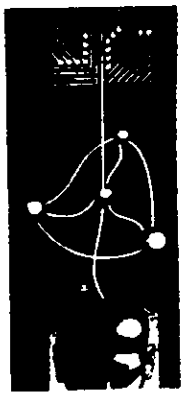

1° Congresso Brasileiro de Redes Neurais

Escola Federal de Engenharia de Itajuba
Itajuba, 24 a 27 de outubro de 1994

Issues on the Complexity of Training Weightless Neural Networks

Marcílio C. P. de Souto
Katia S. Guimarães
Teresa B. Ludermir

Universidade Federal de Pernambuco
Departamento de Informática
Cx. Postal 7851 - 50.732-970 - Recife - PE - Brazil
{mcps,katia,tbl}@di.ufpe.br

Abstract. In this paper, it is extended the Judd's results with respect to learning computational complexity of weighted neural models to include the weightless neural models. It is shown, for example, that also is NP-complete any algorithm that aims to *load any* performable training set in *any* conceivable weightless neural network. It is also conjectured that a specific architecture class, *pyramidal* architectures (for the weightless models), may be a way to overcome the NP-completeness of learning.

1 Introduction

One of the most important features of neural networks[1] is their ability to generalize to new situations. Once trained, a network will compute an input/output mapping which, if the training data was representative enough, will closely match the unknown rule which produced the original data.

The work in this paper deals with basic theoretical questions regarding learning by neural networks, it was inspired by Judd ([7]) who shows the following problem to be NP-complete:

> "Given a neural network and a set of training examples, does there exist a set of edge weights for the network so that the network produces the correct output for all training examples?"

this problem is called the **loading problem**.

Judd developed his work based on McCulloch-Pitts (MPC) neurons ([10]). MCP neurons are implemented by threshold logic gates, where variable input weights play a role analogous to that of synapses in natural neurons. The models used in this paper is based on a different model called the weightless neuron model ([1]). The weightless model is based on the simple operations of look-up table which is best implemented by random access memory (RAM) and where knowledge is directly "stored" in the memory (the look-up tables) of the nodes during learning. Some advantages of this model are: (1) it is straightforward to implement in hardware; (2) learning is not unreasonably slow and (3) error-correction requires only global success signal ([9]).

It is important to point out that there is no study on learning complexity of weightless models, being this paper the first effort towards putting these models in the context of learning computational complexity. Another question is that this kind of study can help to design neural networks, because it identifies underlying problems in learning and tries to find ways to avoid them. Thus, these can yield techniques to neural network design.

Some background on computational complexity is necessary for a better understanding of this work. Here, whenever it is said polynomial time it is meant polynomial time in the length of any binary encoding of the input and problems approached here are always decision problems ([5]).

A problem is in class P when there is a polynomial time algorithm which solves the problem. A problem is in NP when a "guessed" solution for the problem can be verified in polynomial time. A problem (set) H is NP-hard iff for each problem (set) Q in NP, there is a polynomial time transformation f_Q from Q to H, such that given any instance I of Q, $I \in Q$ iff $f_Q(I) \in H$. Then, a problem is NP-complete iff it is both NP and NP-hard. Examples of NP-complete problems are: the Boolean satisfiability problem, the traveling salesperson problem, the set-splitting problem.

The remainder of this paper is divided into three sections. Section 2 presents the weightless neural models and their main characteristics. The main section of this work is Section 3, which puts the weightless models in the context of computational complexity and Judd's work is used as base to discussion ([7]). It is also studied issues regarding loading *pyramidal* archi-

[1] In this paper, the term "neural network" always means feed-forward ones which have binary inputs/outputs and the learning paradigm analyzed is the supervised learning.

tectures, and it is made a parallel between theoretical and empirical results in complexity of learning. To develop this parallel are the backpropagation (weighted models) and pyramidal (weightless models) architectures used. Finally, the last section summarize the discussion in support to our conjecture.

2 Weightless Neural Models

Definition 1 A RAM Neural Network is an arrangement of a finite number of neurons in any number of layers, in which the neurons are RAM (Random Access Memory) nodes. A RAM node is represented in the Figure 1.

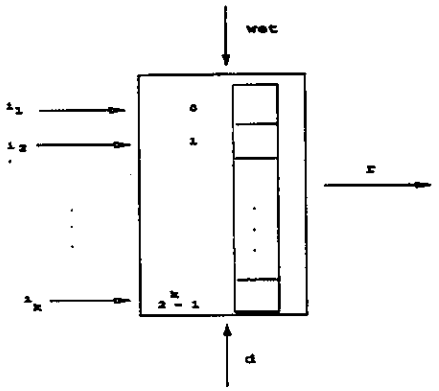

Figure 1: RAM node

The input to each neuron may be external input or outputs of neurons from another layer. The data out may be 0 or 1. The set of connections is fixed and there are no weights in such nets. Instead, the function performed by the neuron is determined by the contents of the RAM - its output is the value contained at the activated memory location. There are 2^{2^N} different functions which can be performed on N address lines and these correspond exactly to the 2^N states that the RAM can be in, that is, a single RAM can compute *any Boolean function* of its inputs.

Seeing a RAM node as truth table (look-up table) the output of the RAM node is described by Equation (1) below:

$$r = \begin{cases} 0 & if\ C[p] = 0 \\ 1 & if\ C[p] = 1 \end{cases} \qquad (1)$$

where C[p] is the contents of the address position associated with the input pattern p.

Definition 2 A PLN Neural Network is an arrangement of a finite number of neurons in any number of layers, in which the neurons are PLN (Probabilistic Logic Node) nodes.

A PLN node differs from a RAM node in the sense that a 2-bit number (rather than a single bit) is now stored at the addressed memory location. The contents of this location (C[p]) can be 0, 1, or u , and it represents one of three possibilities (0, 1, 0.5, respectively) of firing (i.e. generating a 1) at the output. The output of the PLN nodes described by Equation (2) below:

$$r = \begin{cases} 0 & if\ C[p] = 0 \\ 1 & if\ C[p] = 1 \\ random(0,1) & if\ C[p] = u \end{cases} \qquad (2)$$

where C[p] is the contents of address position associated with the input pattern p and random(0,1) is a random function that generates zeros and ones with the same probability.

Besides the RAM and PLN nodes there are many variations and extensions of the RAM node (e.g., MPLN ([11]), cut-point ([9]), GSN ([4]), called RAM-based nodes.

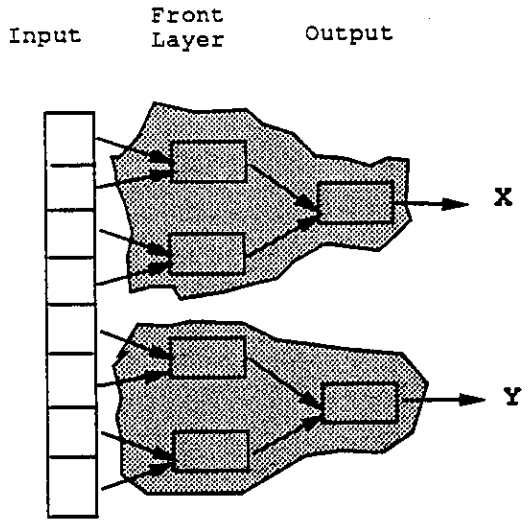

Figure 2: Example of a pyramidal architecture. The shaded and outlined area surrounding output nodes X and Y encompass all the nodes in each support cone. Note that each support cone correspond to an individual pyramid in the architecture.

Although there are several kinds of RAM-based nodes, the architecture used in most experiments developed is the *pyramidal* one (Figure 2). In this topology, neurons are arranged in hierarchical non-overlapping pyramids (trees), where each pyramid culminates in a

single output line. Therefore, if more than one output is associated with the problem, there will be one pyramid for each. As it has already mentioned, the number in the storage locations of RAM-based networks increases exponentially (2^N per node, where N is the size of the node input) with the size of the input problem, then the use of pyramidal architecture allows a decrease in this number.

3 Weightless Models and Complexity of Learning

Judd's work ([7]) was pioneer in the field of learning computational complexity of artificial neural networks. Before his work there was no characterization of learnability in terms of its computational complexity. Notwithstanding, the weightless models still lack this kind of characterization.

3.1 Judd's work in the context of Weightless Models

Judd studies several classes of architectures and shows each one of them to be NP-complete with respect to the loading problem. He does that by producing for each architecture, in its respective class, training examples such that any algorithm performs poorly on some networks and training set in that class (any instance of 3-Satisfiability problem can be transformed into polynomial time to some architecture and training set in this class). The more general case studied precludes only the most ambitious interpretation of the goal in connectionist learning. That is, the connectionist belief of finding an algorithm that is guaranteed to load *any* performable task in *any* conceivable network.

