We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

14

Hybrid Approaches Tabu Learning Algorithm Based Neural Network

Junfei Qiao, Jian Ye and Honggui Han
Beijing University of Technology
China

1. Introduction

Tabu search is a global search algorithm which is popular in recent years [F. Glover, 1989, 1990, 1997]. The main principle of tabu search is that it has some memory of the states that has already been investigated and it does not revisit those states. It considers the set of all possible neighbor states and takes the best one, but it will also take the best move in the neighborhood which might be worse than the current move. The tabu search focuses greatly on expanding the global search area and avoiding the search of the same area. It can always get much better global solutions. The tabu search uses a tabu list to memorize the visited states and keep from recurrent search. Aspiration criterion is set to activate the "tabued state" in the tabu list around which some good global states may be found [D. Cvijovic, 1995].

In the past decade, there has been a growing interest in applying neural network to many areas of science and engineering, such as pattern identification and image management [Jianming Lu, 2007], control[Jian-Xin Xu, 2007] and optimize[Z. S. H. Chan, 2005], communication [Kung S Y, et al. 1998] and so on. Basically, neural network is the computing system characterized by the ability to learn from examples rather than having to be programmed in a conventional way as used in control engineering [K.J. Astrom, B, 1989]. The broad use of neural network in many areas derived from its ability of approximating nonlinear functions. In theory, it has been proved that a three-layered neural network can approximate unknown functions to any degree of desired accuracy [K. Funahashi.1989; and K. Hornik, 1989].

This chapter is focused on the tabu learning algorithm based on neural network in which an unknown function is approximated. The input of the network is given by the values of the function variables and the output is the estimation of the function. In mathematical terms, the objective is to find appropriate values for the weights of the net which approximate the function best.

Gradient-based algorithms, especially the back propagation (BP) algorithm [L.M. Salchenberger, 1992] [P. Werbos, 1993] and its revised version [R. Parisi, 1996; and G. Zhou, 1998], are well known as a type of supervised learning for multilayered neural networks. The method of gradient descent is that a maximal "downhill" movement will eventually reach the minimum of the function surface over its parameter space by moving to the direction of the negative gradient.

Source: Local Search Techniques: Focus on Tabu Search, Book edited by: Wassim Jaziri, ISBN 978-3-902613-34-9, pp. 278, October 2008, I-Tech, Vienna, Austria

However, this method has several inevitable drawbacks. First, it is prone to getting trapped in local minima of the error surface. Second, the training performance is sensitive to the choice of the learning rate and initial values of weights. For these reasons, many researches have been done to overcome these drawbacks especially the local convergence [Y. Izui, 1990 ; and S. Ma,1998].

Recently, some researches were done in the tabu learning algorithm [Junfei Qiao, et al. 2006]. In this chapter, we utilize the tabu search in the NN learning process to help the gradient technique "jump out of" the local minima and get a great improvement on the gradient technique. In order to test the ability of the tabu search to improve the NN learning, we introduce the most basic NN learning algorithm, BP algorithm, as the NN learning algorithm. The tabu search can also be combined with other improved learning algorithm.

The remainder of this chapter is organized as follows. Section 2 first describes the typical artificial neural network including the architecture and the learning algorithm. Then the target function used in this paper approximating the nonlinear function is introduced. In section 3, the tabu-based neural network learning algorithm, TBBP, is described. Section 4 illustrates the experiment and the result. In this section, both of the TBBP and BP are tested to approximate 6 different nonlinear functions. Eventually, section 5 gives the conclusion.

2. Neural network and target function

2.1 Neural network

We use the most employed architecture of NN in this section: a multilayer feed-forward network with a single hidden layer. The schematic representation of the neural network is showed in Fig. 1.

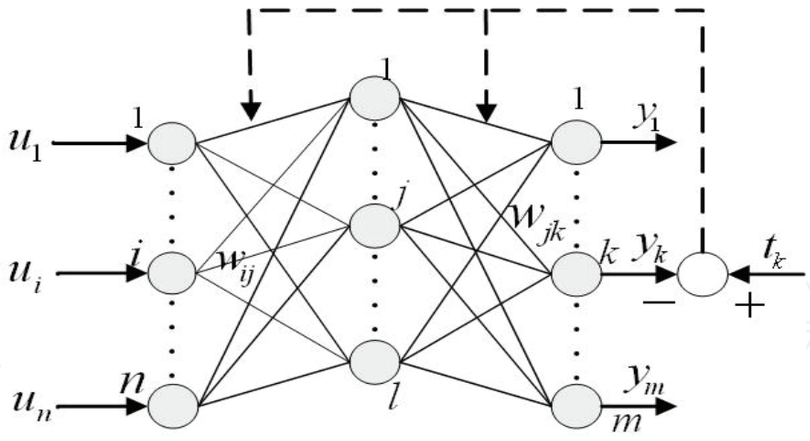


Fig. 1. The structure of the three-layered feed forward ANN

So the NN is presented as (N, W), where N is the set of nodes and W is the set of weights. There are 3 parts in N : N_I , input nodes; N_H , hidden nodes; N_O ,output nodes. If there are n variables in the nonlinear function that we want to approximate, $N_I = n$ .The neural network has l hidden neurons with a bias term in each hidden neuron. In the output layer, there are m nodes. To approximate the nonlinear function, there only has one single node in

the output layer as the approximated value of the function. In Fig.1, w_{ij} connects the i-th node in the input layer and the j-th node in the hidden layer. w_{jk} connects the j-th node in the hidden layer and the k-th node in the output layer.

