Some Performance Comparisons for Self-Generating Neural Tree

W.X. Wen, A. Jennings, H. Liu, and V. Pang
AISS/TSSS, Telecom Research Labs.
Clayton, Victoria 3168, Australia

(in Proc. IJCNN'92, Beijing, China)

Abstract

The performance of Self-Generating Neural Tree (SGNT) method is analysed using some public domain data sets and the Monk problems – a de facto standard benchmark set. Comparisons are performed between SGNT and other methods. The results show that SGNT method is superior to all the unsupervised learning methods and some popular supervised learning methods in both accuracy and speed of learning.

1 INTRODUCTION

The SGNN (Self-Generating Neural Network) method proposed in [10, 1] is an unsupervised learning method. It has been applied to different application areas such as image coding, diagnostic expert system and document/information retrieval system [8].

In this paper, the performance of a kind of SGNN, Self-Generating Neural Trees (SGNT) is analysed using some public domain data sets and the Monk problems [6]– a de facto standard benchmark set for testing supervised/unsupervised learning methods. Comparisons between our method and other (supervised/unsupervised) learning methods are also given.

2 SGNT ALGORITHM

Neural networks are usually designed by human experts. It is quite tricky to choose the right structure of the neural network suitable for a particular application at hand. In this section, we briefly discuss the SGNT method proposed in [10] to generate a neural tree [1] automatically from training examples without any other human intervention.

For this kind of network, not only the weights of the network connections but also the structure of the whole network are all learned from the training examples directly. These include:

1. the number of the neurons in the network,
2. the interconnections among the neurons,
3. the weights on the connections.

A neural tree [1] generated in this way is called a Self-Generating Neural Tree (SGNT). The algorithm to generate an SGNT is basically a hierarchical clustering algorithm. Before we describe the SGNT algorithms, some definitions related are given below:

Definition 1: An *instance* e_i is a real vector of attributes: $e_i = < a_{i1}, ..., a_{in} >$.

Definition 2: A *neuron* n_j is a ordered pair $< W_j, C_j >$, where W_j is the real weight vector of the neuron: $W_j = < w_{j1}, ..., w_{jn} >$, and C_j is the child neuron set of n_j.

Definition 3: An *SGNT* is a tree $< \{n_j\}, \{l_k\} >$ of neurons generated automatically from a set of training instances by the algorithm given below, where $\{n_j\}$ is the neuron/node set and $\{l_k\}$ is the link set of the tree. There is a directed link from neuron n_i to n_j, if and only if $n_j \in C_i$.

Definition 4: A neuron n_k in a neuron set $\{n_j\}$ is called a *winner* for an instance e_i if $\forall j, d(n_k, e_i) \leq d(n_j, e_i)$ where $d(n_j, e_i)$ is the distance between neuron n_j and instance e_i.

Any distance measure can be used, but we use a modified Euclidean distance measure as an example:

$$d(n_j, e_i) = \sqrt{\frac{\sum_{k=1}^{n} \rho_k \cdot (a_{jk} - a_{ik})^2}{n}}$$

where ρ_k is the weight for the k-th attribute. Note that, these weights are different from those weights for the SGNT neurons. The former are determined by a method based on information theory [10] or a human expert prior to the learning phase while the latter are learned from the given training data set.

The SGNT algorithm is a hierarchical clustering algorithm. It is given in a pseudo-C language as follows:

Algorithm 1 (SGNT Generation/Training):

Input:

1. A set of training instances $E = \{e_i\}$, $i = 1, ..., N$.
2. A threshold $\xi \geq 0$.
3. A distance measure for each attribute/weight in instances/neurons.

Output: An SGNT generated from E.

Method:

```
copy(root,e0);
for(i=1,j=1;i<=N;i++) {
    minmumDistance = distance(ex,root);
    winner = oldWinner = root;
    minimumDistance = test(ei,root);
    if(minimumDistance>ξ) {
        if(leaf(winner)) {
            copy(nj,winner);
            connect(nj,winner);
            j++;
        }
        copy(nj,ei);
        connect(nj,winner);
        j++;
    }
    update(winner,ei);
}
```

where the routines are defined as follows:

1. `copy(n,e)`: create a neuron n and copy the attributes/weights in the instance/neuron e to n.
2. `distance(e,n)`: return the distance between instance e and neuron n.
3. `test(e,subRoot)`: find a winner in the current SGNT/sub-SGNT rooted by `subRoot` for instance e and return the distance between the winner and e.
4. `leaf(n)`: check a neuron n to see whether it is a leaf neuron in the current SGNT. A neuron in an SGNT is called a leaf neuron if it has no child neuron.
5. `connect(`n_0, n_1`)`: connect neuron n_0 to n_1 making n_0 as a child neuron of n_1.

