Exploring Optimal Architecture of Multi-layered Feed- forward (MLFNN) as Bidirectional Associative Memory (BAM) for Function Approximation

Manisha Singh
M.Tech. (CSE) student
Noida Institute of Engineering & Technology
Greater Noida, UP, India

Somesh Kumar
Professor
Noida Institute of Engineering & Technology
Greater Noida, UP, India

ABSTRACT

Function approximation is an instance of *supervised learning* which is one of the most studied topics in machine learning, artificial neural networks, pattern recognition, and statistical curve fitting. In principle, any of the methods studied in these fields can be used in reinforcement learning. Multi-layered feed-forward neural networks (MLFNN) have been extensively used for the purpose of function approximation. Another class of neural networks, BAM, has also been studied and experimented for pattern mapping problems and many variations have been reported in literature. In the present study the application of back propagation algorithm to MLFNN has been proposed in such a way that feed-forward architecture behaves like BAM. Various architectures consisting of four-layers have been explored in quest of finding the optimal architecture for the example function.

General Terms

Artificial Neural Networks, Pattern Mapping

Keywords

Neural networks, Multilayered feed-forward neural network (MLFNN), Bidirectional Associative Memory (BAM), function approximation

1. INTRODUCTION

Function approximation is the task of learning or constructing a function that generates approximately the same outputs from input vectors as the process being modeled, based on available training data. The approximate mapping system should give an output which is fairly close to the values of the real function for inputs close to the current input used during learning. There exist multiple methods that have been established as function approximation tools, where an artificial neural network (ANNs) is the one which has been very popular recently. A trained neural network is expected to capture the system characteristics in their weights. Such network is supposed to have generalized from the training data, if for a new input the network produces the same output which the system would have produced. Function approximation is an instance of *supervised learning*, the primary topic studied in machine learning, artificial neural networks, pattern recognition, and statistical curve fitting. In principle, any of the methods studied in these fields can be used in reinforcement learning.

Multilayered feed-forward neural networks (MLFNNs) trained with back-propagation algorithm and the bidirectional associative memory (BAM) trained with Hebb rule are the two categories of ANN which have been extensively used for pattern mapping (and thus for function approximation) task [1-3]. Due to their design, MLFNN with hidden layers are generally used for the classification purposes and suffers with the problem of generalization when used for pattern mapping. To overcome the well-known ill-posed problem [Tikhonov and Arsenin, 1977], a radial basis function network (RBFN) and counter propagation neural network (CPN) has been used. The underlying algorithms which modify weights separate such networks from back propagation. In back-propagation, the error and corresponding weight modification are propagated backwards from the output layer to the input layer. Different training samples may exert opposite forces in increasing or decreasing the same weight, and the net effect is that weight changes are often of extremely small magnitude in networks with learning rates small enough to have some assurance of converging to a local minimum of the mean squared error.

2. RELATED WORK

Substantial work has been cited in the literature for function approximation using feed-forward neural networks (FNNs) (Cybenko, 1989; Hecht-Nielsen, 1989; Carroll and Dickinson, 1989; Hornik, 1990, 1993; Park and Sandberg, 1991, 1993; Barron, 1993). FNNs are established structures capable of approximating generic classes of functions, including continuous and bounded. The structure studied varied in the terms of number of hidden layers for a particular function and different classes of FNNs were defined for approximating different classes of functions as per a set of approximation criteria. It is well known that a two-layered FNN, i.e. one that does not have any hidden layers, is not capable of approximating generic nonlinear continuous functions (Widrow, 1990). On the other hand, five or more layer FNNs are rarely used in practice. Hence, almost all the work deals with the most challenging issue of the approximation capability of three/four-layered FNNs. A three layered architecture was explored in [4] and the results obtained remained interesting.

