Michael R. Anderberg, 1973. Cluster Analysis for Applications. Academic Press, ISBN: 0120576503, pp: 359.
[15] Michael J.A. Berry and Gordon S. Linoff, 2000. Mastering Data Mining. Wiley.

Mutasem Khalil Sari Al Smadi received his BS degree in Software engineering in 2006 from Philadelphia University, Jordan, his MSc degree in intelligent system in 2007 from University Utara Malaysia, Malaysia; He has published paper in IEEE and International Journal of Computer Science and Network Security; currently he is doing PhD in Intelligent System in University Kebangsaan Malaysia in Malaysia.

Khairuddin Omar is an Associate Professor in Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 UKM Bangi, Malaysia. His research interest includes Arabic/Jawi Optical Text Recognition, Artificial Intelligence for Pattern Recognition & Islamic Information System.

Shahrul Azman Noah is currently an associate professor at the Faculty of Information Science and Technology, University Kebangsaan Malaysia (UKM). He received his MSc and PhD in Information Studies from the University of Sheffiled, UK in 1994 and 1998 respectively. His research interests include information retrieval, knowledge representation, and semantic technology. He has published numerous papers related to these areas. He currently leads the knowledge technology research group at ukm.