ISSN 2349-4506
Impact Factor: 2.545

Global Journal of Engineering Science and Research Management

CONSTRUCTIVE TRAINING ALGORITHM FOR DESIGNING FEEDFORWARD NEURAL NETWORKS: A REVIEW

Kiran Khatter*, Jaswinder Kaur
* Ansal University, Gurgaon

DOI: 10.5281/zenodo.57375

KEYWORDS: Neural networks, Constructive algorithm, Cascade correlation algorithm, Cascade 2 algorithm, Recurrent CBP, Casper.

ABSTRACT

In this paper, we review neural networks, models of neural networks, methods for selecting neural network architecture and constructive algorithms for regression problems. Cascade correlation algorithm is the most suitable for solving regression problems. Dynamic Node Creation (DNC) algorithm is a method which automatically grows backpropogation networks until the target problem is solved. Cascade 2 algorithm is a variant of Cascade correlation algorithm. Recurrent CBPmethod addresses the construction of recurrent networks by the use of constructive backpropogation. Casper is known to produce more compact networks with very promising results. Adaptivesigmoidal activation function can be used for better generalization performance and training time can be reduced.

INTRODUCTION

In recent years, many models of neural networks are proposed for regression problem. Multilayer feedforward network is the most popular. Multilayer feedforward network is flexible in structure, has got good representation capabilities and large number of training algorithms are available for it. The generalization ability and learning accuracy of supervised learning in feed forward neural networks depend on several factors such as network architecture which include hidden nodes and connection topology between nodes, choice of activation function for each node and the training parameters like learning rate and weights [1]. These factors of the neural network model are either fixed prior to training or dynamically adjusted or changed during training of the network for a given problem. If these factors are not appropriately chosen then they may cause the under fitting and over fitting of the network which can result in a poor generalization of the network.

Artificial neural Network is a network of artificial neuronsthat are interconnected according to a network architecture. Neuron is the basic processing unit of neural network. Neural networks are composed of simple elements operating in parallel [2].It is inspired by biological nervous systems. The network function is defined by connections between elements. A neural network can be trained to perform a particular function by adjusting the values of the connection between elements.

Artificial neural network contain layers of simple computing nodes which are connected by weighted connection lines and the weights are adjusted during training process based on the comparison of input value and target value. This process is continued until the network output matches the target value.

“A neural Network is massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use [3].” Knowledge is acquired from its environment by the network through a learning process. The interconnections (weights) used to store the knowledge acquired by the network.

The existing neural models have almost no similarities to the biological neural systems. Even the simplest biological brain has the following contra-distinctive features as compared to the von-Neumann architecture of the digital machine [4]. The computation is massively distributed and parallel. Learning replaces a priori program development. Fault-tolerant to a high degree, that is, destruction of a few cells during the computation process does affect the end result of computation.

Artificial neural network can be enumerated as [4]-[9] :

1. Integration and storage of experience.
2. Consideration of new experience in context of new experience.
3. Generalization capability.
4. Fault- tolerant architecture.

Applications of Artificial Neural Network include Signal processing, Pattern Recognition (character recognition, character recognition, speech recognition) [5], Military (Fault detection in complex engineering systems, automatic target recognition, Target tracking), Image processing(image matching, image compression), Decision Support Systems (credit card authorization, stock market analysis), Medicine (image analysis, diagnosis of diseases, cancer cell analysis, ECG and EEG), Artificial Intelligence (expert systems), Function approximation (modeling), Electronics (IC, IC chip layout, analysis of chip failure), Manufacturing, Transportation, Entertainment (special effects, Animation), Oil and Gas Exploration, Telecommunications [2].

FEEDFORWARD NEURAL NETWORK

In layered neural networks, the neurons are organized in the form of layers. The neuron in a layer get input from the previous layer and feed their output to the next layer. This type of networks is called feedforward neural networks and output connections from a neuron to the same or previous layer neurons are not permitted. The input layer is made of special input neurons, transmitting only the applied external input to their outputs. The last layer or neurons is called the output layer, and the layers that are not input or output are called the hidden layers. FeedForward Neural Networks are used in different applications related to problems such as data classification, time series prediction, and identification and control of non-linear objects [14].