Each node i in the input layer has a signal u_i as the input of the hidden layer. Each node j in the hidden layer receives a signal $In(j)$ according to

$$In(j) = \theta_j + \sum_{i=1}^{n} u_i \omega_{ij} \tag{1}$$

The most popular transform function in the hidden layer is the standard sigmoid function $f(x) = e^x/(e^x + 1)$, which is different from the linear function in both the input layer and the output layer. The output of the hidden node is $f(In(j))$. The output of the NN is

$$y_k = \theta_k + \sum_{j=1}^{m} \omega_{jk} f(In(j)) \tag{2}$$

Where θ_l and θ_m are the bias terms of the hidden layer and the output layer.

2.2 Target function

The process of determining the values of the weight parameters of the NN on the basis of a given data set is called supervised learning or training. The data set should contain a representative and sufficient number of input-target relations to allow an appropriate representation of the complex functional mapping of the problem. In the process of training the net, the purpose is to search the values of the weights that minimize the error across the training set. Once the optimization has been performed and the weights have been got, the NN is ready to approximate the function. The error function used in the optimization problem is called the target function.

The target function used in this paper is sum square error (sse) $E(w)$, which is:

$$E(w) = \frac{1}{2} \sum_{q=1}^{m} (T_q - Y_q)^2 \tag{3}$$

Where $Y_q = [y_1, y_2 \cdots y_k \cdots y_m]$ is the q-th output of the NN and $T_q = [t_1, t_2 \cdots t_k \cdots t_m]$ is the q-th target value. In Fig.1, $u_q = [u_1, u_2 \cdots u_i \cdots u_n]$ is the q-th input training data of the training data $\{\{u_1, T_1\}, \{u_2, T_2\} \cdots \{u_q, T_q\} \cdots \{u_n, T_n\}\}$. M is the number of training data.

The gradient descent method searches for the global optimum of the network weights. Each iteration t consists of two steps. First, partial derivatives $\partial E/\partial w$ are computed for each weight in the net. This can be exactly done by starting the computation at the output node k, and then successively computing the derivatives for the weights from w_{jk} to w_{ij} (the gradient information is successively moved from the output layer back towards the input layer). In the second step, weights are modified according to the expression:

$$w(t+1) = w(t) - \alpha \partial E(t) / \partial w(t) \tag{4}$$

Where the α is the learning rate. The error is computed for the new weights, t is increased in one unit and a new iteration begins.

In (3), W is the weight vector of the NN including the weights from the input to the hidden layer, the weights from the hidden to output layer, and the bias factor of the hidden and output nodes. W is represented as:

$$W = \left\{ \begin{bmatrix} \omega_{11} & \cdots & \omega_{1l} \\ \vdots & \ddots & \vdots \\ \omega_{n1} & \cdots & \omega_{nl} \end{bmatrix} \right\}, \begin{bmatrix} \omega_{11} & \cdots & \omega_{1m} \\ \vdots & \ddots & \vdots \\ \omega_{l1} & \cdots & \omega_{lm} \end{bmatrix}, (\theta_1, \theta_2 \cdots \theta_l), (\theta_1, \theta_2 \cdots \theta_m) \tag{5}$$

$E(w)$ can be treated as a nonlinear function of W for the same training data. Because the transform functions in the neural network is continuous, $E(w)$ is in a continuous multidimensional space of W. There are many concaves in this continuous space. The learning algorithm is to find the states at the bottom of the concaves which are the local optimal solutions of the function $E(w)$.

The traditional learning algorithms such as BP which is based on the gradient technique plunge into the local optimization inevitably.

3. The tabu based neural network learning algorithm

3.1 The tabu search in the NN learning

The tabu search (TS) which can be regarded as an iterative descent method is a global search algorithm. In the tabu search, the tabu list and the aspiration criterion are the most important factors for the tabu search to implement the memory and keep from the recurrent search. Tabu list (TL) is used to keep from recurrent search and the aspiration criterion (AC) is used to restart the search in the area in which there may be some superior solutions which can be found in the former chapters.

The simple Tabu Search method is mainly based on a random search. In the neural network learning, we use the weight vector W to compare to be the tabu object and mainly research about the weights. Neighbourhoods are randomly drawn points from uniform distribution. The neighborhood is a restricted region for each weight in the current weight. Hundreds of weights are randomly generated in this region and the best one is chosen. These weights are examined. The best solution in the neighborhood replaces the initial ones and the process is repeated. The weights in the tabu list represent the states which have been searched. The tabu list is generated by adding the last solution to the beginning of the list and discarding the oldest solution from it. If the new generated weight is not in the Tabu list (TL), the search goes on and this weight is added to the Tabu list (TL). To be rejected, all the weights of a new solution would need to be within a tabu area for any of the solutions in the tabu list (TL). The AC is used to activate the states that are tabued but around which there are some superior solutions. If there is one weight in the TL but it satisfies the AC, the complete test is also applied to this "tabu" weight.