The results achieved are negative, because in that more general case (as in almost all subcases analyzed) the loading problem stays NP-complete. The only tractable case studied is, however, trivial, for it is a class of architectures (with *support cone interaction graph* having limited *armwidth* [7, 12]) which seems useless to practical activities. Nevertheless, the NP-completeness results define only the upper bound of the loading problem. Thus, they do not make impossible that in the average case it may be resolved in polynomial time.

It is interesting to verify that all Judd's results found in ([7]), which are related to learning computational complexity of weighted neural models, are also extendable to weightless neural models. This is because, the proofs found there are independent of any particular training algorithm and they are based on the set of Boolean functions[2]. A crucial point in those proofs, which allows this extension, is that neurons are considerate like truth tables (look-up tables). Thus, those proofs are directly extensible to weightless neural model, since weightless neuron models are truth tables as well. Examples of theorems and corollaries extended are:

> **Theorem 3** Loading weightless neural networks, whose node function set are only AND and OR functions, is NP-complete.
>
> **Corollary 3** Loading is NP-complete, independent if one are using weighted or weightless neural models.

This Judd's way to approach the loading problem was criticized by Blum *et. all* ([2]) and Dasgupta *et. all* ([3]) who proposed a study of the loading problem in terms of a *specific* neural network and node function set. Nevertheless, most of their results lead to the NP-completeness of learning.

3.2 Loading pyramidal architectures

One fundamental point not approached until now is the issue of loading deep networks. This issue usually are not analyzed in the weighted models, because the connectionist literature uniformly reports great hardness in loading these kinds of networks ([13, 8]).

In the context of weightless neural models, though, it is interesting to consider deep networks, since the architecture mostly used is the pyramidal, which tends to have high depth.

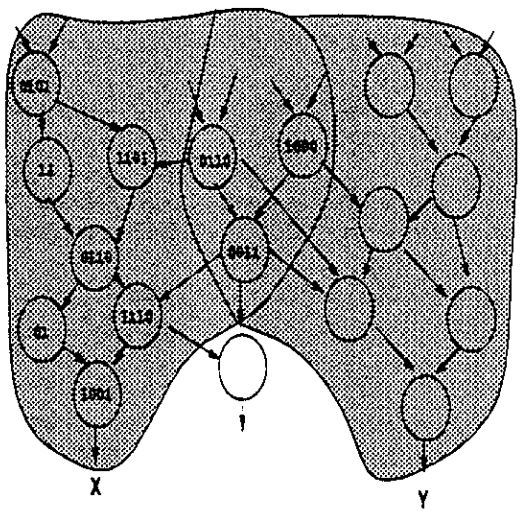

Figure 3: Illustration of support cones. The shaded and outlined area surrounding output node X encompasses all the nodes in its support cone. Likewise the support cone for output Y is shaded in.

[2] And-Or functions, linearly separable functions, all Boolean functions, etc.

Pyramidal networks form a special class of architectures. Given a network of this class it is important to verify that its support cones[3] do not overlap among them. This is a very important characteristic, once one of the causes to the hardness of learning is the interaction among support cones. Such interaction allows that constraints of choosing a configuration of functions in a determined support cone interferes in the choice of the configuration in the other overlapping support cone.

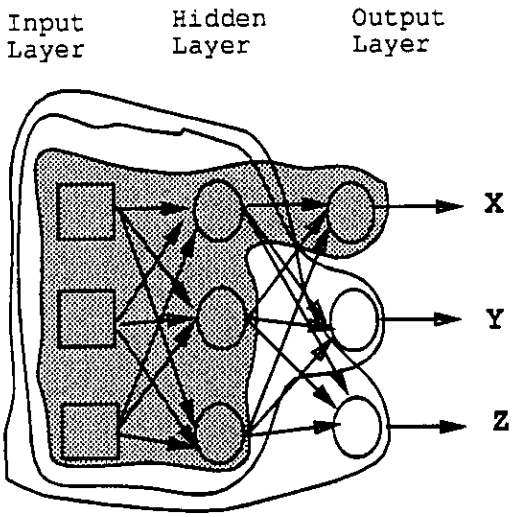

Figure 4: Example of a backpropagation architecture. The shaded and outlined area surrounding output node X encompasses all the nodes in its support cone. All the support cones in this architecture have the same nodes with exception of the output nodes, which is different to each support cone.