Single Layer Feedforward Network

In a network, if there is only the layer of input nodes and a single layer of neurons constituting the output layer, then it is called a single layer network. The number of inputs may not be equal to the number of neurons in a layer. In most applications a feed-forward network with a single layer of hidden units is used with a sigmoid activation function for the units [15].

Multilayer Feedforward Network

In a multilayer network there can be more than one layer of neurons. The data from input units to output units is strictly feedforward. The data processing can extend over multiple (layers of) units, but no feedback connections are present, that is, connections extending from outputs of units to inputs of units in the same layer or previous layers. Also there is no connection within the layer. No processing takes place in the input units. The activation of a hidden unit is a function of the weighted inputs plus a bias. The output of the hidden units is distributed over the next layer of hidden units, until the last layer of hidden units, of which the outputs are fed into a layer of no output units [15].

Multilayer perceptron is wellknown feedforward layered network, on which the Backpropagation learning algorithm [16] is widely implemented. The structures, where connections to the neurons are to the same layer or to the previous layers are called recurrent networks.

Although the backpropagation algorithm (BP) has been a significant milestone in neural network research area of interest, it has been known as an algorithm with a very poor convergence rate [17]. Many attempts have been made to speed up the BP algorithm. Commonly known heuristic approaches [18]-[22] such as momentum [23], variable learning rate [24], or stochastic learning [25] lead only to a slight improvement. Better results have been obtained with the artificial enlarging of errors for neurons operating in the saturation region [26]-[29]. A significant improvement on realization performance can be observed by using various second order approaches namely Newton's method, conjugate gradients, or the Levenberg-Marquardt (LM) optimization technique. Among the mentioned methods, the LM algorithm is widely accepted as the most efficient one in the sense of realization accuracy [30]. It gives a good compromise between the speed of the Newton algorithm and the stability of the steepest descent method, and consequently it constitutes a good transition between these methods.

Problems in FeedForward Neural Network

A. Choice of architecture/size
B. Choice of training algorithm
C. Choice of training parameters (learning rate, momentum etc.)

AUTOMATIC SELECTION OF NN ARCHITECTURE

Recently, various researchers have investigated different approaches that alter the network architecture as learning proceeds. We need a network large enough to learn the mapping and as small as possible to generalize well [31].

Pruning or Destructive Method

This method starts with training a network which is larger than required and then removing the unimportant connection weights or units that are not required. The initial large size allows the network to learn quickly. Approaches include removing the smallest magnitude or insensitive weights or by adding a penalty term to the energy function to encourage weight decay. These algorithms are designed to take advantage of larger systems while avoiding overfitting problem [32]. Pruning has been carried out on networks in three distinct ways. The first is a heuristic approach, identifying which nodes and weights contribute little to the mapping. After these have been removed, additional training leads to a better network than the original. An alternative technique is to include terms in the error function, so that the weights tend to zero under certain circumstances. Zero weights can be removed without degrading the performance of the network. Finally, if we define the sensitivity of the global network error to the removal of a weight or node, and evaluate it for each such parameter in the network, we can then remove the weights or nodes to which the global error is least sensitive. The sensitivity measurement does not interfere with training, and involves only a small amount of extra computational effort [32].

Regularization

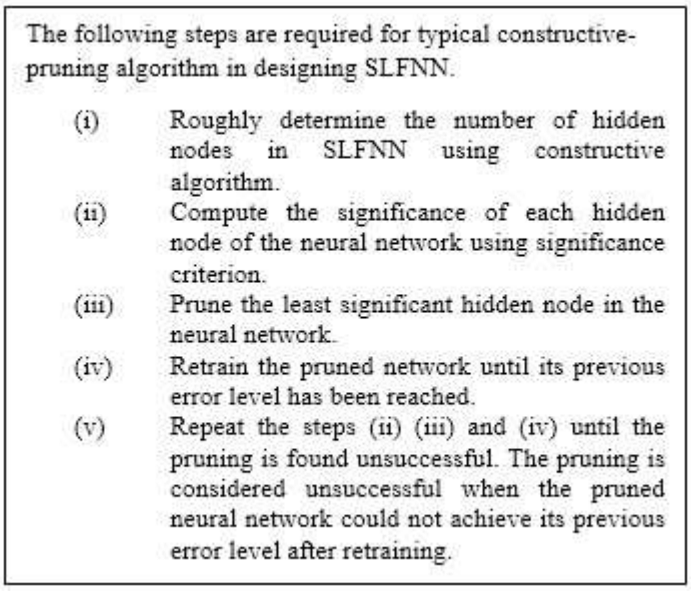
Regularization is a procedure that is used to solve ill-posed problems by introducing additional information to prevent over-fitting. Tikhonov [33] proposed the regularization theory first in the context of functional approximation. One of the major issues in FNN training is to control the complexity and provide good generalization capabilities. One of the assumptions that are generally made in complexity control is that the desired function should be smooth, in the sense that the “nearby” input points lead to “nearby” output points. As pointed out by Gemen [34] the problem of complexity control may be related to the bias / variance dilemma in FNNs. Regularization is the method which tries to strike a balance between the model bias and model variance by incorporating *apriori* knowledge in the model selection (error function). Regularization is also the method of locating an optimum architecture, in terms of number of free parameters and their values. Regularization method imposes constraints on the values of the parameters of the network to control complexity. The regularization method modifies the error function by adding/subtracting a penalty / regularization termto obtain a modified error function.Many regularization terms have been proposed in neural network literature to control the complexity of the network model.