3.2 The tabu based neural network Learning algorithm (TBBP)

The TBBP divides the learning algorithm into two steps, the superficial search (SS) and the deep search (DS). The superficial search mainly finds weights which seem to be big probability of being the "good global solution" in large numbers of neighbour solutions. The deep search trains these "good global solutions" to the end and gets the finial weights used to approximate the function.

The location of the original state has a great effect on the performance of algorithm. The original weight W_0 is randomly generated. If W_0 is trained directly with BP from the original state to the end, it can only get a local optimal solution which is greatly based on the original state W_0. From these different W_0, only some random local optimal solutions are reached. The SS is implemented to filtrate the initial solutions and pick out some good solutions for the deep search. In addition, there is such a kind of weight $W_{01}, W_{02} \cdots W_{0n}$ which are in the same concave of the $E(W)$ space. If the weights $W_{01}, W_{02} \cdots W_{0n}$ are directly searched to the end, the same concave are searched n times and only the same local optimal solution is got. Lots of time is wasted during the recurrent search. The TBBP uses the SS to keep from recurrent search in the same concave.

The superficial search trains the original weight W_0 to a state $W_0^{'}$ which has a depth but is not at the bottom of the concave. $W_0^{'}$ is defined to be the superficial state. TBBP put these superficial states into the tabu list as the tabu object. If $W_0^{'}$ is in the tabu list, it presents that $W_0^{'}$ is in a searched concave and TBBP don't go on to search deeply, otherwise $W_0^{'}$ is considered to be in a new concave and should be deeply searched.

Whether the concave has been searched or not is very important to avoid the recurrent search. The superficial state $W_0^{'}$ which has been deeply searched is in the TL. If a new generated superficial state ($W_i^{'}$) is in the tabu area (TA) of a tabu object ($W_{t(j)}^{'}$) in the TL, TBBP consider $W_i^{'}$ to be in a concave which has been deeply searched with a biggish probability. Then $W_i^{'}$ is given up and the next superficial state will be tested.

There may be some $W_i^{'}$ which are in the TL (corresponding to $W_{t(j)}^{'}$) but satisfy

$$E(W_i^{'}) < (1 - AC) \cdot E(W_{t(j)}^{'}) \; or E(W_i^{'}) > (1 - AC) \cdot E(W_{t(j)}^{'}) \tag{6}$$

where $E(W_i^{1})$ is the sum of error evaluated at the superficial state $W_i^{'}$.

TBBP uses (6) as the aspiration criterion (AC) to activate $W_i^{'}$ which is tabued and continue to train it deeply. If the "tabued" superficial state, $W_i^{'}$, in the tabu area of $W_{t(j)}^{'}$ but fulfils (6), TBBP consider that $W_i^{'}$ jump out from the concave in which $W_{t(j)}^{'}$ is with a biggish probability.

When the SS have picked out the proper weights, the TBBP trains the NN from these weights. This is called the deep search. The DS trains the NN by using the back propagation algorithm.

3.3 The important parameters in TBBP

3.3.1 The tabu area

The probability of finding weights that they are exactly identical is zero because the weights are real values. TBBP uses tabu area (TA) to relax the strict comparison between the new weight and the weights in the tabu list.

The value of TA is very important to keep from recurrent search. If TA is too big, the algorithm may tabu some superior states which are in the concave that don't have been search before and the search area is decreased. Some good global solutions maybe lost. If TA is set to be too small, some solutions in the same concave are searched deeply and the recurrent search TBBP can't be avoided effectively.

In the experiment, many values were tested. In this paper, TA is set in two different values, 0.03 and 0.01, to illustrate how to choose the TA. 100 neighbor solutions $W_{0j}^{'}$ $(j = 1,2\cdots 100)$ are generated in the neighborhood of the same superficial weight $W_0^{'}$. The size of the neighborhood is set to be the TA. TBBP deeply search the 100 neighbor solutions and get the corresponding deep state $W_{0j}^{''}$. The $E(W_{0j}^{''})$ sse_{0j} are compared to $E(W_0^{''})$ sse_0.

Fig.2 shows the different values of D after training 10000 epochs and 100000 epochs when $TA = 0.03$ and $TA = 0.01$.

$$D = (sse_{0j} - sse_0)/sse_0 \tag{7}$$

In Fig.2, whether the TA is set to be 0.03 or 0.01, all the D are within ± 0.03 after training 100000 epochs. Some deviations are big when set TA to be 0.03 after training 10000 epochs. If $TA = 0.01$, many states are considered to be in the different concaves but actually they are in the same concave. Therefore, lots of time is wasted and TBBP can't avoid recurrent search effectively.

According to this experiment, it is obvious that $TA = 0.03$ is better. To be tabued, all the values in the new weight vector should be within this range for any of the tabu objects in the TL.