6. update(n_i, e_{k+1}): update the weight vector of neuron n_i by the attribute vector of e_{k+1} according to the updating rule (1) below.

$$w_{ij,k+1} = w_{ij,k} + \frac{1}{k+1} \cdot (a_{k+1,j} - w_{ij,k}). \tag{1}$$

where $w_{ij,k}$ is the j-th weight of n_i after having seen the first k examples covered by n_i.

After an initial network has been built some optimizations (called horizontal and vertical optimizations) may be performed to make the performance of the network better. As the network is trained repeatedly by the training set the dead branches which stop growing are pruned away to improve the quality (size) of the network. An SGNT branch is called dead if the number of examples covered by its root does not increase during the repeated training. Finally, it is possible to simplify an SGNT by a method similar to Quinlan's method of decision tree simplification [4]. For more detailed information about the SGNT algorithm, optimization, pruning, and simplification, see [10].

3 PERFORMANCE ANALYSIS

Two SGNT systems have been developed on SUN SPARC station and IBM-PC/AT with X-window/MS-window 3 interfaces, respectively. Some public domain data sets for unsupervised learning are used to test the system performance and the results are quite satisfactory:

1. US voting: There are 17 attributes $\times$ 435 examples, 300 training examples and 135 testing examples. Testing result is: 100% correctness for training data. 95% correctness for testing data. This is the same as the best result for unsupervised learning reported previously [5].

2. Mushroom classification: There are 23 attributes $\times$ 8124 examples; 1000 training examples and 7124 testing examples. Testing result: 100% correctness for training data, 99.5% for testing data. The best result reported previously for unsupervised learning was 95% for only about 3000 test examples [5, 2], although for supervised learning, 100% accuracy was reported by Quinlan's C 4.5 method and Kowalczyk's higher order neural network methods [3].

In addition, SGNT method has also been successfully applied to some practical application areas such as image coding, fault diagnosis, and document retrieval [8].

To prove SGNT/SGNN as a useful general purpose learning method, in this section, we conduct a systematic performance analysis for the method and compare the method with many other well-known learning methods. The benchmark we have chosen is the MONK's problems [6]. The reasons are

1. MONK's tests have been performed for many (at least 20) well-known supervised/unsupervised learning methods at the 2nd European Summer School on Machine Learning during summer 1991. This makes a comparison between our method and those well-known methods much easier and less biased.

2. The training and test data and the test results for other well-known methods are easily accessed.

3. The MONK's problems are not too complicated, time-consuming to test but are also reasonably difficult to be solved in different aspects.

The MONK's problems [6] rely on an artificial robot domain, in which robots are described by six different attributes:

x_1:	head_shape	$\in$	round, square, octagon;	x_2:	body_shape	$\in$	round, square, octagon;
x_3:	is_smiling	$\in$	yes, no;	x_4:	holding	$\in$	sword, balloon, flag;
x_5:	jacket_color	$\in$	red, yellow, green, blue;	x_6	has_tie	$\in$	yes, no.

The learning tasks of the three MONK's problems are binary classification tasks, each of them is given by the following logical description of a class.

- Problem M_1: (head_shape = body_shape) or (jacket_color = red). From 432 possible examples, 124 were randomly selected for the training set. No noise is present.

- Problem M_2: Exactly two of the six attributes have their first value. From 432 examples, 169 were selected randomly. No noise is present.

- Problem M_3: (Jacket_color is green and holding a sword) or (jacket_color is not blue and body_shape is no octagon). From 432 examples, 122 were selected randomly. and among them there were 5% misclassifications, ie. noise in the training set.

It is easy to see that M_3 should be the easiest one to learn if there were no noise present because there is only a DNF of first order relations to be learned in the description. M_1 is a bit harder because there is a second order relation, head_shape = body_shpae, to learn. The most difficult one is M_2, one has to examine all the six attributes of any example to tell whether it belongs to the right class.