Constructive Method

In constructivemethods the network starts with a small size and grows when needed [35]. The obvious advantage of this method is that it might require less computation than the destructive methods. In addition, this method frees one from “guessing” a large enough initial network size and topology. However, without proper control of the growth, this process could lead to an oversized network [36]. Constructive algorithms overcome the limitation of searching for a solution in the weight space of a *priori* fixed network architecture by extending the search, in a controlled fashion, to the entire space of neural-network topologies. Further, it has been shown that at least in principle, algorithms that are allowed to add neurons and weights represent a class of *universal learners* [37]. Constructive algorithms search for small solutions first and thus offer a potential for discovering a near minimal network that suitably matches the complexity of the learning task. Smaller networks are also preferred because of their potential for more efficient hardware implementation and greater transparency in extracting the learned knowledge [38].

Constructive Pruning (hybrid) Method

A constructive algorithm adds hidden nodes to minimal network architecture one by one during training. The number of hidden nodes may become ridiculously large in some cases if the addition process is not stopped properly. Some algorithms [39] and [40] try to solve the aforementioned problem by employing a pruning phase in conjunction with a constructive phase. These algorithms first determine the number of hidden nodes and/or weights in neural network roughly using a constructive approach. A pruning approach is then applied for refining the selection, i.e., removing the irrelevant hidden nodes and/or weights.


The following steps are required for typical constructive-pruning algorithm in designing SLFNN.

(i) Roughly determine the number of hidden nodes in SLFNN using constructive algorithm.
(ii) Compute the significance of each hidden node of the neural network using significance criterion.
(iii) Prune the least significant hidden node in the neural network.
(iv) Retrain the pruned network until its previous error level has been reached.
(v) Repeat the steps (ii) (iii) and (iv) until the pruning is found unsuccessful. The pruning is considered unsuccessful when the pruned neural network could not achieve its previous error level after retraining.

This shows how a constructive-pruning algorithm adds and prunes hidden nodes in order to find a near optimal network architecture for a given problem. However, one needs to determine when to stop the pruning process.

The constructive approach has a number of advantages over the pruning approach. First, for constructive algorithms, it is straightforward to specify an initial network whereas for pruning algorithms, one does not know in practice how big the initial network should be. Second, constructive algorithms always search for small network solutions first. They are thus more computationally economical that pruning algorithms, in which the majority of the training time is spent on networks larger than necessary. Third as many networks with different size may be capable of implementing acceptable solutions, constructive algorithms are likely to find smaller network solutions than pruning algorithms. Smaller networks are more efficient in forward computation and can be described by a simpler set of rules. Functions of individual hidden units may also be more easily visualized. Moreover, by searching for small networks, the amount of training data required for good generalization may be reduced. Fourth, pruning algorithms usually measure the change in error when a hidden unit or weight in the network is removed. Due to these advantages constructive algorithms are preferred over pruning algorithms [41].It is common practice to combine different models in order to improve the performance and overcome the problems of a single model [42].

CONSTRUCTIVE METHODS

In general, this class of algorithms starts with a small network and adds units or connections until an adequate performance level is obtained. Many constructive neural network algorithms have been surveyed by many authors but the most popular out of them is Cascade Correlation algorithm [43].