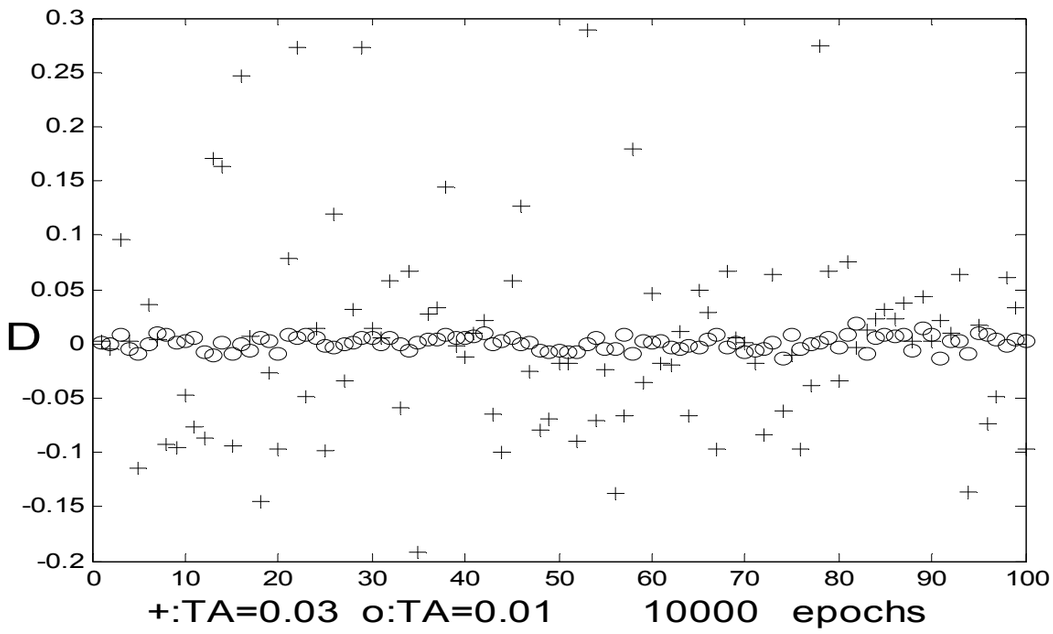
3.3.2 The depth of the superficially search (DSS)

The DSS is the most important parameter in TBBP. If the SS doesn't taken place or the DSS is not deep enough, recurrent search can't be avoided in the global area. If the superficial search is too deep, the neighbor solutions can hardly jump out of the original concave and the search area isn't extended adequately.

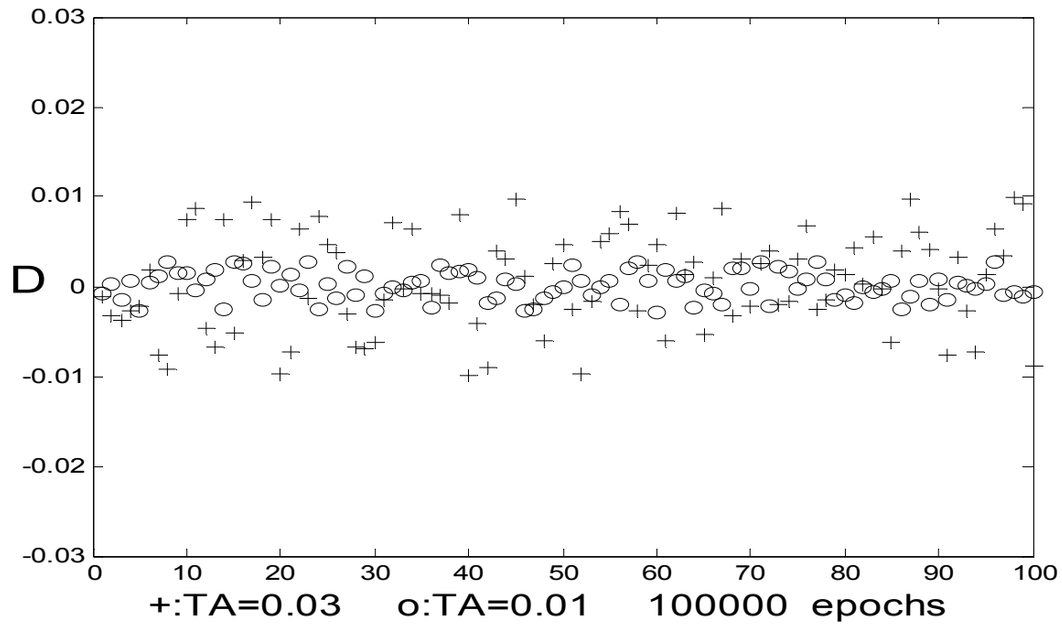
BP algorithm can be simply described as in (4). We set

$$g_k = \partial E(t)/\partial w(t) \tag{8}$$

where g_k is the current gradient.



(a)



(b)

Fig. 2. The different deviations of sse_{0j} from sse_0 .

During the training, g_k may increase in some periods, but there is the trend that g_k decrease when the search depth is increased during the whole training process. Therefore,

TBBP cease the superficial search according to the gradient g. When $g_k = g$, the superficial search stops and the current weight of the NN is set to be the superficial state.

The deep search (DS) is carried out with the 100 neighbor solutions ($W_{ij}^{'}$) which are generated in the neighborhood of the superficial states $W_i^{'}$. Train $W_{ij}^{'}$ deeply and get the deep states $W_{ij}^{''}$. $E(W_{ij}^{''})$ (sse_{ij}) are compared to $E(W_i^{''})$ (sse_i) where $W_i^{''}$ is deeply trained from the original superficial state $W_i^{'}$.

The Fig.3 and Fig.4 show the different deviations of sse_{ij} from sse_i.

$$D = (sse_{ij} - sse_i) / sse_i \tag{9}$$

In the experiment, g is set in three different levers, 0.01, 0.05 and 0.1. If DSS is set to be to small ($g < 0.01$), the algorithm can't jump out of the local minima effectively and some superior solutions may be lost. If DSS is set to be too big ($g > 0.1$), the recurrent search can't be avoided and lots of time is wasted. The deviation is the greatest when $g = 0.05$.

