Taxonomy and Description of Weightless Neural Systems

Teresa B. Ludermir¹
Wilson R. de Oliveira²

Departamento de Informática
Universidade Federal de Pernambuco
Recife - PE - Brazil
CEP: 50.739
e-mails: tbl@di.ufpe.br    wrdo@di.ufpe.br

Abstract.

A taxonomy of weightless neural models is proposed which makes possible a general description for the different models of weightless systems. The main weightless models are described, using the taxonomy proposed, with their main characteristics. Some advantages of weightless system are: (1) systems may be built using conventional digital circuits, without the need to develop special VLSI devices, (2) learning is not unreasonably slow and (3) conventional Theory of Computation tools can be used to analyse their properties.

¹Partially supported by CNPq and FACEPE (Research Foundation of the State of Pernambuco in Brazil).
²Supported by the Brazilian Research Council CNPq-RHAE.
1 Introduction

Neural network models based on weighted-sum-and-threshold have been extensively studied in neural computing. The neural computing model studied in this paper is based on a different kind of artificial neuron called the weightless neuron model or the logical neuron model. The weightless neuron is based on the simple operations of a look-up table which is best implemented by random access memory (RAM) and where the knowledge is directly “stored” in the memory (via “look-up tables”) of the nodes during learning.

The weightless node has its origin on the n-tuple sampling machines of Bledsoe and Browning [7], where the n inputs to the node form the n-tuple which is used to address node memory. Aleksander took a great deal of interest in adaptive learning networks using n-tuple sampling machines and he suggested a universal logic circuit as the node of a learning network (Aleksander [2]). He introduced the SLAM (Stored Logic Adaptive Microcircuit) node, which was produced especially for research purpose before the availability of integrated circuit memories, and the RAM node as in Definition 3 below.

Weightless neural networks (WNN) share most of the main characteristics of Hopfield nets (seeking of energy minima at running time), Boltzmann machines (training of hidden units, escape from local minima) and error back-propagation (learning from errors). Besides, learning from errors in WNN is faster and more direct than in back-propagation. Furthermore, it is straightforward to implement in hardware using digital logic techniques. Tools from Automata Theory and Formal Languages can be used to analyse the theoretical capabilities of WNN (Ludermir [19]) (tractability of analytical studies). In contrast, the other models (Hopfield, Boltzmann) rely heavily on Theoretical Physics, in particular Statistical Mechanics. Another strong point in favour of the WNN approach has recently emerged in (Hong and Tan [16]), where it is shown that an analogue network where individual units output binary values can be simulated by a network of Boolean aggregates, with only a polynomial increase on the number of nodes and a logarithmic time slow down.

This work is part of a research project which aims at the mathematical characterisation of learnability in Artificial Neural Networks (ANN) (Ludermir [21]). Despite the existence of sound results showing that some mathematically well defined classes of function can be implemented \(^3\) in an ANN, the characterisation of what can be learned is still open.

The learnability problem in ANN is approached by, initially, fixing a particular class

\(^3\)To implement a function on an artificial neural network is to present a complete specification of an architecture (units, connections, etc.) in such way that the network behaves in exact correspondence with the given function. For example, (Cybenko [9]) shows how to implement any continuous function on the unit interval; (Hecht-Nielsen [14]) for any \(L_2\) function; and (Ludermir [20]) for the characteristic function of any set recognizable by a probabilistic automata.
of ANN, the Weightless Neural Networks, and we expect to obtain both theoretical and experimental results. The details can be found in (Ludermir [21]).

Thus, in this context, to determine the main components in the mainstreams WNN models is extremely important.

The continuation of this work is pursued elsewhere. A topological approach to learnability can be found in (de Oliveira and Ludermir [11]) and an algebraic approach in (de Oliveira and Ludermir [10]).

A formal definition of an ANN derived from this work is displayed in Section 5.

2 Taxonomy for Weightless Models

All weightless models are based on Random Access Memory node or neuron also called the logic neuron. In this section a taxonomy for this model is proposed which allows a general definition of a weightless neural network.

