On-Line Expansion of Goal Seeking Neuron Networks

Weber Martins[1,2], Carlos Galvão Pinheiro Jr.[2]
[1] Catholic University of Goias, Department of Psychology
[2] Federal University of Goias, School of Electrical Engineering
PIRENEUS Research Group, Goiânia, GO, BRAZIL
Emails: {weber,cgalvao}@pireneus.eee.ufg.br

Abstract

Networks of Goal Seeking Neurons (GSN) [1] are weightless models that were designed to overcome several problems with PLN networks [2]. Weightless (Boolean) models are well-known by their ease to implement on hardware and software. Some hardware implementations and general improvements have been proposed to these models [3, 4, 5] specially to perform better with their main target task: classification. In this work, an improvement is proposed and tested on classifiers that use GSN networks as their building blocks on a hand-written digit recognition task. More specifically, the improvement is related to the expansion of GSN, decreasing saturation problems during learning and therefore allowing the use of more examples during training. GSN uses a fast one-shot learning algorithm and it is not allowed to modify used positions. Therefore, during the training, several examples are refused by the net since it cannot be learned. This behaviour indicates when a GSN network is saturated. By using the idea of expanding the network when it is needed, it is possible to overcome this limitation. The main issue becomes where and when to expand it. A determinist method for training the network is proposed, allowing it to expand where it is necessary. Results of simulations show that the proposed improvement is really promising, leading to significant gains of performance. Moreover, some ideas can be applied on many other neural models.

1. Introduction

The most of papers on neural networks are concerned with analog/continuous models. Nevertheless, weightless (Boolean) models are already famous by their low-cost and ease to implement on hardware and software. Much of the foundation for Boolean neural networks was set by Caianello [6]. Networks of Goal Seeking Neurons (GSN) [1] are weightless models that were designed to overcome several problems with PLN networks [2], the successor of the first logical model. Some hardware implementations and general improvements have been proposed to these models specially to perform better with their main target task: classification. In [4] two improvements are proposed to deal with the output codes and to genetically select the inputs for each classifier network. In [10], GSN networks are combined with Adaptive Logic Networks by providing better initialization for them and in [11] they are used to avoid false examples.

In this work, an improvement is proposed and tested on classifiers that use GSN networks as their building blocks on a hand-written digit recognition task. More specifically, the improvement is related to the expansion of GSN, decreasing saturation problems during learning and therefore allowing the use of more examples during training. GSN uses a fast one-shot learning algorithm and it is not allowed to modify used positions. Therefore, during the training, several examples are refused by the net since it cannot be learned. This behaviour indicates when a GSN network is saturated. By using the idea of expanding the network when it is needed, it is possible to overcome this limitation. The main issue becomes where and when to expand it. A determinist method for training the network is proposed, allowing it to expand where it is necessary, dynamically, on-line. In [3] some experiments were performed by using neurons with higher fan-in (number of inputs) than two. Their topologies are, however, homogeneous and set up at the beginning and off-line (static).

Results of simulations show that the proposed improvement is really promising, leading to significant gains of performance. Moreover, some ideas here described can be applied on many other neural models since a simple constructive strategy is formulated and used.

2. Networks of Goal Seeking Neurons

Bledsoe and Browning [7] proposed the idea of accessing sub-elements of binary input patterns and storing a flag to indicate the appearance of each possible combination (states) to be used in pattern recognition tasks. This gave rise to the n-tuple method where n is the number of observed bits.

Motivated by difficulties in implementing McCulloch-Pitts [8] neurons, the special-purpose SLAM (Storage Logic Adaptive Microcircuit) device was designed, later based on RAM elements. The RAM (Random Access Memory) neuron accepts and produces binary values only. It can be understood as a lookup-table with 2^n addressable

0-7695-0619-4/00 $10.00 © 2000 IEEE

positions, where n is the neuron fan-in, with all the 2^{2^n} Boolean functions computable and therefore no generalisation is obtained at the neuronal level.

Searching for a probabilistic behaviour the PLN (Probabilistic Logic Neuron) was proposed. The main difference between PLN's and RAM's is the inclusion of a third value *"u"* (undefined) *inside* the neuron. When accessed it produces zeros and ones with predefined probabilities. Due to poor results obtained attempting to apply PLN networks on a real-world application the GSN (Goal Seeking Neuron) model was developed. A GSN can generate, store and receive the *"u"* value. This particular ability implies several changes in the algorithm used in PLN networks. There are several architectures of GSN networks but all of them uses a pyramid in the sense that the output of each neuron is forwarded to only one neuron (fan-out = 1). In this article, however, only the feed-forward (supervised) configuration is used. A GSN can be in one of three states: validation, learning and recall.

At validation state, the system aims to check if it is possible to make the requested (current) input-output mapping by feed-forwarding the undefined value as often as possible. While a PLN uses random choice when an *"u"* value is accessed, a GSN sends *"u"* to the next neuron. This value as an input is treated as both logical values, *"0"* and *"1"*.

As the GSN model uses one-shot learning, it assumes that a value once stored can be never replaced by a different value. When the output is equal to *"u"* or to the desired value to be learnt, the pattern (requested input-output mapping) is validated. Otherwise, the pattern is refused, not learned, discarded.

Learning takes place by defining which path should be used for the required (and accepted) mapping. The main constraint in this situation is to re-use as often as possible internal locations where defined values are already stored, avoiding (or postponing at least) saturation. An undefined value is converted to a defined one only if necessary. No attempt to expand the number of inputs (neuron fan-in) is undertaken. The entire process is done backwards, from the output to the input level, so ensuring that each neuron output is defined first. The address of the location chosen for the last neuron specifies the outputs for the neurons associated with its inputs. When several internal positions are addressed and are undefined, a random choice takes place.

When at recall state, the neuron has to produce the binary content value with the highest occurrence in the addressable set in a feed-forwarding operation. If *"1"* occurs as often as *"0"*, the undefined value "u" is propagated forwards.

3. The Expansion Procedure

The challenge on expand an neuron inside a pyramid (tree) is to find out which one was more responsible for the refusing of a pattern. This can be easily accomplished by constraining that it should exist a high correlation between the neuron output in any level (but the first one, where inputs come from the environment, problem under analysis) and the desired output produced by the network.

The standard procedure for storing values in neuron internal positions wouldn't be effective since a random choice is undertaken when there is a doubt on where to store a value. The maintenance of high correlation between neuron and network outputs is not of concern. The proposed procedure, on the other hand, keeps this correlation at high level by assuring that the chosen addressed position has the maximum number of values in the input equals to the desired output. For instance, if the positions 111 and 010 are addressed and a "1" has to be stored, it will occupy the position 111. Once no random choice is made, a (positive) high correlation is imposed to the network and neuron outputs.

To determine when a certain neuron should be expanded, a *block* counter was created, so when the neuron output was different from the desired output, this counter is increased by 1.

A neuron is expanded if the block rate is above 10% of the training set size, its fan-in (number of inputs) is less than 5 and the number of inputs used by the tree after the expansion is equal to or less than the input pattern dimensionality.

In the process of expansion, the neuron receives a new *son* (input) which has the same topology of any other son when the network was first created (the initial topology).

4. Experiments and Results

The total data set is composed by 800 hand-written digits distributed equally on ten classes (0 to 9). The training set is composed of the first 160 examples, remaining 80% of the total just for the evaluation set. A few samples of the data set are given in Figure 1.

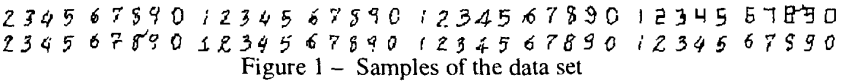

Figure 1 – Samples of the data set

Each example is a binary 16x16 pixel image. The example dimensionality is, therefore, 256. The images were collected from a digitalizer where regions were predefined to mark where segmentation would take place. No centering or scaling pre-processing operation, however, were used.

It's known that the connectivity and coverage area are two major parameter analysis in the design of a GSN based neural network [9]. It was, therefore, employed neurons which fan-in 2, 3 or 4. The coverage area was always less than or equal to half the pattern dimension (128). During the expansion procedure, it was however allowed the tree to grow and cover the whole input space. For each architecture, we used 3, 5 and 9 trees to each bit in the output code. A 16x16 Hadamard matrix was used to create output codes. This particular code has the advantage of keeping each class code with Hamming distance equals to 8 to every other class code. The initial networks (pyramids, trees) are homogeneous. Each result displayed is the average on 50 simulations. Figure 2 shows the performance at evaluation. Table 1 focus on an interesting result (an average result too): two configurations that ended up with similar number of inputs per network. One of them uses homogeneous (static, not dynamically expanded) networks with fan-in = 4 while the other one uses the proposed expansion procedure applied to networks initially with fan-in = 2. Both of them uses four levels at the pyramid structure.