TBBP set DSS to be $g = 0.05$. It considers that the search area are extended most widely from the original superficial state when DSS=0.05. Meanwhile, the recurrent search can be kept from most effectively when DSS=0.05.

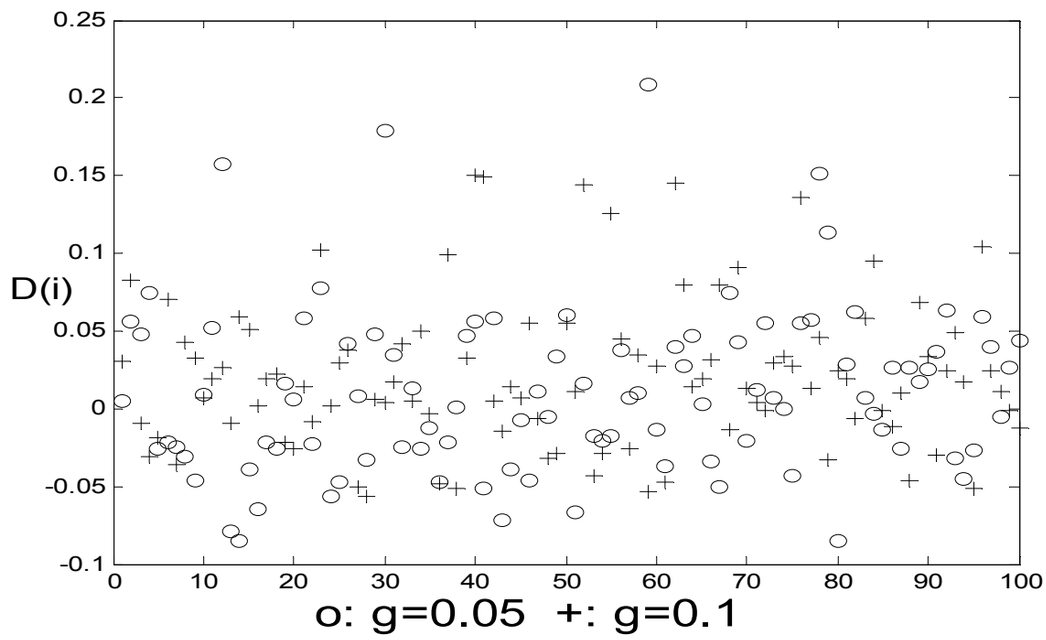


Fig. 3. The different deviations of sse_{0j} from sse_0 when g is set to be too big

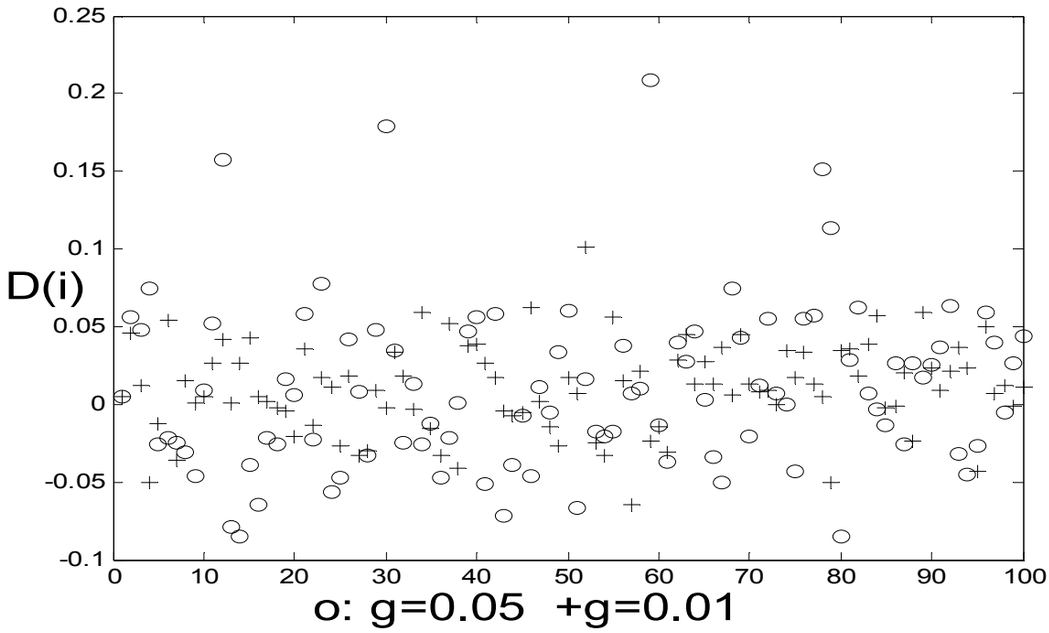


Fig.4. The different deviations of sse_{0j} from sse_0 when g is set to be too small

3.4 The Basic Step of TBBP

In this section, we give the detailed description of TBBP. It mainly consists of seven steps as follows:

(1) Initialization.

This step generates an original weight vector W_i which is described in (5) randomly.