Cascade Correlation NN

Cascade correlation is a powerful method of training neural networks. Cascade Correlation starts with a minimal network in which new hidden units are trained and added one by one. Cascade-Correlation consists of two steps.

Cascade architecture is the first step in which hidden units are added one at a time to the network. They do not change after they are added. Learning algorithm is the second step in which new hidden units are created and installed. We try to maximize the magnitude of correlation between the new unit output and residual error signal [44].

In this there are some inputs and one or more output units with no hidden units. Each input unit is connected to each output unit with a connection whose weight can be adjusted. Bias input is set to 1 permanently. Hidden units are added one by one to the network and each unit receives a connection from original inputs of the network and also from pre-existing hidden unit. When the units are added to the network, the weights of hidden units are frozen .Output units are repeatedly trained. A new one unit layer is added to the network when new unit is added. This is done until some of the incoming weights are zero. In a single layer network we can use Delta rule or Windrow-hoff with no need to back propogate through hidden units. Quickprop algorithm can be used to train output weights. New hidden unit is added to the network and the input weights are frozen and the output weights are frozen and all the output weights are trained once. This is repeated until error is small.Whenthe weights in the output layer are trained the other weights in the active network are frozen. When the candidate weights are trained none of the weights in the active network are changed. In a machine with plenty of memory, it is possible to record the unit-values and the output errors for an entire epoch, and then to use these cached values repeatedly during training, rather than recomputing them for each training case. This can result in a tremendous speedup, especially for large networks.

Dynamic Node Creation Algorithm

Dynamic Node Creation Algorithm (DNC) addresses the problem of training large networks and testing networks with different numbers of hidden layer units. This is a new method called Dynamic Node Creation (DNC) which automatically grows backpropogation networks until the target problem is solved [45]. This method starts with a network topology having a small number of hidden units and automatically grows backpropagation network by adding one hidden unit at a time until a network that can solve the application problem has been found. In this model all the connections between the nodes of each layer to the nodes of layer above it are feedforward. There are no backward connections. Connections in the network are allowed only between the input units and the hidden nodes, and between the hidden units and output units. At all the layers above the input, output (I) = $\frac{1}{1+e^{-1}}$ logistic activation function [46] is used. New node should be added if both the following conditions are met $\frac{|a_t-a_{t-1}|}{a_{t0}} < \Delta_T$ and $t - w \geq t_0$ where a_t is average squared error at time t, t is current time, t_0 is the time when last node was added, Δ_T is the trigger slope and w is the width of window over which trigger slope is determined. A new node can be added after at least w trials. Each time a hidden node is added to the network topology, the new network is trained by using the backpropagation method. Now when to stop the addition of node is determined by the following conditions. $a_t \leq c_a$ and $m_t \leq c_m$ where a_t is average squared error at time t, c_a is the desired cutoff average squarederror, c_m is the desired cutoff for maximum squared error, m_t is the maximum squared error at the time t on any output at node. If these conditions are satisfied then we can stop adding nodes. This method yields a solution for every problem tested, in particular the method successfully generated networks that solve the parity problems with 5 and 6 input units using only 5 and 6 hidden units, respectively. Note that it is extremely difficult to train a network with 5 (or 6) hidden units to solve the 5 (or 6) bit parity problem, if we fix the number of hidden units in advance.

VARIANTS OF CASCADE CORRELATION NEURAL NETWORK

Cascade 2 algorithm

Cascade 2 algorithm was also first proposed by Fahlman, who has implemented the CCA. Cascade 2 algorithm differs from CCA by training a new hidden node to directly minimize the residual error rather than tomaximize the covariance between hidden node output and residual error at output nodes. Besides, hiddennode has adjustable output connections to all of the output nodes and all other things are common in bothalgorithms. Several authors have demonstrated that CCA is effective for classification but not very successfulon regression tasks. This is due to the correlation term tending to drive the hidden node activations to theirextreme values, thereby, making it hard for the network to produce a smoothly varying output [47], [48], [49].