The unsupervised learning results of SGNT and some well-known learning methods [6] are given in Table 1[1]. It turns out that SGNT always ranks the top for the easier tasks such as M_1 and M_3. For

Learning method tested	Test results				Tester(s)
	M_1	M_2	M_3	**Average**	
SGNT (unsupervised)	82.6%	78.2%	84.5%	**81.8%**	Wen et. al.
8×8 Kohonen LVQ Net	76.1%	63.0%	81.2%	**73.4%**	Pang et. al.
CLASSWEB 0.10	71.8%	64.8%	80.8%	**72.5%**	Kreuziger et. al.
CLASSWEB 0.15	65.7%	61.6%	85.4%	**70.9%**	Kreuziger et. al.
CLASSWEB 0.20	63.0%	57.2%	75.2%	**65.1%**	Kreuziger et. al.
ECOBWEB leaf prediction	71.8%	67.4%	69.1%	**72.5%**	Reich & Fisher
ECOBWEB l.p & information utility	82.7%	71.3%	68.0%	**74.0%**	Reich & Fisher

Table 1: Accuracy comparisons for unsupervised learning methods

the harder task M_2, the performance of SGNT is much better than the others and so is the average performance of SGNT. Actually, the performance of SGNT for the harder problem is even better than many popular supervised learning methods (see Table 3). As for the training speed, SGNT is significantly faster than its competitors. For the MONK's problems the speed comparisons are given in Table 2.

Learning method tested	Training time (in seconds)			Platform used for testing	Tester(s)
	M_1	M_2	M_3		
SGNT (unsupervised)	1.47	2.37	2.69	SUN SPARC 1	Wen et. al.
8×8 LVQ Net	62.74	85.72	62.04	SUN SPARK 1^+	Pang et. al.
CLASSWEB 0.10	1406.47	2013.78	1311.25	SUN SPARC 1^+	Kreuziger et. al.
CLASSWEB 0.15	867.47	977.04	822.09	SUN SPARC 1^+	Kreuziger et. al.
CLASSWEB 0.20	499.94	646.06	521.21	SUN SPARC 1^+	Kreuziger et. al.
COBWEB's	Not available				

Table 2: Speed comparisons for unsupervised learning methods

Another advantage of the SGNT method over the conventional unsupervised learning methods is that the class information (if available) can be used to improve the classification performance significantly. For the MONK's problems, if we use the class information to calculate the first and second order information gains [9] of the attributes and use the fist order information gains selected according to both first and second order information gain features as the attribute weights, we obtain the results given in Table 3. The selective first order information gains used as the attribute weights are 0.0753, 0.0058, 0.0, 0.0, 0.2870, 0.0 for M_1 and 0.0, 0.2937, 0.0, 0.0029, 0.2559, 0.0 for M_3. For M_2, because all first and second order information gains are less than 0.1, meaning that all single attributes and pairs of attributes have little contribution to the classification result, we use attribute weights proportional to the reciprocal of the first order information gains to reduce the deviation caused by the training examples.

[1] All test results except those for SGNT and LVQ are given by [6].

Learning method tested	Test result			Tester(s)
	M_1	M_2	M_3	
SGNT (supervised)	100%	87%	100%	Wen et.al.
AQ17-DCI	100%	100%	94.2%	Bala et. al.
AQ17-HCI	100%	93.1%	100%	Bala et. al.
AQ15-GA	100%	86.8%	100%	Bala et. al.
Assistant Professional	100%	81.3%	100%	Cestnik et. al.
mFOIL	100%	69.2%	100%	Dzeroski
ID5R	81.7%	61.8%	N/A	Van de Velde
IDL	97.2%	66.2%	N/A	Van de Velde
ID5R-hat	90.3%	65.7%	N/A	Van de Velde
TDIDT	75.7%	66.7%	N/A	Van de Velde
ID3	98.6%	67.9%	94.4%	Kreuziger et.al.
ID3 (no windowing)	83.2%	69.1%	95.6%	Kreuziger et.al.
ID5R	79.7%	69.2%	95.2%	Kreuziger et.al.
AQR	95.9%	79.7%	87.0%	Kreuziger et.al.
CN2	100%	69.0%	89.1%	Kreuziger et.al.
PRISM	86.3%	72.7%	90.3%	Keller
Backpropagation	100%	100%	93.1%	Thrun
Backprop. with weight decay	100%	100%	97.2%	Thrun
Cascade Correlation	100%	100%	97.2%	Fahlman

Table 3: Comparisons for supervised methods

Although the SGNT/SGNN method was mainly developed for the purpose of unsupervised learning and the class information is only used to assign weights to the attributes (never used in training process), Table 3 shows that the performance of SGNT is still quite impressive.

The speed comparisons in Table 4 show again the SGNT method is much faster than any other methods for supervised learning. The time spent by calculating the first and second order information gains for all three MONK's problems are the same, 0.2 second, and is not included in the training time given in Table 4.