(2) Superficial Search.

i. The initial weight W_i is trained with BP algorithm.

ii. The superficial state $W_i^{'}$ and $E(W_i^{'})$ are got.

iii. The $W_i^{'}$ is tested to decide whether the deep search is implemented or not. If $W_i^{'}$ is tabued and doesn't satisfy AC, go to Step (1) to generate the next initial weight; otherwise, go to Step (3) and add it into the Tabu List.

(3) Deeply search the superficial state $W_i^{'}$.

i. Deeply search the $W_i^{'}$ and get the corresponding deep state $W_i^{''}$.

ii. Evaluate $E(W_i^{''})$ (sse_i) and set it to be the best weight in this neighborhood. $W_{best(i)} = W_i^{''}$.

(4) Generate a neighbor solution $W_{ij}^{'}$ around the superficial state $W_i^{'}$ randomly and evaluate $E(W_{ij}^{'})$.

(5) Test the superficial state $W_{ij}^{'}$ in the neighbourhood of $W_i^{'}$.

If $W_i^{'}$ is in the TL and doesn't satisfy the AC, go to Step (4) to generate the next $W_{ij}^{'}$; otherwise, go to Step (6) to search $W_{ij}^{'}$ deeply.

(6) Deeply search $W_{ij}^{'}$.

i. Train $W_{ij}^{'}$ with BP algorithm and get $W_{ij}^{''}$.

ii. If $E(W_{ij}^{''}) < E(W_{best(i)})$, set $W_{best(i)} = W_{ij}^{''}$.

iii. If $W_{ij}^{'}$ have been generated, go to Step (1) to get another original weight W_i; otherwise, go to Step (4) to generate the next $W_{ij}^{'}$.

iv. Go to Step (7) if the maximal num of the neighborhood is reached.

(7) Get the weight generated by TBBP.

Compare all the $E(W_{best(i)})$ and get the best one as the solution of TBBP.

4. Experimental evaluation

In this section, we first introduce our experiments and then give the results.

4.1 Experiments

The TBBP and BP algorithm both are written in C and all the programs are complied in VC++ 6.0. All the figures (including the Figures in Section 3) are drawn by Matlab 6.0 using the data generated by the program.

The important parameters of TBBP such as the DSS and TA are discussed in Section 3. In this subsection, the experiments are performed using the value selected in Section 3.

In order to test the ability of TBBP to approximate the nonlinear function, we choose 6 nonlinear objective functions to test. They are

$$y = x_1 + x_2 \tag{a}$$

$$y = x_1 x_2 \tag{b}$$

$$y = x_1 / |x_2| + 1 \tag{c}$$

$$y = x_1^2 - x_2^3 \tag{d}$$

$$y = x_1^3 - x_1^2 \tag{e}$$

$$y = \alpha_1 \arctan(x_1 - \beta_1) + \alpha_2 \arctan(x_2 - \beta_2) + \alpha_3 \arctan(x_3 - \beta_3) + \lambda \tag{f}$$

Where $\lambda = -\alpha_1 \arctan(-\beta_1) - \alpha_2 \arctan(-\beta_2) - \alpha_3 \arctan(-\beta_3)$.

Because TBBP is designed to test the superior effect of tabu search used in NN learning, a simple 3-layered feed forward neural network which is described in Fig.1 is chosen for all

the 6 nonlinear functions. There are six nodes in the hidden layer. All the nets have only one output node whose output is severed as the approximating function value.

The training data of function (a)(b)(c)(d)(e) are all generated randomly from $x_i \in [-1,1]$. The training data of function (f) is generated from $x_i \in [0,200]$.

In order to test the TBBP's ability of forecasting the nonlinear function in the unknown areas, the interpolation and extrapolation data are set to be tested. The interpolation data are generated randomly in the same range of the training data. The extrapolation data are generated outside the range of the training data. It is $x_i \in [-2,-1]$ or $x_i \in [1,2]$ for function (a)(b)(c)(d)(e), and $x_i \in [200,400]$ for function (f).

In TBBP, 200 superficial states are tested for each function and 100 neighbor solutions are generated in the neighborhood of each superficial state. In BP, We set the momentum to be 0.9 and train the neural network 50000 iterations.

In order to show how greatly the tabu search works to avoid recurrent search, counters are set to record the times the tabu take place.

4.2 Experimental results

For all the 6 nonlinear functions, the best sum square error (sse) generated in both TBBP and BP are shown in Table1.

		y_1	y_2	y_3	y_4	y_5	y_6
Training samples	TBBP	7.41E-05	7.65E-04	4.29E-02	2.82E-03	8.91E-04	5.41E+01
	BP	2.18E-02	2.46E-02	1.12E-01	1.07E-01	1.78E-01	4.39E+02
A	TBBP	3.42E-03	9.82E-03	1.90E-01	3.58E-01	8.12E-02	6.28E+02
	BP	9.48E-01	3.28E-01	4.57E+00	2.07E+00	3.77E+00	9.27E+03
B	TBBP	6.71E-01	5.34E-01	4.21E+01	1.85E+01	4.33E+01	3.42E+04
	BP	3.75E+01	6.71E+01	1.92E+02	4.09E+01	1.70E+02	9.85E+05

Table 1. The comparison of sum square error (sse) between the two algorithms

Results in Table 1 present that the sse generated by TBBP are obvious smaller than the BP's. It illustrates that the TBBP can approximate the function in all the 3 data including training data, interpolation data and extrapolation data.