Recurrent CBP

Recurrent Constructive backpropogation network was proposed by [50]. This method addresses the construction of recurrent networks by the use of constructive backpropogation. This is a similar approach to Cascade correlation algorithm. A fully recurrent network with arbitrary number of layers can be constructed efficiently. Even though there is no need to back propogate the error through not more than one layer of hidden neurons, recurrent constructive backpropogation algorithm is as efficient as cascade correlation algorithm. We can continue the adaptation of network weights as well as continue structure adaptation after the network is constructed. It includes the addition and deletion of neuron and layers in a computationally efficient manner [50]. The recurrent cascade correlation algorithm has better modeling capabilities as compared to cascade correlation algorithm. The constructive backpropagation procedure for recurrent network construction can be described by the following steps:

1. Initialization: Train the first hidden unit by minimizing the cost function (1). At this point there is only one recurrent connection, that is, the self-connection. If one hidden unit does not yield acceptable solution, then go to step 2. Otherwise stop the training.

2. Train a new hidden unit and adapt the previous ones. Connect network inputs, feedforward connections from previous units and recurrent connection from previous units to the new unit and connect its output to the output units. Note that in the new neuron perspective the previous units are fixed. In addition, make a recurrent connection from the new hidden unit output to all the currently existing hidden units (including the recurrent self-connection). Train the new and the previous hidden units hierarchically by minimizing the cost function (l), that is for i^{th} neuron the cost function is SSE,

$$SSE_i = \sum_{l,k}\left(d_{kl} - \sum_{j=1}^{i-1}(v_{jk}h_{jl} + v_{jk}^0) - (v_{ik}h_{il} + v_{ik}^0)\right)^2$$

3. Test for convergence: If the current number of hidden units yields acceptable solution, then stop the training. 0therwise go back to step 2.

Casper – Cascade network algorithm employing Progressive RPROP

Casper is a constructive learning which builds cascade networks [51]. It is a modified version of RPROP algorithm for network training. RPROP is a gradient descent algorithm which uses separate adaptive learning rates for each weight [52], [53]. It is constructed in a manner similar to cascade correlation. It starts with a single hidden neuron and successively adds single hiddenneurons. Casper starts with a single hidden neuron and successively adds single hidden neurons. RPROP is used to train the whole network each time a hidden neuron is added. RPROP is modified, however, such that when a new neuron is added the initial learning rates for the weights in the network are reset to different values, depending on the position of the weight in the network. The network is divided into three separate regions, each with its own initial learning rate L1, L2 and L3. The first region is made up of all weights connecting to the new neuron from previous hidden and input neurons. The second consists of all weights connecting the output of the new neuron to the output neurons. The third is made up of the remaining weights, which consist of all weights connected to, and coming from, the old hidden and input neurons.

Casper is known to produce more compact networks with very promising results.

Adaptive slope sigmoidal function Constructive Algorithm

Sudhir et al. [54] developed the adaptive slope sigmoidal function for constructive algorithm. Depending on the problem to be solved the number of nodes in the input and output layers are defined. It starts with minimal architecture and one hidden node is added during training at one time to the network. This method able to solve the step size of weight update problem. The nonlinear hidden nodes are prevented from going into saturation.

CONCLUSION

This paper presents an overview of constructive neural networks. In constructive method, training starts with minimal structure and then more layers of neurons are added according to some rule which is predefined.

Constructive algorithms provide a natural framework for incorporating problem-specific knowledge into initial network configurations. In this paper we have enlightened various constructive algorithm methods that construct feedforward architecture for regression problems.