**Definition 1** A \(q\)-RAM neuron or node, \(q \in \{1, 2, \ldots \}\), is as depicted in Figure 1.

- The input terminals, \(i_1, i_2, \ldots, i_k\), may represent external input or the output of neurons from another layer or a feedback input.
- The data out, \(r\), may be 0 or 1.
- Each one of the \(2^k\) memory locations stores a \(q\)-value i.e. a value in the range \(\{0, 1, \ldots, 2^q - 1\}\).
- The write-enable terminal can be 0 or 1. If it is 0 the network is said to be in the test phase and no changes will be made on memory locations of the node. If it is 1, the network is in the training phase, changes will occur in the memory locations accordingly to the activation function.
- There are no weights associated to the input terminals, instead the function performed by the neuron is determined by the contents of the memory locations and the result of the boolean valued activation function\(^4\) \(g\). That is, \(r\) is the result of applying \(g\) to the \(q\)-value stored in the memory location determined by the input terminals.

**Definition 2** A weightless neural network (WNN) is an arrangement of a finite number of interconnected \(q\)-RAMs in several layers. WNN are also known as logical neural networks.

\(^4\)A boolean valued function is a function from any set \(X\) to \(\{0, 1\}\), \(f : X \rightarrow \{0, 1\}\).
Notation. A weightless neural networks composed of $q$-RAMs with $q$ fixed is called a $q$-WNN for short.

With the general definitions above the most important development of the subject can now be followed and the differences in each phase can be made explicit.

3 Weightless Models

3.1 RAM Model

Definition 3 A RAM neural network [3] is a 1-WNN in which the neurons are 1-RAMs. The 1-RAM is just called a RAM node.

The RAM node is represented in the Figure 1:

![Figure 1: RAM node](image-url)
The activation function is obviously the identity function. There are $2^N$ 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 function of its inputs.

Viewing a RAM node as a look-up table (truth table), its output is described by equation (1) below:

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

(1)

$C[p]$ is the content of the address position associated with the input pattern $p = i_1i_2...i_k$.

Learning in a RAM node, which is entered by activating the write-enable terminal, $wet$, takes place simply by writing to it. This learning process is much simpler than the adjustment of weights as in a weighted-sum-and-threshold network.

The RAM node, as defined above, can compute all binary functions of its input while the weighted-sum-and-threshold nodes can only compute linearly separable functions of its input. For example, the weighted neuron cannot compute the exclusive OR boolean function.

There is no generalisation in the RAM node itself (the node must store the appropriate response for every possible input), networks composed of RAM nodes do generalise (Aleksander [3]). Generalisation in logical networks is affected first by the diversity of the patterns in the training set - in other words, the more diverse the patterns in the training set, the greater will be the number of subpatterns seen by each RAM, resulting in a larger generalisation set. Secondly, the connection of RAMs to common features in the training set reduces the generalisation set.

A simple learning algorithm for RAM networks is summarised below:

**RAM learning algorithm.**

1. Present an input pattern to the input terminals.
2. Select the RAM nodes which should learn ($\alpha$ nodes are going to learn, $\alpha$ is a parameter set before training called the learning rate) and present the desired output to the vector $D$.
3. Mark the write enable terminals, $wet$, of the nodes that are going to learn in order to store in the memory positions addressed the desired outputs.
4. Repeat by going back to step 1 for all the N training patterns.
5. The algorithm halts when the error of the solution is acceptable (this parameter is also set before training). Otherwise, repeat the whole procedure by going back to step 1 again.
3.2 PLN Model

The introduction of a probabilistic element into the RAM node was proposed by Aleksander [4] after attempting a rapprochement between Boltzmann machines and RAM networks. He called the node with this probabilistic element a probabilistic logic node (PLN). The main feature of the PLN model is the unknown state, \( u \), which responds with a randomly generated output for inputs on which it has not been trained. A definition of a PLN node follows:

**Definition 4** A *PLN neural network* is a 2-WNN where the activation function is a probabilistic function.

The output of the PLN node is described by equation 2:

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

where \( C[p] \) is the content of the address position associated with the input pattern \( p \), and \( \text{random}(0,1) \) is a random function that generates zeros and ones with equal probability.