Multilayer feed-forward neural networks have simpler dynamics compared to feedback or recurrent neural networks [5-7] and BAMs. BAMs has limited storage and mapping capability due to the use of Hebbian learning rule. There has been reported substantial work [8]-[25] in the literature to overcome this limitation. In the high-order BAM [8], high-order nonlinearity is applied to forward and backward information flows to increase the memory capacity and improve error correction capability. The exponential BAM [9] employs an exponential scheme of information flow to exponentially enhance the similarity between an input pattern

and it's nearest stored pattern. It improves the storage capacity and error correcting capability of the BAM, and has a good convergence property. In [10], a high capacity fuzzy associative memory (FAM) for multiple rule storage is described. A weighted-pattern learning algorithm for BAM is described in [11] by means of global minimization. As inspired by the perceptron-learning algorithm, an optimal learning scheme for a class of BAM is advanced [12]-[13], which has superior convergence and stability properties. In [14], the synthesis problem of bidirectional associative memories is formulated as a set of linear inequalities that can be solved using the perceptron training algorithm. A multilayer recursive neural network with symmetrical interconnections is introduced in [15] for improved recognition performance under noisy conditions and for increased storage capacity. In [16], a new bidirectional hetero-associative memory is defined which encompasses correlational, competitive and topological properties and capable of increasing its clustering capability. Other related work can be found in [17]-[20] in which either by adding dummy neuron, increasing in number of layers or manipulating the interconnection among neurons in each layer, the issue of performance improvement of BAM is addressed. Even some new learning algorithms were introduced to improve the performance of original BAM [21]-[25].

3. PROPOSED MODEL

Various algorithms used in the methods discussed in section 2, work only in one direction i.e. forward direction and the error thus calculated is propagated backward and weights are adjusted accordingly. A new method based on Back Propagation (BP) algorithm is proposed. This works in two phases: ***forward phase*** and ***backward phase*** (now onwards **Phase-I** and **Phase-II**). Since in both the phases the error is calculated and weights are adjusted accordingly, once in forward direction and then in backward direction, therefore there is a significant improvement in convergence. In this section outline of the theory behind the proposed approach is presented.

3.1 Bidirectional Associative Memory

In Bidirectional associative memory, neurons in one layer are fully connected to the neurons of second layer. It is a hetero-associative, content-addressable memory.

If a feed-forward network with two hidden layers is created and connections among various neurons of these layers are done in both directions as shown in **Figure 1** and error minimization using back propagation algorithm is performed on this architecture, then this architecture corresponds to the BAM. Here the weights are taken symmetrical, means

$W_{ij} = W_{ji}$ for any two nodes *i* and *j*

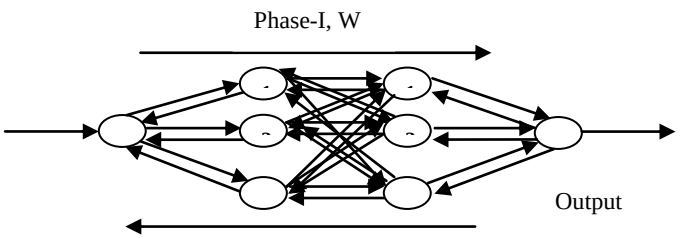

Fig 1: A typical 1-n_1-n_2-1 feed-forward architecture as Bidirectional Associative Memory

3.2 The standard BP Algorithm for 1-n_1-n_2-1 feed-forward architecture

Phase-II, W'

- *Create a feed-forward network with one input, 'h' hidden units and one output.*
- *Initialize all the weights to small random values.*
- *Repeat until* $(E_{avg} = \sum_{k=1}^{Q} E_k < \tau)$ *where* τ *is the tolerance level of error*

1. *Input pattern* X_k *from training set and compute the corresponding output* S_y
2. *For corresponding output, calculate the error as* $\delta_o = S_y (D - S_y)(1 - S_y)$
3. *For each hidden unit 'h', calculate*

$$\delta_h = S_h (1 - S_y) W_{hj} \delta_o$$

4. *Update each network weight between hidden and output layer* W_{hj} *as follows:*

$$W_{hj} = W_{hj} + \eta \Delta W_{hj} + \alpha W_{hj} old$$

where, $W_{hj} = S_y * \delta_o$

and $W_{hj} old = \Delta W_{hj}$

5. *Update each network weight between input and hidden layer* W_{ih} *as follows:*

$$W_{ih} = W_{ih} + \eta \Delta W_{ih} + \alpha W_{ih} old$$

where $W_{ih} = \delta_h * X_k$

and $\Delta W_{ih} old = \Delta W_{ih}$

3.3 The proposed two- phase BP Algorithm

Phase-I (forward phase):

