A Comparative Study between SGNT and SONN

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

(To apear in Proc. AI'92, Hobart, Australia, Nov. 1992)

Abstract

Comparisons between the Self-Generating Neural Network (SGNN) and the Self-Organizing Neural Network (SONN) are performed. Although the SGNN concept was developed from SONN, the results obtained show that it has quite a few significant advantages when compared with SONN. These include simplicity of design methodology, greater speed for both training and testing, higher accuracy of clustering/classification, and better generalization capability. An analysis is also conducted to investigate why SGNN is superior to SONN.

1 INTRODUCTION

The SGNN (Self Generating Neural Network) method proposed in [11, 8, 1] is based on both SONN (Self-Organizing Neural Network) [4] and traditional AI unsupervised learning methods, such as COBWEB and ARACHINE [2, 5]. It has been applied to different applications such as image coding, diagnostic expert systems and document/information retrieval systems [10]. Two SGNN systems have been developed on SUN SPARC stations and IBM-PC/AT with X-window/MS-window 3 interfaces, respectively. Comparisons have been given in [9] to show the advantages of the SGNN method over COBWEB/CLASSWEB/ARACHINE. The purpose of this paper is to give a comparative analysis between the performances of a type of SGNN, Self-Generating Neural Trees (SGNT), and SONN to show the difference between them and the reasons for this difference. The MONK problems [7] – a de facto standard benchmark set for testing supervised/unsupervised learning methods are used for comparison. The result of the comparison shows that SGNN/SGNT is superior to SONN in aspects such as the simplicity of design, speed of training and testing on conventional computer systems, better clustering/classification results, and better generalization capability.

For SGNN/SGNT generation, optimization, pruning, and simplification algorithms see [11, 8].

2 SGNT vs. SONN

Although the SGNN/SGNT bears much resemblance to SONN, it has many of its own features. To compare SGNT/SGNN with SONN, we conducted a systematic performance analysis for both methods. The benchmark that we chose was the MONK's problems [7]. The reasons for this choice are 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 the summer of 1991. This makes an independent comparison of different methods possible. The training and test data and the test results for other well-known methods are easily accessed. The MONK's problems are not too complicated or time-consuming to test but are reasonably difficult to solve. The MONK's problems [7] 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.

The results given in [9] show that SGNN/SGNT outperforms the most popular unsupervised learning methods, such as ECOBWEB and CLASSWEB, and is significantly faster than all those competitors. In this paper, we will use MONK's problems again to compare SGNN/SGNT and SONN. The comparison is conducted in three aspects: simplicity of network design, accuracy of classification/clustering and generalization capability, and the training/testing speed.

2.1 Design Efforts

As implied by the name Self-Generating, the design of an SGNT/SGNN requires much less effort from the designer. For the design of an SONN, such as LVQ [4], to handle a particular application, the designer has to decide

1. how many neurons the network needs,
2. how the neurons are arranged in the networks,
3. what is the neighbourhood size to use and how the neighbourhood should shrink during the training,
4. the usage count arrangement to make better use of the neurons, and
5. the appropriate learning rate for training the network and the strategy to reduce it.

Some of these system parameters affect the performance of the SONN significantly. This means that if they are selected improperly, the final system performance may not be as good as expected. Too many neurons or links will obviously waste the system resources and also cause unnecessary delay because of the exhaustive search strategy of SONN during both training and testing. Once the structure of the network has been determined, it remains fixed. Any change of the application requirement will lead to a complete re-design and re-training of the network. The rest of the decisions that the network designer has to make are very tricky and there is no standard way to choose these parameters convincingly. The usual practice is to perform multiple experiments to choose them empirically. Therefore, this is often a tedious and fruitless design task.

In the case of MONK's problems, we performed experiments with 6×6, 8×8, 12×12, and 14×14 LVQ grids. The 12×12 and 14×14 networks did not give significantly better performance than that of 8×8 network for MONK's problems, but 8×8 network was significantly better than 6×6 network. There was a severe penalty when increasing the network size because training time would become longer due to the number of nodes needed to be searched. In addition, the neighbourhood size for training must be initially large and this will further add to the length of the training schedule. Therefore, we decided to use the 8×8 network for our comparisons with SGNT/SGNN. Because there is no standard way of choosing the diameters of the neighborhoods and their shrinking strategy for LVQ network training, we chose them empirically according to experience accumulated in previous experiments (see Table 1). The learning rates used in the LVQ networks is also given in Table 1. Neuron usage counters were not used because we were not concerned with making the best use of the neurons. We used two metrics, city block and Euclidean distance, for LVQ net training and testing. Our results seem to show that city-block is on average slightly better for the MONK's problems. This might be because an Euclidean metric requires the computation of a square of the sum followed by a square root, whereas a city block requires only the sum of the absolute differences. The networks were trained using 110 epochs in total.

learning rate	neighbourhood diameter	epochs
0.06	4	10
0.06	3	30
0.05	2	20
0.04	1	20
0.025	0	30

Table 1: The training schedule for LVQ nets

In the case of SGNN/SGNT, all that needs to be done is to choose an appropriate parameter ξ and if we are lazy to do so, we can always choose $\xi = 0$. The SGNN/SGNT system will do everything else for the development of the neural network. According to our experiences, the system does this quite well. For the MONK's problems, SGNT generated three networks with 156, 168, and 82 neurons, respectively. These figures are greater than the 64 neurons used by an 8×8 LVQ network, but it should be noted

that, For MONK problems, there was no significant difference in performance between 8×8 and $14 \times 14 (= 196)$ LVQ networks except that the former was significantly faster to train/test than the latter. The generated SGNT networks had 5 levels and each of them was generated in only 5 epochs.