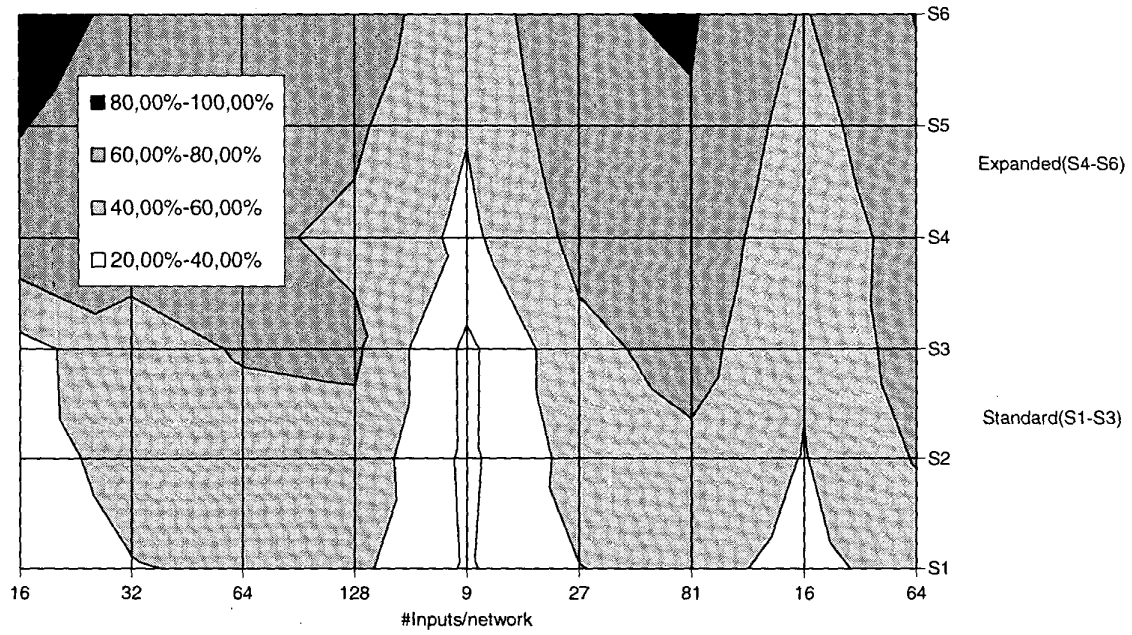

Figure 2: Performance at evaluation set. S1, S2, S3:standard GSN; S4, S5, S6: expanded GSN. S1 and S4: 3 nets/output code bit; S2 and S5: 5 nets/output code bit; S3 and S6: 9 nets/output code bit. # Inputs/network: 16, 32, 64, 128 (fan-in = 2); 9, 27, 81 (fan-in = 3); 16, 64 (fan-in = 4).

	# inputs/net	# net/bit	Performance
Expanded	240,3	3	75,43%
Initial fan-in = 2	238,4	5	80,58%
Level = 4	238,8	9	82,38%
Static (standard)	256	3	50,75%
Fan-in = 4	256	5	61,73%
Level = 4	256	9	70,68%

Table 1 – Average performance at evaluation of two selected architectures (expanded and static) with same level and similar coverage areas.

In general, by looking at Figure 2, one can observe how the expansion has often increased the performance of the classifier system at evaluation. Of course, this increased performance has cost the expansion of the topology.

More interesting, however, is to compare two similar architectures as it was shown by Table 1. On average, it is better to evolve/expand a small network than start with a large one. This result supports claims of constructive neural models where learning controls of topology.

5. Conclusions

This paper has presented a strategy to decrease saturation in GSN networks and increase the number of used patterns (examples) during training. As a consequence, performance is also increased. Results were shown on a real-world task.

The main idea is to identify which neurons are responsible for the example blocking and to expand them by increasing their number of inputs. To measure the neuron responsibility, it was employed counters and the concept of high correlation on all levels (but the first one) between neuron and network outputs.

Of course, this expansion is a price to pay but it is better to begin with small networks and evolve them dynamically than to start with large homogenous topologies. This constructive approach could be adapted on other neural models as well as other ideas that have emerged on weightless neural models.

References

[1] Filho, E., Bisset, D. L., and Fairhurst, M. C., 1990, "A goal-seeking neuron for Boolean neural networks", in *Proc. of the IEEE Internetional Neural Networks Conference*, Paris, France.

[2] Kan, W. K., and Aleksander, I., 1987, "A probabilistic logic neuron network for associative learning", in *Proc. IEEE Intl Conference on Neural Networks*, vol II, 541-548, San Diego, USA.

[3] Bowmaker, R. G., and Coghill, G. G., 1992, "Improved Recognition Capabilities for Goal Seeking Neurons", in *Electronic Letters*, vol 28, 3, 220-221.

[4] Martins, W., and Allinson, N. M., 1993, "Two improvements for GSN neural network", in *Proc. Weightless Neural Network Workshop '91*, 58-63, York, UK.

[5] Martins, W. and Allinson, N. M., 1994, "Optimization of Presentation Order in Networks of Goal Seeking Neurons", in *Proc. World Congress on Neural Networks*, vol IV, 110-115, San Diego, USA.

[6] Caianello, E. R., 1961, "Outline of a theory of thought-proccess and thinking machines", in *Journal of Theorical Biology*, vol 2, pp 204-235

[7] Bledsoe, W. W. and Browning, I.,1959, "Pattern recognition and reading by machine", *in Proc. of The Eastern Joint Computer Conference*, pp. 225-232

[8] McCulloch, W. S., and Pitts, W. H., 1943, "A logical calculus of the ideas immanent in nervous activity", in *Bulletin of Mathematical Biophysics*, 5,1943, pp. 115-133

[9] Filho, E., 1990, "Investigation of Boolean Neural Network Based On A Novel Goal-Seeking Neuron", PhD Thesis, Electronic Engineering Labs, University of Kent, UK.

[10] Martins, W. and Allinson, N. M., 1994, "Combining Adaptive Logic Networks and Goal Seeking Neurons", in *Proc. European Conference on Artificial Neural Networks*, vol II, 863-866, Sorrento, Italy.

[11] Martins, W. and Allinson, N. M., 1996, "Avoiding False Examples By Use of Cooperation", in *Proc. World Congress on Neural Networks*, 795-798, San Diego, USA.