Proceedings of the 9th International Conference on Neural Information Processing (ICONIP'02), Vol. 1
Lipo Wang, Jagath C. Rajapakse, Kunihiko Fukushima,
Soo-Young Lee, and Xin Yao (Editors)

FINDING OPTIMAL ARCHITECTURE AND WEIGHTS USING EVOLUTIONARY LEAST SQUARE BASED LEARNING

Ranadhir Ghosh and Brijesh Verma

School of Information Technology
Griffith University, PMB 50, Gold Coast Mail Center
QLD 9726, Australia
E-mail: {r.ghosh, b.verma}@gu.edu.au

ABSTRACT

In this paper, we present a novel idea of implementing a growing neural network architecture using an evolutionary least square based algorithm. This paper focuses mainly on the following aspects, such as the heuristics of updating weights using an evolutionary least square based algorithm, finding the number of hidden neurons for a two layer feed forward multi-layered perceptron (MLP), the stopping criteria for the algorithm and finally comparisons of the results with other traditional methods for searching optimal or near optimal solution in the multidimensional complex search space comprising the architecture and the weight variables. We applied our proposed algorithm for XOR data set, 10 bit odd parity problem and many real bench mark data set like handwriting dataset from CEDAR and breast cancer, heart disease data set from UCI ML repository. The comparison results, based on classification accuracy and the time complexity are discussed. We also discuss the issues of finding a probabilistic solution space as a starting point for the least square method and address the problems involving fitness breaking .

KEY WORDS

ANN, evolutionary algorithm, least square method

1 INTRODUCTION

1.1 Background

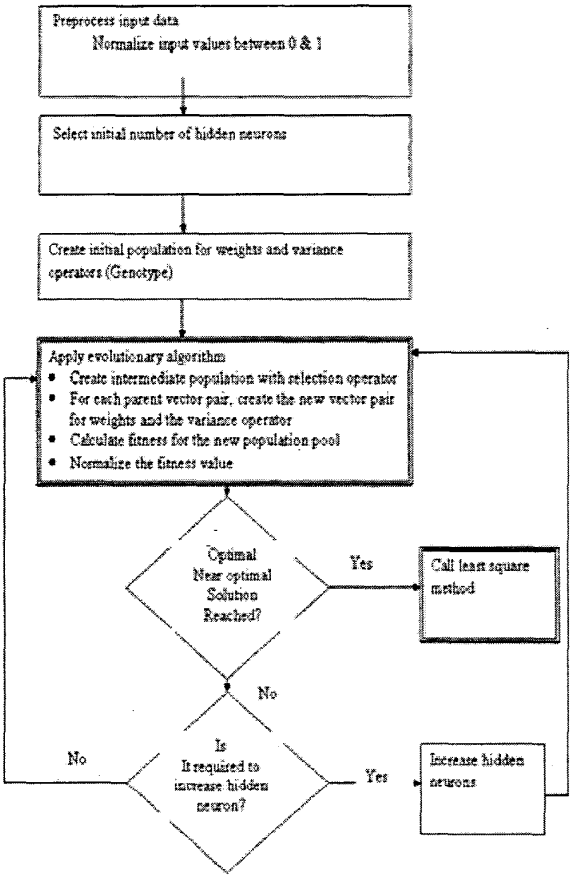
The aspect of learning for artificial neural network (ANN) has always been a major challenge to the researchers due to its various complexities and trade off characteristic for classification accuracy and time complexity. Quite often the problem becomes multitude with the additional problem of generalization ability. The most popular learning algorithms use the concept of gradient descent. Still after few decades of active research in the ANN learning area, one of the most popular weight-updating rule or learning (training) algorithms is Error BackPropagation (EBP). However, most of the EBP based neural learning algorithms including EBP strictly depends on the architecture of the ANN and there are many problems associated with the currently existing algorithms based on EBP and its variations [2-5]. There were a number of hybrid techniques proposed to improve EBP type learning algorithms by using least square methods (LSM), evolutionary algorithms (EA), etc. [6-11].

Earlier work by Verma and Ghosh [1], suggested an alternative learning methodology, which uses a hybrid technique by using evolutionary learning for the hidden layer weights and least square based solution method for the output layer weights. However the suggested algorithm could only modify the weights, hence a topology of the ANN architecture is obviously an area of concern. The other problem reported for GALS was of its high memory complexity nature. It was shown in some preliminary study that with an input matrix of order 1500 X 100 (row and column respectively) on a pc with 128 MB RAM and CPU speed of 512 MHz , the memory allocation was a problem to call the least square solution routine.

