Cover Letter

Weightless Neuron Models

Edson Costa de Barros Carvalho Filho
Universidade Federal de Pernambuco
Departamento de Informática
Recife — PE — Brazil
CEP 50.739
e-mail:edson@di.ufpe.br

Topic: Neural Networks

Paper accepted to the XVIII Latinamerican Informatics Conference to be held in Las Palmas de Gran Canaria, SPAIN, from August 31 to September 4, 1992.

Keywords: Weightless Neurons, Boolean Networks, RAM-Based, GSN

Corresponding Author:

Dr. Edson Costa de Barros Carvalho Filho
Universidade Federal de Pernambuco
Departamento de Informática
Av. Prof. Luís Freire
Recife — PE — Brazil
CEP 50.739

E-mail: edson@di.ufpe.br

Weightless Neuron Models †

Edson Costa de Barros Carvalho Filho
Universidade Federal de Pernambuco
Departamento de Informática
Recife — PE — Brazil
CEP 50.739
e-mail:edson@di.ufpe.br

Abstract

The study of connectionist processing and artificial neural networks has generated a wide diversity of models, both in terms of the nature of the computational nodes employed and the structural architecture for implementation. While many models are based on nodes which compute the familiar sum-of-products function, and alternative class of neural networks use adaptive Boolean logic elements for the processing node. In principle, such networks offer a number of advantages relating particularly to their potential for high-speed processing, simplicity of implementation in VLSI, tractability of analytical studies and real world practical tasks. This class of nodes is known as weightless or RAM-Based neurons because they present a structure similar to a look-up table or a Random-Access-Memory. This paper review the Random-Access-Memory(RAM) Model[1], the Probabilistic Logic Neuron Model(PLN)[5], the Multi-Valued PLN[7] and the Goal-Seeking Model(GSN)[4].

Keywords: Weightless Neurons, Boolean Networks, RAM-Based, GSN

1 RAM Neurons

Motivated by difficulties in implementing McCulloch-Pitts neurons[6] and the interesting properties of Boolean nodes, Aleksander[1] designed the the special-purpose SLAM(Stored Logic Adaptive Microcircuit) device, later based on RAM(Random Access Memory) elements, as an electronic device capable of providing hardware implementation of universal sets of Boolean functions.

The RAM neuron accepts and produces binary values. A RAM neuron is a device, as illustrated in Figure 1, capable of computing any Boolean function with a given number of inputs. A RAM neuron can be described as consisting of a *set of input terminals* which repre-

† This work has been supported by the Brazilian Research Council(CNPq) and the Research Foundation of the State of Pernambuco(FACEPE).

sents the input of the neuron; a *set of weighted powers* which encode inputs in the form of addresses(the weighted powers never change and are used only to label input terminals), a *set of cells* which store the contents or learned information; an *output terminal* which returns an addressed content given by the input terminals; a *teach terminal* which provides the desired output; and a *operation mode terminal* which indicates if the neuron is in the learning or recalling phase.

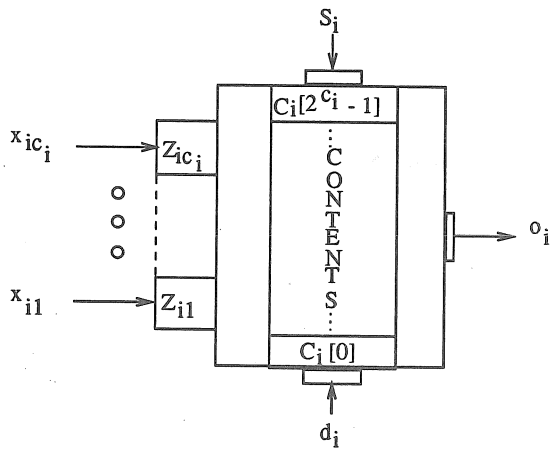

Figure 1: The RAM Model

A RAM neuron, n_i, with c_i input terminals has 2^{c_i} addressed contents and requires 2^{c_i} bits to be implemented. As all $2^{2^{c_i}}$ functions are computable, RAM neurons do not generalise. A RAM neuron has two states or operational modes: *the learning state* in which the neuron assigns the value on the teach terminal to the addressed content, and *the recall state* in which the neuron puts the addressed content on the output terminal. The output function is described by Equation 1.

$$o_i = \begin{cases} 0 & \text{iff } C_i[a_{i_m}] = 0 \\ 1 & \text{iff } C_i[a_{i_m}] = 1 \end{cases} \tag{1}$$

where $C_i[a_{i_m}]$ is the content of the address a_{i_m};

If the weighted powers are labelled as 2^k where $0 \leq k \leq c_i - 1$, then the address, a_{i_m}, accessed by an input, $x_i = x_{i1}, x_{i2}, ..., x_{ic_i}$, is defined by Equation 2

$$a_{i_m} = \sum_{j=1}^{j=c_i} x_{ij} z_{ij} \tag{2}$$

where x_{ij} is the input value of the j^{Th}-terminal and z_{ij} is the weight of the j^{Th}-terminal.