The advantage of the approximation in the extrapolation data of TBBP is smaller than the corresponding one in the interpolation data. This is because the interpolation data are generated in the same range of training data. But the extrapolation data is outside the range of the training data.

The sse of TBBP is not very small, Especially in extrapolation data for function (c) (d) (e) (f). Why is the sse of the TBBP not smaller enough (maybe bigger than the sse of GA)? TBBP also uses the BP algorithm as the learning strategy. Actually, we only use tabu search in BP

to test its ability in NN learning. We can also corporate tabu search with other methods such as GA and compare the result with the GA algorithm.

Table 2 shows the times the tabu take place in every 100 neighbor searches of each superficial state. In the table, max times, min times and the mean times the tabu take place are given. From Table 2, we can conclude that the tabu search have played an important role in the recurrent search. Because neighbor solutions are generated randomly, there is no rule of the data in the Table 2.

	(a)	(b)	(c)	(d)	(e)	(f)
min	15	18	23	7	20	14
max	41	43	39	28	32	44
mean	29	31	32	17	26	22

Table 2. Times the tabu take place in every 100 neighbor solutions of each superficial state

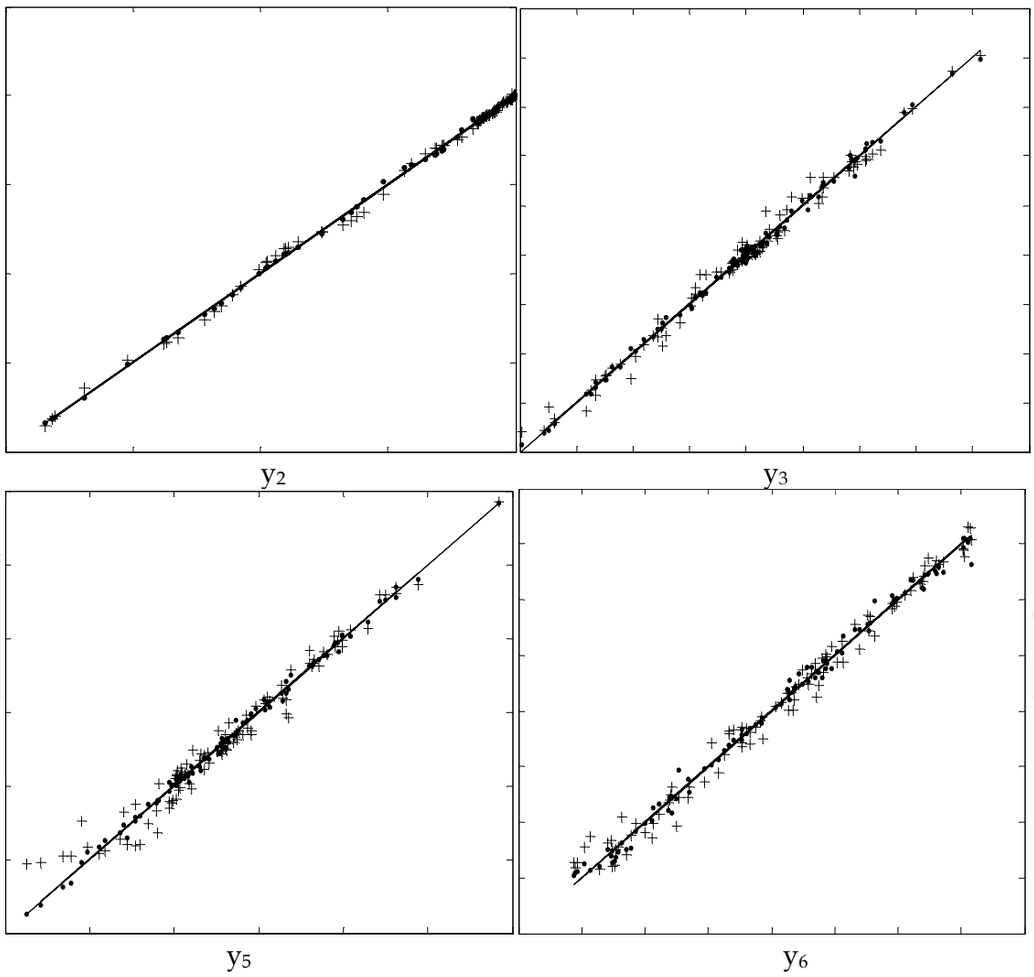


Fig. 5. The comparison of two algorithms in forecasting the interpolation data for different functions. '+' represent BP and '.' represent TBBP

Since the ability of forecasting the function in the non-training data is the most important thing, Fig. 5 is provided to illustrate the different values forecasted by TBBP and BP in the interpolation data for function (b)(c)(e)(f). X axis represents the true value for the interpolation data, Y axis represents the value forecasted by the NN. The line denotes the destination values.

$E(W)$ is smaller when the point is closer to the line. In Fig.5, the point forecasted by TBBP is much closer than those forecasted by BP. Consequently the TBBP can approximate the nonlinear function much more accurately.