2.2 The accuracy comparison

The results of SGNT and SONN methods for the MONK problems are given in Table 2 where CB stands for City Block, and Eu for Euclidean. The

Learning method tested	Test results		
	M_1	M_2	M_3
SGNT	82.6%	78.2%	84.5%
8×8 LVQ (CB)	76.1%	63.0%	81.2%
8×8 LVQ (Eu)	72.5%	64.4%	75.7%

Table 2: Accuracy comparisons for unsupervised learning methods

results of LVQ nets show the averages of 10 runs with different initial weight sets. The results show that SGNT is better than SONN for the easier tasks such as M_1 and M_3. For the harder task M_2, the performance of SGNT is much better than SONN, as is the average performance of SGNT. This comparison shows that the fixed uniform structure (such as a grid structure) of SONN cannot be a perfect panacea for all applications. For different applications, ideally, different networks structures should be carefully designed. SGNN/SGNT gives a possible alternative to heavily handcrafted network methods, such as Neocognitron[3]. It automatically adjusts the neuron/link density in different parts of the network according to the training sample distribution for a particular application, whereas SONN treats all applications uniformly and consequently, wastes neurons/links in some parts of the network while has too few neurons/links to use in other parts even in a big network.

2.3 The training speed

For the MONK's problems, speed comparisons between the SGNT and SONN are given in Table 3. The SGNT results were obtained on SUN SPARC 1 workstation while the SONN results obtained on SUN SPARC 1^+. The results show that the SGNT is significantly faster than SONN. The problem here for SONN is that an exhaustive search of the whole network is performed to find a winner no matter how large the network is. Furthermore, as mentioned earlier, there are often redundant neurons in

Learning method tested	Training time (in sec)		
	M_1	M_2	M_3
SGNT	1.47	2.37	2.69
8×8 LVQ (CB)	62.74	85.72	62.04
8×8 LVQ (Eu)	86.12	117.01	84.56

Table 3: Speed comparisons for unsupervised learning methods

SONN, thus extra delays are added to the search. SGNT/SGNN avoids the pitfalls of SONN by only searching a very small part of the network hierarchically. For SGNT, this becomes a very efficient tree search ($O(\log_b N)$). The SGNN/SGNT algorithm generates neurons/links only when they are really needed and the pruning and simplification of the network further reduce redundancy. Therefore, it is not surprising that SGNN/SGNT should be much faster to train/test than SONN.

3 DISCUSSION

> "The thought that my mind is really nothing but an empty sieve - often this, too, disconcerts me."
>
> Logan Pearsall Smith, "All Trivia"

There are a great many neural network researchers concerned with the issue of "scaling up" neural networks to tackle large problems, and more difficult problems than pattern recognition. A task such as target identification in the presence of noise and clutter is fairly well treated by current networks, but a more difficult task such as handwritten text is still fairly difficult. In all practical cases of network application to large problems, a strict network structure is constructed by the network designers. For example the AT&T zip code network has seven layers and a complex hierarchical structure [6]. More well known is the Neocognitron [3]. The success of the neocognitron derives very much from its structure and its careful training regime wherein the feature set is at a progressively higher level of abstraction towards the top of the network.

So it appears that to tackle large problems some structuring of the problem is necessary. The key question then becomes: how do such networks arise in nature? There is no designer present carefully feeding examples to the network in such a way that it has tractable and computationally feasible learning tasks at each layer. To resolve this question we can proceed in two ways: we can look in nature for how the networks develop, or attempt to construct

models that emulate the biological capability. Our work on SGNT follows this second path, and the results on standard problems show that it is a powerful mechanism. SONN shows some promise in delivering a network solely on the basis of the sensory data. For example Kohonen's network [4] is quite successful at developing a network for pattern recognition. It gives the appearance of "principle free" learning: that the network structure arises solely on the basis of the training data.

However Kohonen's network incorporates a neighbourhood measure that is reduced as the training of the network proceeds, which plays a major part in the network structure. Even though this is not a "principle based" method, it is still giving the network designer considerable control over the end result. So we argue that SONN's are developed with influence from the designer in selecting this schedule.

The SGNT approach is quite different: there are guiding principles and constraints that influence the formation of the network structure. These principles are derived from traditional AI concept formation/decision tree methods, so they have apparently little in common with biological mechanisms.

What are the biological equivalents of our pruning rules? Given that artificial neural networks are a greatly abstracted view of neural networks, we cannot say at this stage. But on the surface it does appear reasonable to limit the number of poorly formed hierarchies. Perhaps we can discover biological mechanisms that are similar. Even though speed of training is not of great importance in itself, it does show economy of computation which may imply evolutionary advantage for the corresponding biological network. There is considerable evidence that network development is not from a "blank sheet".

Acknowledgement

Discussions with Maureen Molloy assisted in the preparation of the discussion. 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] D. Fisher. Knowledge acquisition via incremental conceptual slustering. *Machine Learning*, 2:139–172, 1987.

[3] K. Fukushima and N. Wake. Handwritten alphanumeric character recognition by the neocognition. *IEEE Trans. on Neural Networks*, 2, No. 3:355–365, 1991.

[4] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.

[5] K. McKusick and P. Langley. Constrains on tree structure in concept formation. In *Proc. IJCAI'91*, volume 2, pages 810–816, Sydney, Aug. 1991. Morgan Kaufmann.

[6] E. Sackinger et al. Application of the ANNA neural network chip to high-speed character recognition. *IEEE Trans. on Neural Networks*, 3, No. 3:498–505, 1992.

[7] 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.

[8] W. X. Wen, A. Jennings, and H. Liu. Learning a neural tree. In *Proc. IJCNN'92 (International Joint Conf. on Neural Networks), Beijing, China*, Nov. 1992.

[9] 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.

[10] 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.

[11] 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.