The standard BP algorithm

Phase-II (backward phase):

- *Repeat until* $(E_{avg} = \sum_{k=1}^{Q} E_k < \tau)$ *where* τ *is the tolerance level of error*

1. $W_{hi} = W'_{ih}$ *and* $W_{jh} = W'_{jh}$
2. *Input pattern* D_k *for corresponding* X_k *and compute the corresponding output* S_u
3. *For corresponding output* S_u *, calculate the error as*

$$\delta_i = S_u (X - S_u)(1 - S_u)$$

4. *For each hidden unit 'h', calculate*

$$\delta_h = S_h (1 - S_h) W_{hi} \delta_i$$

5. *Update each network weight between hidden and input layer* W_{hi} *as follows:*

$$W_{hi} = W_{hi} + \eta \Delta W_{hi} + \alpha W_{hi} old$$

$$where, W_{hi} = S_h * \delta_i \; and \; W_{hi}old = \Delta W_{hi}$$

6. Update each network weight between output and hidden layer W_{jh} as follows:

$$W_{jh} = \Delta W_{jh} + \eta \Delta W_{jh} + \alpha W_{jh}old$$

$$where, \Delta W_{jh} = \delta_i * D_k \; and \; \Delta W_{jh}old = \Delta W_{jh}$$

4. SIMULATION DESIGN AND EXPERIMENTS

A four-layered feed-forward neural network has been created for the experimentation. At the input and output layers, only one neuron has been considered since the function {***sin(x)***} under consideration is a univariate function. Since there does not exist any thumb rule as far as the number of hidden layers and number of neurons therein. It is also well known that a two-layered FNN, i.e. one that does not have any hidden layers, is not capable of approximating generic nonlinear continuous functions (Widrow, 1990). On the other hand, five or more layer FNNs are rarely used in practice. Hence, almost all the work deals with the most challenging issue of the approximation capability of three/four-layered FNN. Various combinations of neurons ranging from 2 to 35 in both hidden layers are experimented. The summary of the various parameters like the learning rate, momentum etc. used is shown in **Table 1.**

At different learning rates and fixed momentum value (0.6), the convergence of the function has been studied with tolerance value .005 by both algorithms discussed in **section 3.2 & 3.3** - standard back propagation and the proposed two-phase back propagation algorithm (which let the MLFNN work as BAM). For each architecture five runs for both the above algorithms were executed and the average number of epochs was recorded at which the convergence was achieved for the taken error tolerance. The results have been tabulated in **Table 2 (a), (b) and (c).** One epoch in standard BP algorithm means when all the training patterns are presented once, while one epoch in proposed two-phase BP algorithm means when for all patterns both forward and backward phases run once. The same results has been shown graphically in **Figure 2.**

Table 1: The parameters used in simulation.

Parameters	***Number/Values***
Input Neurons	1
Hidden Neurons in hidden layer 1 ***(n_1)***	2……35
Hidden Neurons in hidden layer 2 ***(n_2)***	2……35
Number of Patterns	5
Learning rate, ***η***	0.7/0.8/0.9/1.0
Momentum, ***α***	0.6
Tolerance, ***τ***	0.005

Table 2: Number of epochs required for convergence for approximating the function y=sin(x)

(a) Case I: 1-n_1-n_2-1, where n_1<n_2

Architectures	*Learning rates*							
	η=0.7		***η=0.8***		***η=0.9***		***η=1.0***	
	FFN	BAM	FFN	BAM	FFN	BAM	FFN	BAM
1-2-3-1	9692	1191	5467	857	6459	1048	5600	901
1-2-4-1	7016	919	4847	910	3292	1121	8426	911
1-2-5-1	4837	965	4842	798	3573	714	4046	863
1-3-4-1	5238	738	3908	667	4260	597	5120	692
1-3-5-1	4284	734	4758	657	8299	569	5717	631
1-4-5-1	3443	687	3813	583	4257	489	2817	537
1-5-10-1	2242	508	3331	523	2543	411	1997	417
1-10-15-1	1267	375	1013	372	1713	340	1042	270
1-15-20-1	843	313	816	255	690	246	632	251
1-20-25-1	678	235	560	236	591	203	473	223
1-25-30-1	542	227	606	203	520	200	463	191
1-30-35-1	576	204	540	295	448	185	511	364

(b) Case II: 1-n_1-n_2-1, where $n_{1>}n_2$

Architecture	*Learning rates*							
	η=0.7		***η=0.8***		***η=0.9***		***η=1.0***	
	FFN	BAM	FFN	BAM	FFN	BAM	FFN	BAM
1-3-2-1	5472	1146	5109	1490	8954	765	3315	1250
1-4-2-1	5201	1466	8167	779	3214	954	3913	925
1-4-3-1	4054	774	4576	1034	6044	745	3408	549
1-5-2-1	4156	767	4369	898	6536	743	2457	760
1-5-3-1	3144	760	5785	729	2850	718	3399	728
1-5-4-1	5673	732	4789	722	4432	626	2337	715
1-10-5-1	1625	464	1578	488	1475	426	1489	442
1-15-10-1	1204	355	886	362	1090	294	808	276
1-20-15-1	799	274	741	298	583	269	687	228
1-25-20-1	688	270	521	242	548	201	436	213
1-30-25-1	571	229	461	223	466	200	374	192
1-35-30-1	493	287	431	275	407	232	419	363

(c) Case III: 1-n_1-n_2-1, where n_1=n_2

Architecture	*Learning rates*							
	η=0.7		***η=0.8***		***η=0.9***		***η=1.0***	
	FFN	BAM	FFN	BAM	FFN	BAM	FFN	BAM
1-2-2-1	7993	1186	7124	1311	8473	906	6202	1123
1-3-3-1	4134	982	5613	804	3778	807	4609	658
1-4-4-1	5614	806	5550	635	5008	669	3556	567
1-5-5-1	4168	560	2269	581	3138	561	2122	499
1-10-10-1	1986	420	1202	403	1425	363	1058	345
1-15-15-1	986	353	903	308	685	268	679	255
1-20-20-1	757	267	703	246	614	227	533	191
1-25-25-1	819	211	537	223	485	213	456	185
1-30-30-1	522	219	516	204	488	198	445	182
1-35-35-1	446	244	437	287	482	346	513	307

5. RESULTS AND DISCUSSION

The results shown in **Table 2(a), (b),** and **(c)** clearly indicate that the proposed two-phase back propagation algorithm for BAM architecture, in general, outperform the MLFNN architecture with standard back propagation algorithm. **Table 2(a)** shows the results for the architectures where first hidden layer has lesser number of neurons compared to the second hidden layer (**1-n_1-n_2-1, where n_1<n_2**), while **Table 2(b)** contains for the other case where the number of neurons in first hidden layer are greater than those of second hidden layer (**1-n_1-n_2-1, where n_1>n_2**) while **Table 2(c)** is for the case (**1-n_1-n_2-1, where n_1=n_2**). In every case, the number of epochs taken for the convergence of the taken function is substantially low in the case of proposed two-phase BP algorithm, in almost every architecture studied. It is observed that the number of epochs kept on lessening when one move on increasing the number of neurons in the hidden layers and various combinations are explored. And the best architecture is when the number of neurons in the two hidden layers is between 30 and 35, which can be seen that at the architecture 1-35-35-1, the number of epochs required for convergence starts increasing. Also there are some architectures in between where a larger number of epochs were recorded for convergence (e.g. 1-2-4-1, 1-4-2-1 etc.). Since both the algorithms work on the randomness of initial weight matrices, this may be one of the reasons for such results. Another obvious observation comes out that on increasing the learning rate from 0.7 to 1.0 the convergence is being achieved in lesser number of epochs, in general.

The results show that the best suited BAM architecture for the taken function ***sin(x)*** is 1-30-30-1 with learning rate 1.0 and momentum 0.6.