5. Conclusions

In this paper, we propose an approach, TBBP, to solve the local convergence of the gradient method. The tabu search can jump out of the local minima, expand the search area and avoid the recurrent search. By using the tabu search in the NN learning, TBBP shows a great improvement of jumping out from the local minimal. It can also approximate the nonlinear function in unknown area. Experiment results show that:

1) TBBP can jump out of the local minimal to extend the search in the global space;

2) Recurrent search can be kept from effectively and lots of time is saved.

This chapter shows the efficiency of integrating the tabu search into the neural network learning. It also gives us a clue that the tabu search is feasible to cooperate with other NN learning algorithms (not only BP).

6. Acknowledgements

This work was supported by the National Science Foundation of China under Grant 60674066; and by the Beijing Education Committee Technology Development Foundation under Grant KM20061005019; and the Beijing Elitists Foundation under Grant 20061D0501500203; and the Beijing Municipality 'New Star' program.

7. References

F. Glover. (1989). Tabu search—Part I, ORSA J. Comput, Vol 1. NO.3, 190–206.

F. Glover. (1990). Tabu search—Part II, ORSA J. Comput, Vol 2. NO.1, 4–32.

F. Glover, M. Laguna.(1997). Tabu search, Boston: Kluwer.

D. Cvijovic, J. Klinowski. (1995). Taboo search: an approach to the multiple minima problem, Science Vol 267. NO.3, 664–666.

Jianming Lu, Xue Yuan, and Takashi Yahagi.(2007). A Method of Face Recognition Based on Fuzzy c-Means Clustering and Associated Sub-NNs. IEEE Trans on Neural Network, vol. 18, NO. 1, 150-151.

Jian-Xin Xu, Ying Tan.(2007). Nonlinear Adaptive Wavelet Control Using Constructive Wavelet Networks. IEEE Trans on Neural Network, vol. 18, NO. 1, 115-128.

Z. S. H. Chan and N. Kasabov.(2005). Fast neural network ensemble learning via negative-correlation data correction. IEEE Trans. Neural Network., Vol. 16, NO. 6, 1707–1710.

Kung S Y, et al.(1998). Neural Networks for Intelligent Multimedia Processing. Proc IEEE, 1244-1272.

K.J. Astrom, B. Wittenmark.(1989). Adaptive Control. Reading, MA: Addison-Wesley.

K. Funahashi.(1989).On Approximate Realization of Continuous Mappings by Neural Network, IEEE Trans. Neural Netwoks. Vol2. NO.3, 183-192.

K. Hornik, M. Stinchcombe, H. White. (1989). Multilayer feed-forward network are universal approximators, IEEE Trans. Neural Networks.Vol2. NO.5, 359-366.

L.M. Salchenberger, E.M. Cinar, N.A. Lash. (1992). Neural networks: a new tool for predicting thrift failures, Decision Sciences 23,899-916.

P. Werbos.(1993).The roots of backpropagation from ordered derivatives to neural networks and political forecasting, John Wiley & Sons Inc, New York.

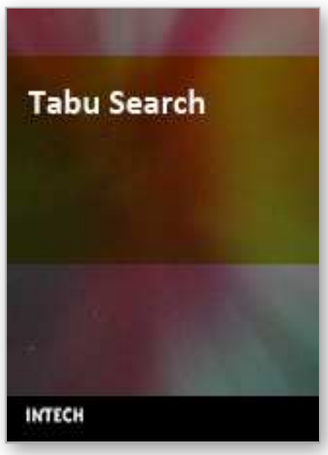
R. Parisi, E.D.Di Claudio. (1996). A generalized learning paradigm exploiting the structure of feedforward neural networks, IEEE Trans. Nueral Networks Vol7.NO.6, 1450-1459.

G. Zhou, J. Si. (1998).Advanced neural network training algorithm with reduced complexity based on Jacobian deficiency, IEEE Trans. Neural Networks .Vol9.NO.3, 448-453.

Y. Izui, A. Pentland. (1990).Analysis of neural networks with redundancy, Neural Computation Vol2, 226-238.

S. Ma, C. Ji..(1998).A unified approach on fast training of feedforward and recurrent networks using EM algorithm, IEEE Trans. Signal Processing, Vol 46,2270-2274.

Jian Ye, Junfei Qiao, Xiaogang Ruan.(2006) A Tabu Based Neural Network Learning Algorithm. Neurocomputing, Vol 10.



Tabu Search

Edited by Wassim Jaziri

ISBN 978-3-902613-34-9

Hard cover, 278 pages

Publisher I-Tech Education and Publishing

Published online 01, September, 2008

Published in print edition September, 2008

The goal of this book is to report original researches on algorithms and applications of Tabu Search to real-world problems as well as recent improvements and extensions on its concepts and algorithms. The book' Chapters identify useful new implementations and ways to integrate and apply the principles of Tabu Search, to hybrid it with others optimization methods, to prove new theoretical results, and to describe the successful application of optimization methods to real world problems. Chapters were selected after a careful review process by reviewers, based on the originality, relevance and their contribution to local search techniques and more precisely to Tabu Search.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Junfei Qiao, Jian Ye and Honggui Han (2008). Hybrid Approaches Tabu Learning Algorithm Based Neural Network, Tabu Search, Wassim Jaziri (Ed.), ISBN: 978-3-902613-34-9, InTech, Available from: http://www.intechopen.com/books/tabu_search/hybrid_approaches_tabu_learning_algorithm_based_neural_network

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen