



The deterministic evolutionary learning algorithm

Ray Tsaih

Department of Management Information Systems, National Chengchi University, Taipei, Taiwan, R. O. C.

Abstract

Most of promising evolutionary ANNs involve a global search algorithm that is a stochastic one. We have introduced a heuristic learning algorithm for the application problems with non-binary desired outputs. The learning algorithm is named as the SBP learning algorithm, where the softening algorithm is first used to decide network's structure, and the obtained NS is then trained via BP. The design of the SBP learning algorithm does not follow the ideas of genetic inheritance and natural selection; however, during its learning process, NS develops in a deterministic way by degrees to a more advanced or mature stage. Thus the SBP learning algorithm is a deterministic evolutionary learning algorithm. The arrangement of the SBP learning algorithm is acceptable provided the ability of successfully classifying the training patterns via the softening algorithm and there are accordance between the original non-binary desired outputs and the transformed binary desired outputs.

1 Introduction

To apply successfully an artificial neural networks (ANN) to practical problems, there is a series of considerations. Those considerations include the type of ANN, the network structure (NS, including the number of hidden layers, the number of hidden nodes, and the connections), the type of activation function, the learning algorithm. None of these considerations is trivial and most of them have no theoretical answer at this moment. Even a well-known ANN is chosen, some considerations confuse even the experts. Take the multi-layered feed-forward network (MLP) with the back propagation (BP) learning algorithm [7] as the illustration. The management of less hidden nodes results in the non-existence



of the globally optimal learning results, and the management of excess hidden nodes promises most likely a sluggish learning process and a locally optimal learning result. Yet, the proper number of hidden nodes corresponding to some problem is usually unknown and determined by trial-and-error. To do well the learning job under the dilemma of the amount of hidden nodes, a simple, plausible way is adopting excess hidden nodes and fixing BP to avoid the predicaments of sluggish learning processes and locally optimal learning results. This leads to the development of some direct modifications of BP, for example, those in [8]. Unfortunately, none of them provided a generalized solution, which surprised nobody because they tried to tune up the learning without changing the fundamental design of BP: the algorithm has the power of altering the weights, yet no power of adding/deleting hidden nodes. ANN's NS has a significant impact on ANN's information processing and learning abilities. Therefore, one may favor the learning algorithm which has the power to determine autonomously not only the weights, but also the amount of adopted hidden nodes. However, none of the answers to such favor provided a generalized solution. It seems, at this moment, those considerations mentioned above involve and there is no solid suggestion.

Above discussions give some idea about the current researches. For instance, many researchers are motivated to derive the learning algorithm which can autonomously search for the optimal design of ANN's NS, a NS which performed best on a specified task according to some optimality criteria. With such learning algorithm, the learning is like searching along the surface defined by the performance level of ANN's NS in the NS space, which is composed of all NSs. But, those learning algorithms are heuristic due to the characteristics associated with such a surface: 1) the NS space is unbounded since the number of possible NS is infinite; 2) the surface is non-differentiable since, for example, changes in the number of nodes is discrete and can have discontinuous effect on ANN's performance; 3) the surface is non-analyzable since the mapping from ANN's NS to ANN's performance is not analyzable yet; 4) the surface is complex and deceptive since ANNs with similar NS may have dramatically different performances, and ANNs with quite different NS can have very similar performances. Those characteristics lead to the learning algorithms hardly having concrete theoretical supports. From literatures, there are two categories of such learning algorithm: the evolutionary ANN (EANN) algorithms, which are stochastic, and the weight-and-structure-change learning algorithms, which are deterministic.

Over the past years, researchers have applied the evolutionary computation to problems whose solution space is too large and highly complex that it is difficult to employ conventional optimization procedures to search for the global optimum. Evolutionary computation refers to a collection of stochastic search algorithms whose designs are based upon the ideas of genetic inheritance and the Darwinian principle of the survival-of-the-fittest (natural selection). There are several different styles of evolutionary algorithms: evolutionary strategies (ES), evolutionary programming (EP), genetic algorithms (GA), and genetic programming (GP). All of them model the search process over the solution space by mimicking a biological evolution process; they are different mainly in



the evolution operators involved and the representation of the solution space. Most researchers believe that, basically, evolutionary computation should not be considered as a kind of optimization technique to compete with other alternative techniques, but an optimization principle to be incorporated with existing techniques. Thus, they propose to apply EP, GA and GP to the determination of the network structure of ANN. This gives rise to three classes of ANN, namely, EPNN, GANN, and GPNN; all of those are portions of EANN. Most promising EANN involve a global search algorithm which is a stochastic one [13].

In contrast with the stochastic way of EANN, all of weight-and-structure-change learning algorithms alter the weights and the amount of adopted hidden nodes in a deterministic way. For instance, there are the tiling algorithm [6], the cascade-correlation (CC) algorithm [3], the upstart algorithm [4], the W&S algorithm [12], CTN algorithm [1], and the softening algorithm [9][10]. They adjust the network structure in one of the following ways:

- (1) Destructive way: use excess hidden nodes initially and prune (remove) least effective hidden nodes during the learning process; e.g., W&S algorithm and CTN algorithm;
- (2) Constructive way: use less hidden nodes initially and recruit (add) more hidden nodes during the learning process; e.g., the tiling algorithm, CC algorithm, and the upstart algorithm; and
- (3) Constructive/Destructive way: use only one hidden node initially, and autonomously recruit as well as prune hidden nodes during the learning process; e.g., the softening algorithm.

The designs of those weight-and-structure-change learning algorithms do not follow the ideas of genetic inheritance and natural selection. However, the processes of those algorithms are the ones in which NSs develop in a deterministic way by degrees to a more advanced or mature stage. Thus those weight-and-structure-change learning algorithms are referred as the deterministic evolutionary learning algorithms.

Among those deterministic evolutionary learning algorithms, the softening algorithm is straightforwardly favored due to its ability of autonomously recruiting and pruning hidden nodes during the learning process. There are also other distinguished features in the softening algorithm; for instance, it is applied to the MLP, and it guarantees a perfect learning result for the cases of the binary output patterns. From the empirical results in [2] [9] [5] [11], it seems that the softening algorithm outperforms BP. It seems that the ability to recruiting and pruning hidden nodes during the learning process does help to overcome the predicaments of BP. Unfortunately, the softening algorithm is designed to deal only with the case of the binary desired outputs. When working with the case of the non-binary desired outputs via the softening algorithm, the non-binary numbers have to be first converted into binary digits. This increases the number of output nodes and the learning complexity, requiring a longer learning time. Thus, a learning algorithm that can deal with the case of the non-binary desired outputs is further desired.

Here we introduce such a learning algorithm, which can deal with the case of the non-binary desired outputs. We name it the SBP learning algorithm.

The SBP learning algorithm integrates the softening algorithm and BP, and it can autonomously adjust the weights, the number of hidden nodes, and the amount of hidden layers.

In latter part of this paper, the softening algorithm is first reviewed. Then, the proposed learning algorithm, the SBP learning algorithm, is introduced. Conclusions and future work are presented in the last.

2 The review of the softening algorithm

The block diagram of the softening algorithm is demonstrated in Figure 1. There are four distinguished aspects in this structure: the linearly separating condition (LSC), the thinking mechanism, the cramming mechanism, and the reasoning mechanism. These aspects are explained in the following paragraphs.

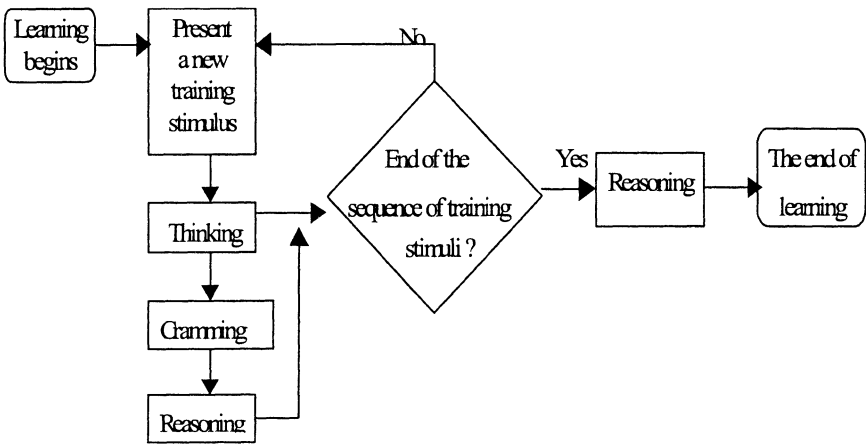


Figure 1. The block diagram of the softening algorithm

The softening algorithm is used as a classifier which learns to distinguish, concerning each output node, if the stimulus is a member of one class of stimuli, called class 1, or of a different class, called class 2. This task is a 2-classes categorization learning task. LSC is designed to make such task easier to accomplish. At the learning stage, the training stimuli are presented one by one. Let $K(k) \equiv \{1, \dots, k\}$ and, concerning the l th output node, $K(k) \equiv K_{1l}(k) \cup K_{2l}(k)$ where $K_{1l}(k)$ and $K_{2l}(k)$ are the sets of indices of the first k training stimuli in class 1 and class 2, respectively. Let the desired output values of classes 1 and 2 be 1.0 and -1.0, respectively. The LSC(k, l) is defined as follows:

$$\min_{c \in K_{1l}(k)} net(B_c, Y_l, X) > \max_{c \in K_{2l}(k)} net(B_c, Y_l, X) \quad (1)$$

Table 1 displays the definition of notations used here. When the LSC(k, l) is satisfied, we can use the following threshold value to accomplish the 2-classes categorization learning task:

$$v_l = \frac{\min_{c \in K_{l1}(k)} \text{net}(B_c, Y_l, X) + \max_{c \in K_{l2}(k)} \text{net}(B_c, Y_l, X)}{2} \quad (2)$$

That is, with respect to the l th output node,

$$B_c \in \begin{cases} \text{Class } 1 & \text{if } \text{net}(B_c, Y_l, X) \geq v_l \\ \text{Class } 2 & \text{if } \text{net}(B_c, Y_l, X) < v_l \end{cases} \quad (3)$$

Table 1. The definition of the used notations

m, p, and q : the amounts of input, hidden, and output nodes, respectively. B_c : the cth given stimulus input pattern, $c=1, 2, \dots, k$. b_{cj} : the input value received by the jth input node when B_c is presented to the network. w_{ij} : the weight of connection between the jth input and the ith hidden node. w_i : the vector of weights of the connections between all input nodes and ith hidden node; $w_i \alpha (w_{i1}, w_{i2}, \dots, w_{im})$. θ_i : the negative of the threshold value of the ith hidden node. $X_i^t \equiv (\theta_i, w_i^t)$ and $X^t \equiv (X_1^t, X_2^t, \dots, X_p^t)$. $h(B_c, X_i)$: the activation value of the ith hidden node given the stimulus B_c, and $h(B_c, X_i) \equiv \tanh\left(\theta_i + \sum_{j=1}^m w_{ij} b_{cj}\right)$ $h(B_c, X)$: the activation value vector of all hidden nodes given the stimulus B_c, and $h(B_c, X) \equiv (h(B_c, X_1), h(B_c, X_2), \dots, h(B_c, X_p))^t$ r_{li} : the weight of the connections between ith hidden nodes and lth output node. r_l : the vector of weights of the connections between all hidden nodes and lth output node; $r_l \equiv (r_{l1}, r_{l2}, \dots, r_{lq})$. s_l : the negative of the threshold value of the lth output node. $Y_l^t \equiv (s_l, r_l^t)$, $Y^t \equiv (Y_1^t, Y_2^t, \dots, Y_q^t)$, and $Z^t \equiv (Y^t, X^t)$. $\text{net}(B_c, Y_l, X)$: the net input value of the lth output node given the stimulus B_c, and $\text{net}(B_c, Y_l, X) \equiv s_l + \sum_{i=1}^p r_{li} h(B_c, X_i)$ $O(B_c, Y_l, X)$: the activation value of the lth output node given the stimulus B_c, and $O(B_c, Y_l, X) \equiv \tanh(\text{net}(B_c, Y_l, X))$ d_{cl} : the desired output value of the lth output node when B_c is presented to the network.

The thinking mechanism is used to adjust the weights, and, currently, it is implemented with the momentum version of the generalized delta rule and the decay mechanism. This thinking mechanism, however, does not guarantee a globally optimal learning result. Therefore, there is the cramming mechanism whose job is recruiting hidden nodes. By recruiting proper hidden nodes, the

dimension of the weight space and the surface defined by the performance level of NS extend, and the defected learning result could be no more an attractor. In other words, recruiting hidden nodes could render the learning escape from the encountered relatively optimal learning result.

It is possible to accomplish the goal of learning via merely using the cramming mechanism; however, it may eventually recruit too many hidden nodes that the generalization ability of the network is bad. Thus, there is the reasoning mechanism, which includes the thinking mechanism and the pruning mechanism which is designed for pruning the irrelevant hidden nodes. In a Z , the i th hidden node is irrelevant with respect to the l th output node if $LSC(k,l)$ is still satisfied with the same Z except $r_{\bar{h}} = 0$, and a hidden node is irrelevant if it is irrelevant with respect to all output nodes.

3 The proposed learning algorithm

For the application problems with non-binary desired outputs, the following learning procedure is proposed: (1) the softening algorithm is first used to generate a MLP network, named as the S network, (2) the S network is trained via BP. This is called as the SBP learning procedure. First, the SBP learning procedure is introduced. Then, the theoretical and heuristic parts of work are shown.

3.1 The SBP learning algorithm

The detailed SBP learning procedure is illustrated in Figure 2 and explained in the following paragraphs.

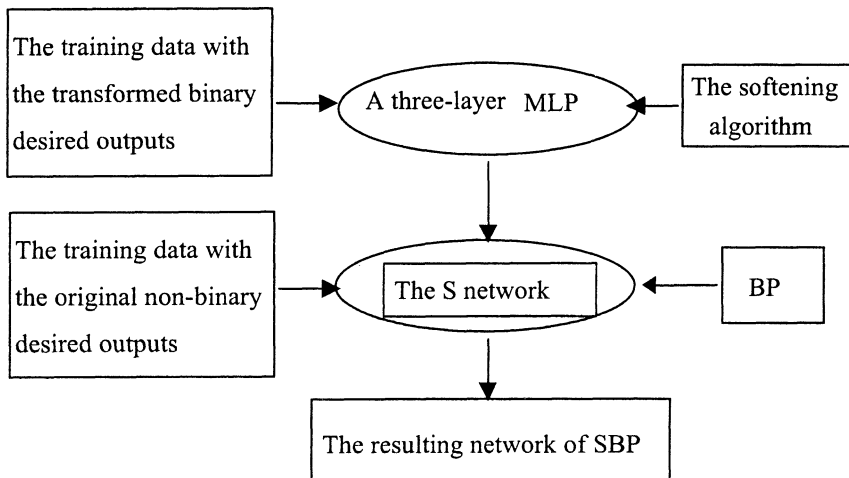


Figure 2. The SBP learning procedure

Without losing generalization, assume there is only one output node. To

fit in the characteristic of the binary desired outputs of the softening algorithm, the non-binary desired output of the training data shall be transformed into a value of binary digits before feeding in the softening algorithm. We can manually classify the training patterns based upon their desired output values. Assume it results in G groups, and $N = \min_n \{n \mid 2^n \geq G\}$. Then, concerning

the training pattern of any classification, said the g th classification, a binary code with N bits corresponding to the number $g-1$ is assigned as its transformed binary desired output. In other words, the desired output of any training pattern of this g th classification will be replaced by such an associated binary code. The softening algorithm and those new input/output pairs are used to train an MLP network with one hidden layer and N output nodes. After the learning, the resulting MLP network is denoted as the S network.

If $N = 1$, then BP and the original input/output training patterns are used directly to train the S network. If $N > 1$, then a new four-layer MLP network with one output node is built from the S network, and such a four-layer MLP network is trained via BP and the original input/output training patterns. That is, the S network is directly used as the input layer, the first hidden layer and the second hidden layer of the new MLP network, and an extended layer with one node is introduced as the output layer of the new MLP network. The arrangement of weights between the input layer and the first hidden layer, and between the first and second hidden layers of the new MLP network, is the same as those of the S network; the weights between the output node and all hidden nodes of the second hidden layer of the new MLP is assigned subjectively based upon the transformed binary desired output values. After the learning, the resulting MLP network is what we obtained from the SBP learning procedure.

3.2 The promise of the SBP learning algorithm

3.2.1 The theoretic work of the SBP learning algorithm

The softening algorithm can be theoretically designed to guarantee a perfect learning result for the 2-classes categorization learning task, in which the desired outputs are binary numbers. This can be shown from the following lemma, which show the $LSC(k, l)$ for all l can be satisfied immediately via properly assigning weights associated with the new recruiting hidden node(s).

Lemma 1 (for the case of binary input). With respect to the new recruiting p th hidden node, there exist X_p and r_{lp} s that render the $LSC(k, l)$ for all l satisfied.

Proof: Let $\Omega(k) \equiv \{l \mid LSC(k, l) \text{ is not satisfied before introducing the new } p\text{th hidden node}\}$. If $l \notin \Omega(k)$, let $r_{lp} = 0$. Then, for every $l \notin \Omega(k)$, $LSC(k, l)$ is still satisfied after adding the p th hidden node. As for the $l \in \Omega(k)$, take an l as the illustration, and, without losing generality, let $d_{kl} = 1.0$. Thus $K_{l1}(k) = K_{l1}(k-1) \cup \{k\}$, $K_{l2}(k) = K_{l2}(k-1)$, and

$$\min_{c \in K_{l1}(k-1)} \sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) > \max_{c \in K_{l2}(k-1)} \sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) \quad (4)$$

$$\sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_k, \mathbf{X}_i) < \max_{c \in K_{l2}(k-1)} \sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) \quad (5)$$

After introducing the new p th hidden node, we need to show that there exist $\mathbf{X}_p$ and r_{lp} such that

$$\min_{c \in K_{l1}(k)} \sum_{i=1}^p r_{li} h(\mathbf{B}_c, \mathbf{X}_i) > \max_{c \in K_{l2}(k)} \sum_{i=1}^p r_{li} h(\mathbf{B}_c, \mathbf{X}_i) \quad (6)$$

$$\text{Let } r_{lp} = 2 \left(\max_{c \in K_{l2}(k-1)} \sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) - \sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_k, \mathbf{X}_i) \right), \quad \mathbf{w}_p = \lambda \mathbf{B}_k, \text{ and } \theta_p$$

$= \lambda - \lambda \mathbf{B}_k^t \mathbf{B}_k$, where λ is a large number and equals to 2^τ . Therefore, $h(\mathbf{B}_k, \mathbf{X}_p) = \tanh(2^\tau)$. Also, $h(\mathbf{B}_c, \mathbf{X}_p) \leq -\tanh(2^\tau) \forall c \in K(k-1)$, and $|h(\mathbf{B}_v, \mathbf{X}_p) - h(\mathbf{B}_u, \mathbf{X}_p)| \leq 1.0 - \tanh(2^\tau) \forall v \in K_{l1}(k-1) \forall u \in K_{l2}(k-1) \forall l \in \Omega(k)$, due to $\mathbf{B}_c \in \{-1, 1\}^m \forall c$. Furthermore, there exists an integer τ such that $0.5 < \tanh(2^\tau)$ and

$$r_{lp}(1.0 - \tanh(2^\tau)) < \min_{v \in K_{l1}(k-1)} \left(\sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_v, \mathbf{X}_i) - \sum_{i=1}^{p-1} r_{li} h(\mathbf{B}_u, \mathbf{X}_i) \right) \forall l \in \Omega(k)$$

With such τ and the facts of (4) and (5), equation (6) is satisfied. **Q.E.D.**

Lemma 2 (for the case of input being real numbers). With respect to the two new recruiting hidden nodes, the $p-l$ th and p th hidden nodes, there exist $\mathbf{X}_{p-1}$, $\mathbf{X}_p$, r_{lp-1} s and r_{lp} s that render the LSC(k, l) for all l satisfied.

Proof: Let $\Omega(k) \equiv \{l \mid \text{LSC}(k, l) \text{ is not satisfied before introducing two new hidden nodes}\}$. If $l \notin \Omega(k)$, let $r_{lp-1} = 0$ and $r_{lp} = 0$. Then, for every $l \notin \Omega(k)$, LSC(k, l) is still satisfied after introducing two new hidden nodes. As for the $l \in \Omega(k)$, take an l as the illustration, and without losing generality, let $d_{kl} = 1.0$. Thus $K_{l1}(k) = K_{l1}(k-1) \cup \{k\}$, $K_{l2}(k) = K_{l2}(k-1)$, and

$$\min_{c \in K_{l1}(k-1)} \sum_{i=1}^{p-2} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) > \max_{c \in K_{l2}(k-1)} \sum_{i=1}^{p-2} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) \quad (7)$$

$$\sum_{i=1}^{p-2} r_{li} h(\mathbf{B}_k, \mathbf{X}_i) < \max_{c \in K_{l2}(k-1)} \sum_{i=1}^{p-2} r_{li} h(\mathbf{B}_c, \mathbf{X}_i). \quad (8)$$

After introducing the $p-l$ th and p th hidden nodes, we need to show that there exist $\mathbf{X}_{p-1}$, $\mathbf{X}_p$, r_{lp-1} and r_{lp} such that

$$\min_{c \in K_{l1}(k)} \sum_{i=1}^p r_{li} h(\mathbf{B}_c, \mathbf{X}_i) > \max_{c \in K_{l2}(k)} \sum_{i=1}^p r_{li} h(\mathbf{B}_c, \mathbf{X}_i) \quad (9)$$

Let $\mathbf{w}_{p-1} = -\lambda \boldsymbol{\alpha}$, $\mathbf{w}_p = \lambda \boldsymbol{\alpha}$, $\theta_{p-1} = \lambda \varepsilon + \lambda \boldsymbol{\alpha}^t \mathbf{B}_k$, and $\theta_p = \lambda \varepsilon - \lambda \boldsymbol{\alpha}^t \mathbf{B}_k$, where $\boldsymbol{\alpha}$ is a vector with $\|\boldsymbol{\alpha}\| = 1$, ε is a tiny positive number, and λ is a large number and equals to 2^τ . Thus, $h(\mathbf{B}_k, \mathbf{X}_{p-1}) = h(\mathbf{B}_k, \mathbf{X}_p) = \tanh(\varepsilon 2^\tau)$, and $h(\mathbf{B}_c, \mathbf{X}_{p-1}) = \tanh(2^\tau(\varepsilon - \boldsymbol{\alpha}^t \mathbf{B}_c + \boldsymbol{\alpha}^t \mathbf{B}_k))$ and $h(\mathbf{B}_c, \mathbf{X}_p) = \tanh(2^\tau(\varepsilon + \boldsymbol{\alpha}^t \mathbf{B}_c - \boldsymbol{\alpha}^t \mathbf{B}_k)) \forall c \in K(k-1)$.

Also, let $r_{lp-1} = r_{lp} = \max_{c \in K_{l2}(k-1)} \sum_{i=1}^{p-2} r_{li} h(\mathbf{B}_c, \mathbf{X}_i) - \sum_{i=1}^{p-2} r_{li} h(\mathbf{B}_k, \mathbf{X}_i)$.

Furthermore, with a reasonable assumption that the amount of training data is finite, there exist proper $\boldsymbol{\alpha}$ and ε such that $|\boldsymbol{\alpha}^t \mathbf{B}_c - \boldsymbol{\alpha}^t \mathbf{B}_k| > \varepsilon \forall c \in K(k-1)$, and



an integer τ such that $0.5 < \tanh(\varepsilon 2^\tau)$ and $|h(\mathbf{B}_c, \mathbf{X}_{p-1}) - h(\mathbf{B}_c, \mathbf{X}_p)| \cong 0.0 \quad \forall c \in K(k-1)$. With such α , ε , τ , r_{lp-1} , and r_{lp} , and the facts of (7) and (8), equation (9) is satisfied. **Q.E.D.**

3.2.2 The heuristic work of the SBP learning algorithm

With the accordance between the original non-binary desired output values and the transformed binary desired output values as well as the ability of successfully classifying the training patterns via the softening algorithm, we expect the softening algorithm can acquire some useful NS information for BP. Thus, in the second part of SBP procedure, BP may proceed on a better network structure. The meaning of being better is being more efficient and more effective in learning as well as more effective in forecasting ability (generalization).

The λ s mentioned in Lemma 1 and 2 are like gain parameters; their largeness render the activation function of the new recruiting hidden node like a threshold one, and the resulting saturation phenomenon of the activation function renders the following learning process much more inefficient. In contrast, the softening algorithm can be designed loosely. Heuristically, it is also feasible to recruit hidden node(s) with the weights mentioned in Lemma 1 and 2, and with the value of λ being 1.0, and then use the thinking mechanism to tune the weights. And such process is repeated till LSC(k,l) for all l are satisfied. Such arrangement make the softening algorithm a state of art; however, the empirical results show that the above repeated strategy works well, resulting in a finite amount of new hidden nodes.

4 Conclusion and Future Work

We have introduced a heuristic learning algorithm for the application problems with non-binary desired outputs. The learning algorithm is named as the SBP learning algorithm, where the softening algorithm is first used to decide network's structure, and the obtained NS is then trained via BP. The design of the SBP learning algorithm does not follow the ideas of genetic inheritance and natural selection; however, during its learning process, NS develops in a deterministic way by degrees to a more advanced or mature stage. Thus the SBP learning algorithm is a deterministic evolutionary learning algorithm.

The arrangement of the SBP learning algorithm is acceptable provided the ability of successfully classifying the training patterns via the softening algorithm and there are accordance between the original non-binary desired outputs and the transformed binary desired outputs. We may classify the input patterns of each training set into two categories via the mean value (or the median) of their desired output values. One future work is to study a better way of digitally transforming the desired output values.

From some simple experimental results, it seems that the heuristic softening algorithm can successfully classify the training patterns, and then BP proceeds on a better MLP network. The meaning of being better is regarding the better forecasting ability, while there is no significant improvement in efficiency. Compared with BP, however, SBP is less liable of encountering with the rela-



tively optimal learning result. In other words, the heuristic softening algorithm renders the following learning of BP less liable of encountering with the relatively optimal learning result (than the original BP's learning), although the following learning of BP may be more complicated. Although those are shown in the empirical results, the above facts have not been proved mathematically. Besides, some arrangements of the softening algorithm are rather a state of art. Another future work is to further investigate theoretically to verify those facts.

References

- [1] Chen, Y., Thomas, D. & Nixon, M. Generating-Shrinking Algorithm for Learning Arbitrary Classification. *Neural Networks*, 7, pp. 1477-1489, 1994.
- [2] Chiou, Y., Liu, S. & Tsaih, R. Applying Reasoning Neural Networks to the Analysis and Forecast of Taiwan's Stock Index Variation. *Taipei Economic Inquiry*, Vol. 34, No. 2, pp. 171-200, 1996.
- [3] Fahlman, S. & Lebiec, C. The Cascade-Correlation Learning Architecture. In Touretzky, D. (Eds.), *Advances in Neural Information Processing Systems II* (Denver, 1989), San Mateo: Morgan Kaufmann, 1990.
- [4] Frean, M. The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks. *Neural Computation*, 2, pp. 198-209, 1990.
- [5] Lin, H., Tsaih, R. & Jee, R. Exploring the Relative Abilities of Neural Networks and VAR Models in Forecasting Taiwan Bond Prices, *Review of Securities and Futures Markets*, Vol. 9, NO. 1, pp. 63-113, 1997.
- [6] Me'zard, M. & Nadal, J. Learning in Feedforward Layered Networks: The Tiling Algorithm. *Journal of Physics*, A 22, pp. 2191-2204, 1989.
- [7] Rumelhart, D., Hinton, G. & Williams, R. Learning Internal Representations By Error Propagation. In Rumelhart, D. & McClelland, J. (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, pp. 318-362, Cambridge, MA: MIT Press, 1986.
- [8] Sarkar, D. Methods to speed up error back-propagation learning algorithm. *ACM Computer Surveys*, Vol. 27, No. 4, Dec., pp. 519-542, 1995.
- [9] Tsaih, R. The Reasoning Neural Networks. In Ellacott S., Mason, J. & Anderson, I. (Eds.), *Mathematics of Neural Networks: Models, Algorithms and Applications*, Kluwer Academic publishers, London, pp. 366-371, 1997.
- [10] Tsaih, R. An Explanation of Reasoning Neural Networks, *Mathematical and Computer Modelling*, Vol. 28, No. 2, pp. 37-44, 1998.
- [11] Tsaih, R., Hsu, Y. & Lai, C. Forecasting S&P 500 Stock Index Futures with the Hybrid AI system, *Decision Support Systems*, Vol. 23, No. 2, pp. 161-174, 1998.
- [12] Watanabe, E., & Shimizu, H. Algorithm for Pruning Hidden nodes in Multi-Layered Neural Network for Binary Pattern Classification Problem. *Proceedings of 1993 International Joint Conference on Neural Networks*, I, pp. 327-330, 1993.
- [13] Yao, X. A Review of Evolutionary Artificial Neural Networks. *International Journal of Intelligent Systems*, 8 (4), pp. 539-567, 1993.