A PLN node differs from a RAM node in that a 2-bit number (rather than a single bit) is stored at the addressed memory location. Although it is possible to store four states in a 2-bit number only three states are used in the definition of the activation function by identifying 01 with 10. So in equation 2, 0 represents state 00, 1 represents state 11 and \( u \) represents states 01 and 10. The content of the memory location is turned into the probability of firing (i.e., generating a 1) at the overall output of the node. As in a simple RAM node, the \( k \) binary inputs to a node form an address into one of the \( 2^k \) RAM locations. Simple RAM nodes then output the stored value directly. In the PLN, the value stored at this address is passed through the activation function which converts it into a binary node output. Thus each stored value may be interpreted as affecting the probability of outputting a 1 for a given pattern of node inputs.

The contents of the nodes are 0's, 1's and \( u \)'s, where \( u \) means the same probability of producing a 0 or 1 output.

Learning with PLN networks becomes a process of replacing \( u \)'s with 0's and 1's so that the network consistently produces the correct output pattern in response to training pattern inputs. At the start of training, all stored values in all nodes are initialised to \( u \), and thus the network behaviour is completely unbiased. In a fully converged PLN net, every addressed location should contain a 0 or a 1, and the net's behaviour will be completely deterministic.\(^5\)

\(^5\)There may be PLN locations which are never addressed in the training process, such as addresses to nodes in the input layer which represent n-tuples not present in the training set, or to nodes in higher layers
A very simple learning algorithm for PLN networks is summarised as follows:

**PLN learning algorithm.**

1. Set the contents of all the memory locations to $u$.
2. Present an input pattern to the input terminals together with the desired response to the output terminals.
3. The contents of the memory locations are propagated forward through the layers of the network until they reach the output node.
4. The response of the network is compared with the desired output.
   
   (a) if they are similar, then the contents of the addressed memories will assume their current values, (i.e, the addressed memories will be rewarded), or
   
   (b) if they are different then the network is allowed to run again until:
   
   - The output matches the desired response, then reward.
   - The output mismatches the desired response $\beta$ times ($\beta$ is a parameter which should be set before training). In this case the addressed memories are punished by reverting their contents back to the $u$ state.

5. Repeat by going back to step 2 for all the $N$ training patterns.

6. The algorithm halts when consistent success (the correct output produced $N$ times consistently) indicates that all patterns have been learned.

### 3.3 MPLN Model

The Multiple-valued Probabilistic Logic Neuron (MPLN) (Myers [22]) is an extension of the PLN. A MPLN node differs from a PLN node in two ways. Firstly a q-bit number (rather than two bits) is now stored at the addressed memory location and secondly the probabilistic activation function is slightly changed. The contents of this location is also the probability of firing at the overall output of the node.

**Definition 5** A *MPLN neural network* is a weightless neural network where the nodes are q-RAMs and the activation function is a probabilistic function.

The output of the MPLN node is determined slightly different from the PLN node. The difference is that the q-value, say $m$, represents the firing probability of, say, $m/(2^q-1)$. For example, consider $q$ to be 3. Then numbers from 0 to 7 can be stored at each location of the if some combinations of lower node outputs never occur. These unaddressed locations may contain $u$ without affecting the status of the converged network.
MPLN. The actual number stored is as a direct representation of the firing probability, by treating the number as a fraction of 7. Thus a stored 2 would cause the output to fire with a probability of 2/7.

One result of extending the PLN to MPLN is that the node locations may now store output probabilities more finely gradated than in the PLN. For example, it is possible that a node will output 1 with 28.57% (2/7) probability under a certain input. The second result of the MPLN extension is that learning can allow incremental changes in stored values. In this way, one reset does not erase much information. Erroneous information is discarded only after a series of errors. Similarly, new information is only acquired after a series of experiences indicate its validity.

3.4 Remarks

There are three important parameters involved in the weightless model:

- the number $k$ of inputs to the nodes: has a direct influence on the memory requirements, and on the tradeoff between generalisation and memorisation.

- the number $q$ of bits used in the nodes: the number $2^q$ of possible stored valued numbers will influence the speed of learning.

- the activation function which is applied on the addressed content: the activation function may be probabilistic (PLN, MPLN) and may also be linear, step or sigmoid function (Myers [22]), as for weighted nodes.