Learning method tested	Training time (in seconds)			Platform used for testing	Tester(s)
	M_1	M_2	M_3		
SGNT (supervised)	1.50	3.06	0.85	SUN SPARC 1	Wen et. al.
ID3	35.51	154.02	23.04	SUN SPARC 1^+	Kreuziger et.al.
ID3 no windowing	4.98	7.61	3.74	SUN SPARC 1^+	Kreuziger et.al.
ID5R	99.20	407.61	78.91	SUN SPARC 1^+	Kreuziger et.al.
AQR	4.17	9.45	4.00	SUN SPARC 1^+	Kreuziger et.al.
CN2	4.48	74.04	10.25	SUN SPARC 1^+	Kreuziger et.al.
Cascade correlation	5.11	7.75	12.27	DECSTATION 3100	Fahlman
Backprop.	10 to 30			SUN SPARC	Thrun

Table 4: Speed comparisons for supervised learning methods

4 DISCUSSION

It turns out that the SGNT method has quite a few attractive features comparing with the conventional neural networks and the other unsupervised learning techniques in conventional AI:

1. The structure is determined adaptively according to the task at hand. The performance is quite satisfactory. Whenever it is difficult to determine the right network structure, this method may well

be considered as an alternative to conventional neural network methods such as backpropagation for which the network structures have to be magically determined by human experts.

2. There is an efficient use of neurons and links to solve the current task. Therefore, some large scale applications such as mushroom classification, image coding, and information retrieval can be implemented on personal computers without difficulty. The optimization, pruning, and simplification techniques make our method more practical for large scale applications.

3. There is no delay caused by redundant neurons in conventional neural nets. Only a small part of the SGNT hierarchical structure needs to be searched to find the right place to update the network. This provides the main speed advantage over most other learning methods, and makes it possible to apply our method to real time applications.

4. The method can be used for supervised, unsupervised or even hybrid learning tasks depending on whether the class information is available or not.

Because our method is mainly an unsupervised learning method, the results for supervised learning may be not as attractive as those for unsupervised learning, but considering that there is nearly no structure design effort, which cannot be avoided for methods like backpropagation, the advantage of SGNT is still obvious.

Future work[7] will give a performance analysis for Self-Generating Neural Networks (SGNN) allowing an acyclic directed network structure which is more general than tree structure. The SGNN method performs better than SGNT but is slightly slower than SGNT. The further research issues include recurrent network structure and applying the method to some application areas, such as speech recognition, and natural language understanding, etc.

Acknowledgement

The permission of the Director of Telecom Australia Research Laboratories to publish this paper is acknowledged.

References

[1] L. Fang, A.Jennings, W.Wen, K.Li, and T.Li. Unsupervised learning in a self-organizing tree. In *Proc. IJCNN'91 (International Joint Conf. on Neural Networks)*, Singapore, Nov. 1991.

[2] W. Iba, J.Wogulis, and P.Langley. Trading off simplicity and coverage in incremental concept learning. In *Proc. 5th International Conference on Machine Learning*, pages 73–79, 1988.

[3] A. Kowalczyk et al. Discovering production rules with higher order neural networks: a case study. In *Proc. 8th Workshop on Machine Learning*. Morgan Kaufmann, 1991.

[4] J. Quinlan. Simplifying decision trees. *Int.J.Man-Machine Studies*, 27:221–234, 1987.

[5] J. Schlimmer. Concept acquisition through representational adjustment. Technical report 87-19, Department of Information and Computer Science, University of California, Irvine, 1987.

[6] S. Thrun et al. The MONK's problems: A performance comparison of different learning algorithms. Tech Report CMU-CS-91-197, Carnegie Mellon University, Dec. 1991.

[7] W. X. Wen, A. Jennings, and H. Liu. A performance analysis for self-generating neural networks. In *Proc. IJCNN'92 (International Joint Conf. on Neural Networks), Beijing, China*, Nov. 1992.

[8] W. X. Wen, A. Jennings, and H. Liu. Self-generating neural networks and their applications to telecommunications. In *Proc. ICCT'92 (International Conference on Communication Technology)*, Beijing, China, 9 1992.

[9] W. X. Wen, A. Jennings, and H. Liu. Supervised learning in self-generating neural networks. Research Labs. Report, in preparation, May 1992.

[10] W. X. Wen, H. Liu, and A. Jennings. Self-generating neural networks. In *Proc. IJCNN'92 (International Joint Conf. on Neural Networks)*, Baltimore, June 1992.