The main aim of the research presented in this paper was to investigate a growing neural network architecture for an evolutionary hybrid learning for GALS and conduct a comparative study between the existing learning algorithms that modifies the weights and architecture of the ANN using evolutionary technique with our new proposed algorithm, and then the earlier proposed algorithm was modified further to decrease the memory complexity. Some simulation results were analyzed to find a proper range of weights as a starting for the hidden layer weights, before applying the evolutionary algorithm.

2 RESEARCH METHODOLOGY

Following is the flowchart for the overall algorithm[1]



2.1 Modification of GALS

To improve the memory complexity of our original GALS [1], we call the least square method after the convergence property of the evolutionary algorithm is over. Earlier, the algorithm used to find the best set of weights from the initial generation of the population pool. In that case for n number of population the least square method was called n times. This problem was overcome by calling it after the convergence of the evolutionary algorithm. This could have lead to the risk of a potential problem of fitness breaking of the chromosome. We did some experiment based on the rank of the population pool to test whether breaking the chromosome into two halves for calling the least square method causes any major setback to the fitness of the gene. Following, we describe the stopping criteria for the convergence of the evolutionary algorithm.

If (best_RMS_error[2] < goal_RMS_error) then
    Stop
Else if (number_of_generation = total_number_of_generation[3]) then
    Stop
Else if (train_classification_error is increased in #m[4] consecutive generation) then
    Stop

2.2 Finding optimal number of hidden neurons

We used two different types of experiments – Linear incrementing for GALS (LIGALS) and binary tree search type for GLAS (BTGALS) to find the number of hidden neurons-

1. Starting with a small number, and then incrementing by 1 (LIGALS)
2. Using a binary tree search type (BTGALS)

2.2.1 Experiment A (LIGALS)

In experiment A, we start with a small number of hidden neurons and then increment by one. The stopping criteria for this experiment was as follows:

If (train_classification_error = 0) then
    Stop
Else If (the test classification error is high in #n[5] consecutive generation) then
    Stop

2.2.2 Experiment B (BTGALS)

In experiment B, we use a binary tree search type to find the optimal number. The pseudo code of the algorithm is given below:

Step 1: Find the % test classification error & train_classification_error (error_min) for #min number of hidden neurons
    error_min = (train_classification_error (%)+ test_classification_error (%)) / 2

Step 2: find the % test classification error & train classification error (error_max) for #max number of hidden neurons

[1] The details stepwise algorithm can be found in our earlier work [1].

[2] The best_RMS_error is the best of the RMS error from the population pool

[3] Total number of generations is considered as 30

[4] m is considered as 3

[5] We use n = 3 for our experiments, which was determined by trial and error method

error_max = (train_classification_error (%) + test_classification_error (%)) / 2

Step 3: Find the % test classification error & train classification error (error_mid) for #mid (mid = (min + max) / 2) number of hidden neurons

error_mid = (train_classification_error (%) + test_classification_error (%)) / 2

Step 4: if (error_mid < error_min) and (error_mid > error_max) then

min = mid
mid = (min + max / 2)
else
max = mid
mid = (min + max / 2)
end if

Step 5: Go to Step1, if (mid > min) and (mid < max)
Else go to Step 6

Step 6: #number of hidden neurons = mid

3 EXPERIMENTAL RESULTS

Table 1 shows the result of the fitness breaking for the chromosome based on their ranks. We consider for handwriting data set of pattern length 100. After every generation of the evolutionary algorithm, we calculate the fitness for the population pool, as calculated by our evolutionary algorithm. Then we calculate the fitness of the gene by breaking it into two halves, and taking the first halves and then combining with the output layer weights (by calling the least square function). The comparisons of the fitness values are reported. We only consider first 10 ranking of the population pool from the set of 50. The result shows that the fitness of the gene (specially for the chromosome with very high fitness) are not affected much when it is broken and combined with the least square method.

Table 1 Analysis of fitness breaking for calling least square method

Generation	Fitness for full length gene		Fitness for half length gene with least square	
	Population Number	Ranking	Population Number	Ranking
1	10	1	10	2
	11	2	11	1
	9	3	9	3
	33	4	33	5
	37	5	37	7
	1	6	1	4
	41	7	41	6
	36	8	36	8
	17	9	17	9
	19	10	19	14
2	17	1	17	1
	3	2	3	2
	8	3	8	4
	1	4	1	3
	46	5	46	6
	47	6	47	10
	11	7	11	8
	19	8	19	7
	32	9	32	13
	17	10	17	9

Table 2 shows the effect on finding the inverse function for QR decomposition method, which is used to find the weight matrix. The other simulation results suggested that the difference of order in hidden matrix and the difference in range gives a better result for calculating the inverse function for the R matrix., which in turns becomes the weight matrix. We use handwriting data set as input. The length of the pattern was 100. After initializing the hidden layer weights with specific range as closed interval, the output of the hidden neurons were tested. The experimental result shows that an initial close range of (0.01 –0.09) was the best choice for initialization of the hidden layer weights.

Table 2 Analysis of initial weight range for the hidden layers

Range	Max	Min	Std. dev
0 – 9	1	1	0
0.1 – 0.9	1	0.9979	0.000156
0.01 – 0.09	0.83648	0.64958	0.0337
0.001 – 0.009	0.541417	0.515473	0.00455
0.0001- 0.0009	0.706357	0.5706	0.0237

The length of training and testing for all the data set was considered as described in table 3.

Table 3 Length of pattern for training & testing data set

Data set	Training length	Testing length
XOR	4	2
10 bit odd parity	900	124
Handwriting data set[6]	1000	200

[6] For handwriting data set, features with vector length 100 was used as input to the ANN after extracting from

Breast cancer	400	299
Heart disease (Cleveland)	250	53

Tables 4, 5 & 6 show the results for training & testing RMS and classification error using EBP and the GALS (applying two different stopping criteria)[7]. The time complexity of all the algorithms was found by the total time taken by each algorithm to find the optimal number of hidden neurons and the learning.

Table 4 RMS & Classification error results for EBP[8]

Data set	Train Error		Test Error		#Hidden	Time
	RMS	Class (%)	RMS	Class (%)		
XOR	0.03	0	0.02	50	3	13m[9]
10 bit odd parity	0.368	9	0.042	19	23	80m
Handwriting data set	0.537	18	0.154	34	27	128m
Breast cancer	0.342	19	0.083	32	19	87m
Heart disease	0.317	11	0.068	23	14	68m

Table 5 RMS & Classification error results for GALS (LIGALS)

Data set	Train Error		Test Error		#Hidden	Time
	RMS	Class (%)	RMS	Class (%)		
XOR	0.003	0	0.002	25	2	5m
10 bit odd parity	0.068	9	0.009	21	56	67m
Handwriting data set	0.048	11	0.046	17	101	113m
Breast cancer	0.058	14	0.037	21	46	82m
Heart disease	0.048	9	0.005	12	41	77m

Table 6 RMS & Classification error results for GALS (BTGALS)

Data set	Train Error		Test Error		#Hidden	Time
	RMS	Class (%)	RMS	Class (%)		
XOR	0.003	0	0.002	25	2	5m
10 bit odd parity	0.059	10	0.009	22	48	51m
Handwriting data set	0.061	13	0.037	16	86	92m
Breast cancer	0.047	11	0.036	19	38	77m
Heart disease	0.039	8	0.003	13	34	64m

Figure 1 shows the comparison of classification accuracy for the training data set. The graph shows that, in terms of classification accuracy GALS outperforms the traditional EBP.

Figure 1 Classification accuracy for training data set

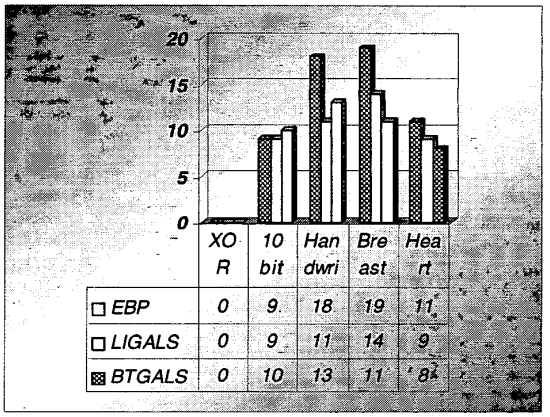


Figure 2 shows the comparison of classification accuracy for the testing data set. The graph shows that, in terms of time complexity GALS outperforms the traditional EBP in almost all the cases.

the input image files for upper case characters (A-Z) using chain code feature extractor.

[7] All the experiments were conducted on SP2 supercomputer at Griffith University, which consists of eight RS/6000 390 machines and 14 RS/6000 590 machines connected by a high speed switch.

[8] The training time for all the algorithms was considered within some specific range for comparison purpose. The training for EBP was forcibly stopped after it crossed the time range limit from the other two algorithms.