The learning algorithm for PLN can be generalised to the weightless model, which changes the function $C[p]$ as follows:

$$C[p] = \begin{cases} C[p] - \eta g(r) & \text{punishment} \\ C[p] + \eta g(r) & \text{reward} \end{cases}$$

where $\eta$ is the learning rate and

$$g(r) = \begin{cases} +1 & \text{if } r = 1 \\ -1 & \text{if } r = 0 \end{cases}$$

4 Others Weightless Models

There are many variations and extensions of the basic learning algorithm for q-WNN. All of them can be generally described through our definition 1. In fact, it must be stressed
that the others weightless models are only variations on the learning algorithm and use of the network but the hardware is the same. In this section five such variations are briefly described.

In order to make PLN less susceptible to noise and more able to generalise, Aleksander [5] suggested an extension of the PLN, the G-RAM, which once trained, spreads stored information to those neighbouring locations which still are in the unknown state u. So a G-RAM is 2-WNN with a probabilistic activation function (the same as the PLN one) with different training and using algorithms.

Another extension of PLN was suggested by Gorse and Taylor [13]. They developed a model of a noise neuron which incorporates and formalises many known properties of living neurons. They called their node model a p-RAM. The p-RAM in its simplest form is a lookup table in which each address stores a value \( \varepsilon \in [0, 1] \). This \( \varepsilon \) value is the probability of firing (i.e., generating a 1) at the overall output of the node. The main difference between p-RAM and MPLN is that p-RAMs allow continuous values to be stored in the memory of the nodes; MPLNs allow only discrete values. Note that the value \( \varepsilon \) is not to be understood as a weight, as one might be inclined to believe, but represents a probability of firing. A hardware implementation of a 2-node network of 2-input p-RAMs has been successfully constructed (Clarkson et al. [8]). Obviously, an implemented p-RAM is nothing but a MPLN.

The cut-point node ([17], [18]) is an extension of MPLN, based on ideas from the Theory of Probabilistic Automata, where the nodes output 1 or 0, depending on both the values stored in the memory position addressed by the input, and also on a probability. The probabilities outputted by the nodes are accumulated until the output layer of the network is reached, then this probability is compared with a threshold associated with every network. A pattern is accepted or recognised if, and only if, the probability computed during the running process is greater than the network threshold. This procedure simulates the behaviour of a probabilistic automata in an extended \( q \)-WNN. Networks constructed with cut-point nodes extends the power of computation of a MPLN once a MPLN is computationally equivalent to a (non-deterministic) finite state automata. A study of the computability capacity of WNNs can be found in [20] and [19].

The Goal-Seeking Neuron (GSN) [12] is a weightless node that uses local low level goals to govern its behaviour, and it is capable of one-shot learning. In such learning, only one presentation of the training set is required for the network to learn, resulting in a decrease of the total learning time required by the network. The major difference between the GSN and the PLN is that the former allows propagation of all three stored values through its output, whereas the latter only allows values of 0 and 1 to be propagated. So a GSN is also a 2-WNN with a probabilistic activation function with
different training and using algorithms.

The hysterectic neuron [1] was inspired both by the MPLN model and by the hysteresis mechanism. This model is constrained to a fixed pair of ascending and descending transition functions which define a shape similar to hysteresis loops. This model corresponds to a $q$-WNN endowed with an energy function for learning (energy is a fundamental concept in dynamical systems behaviour).

5 Formal Definition of a Neural Networks

From the discussion above we reach the conclusion that a WNN is essentially a vertex labelled digraph where the labels are the set of (finite) boolean functions (look-up table). Conventional learning algorithms may be seen as functions acting on the node labels of the underlying digraph whilst optimal network architecture searching (Hertz at.al. [15, Section 6.6, page 156]) as functions acting both in the node labels and on the vertex set. More formally:

Definition 6 1. A digraph, $G = (V, E)$, is a pair of finite sets where $E \subseteq V \times V$. A member of $V$ is called a vertex or node and a member of $E$ an edge or arc.

2. A labelled digraph, $G = (V, E, l)$, is a digraph $G = (V, E)$ together with a function $l : V \rightarrow L$, from the vertex set to an arbitrary set $L$, called the set of labels.