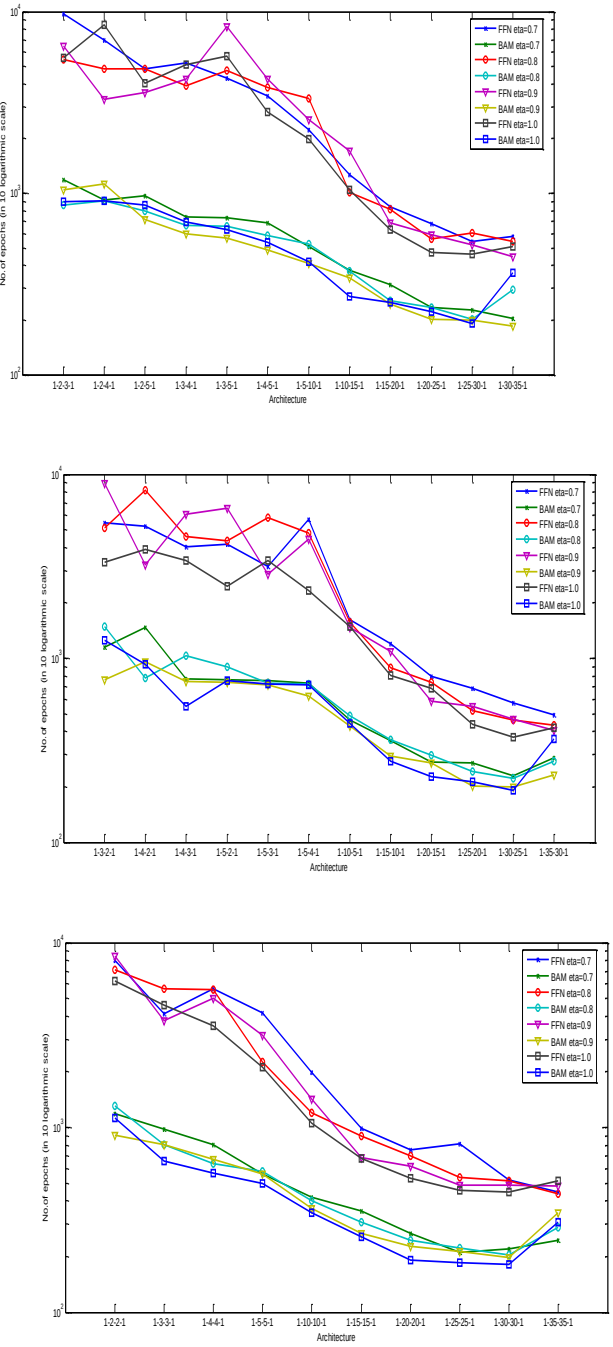

Fig. 2: ***Comparative graphs of the number of epochs required for convergence in MLFNNs and BAM for the architectures*** ***(i)*** **1-n_1-n_2-1, where n_1<n_2** ***(ii)*** **1-n_1-n_2-1, where n_1>n_2** ***(iii)*** **1-n_1-n_2-1, where n_1=n_2**

6. CONCLUSION AND FUTURE SCOPE

A new variation of back propagation algorithm, the Two-Phase BP, has been presented which when applied to multi-layered feed-forward neural network (MLFNN) let it behave like bidirectional associative memory (BAM). The four-layered architectures having various combinations of neurons in the two hidden layers have been experimented for a single valued-function. The results presented above suggest that proposed algorithm is better suited for function approximation and results obtained are quite encouraging. In future, the proposed algorithm can be applied for approximating variety of functions and the detailed results thus obtained may be compared with traditional algorithms and the effect on convergence could be studied.

7. REFERENCES

[1] Rumelhan D.E., Hinton G.E., and Williams R. J**.,** "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing*, Vol. I Foundations, MIT Press, Cambridge, MA, pages 318-364. 1986.Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.

[2] Tang C. Z**.** and Kwan H**.** K., "Parameter effects on convergence speed and generalization capability of backpropagation algorithm". *International Journal of Electronics***,** 74(1): 35-46, January 1993.

[3] Kosko B., *Neural Networks and Fuzzy Systems***,** Prentice-Hall. Inc., New Jersey, 1992.

[4] Singh, M. and Kumar, S., "Using Multi-layered Feed-forward Neural Network (MLFNN) Architecture as Bidirectional Associative Memory (BAM) for Function Approximation", *IOSR Journal of Computer Engineering (IOSR_JCE)*, vol. 13, Issue 4, pp 34-38

[5] Pearlmutter B.A., "Gradient calculations for dynamic recurrent neural networks: a survey". *IEEE Trans. On Neural Networks***,** 6 (5):1212-1228, Sept. 1995.Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.