These negative results about the complicated interaction among support cones can be verified empirically, for example, looking at the features of the well-known and widely used backpropagation architectures ([6]). These architectures usually are fully connected, that is, a determined neuron of a level is connected to all neurons of the former. Therefore, they have the worst case in the interaction among support cones (Figure 4), for each support cone overlaps to all others. Empirically, it is acknowledged that as the network gets larger and deeper, the amount of time required for them to load the training data grows prohibitively ([8]). Looking at these results, in terms of backpropagation architectures, they reflect the increasing in the size of support cones. Such increase produces an explosion in the number of constraints that is necessary to deal with, hence in these networks learning becomes very hard.

On the other hand, there are experiments which weightless neural model using pyramidal architectures, when compared to backpropagation methods, can learn orders of magnitude faster ([11]). The weightless neural models often use pyramidal architectures, where each pyramid sees a part of the total input and there are no overlappings among pyramids (Figure 2). Consequently, each output neuron has its own support cone (in the case the total pyramid which it belongs to) and the choice of its support cone configuration will not interfere with the other support cones. Thus, in these architectures there are less constraints to deal with (since that the only concern is about individual pyramids) than the myriad of them that exist in the backpropagation architectures, such that learning becomes easier.

4 Final remarks

In this paper it has been sketched the first connections between the theory of computational complexity and weightless neural models learnability. The learning computational complexity in the weighted models have been reviewed, and one can verify that, in most cases, the results lead to the NP-completeness. With base on Judd's work these results were extended to include the weightless models.

Also, a parallel was made between backpropagation (weighted neural models) and pyramidal (weightless neural models) architectures in terms of interactions among support cones. It has been verified that there are experiments which show that weightless neural models using pyramidal architecture can learn orders of magnitude faster than the backpropagation ones.

Based on this it is conjectured that the class of pyramidal architectures may be a way to overcome the hardness of learning and it is being investigated if this is true. It is important to point out that the characterization of a polynomial architecture class (an architecture class that all tasks it can perform are loading in polynomial time) is a fundamental guide to neural networks design.

[3] This is the set of all nodes that can affect the behavior of an output node (Figure 3) [7]. Note that in Figure 3 the two cones overlap in three nodes. The binary numbers name node functions and as group they constitute a partial configuration for output node X.

References

[1] I. Aleksander and H. Morton. *An introduction to neural computing.* Chapman and Hall. London. Great Britain. 1 edition, 1990.

[2] A. L. Blum and R. L. Rivest. Training a 3-node neural network is NP-complete. *Neural Networks*, 5:117–127, 1992.

[3] B. Dasgupta, Siegelmann. H. T., and E. Sontag. On the complexity of training neural networks with continuos activation functions. Technical report, Rutgers University, New Brunswick, NJ, December 1993.

[4] C. B. C. E. Filho. *Investigation of Boolean Neural Network Based on a Novel Goal-Seeking Neuron.* PhD thesis, University of Kent at Canterbury, 1990.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, San Francisco, California, 1979.

[6] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Anais of International Joint Conference on Neural Network*, Washington, June 1989.

[7] J. S. Judd. *Neural network design and the complexity of learning.* The MIT Press, Cambridge, Massachusetts, USA, 1990.

[8] J. F. Kolen and A. K. Goel. Learning in parallel distributed processing networks: computational complexity and information content. 1990. To appear in IEEE Systems, Man, and Cybernetics.

[9] T. B. Ludermir. Learning algorithms for cut-point neural networks. In *Anais do X SBIA*, pages 433–443, Porto Alegre, RS, October 1993.

[10] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. In J. A. Anderson and E. Rosenfeld, editors, *Neurocomputing: Foundations of Research*, chapter 2, pages 18–27. The MIT Press, USA, 5 edition, 1990.

[11] C. Myers and I. Aleksander. Learning algorithms for probabilistc neural nets. In *IEE International Conference on Artificial Neural Networks*, pages 310–314, London, UK, October 1989. IEE.

[12] N. Robertson and P. D. Seymour. Graph Minor. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[13] G. Tesauro and R. Janssens. Scaling relationships in backpropagation learning: dependece on predicate order. *Complex Systems*, 2:39–44, 1988.