REFERENCES

1. Sharma, Sudhir Kumar, and Pravin Chandra. "Constructive neural networks: a review." International Journal of Engineering Science andTechnology 2.12 (2010): 7847-7855.
2. Demuth, H., Beale, M., & Hagan, M. (2008). Neural network toolbox™ 6.User's guide.
3. Haykin, S. (1994). Neural networks: a comprehensive foundation. Prentice Hall PTR.
4. N. K. Bose and P. Liang. "Neural Network Fundamentals with Graphs,Algorithms, and Applications". Tata McGraw–Hill, New Delhi, 1998
5. R.J. Schalkoff. Artificial Neural Networks. McGraw-Hill Co., New York, 1997.
6. S.K. Basu. "A cursory look at parallel and distributed architectures and biologically inspired computing". In N. K. Sinha and M. M. Gutpta, editors, Soft Computing and Intelligent Systems Theory & ApplicationsAcademic Press, London, 2000.
7. M. Bianchini, P. Fasconi, M. Gori, and M. Maggini. "Optimal learning in artificial neural networks: A theoretical view". In C. T. Leondes, editor, Optimization Techniques, volume II of Neural Network SystemsTechniques and Applications. Academic Press, California, 1998.
8. V. Cherkassky and F. M¨ulier. Learning from Data – Concepts, Theoryand Methods. John Wiley, New York, 1998.
9. P. Chandra, "Learning in Sigmoidal Feedforward Artificial Neural Networks", Ph.D. Thesis, University School of Information Technology, Guru Gobind Singh Indraprastha University, Delhi 06, 2004.Yegnanarayana, B. Artificial neural networks. PHI Learning Pvt. Ltd.,2009
10. McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics5.4 (1943): 115-133.
11. Rosenblatt, Frank. "The perceptron: a probabilistic model for information storage and organization in the brain." Psychological review 65.6 (1958): 386.
12. Widrow, Bernard. "Generalization and information storage in network of adaline'neurons'." Self-organizing systems-1962 (1962): 435-462.
13. Skorohod, B. A. "Diffusion learning algorithms for feedforward neuralnetworks." Cybernetics and Systems Analysis 49.3 (2013): 334-346.
14. Krose, Ben, and P. A. T. R. I. C. K. van der Smagt. "An introduction to Neural Networks. 1996." Am sterdam, Amsterdam: The University of Amsterdam.
15. Haykin S., "Neural Networks-A Comprehensive Foundations", Prentice-Hall International, New Jersey, 1999.
16. Bello, Martin G. "Enhanced training algorithms, and integrated training/architecture selection for multilayer perceptron networks." Neural Networks, IEEE Transactions on 3.6 (1992): 864-875.
17. Samad, Tareq. "Back-propagation improvements based on heuristicarguments." Proceedings of International Joint Conference on NeuralNetworks. Vol. 1. 1990.
18. Solla, S. A., E. Levin, and M. Fleisher. "ªAccelerated Learning in Layered Neural Networks, º Complex Systems, vol. 2." (1988): 625-639.
19. Sperduti, Alessandro, and Antonina Starita. "Speed up learning and network optimization with extended back propagation." Neural networks 6.3 (1993): 365-383.
20. Van Ooyen, Arjen, and Bernard Nienhuis. "Improving the convergenceof the back-propagation algorithm." Neural Networks 5.3 (1992): 465-471.
21. Miniani, A. A., and Ronald D. Williams. "Acceleration of back-propagation through learning rate and momentum adaptation." Proceedings of International Joint Conference on Neural NetworksVol. 1. 1990.
22. Jacobs, R. A., "Increased rates of convergence through learning rateadaptation", Neural Networks, vol. 1, no.4, pp. 295-308, 1988.
23. Salvetti, A. and Wilamowski, B. M., "IntroducingStochastic Process within the Backpropagation Algorithm for Improved Convergence", presented at ANNIE'94 - Artificial Neural Networks in Engineering, St. Louis, Missouri, USA, November 13-16, 1994; also in IntelligentEngineering Systems Through Artificial Neural Networks vol 4, pp. 205-209, ed. C. H. Dagli, B. R. Fernandez,J. Gosh, R.T. S. Kumara, ASME PRESS, New York 1994.