[6] Prokhorov D.V., Feldkamp L.A., and Tyukin I.Y**.,** "Adaptive behavior with fixed weights in RNN: an overview". *Proceedings* of *International Joint Conference on Neural Networks, 2002*, pages 201 8-2022.

[7] Kwan H. K. and Yan J., "Second-order recurrent neural network for word sequence learning", *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing***,** Hong Kong, May 2-4. 2001, pages 405-408.

[8] Tai H.-M.. Wu C.-H., and Jong, T.-L., 'High-order bidirectional associative memory': *Elecrronics krlers*, 25(21):1424-1425, 12th Oct. 1989.

[9] Jeng Y.-J., Yeh C.-C., and Chiueh T.D., 'Exponential bidirectional associative memories", *Electronics Letters*, 26(11):717-718,24th May1990.

[10] Chung F.L. and Lee Tong, **'**On fuzzy associative memory with multiple-rule storage capacity". IEEE Trans. on *Fuzzy Systems***,** 4(3):375-384, August 1996.

[11] Wang T.. Zhuang X., and Xing X., 'Weight learning of bidirectional associative memories by global minimization", *IEEE Trans. on Neural Networks*, 3(6):1010-1018, Nov. 1992.

[12] Shanmukh K**.** and Venkatesh Y.V., 'Generalized scheme for optimal learning in recurrent neural networks': *IEE Proc: Vision, Image, and Signal Processing*. 142(2):71-77, April 1995.

[13] Salih, I. Smith, S.H., and Liu**,** D., 'Design of bidirectional associative memories based on the perceptron training technique". *Proceedings* ***of*** *IEEE International Symposium* ***on*** *Circuits and Systems*, 1999, Vol. ***5,*** pages 355-358.

[14] Kwan H. K**.** and Tsang P. C., 'Multi -layer recursive neural network': *Proceedings of Canadian Conference on Electrical and Computer Engineering, Montreal, Canada*, September 17-20, 1989, Vol. **1.** pages 428-431.

[15] Widrow B**.** and Winter R., 'Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition'', *IEEE Computer Magazine*, pages 25-39. March 1988.

[16] Chartier S., Giguere G. and Langlois D., "A new bidirectional heteroassociative memory encompassing correlational, competitive and topological properties", *Neural Networks 22* (2009), pp 568-578, 2009

[17] Wang Y.F., Cruz J.B. and Mulligan J.H., "Two coding strategies for bidirectional associative memory", *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp 81-92, 1990

[18] Wang Y.F., Cruz J.B. and Mulligan J.H., "Guaranteed recall of all training pairs for bidirectional associative memory", *IEEE Trans. Neural Networks*, vol. 2, no. 6, pp 559-567, 1991

[19] Kang H., "Multilayer associative neural network (MANN's): Storage capacity versus perfect recall*", IEEE Trans. Neural Networks*, vol. 5, pp 812-822, 1994

[20] Wang Z., "A bidirectional associative memory based on optimal linear associative memory", *IEEE Trans. Neural Networks*, vol.45, pp 1171-1179, Oct.1996

[21] Hassoun M.H. and Youssef A.M., "A high performance recording algorithm for Hopfield model associative memories", *Opt. Eng*., vol. 27, no. , pp 46-54, 1989

[22] Simpson P.K., "Higher-ordered and interconnected bidirectional associative memories", *IEEE Trans. Syst. Man. Cybern*., vol. 20, no. 3, pp 637-652, 1990

[23] Zhuang X., Huang Y. and Chen S.S., "Better learning for bidirectional associative memory", Neural Networks, vol.6, no.8, pp1131-1146, 1993

[24] Oh H. and Kothari S.C., "Adaptation of the relaxation method for learning bidirectional associative memory*", IEEE Trans. Neural Networks*, vol.5, pp 573-583, July 1994.

[25] Wang C.C. and Don H.S., "An analysis of high-capacity discrete exponential BAM", *IEEE Trans. Neural Networks*, vol. 6, no. 2, pp 492-496, 1995.