[9] m denotes minutes

Figure 2 Classification accuracy for testing data set

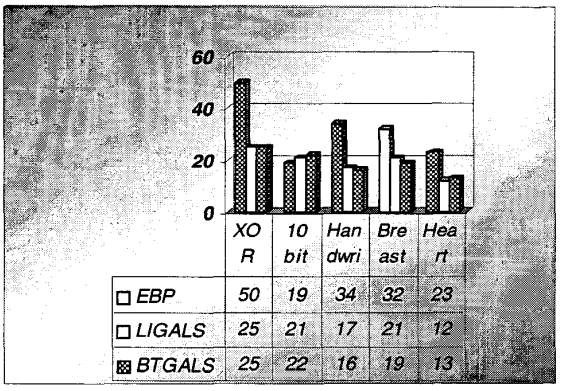
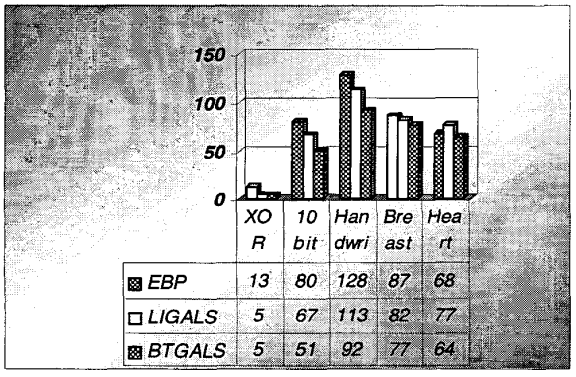


Figure 3 shows the comparison of time complexity for all the data set. The result shows that in terms of time complexity also, GALS is slightly better than the traditional EBP.

Figure 3 Time complexity for all the data set



4 CONCLUSION AND FURTHER RESEARCH

In this paper, we proposed a novel approach for finding optimal number of hidden units. We have discussed with experimental results, how to achieve that with two different stopping criteria approach. We also improved our earlier GALS algorithm, to reduce the memory complexity & also the choice of parameter setting for the algorithm. From the experimental results, we show that the new approach outperforms some other traditional approaches in terms of its ability of better classification accuracy & time. It should be noted here that had we considered a fully such automated system for EBP, the time complexity would have even be higher. When compared to the traditional EBP, in terms of required number of hidden neurons, it is shown that GALS requires more number of hidden neurons. In future, we would like to improve the memory complexity of the algorithm further by introducing a clustering technique for the feature vector of the input data set. So that the training can be performed for not the whole data set but for the data set equal to the number of classes, where each of the vector will be representing a single class.

5 REFERENCES

[1] B. Verma and R. Ghosh, "A Novel evolutionary Neural Learning Algorithm", WCCI 2002, CEC 7302, pp. 1884-89, Honolulu, Hawaii, USA, 2002.

[2] P. Whittle, "Prediction and regularization by linear least square methods", Van Nostrand, Princeton, N.J., 1963.

[3] S. D. Goggin, K.E. Gustafson, and K. M. Johnson,"An asymptotic singular value decomposition analysis of nonlinear multilayer neural networks," International Joint Conference on Neural Networks, pp. I-785-I-789, 1991.

[4] S. A. Burton, "A matrix method for optimizing a neural network," Neural comput. Vol 3, no 3.

[5] S. Lawrence, C.L. Giles, Ah Chung Tsoi, "What size neural network gives optimal generalization? Convergence properties of backpropagatioin", UMIACS-TR-96-22.

[6] V. Petridis, S.Kazarlis, A.Papaikonomu and A.Filelis. "A hybrid genetic algorithm for training neural network". Artificial Neural Networks, 2, pp. 953-956, 1992.

[7] D. Whitley, T. Starkweather, & C. Bogart" Genetic algorithms and neural networks - optimizing connections and connectivity". Parallel Computing, vol. 14, pp. 347-361, 1990.

[8] D. Montana, & L. Davis, "Training feedforward neural networks using genetic algorithms". Proceedings of the Eleventh International Joint Conference on Artificial Intelligence IJCAI-89, 1, 1989.

[9] D.R. Hush and B.G.Horne, "Progress in supervised neural networks," IEEE signal processing Magazine, vol. 10, no. 1, pp. 8-39, Jan 1993.

[10] M.F. Moller, "A scaled conjugate gradient algorithm for fast supervised learning," Neural networks, vol. 6, pp. 525-523, June 1993.

[11] A.P. Topchy and O.A .Lebedko, "Neural network training by means of cooperative evolutionary search," Nuclear Instruments & Methods in Physics Research, Section A: Accelerators, Spectometers, Detectors and Associated Equipment, vol. 389, no. 1-2, pp. 240-241, 1997.