3. The fan-in of a vertex $v$ is the cardinality of $\operatorname{pred}(v) = \{u \mid (u, v) \in E\}$ whilst the fan-out is the cardinality of $\operatorname{succ}(v) = \{u \mid (v, u) \in E\}$.

We assume that $\mathbb{N}$ is the set of natural numbers.

Definition 7 1. A network architecture or neural net is a triple $\mathcal{N} = (G, I, O)$, where $G = (V, E, l)$, is a labelled digraph, $I, O \subseteq V$ and $L$, the range of $l$, is of the form $L = \mathbb{N} \cup \mathcal{F}$ such that:

- $l(I \cup O) \subseteq \mathbb{N}$
- $\mathcal{F}$ is a set of activation functions, $\mathcal{F} = \{f : D' \rightarrow D \mid D$ is a fixed set, $r > 0\}$.
- We denote by $\mathcal{F}[g]$ the function in $\mathcal{F}$ assigned to the vertex $g \in V$.

2. $G$ is called the underlying digraph, $I$ the input nodes, $O$ the output nodes and $C = V \setminus (I \cup O)$ the computation nodes.
Remarks 1

1. We may assume \textit{wlog} that the labels of the input nodes are \{1, 2, \ldots, n\}, where \(n\) is the cardinality of \(I\), and the labels of the output nodes are \{1, 2, \ldots, m\}, where \(m\) is the cardinality of \(O\).

2. If \(G\) is acyclic, \(I \cap O = \emptyset\) and \(\text{pred}(I) = \emptyset = \text{succ}(O)\), then \(\mathcal{N}\) is a feedforward network architecture.

3. \(\mathcal{N}\) is single layered if besides \(\text{pred}(O) = V \setminus (I \cup O) = \text{succ}(I)\).

4. A Hopfield network is a network with \(I = O = \emptyset\) and \(G\) is symmetric digraph (just a graph) which is complete.

In what follows the neural nets are assumed to be feedforward. For the general case the reader is referred to (de Oliveira and Ludermir [10]).

Definition 8

Given a net \(\mathcal{N}\) with \(n\) input nodes and \(m\) output nodes. A function \(f : D^n \rightarrow D^m\) is computed by \(\mathcal{N}\) by assigning a value \(x \in D^n\) to the input nodes and taking the result \(f(x)\) as being the value \(y \in D^m\) assigned to the outputs nodes. The intermediate steps in the calculations of \(y\) are of the form: if a computation node \(g\) has fan-in \(r\) and the nodes in \(\text{pred}(g)\) has output \(z \in D^r\), then \(g\) outputs \(\mathcal{F}[g](z)\).

Remarks 2

1. The definition of a neural net above is sufficient general to cover both weighted and weightless models by just choosing the appropriate set of activation functions.

2. For the weightless:

\[
\mathcal{F} = \{ f : \{0,1\}^k \rightarrow \{0,1\} \mid f \text{ is a look-up table or boolean function} \}
\]

3. For the weighted models there is one for each kind, for example:

(a) binary valued:

\[
\mathcal{F} = \{ f : \{0,1\}^n \rightarrow \{0,1\} \mid \exists w \in \mathbb{R}^n, \theta \in \mathbb{R} \sum_{i=1}^{n} w_i x_i > \theta \leftrightarrow f(x) = 1 \}
\]

(b) continuous valued:

\[
\mathcal{F} = \{ f : \mathbb{R}^n \rightarrow \mathbb{R} \mid \exists w \in \mathbb{R}^n, \theta \in \mathbb{R} \ f(x) = g(\sum_{i=1}^{n} w_i x_i) \}
\]

where \(e.g.\) \(g\) can be a sigmoidal function.
6 Summary

This paper has proposed a taxonomy for weightless neural models and showed that through this taxonomy it is possible to generally describe the basic processing mechanisms of weightless neurons. A short review, together with some advantages, of RAM, PLN and MPLN models has been presented. The main contributions of the paper are to identify the main components in the mainstreams WNN models, which is the RAM neuron employed, and a mathematical taxonomy for them. The variations in the model are in learning procedures and the way of using the network. And is further studied in (de Oliveira and Ludermir [10]).

References