24. Balakrishnan, Karthik, and Vasant Honavar. "Improving convergence of back propagation by handling flat-spots in the output layer." Proceedings of Second International Conference on Artificial Neural Networks. 1992.
25. Krogh, Anders, C. I. Thorbergsson, and John A. Hertz. "A cost functionfor internal representations." Advances in neural information processing systems. 1989.
26. Parekh, Rajesh, Karthik Balakrishnan, and Vasant Honavar. "An empirical comparison of flat-spot elimination techniques in back-propagation networks." PROCEEDINGS-SPIE THE INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING. SPIE INTERNATIONAL SOCIETY FOR OPTICAL, 1992.
27. Torvik, L. and Wilamowski, B. M., “Modification of the Backpropagation Algorithm for Faster Convergence”, presented at 1993 International Simulation Technology Multiconference November 7-10, San Francisco; also in proceedings of Workshop on Neural NetworksWNN93 pp. 191-194, 1993.
28. Hagan, M. T. and Menhaj, M., “Training feedforward networks with theMarquardt algorithm”, IEEE Transactions on Neural Networks, vol. 5,no. 6, pp. 989- 993, 1994.
29. D.W. Opitz and J.W. Shavlik, “Dynamically adding symbolically meaningful nodes to knowledge based neural networks,” Knowledge-Based Syst., vol. 8, no. 6, pp. 301–311, 1995.
30. Huyser, Karen A., and Mark A. Horowitz. "Generalization in digitalfunctions." Neural Networks 1 (1988): 101.
31. R.Reed, “Pruning algorithms- A survey.” IEEE Trans. Neural Networks,vol. 4,pp 740-747, Sept.1993.
32. Tikhonov, Andrei Nikolaevich. "Regularization of incorrectly posedproblems." Soviet Math. Dokl. Vol. 4. No. 6. 1963.
33. Geman, Stuart, Elie Bienenstock, and René Doursat. "Neural networksand the bias/variance dilemma." Neural computation 4.1 (1992): 1-58.
34. Kwok, Tin-Yau, and Dit-Yan Yeung. "Constructive algorithms for structure learning in feedforward neural networks for regression problems." Neural Networks, IEEE Transactions on 8.3 (1997): 630-645.
35. S. Fahlman and C. Lebiere. “The Cascade Correlation Architecture”, InAdvances in Neural Information Processing System 2, pages 524-532
36. Morgan Kaufman, San Mateo, CA, 1990Baum, Eric B. "A proposal for more powerful learning algorithms." Neural Computation 1.2 (1989): 201-207.
37. Parekh, Rajesh, and Vasant Honavar. "Constructive theory refinement inknowledge based neural networks." Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998IEEE International Joint Conference on. Vol. 3. IEEE, 1998.
38. Hirose, Yoshio, Koichi Yamashita, and Shimpei Hijiya. "Back-propagation algorithm which varies the number of hidden units." Neural Networks 4.1 (1991): 61-66.
39. Islam, MdMonirul, and Kazuyuki Murase. "A new algorithm to design compact two-hidden-layer artificial neural networks." Neural Networks 14.9 (2001): 1265-1278.
40. Guresen, Erkam, and GulgunKayakutlu. "Topology of Constructive Neural Network Algorithms." Global Journal on Technology 3 (2013).
41. Khashei, Mehdi, Ali ZeinalHamadani, and Mehdi Bijari. "A novel hybrid classification model of artificial neural networks and multiple linear regression models." Expert Systems with Applications 39.3 (2012): 2606-2620.
42. doCarmoNicoletti, Maria, et al. "Constructive neural network algorithms for feedforward architectures suitable for classification tasks." Constructive neural networks. Springer Berlin Heidelberg, 2009. 1-23
43. Fahlman, Scott E., and Christian Lebiere. "The cascade-correlationlearning architecture." (1989).Networks 14.9 (2001): 1265-1278.
44. Ash, Timur. "Dynamic node creation in backpropagation networks."Connection Science 1.4 (1989): 365-375.
45. Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. No. ICS-8506.CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1985.
46. Prechelt, Lutz. "Investigation of the CasCor family of learning algorithms." Neural Networks 10.5 (1997): 885-896.

47. Nechyba, Michael C., and Yangsheng Xu. "Neural network approach tocontrol system identification with variable activation functions." Intelligent Control, 1994. Proceedings of the 1994 IEEE InternationalSymposium on. IEEE, 1994.
48. Hwang, Jenq-Neng, et al. "The cascade-correlation learning: a projectionpursuit learning perspective." Neural Networks, IEEE Transactions on7.2 (1996): 278-289.
49. Lehtokangas, Mikko. "Constructive backpropagation for recurrent networks." Neural processing letters 9.3 (1999): 271-278.
50. Treadgold, Nick K., and Tamás D. Gedeon. "A cascade network algorithm employing progressive RPROP." Biological and ArtificialComputation: From Neuroscience to Technology. Springer BerlinHeidelberg, 1997. 733-742.
51. Riedmiller, Martin, and Heinrich Braun. "A direct adaptive method forfaster backpropagation learning: The RPROP algorithm." Neural Networks, 1993, IEEE International Conference on. IEEE, 1993.
52. Riedmiller, Martin, and I. Rprop. "Rprop-description and implementation details." (1994).
53. Sharma, Sudhir Kumar, and Pravin Chandra. "An adaptive slope sigmoidal function cascading neural networks algorithm." Emerging Trends in Engineering and Technology (ICETET), 2010 3rd International Conference on. IEEE, 2010.