There are three major differences between the RAM and McCulloch-Pitts-like neurons:

1. McCulloch-Pitts like-neurons can have inputs, outputs and weights assuming any real number values, while the RAM neuron is essentially discrete;

2. The adaptive process in the McCulloch-Pitts neuron is related to the strength of the connections, while in the RAM model it is related to the addressable contents;

3. The McCulloch-Pitts model generalises in the sense that only linearly separable functions are computable, while the RAM neuron does not generalise, and all functions are computable.

2 Probabilistic Logic Neurons

After attempting a rapprochement between Boltzmann machines and logic node networks[2], in which the probabilistic nature of nodes is claimed to be important in solving pattern completion tasks, Kan and Aleksander proposed the Probabilistic Logic Neuron(PLN)[5].

The PLN model is a probabilistic version of the RAM neuron. The main difference between PLNs and RAMs is the inclusion of a third logic value such that when addressed produces a probabilistic binary output. This third logic value is equivalent to a "don't care", unknown or undefined value and is generally labelled 'u'.

The probabilistic output function that generates the neuron's output can be defined to produce zeros or ones with any probability, but it is normally considered to be the probabilistic function with equal likelihood of generating zeros and ones as defined by Equation 3.

$$o_i = \begin{cases} 0 & \text{iff } C_i[a_{i_m}] = 0 \\ 1 & \text{iff } C_i[a_{i_m}] = 1 \\ Ran(0,1) & \text{iff } C_i[a_{i_m}] = u \end{cases} \tag{3}$$

where $C_i[a_{i_m}]$ is the content of the address a_{i_m} and $Ran(0,1)$ is a random function that generates zeros and ones with the same probability.

In terms of information theory, the inclusion of the undefined value duplicates the memory requirements from 2^{c_i} bits for a neuron with c_i inputs, to 2×2^{c_i}.

The undefined value in a cell is used to identify untrained inputs, so if an untrained input appears and addresses an undefined cell the neuron simply produces a random output that is claimed as a form of internal generalisation. At the neuron level, the internal generalisation in the recall phase is totally independent of the learning experience since the random process does not consider past experiences(It is random). Even without training, the same untrained input can cause a PLN neuron to produce opposite output values, and this reflects the proba-

bilistic behaviour of PLN neurons. Some particular problems associated with the probabilistic behaviour of PLN neurons are discussed in [3].

3 Multiple-valued Probabilistic Logic Neurons

A cell in the PLN model is able to store zero, one or the undefined value. The probabilistic output function produces a zero or one with equal probability when the u value is addressed. In the Multiple-valued Probabilistic Logic Neuron(MPLM) proposed by Myers and Aleksander[7], a cell stores a range of values between 0 to 1, and the probabilistic output functions accepts the stored content as the probability of producing a 1 value. Stored content values vary from the complete randomness of the output at 0.5 to the absolute certainty of the output for the extreme values 0 and 1. There are two important parameters involved in the MPLM model, which are:

1. *The number of elements representing possible stored values* which is defined by the number of bits used in cells. After a punish/reward learning, the cells tend to acquire a zero or a one value. If a large number of elements is used to represent the probability of outputs, the learning process will lead showly to the internal saturation(evolution) towards one of the extremes. This is valuable when changes in the stored content are going in the right direction, however it can be undesirable when it is going in the wrong way. Also, there is a memory penalty to represent a cell as a word of B_l bits, that is $B_l 2^{c_i}$ as well as longer convergence times. If a smaller number of storage elements is used, as with the PLN neuron which only stores three values, the main problem is that a cell gets quickly saturated.

2. *The output function* which is applied on the addressed content. The output function may be a probabilistic, linear, step, or sigmoid function as described for the McCulloch-Pitts neuron. Some experimental results[7] has investigated the application of the sigmoid function, $\Phi(C_i[a_{i_m}])$ defined by Equation 4.

$$\Phi(C_i[a_{i_m}]) = \frac{1}{1 + \epsilon^{\alpha(-2C_i[a_{i_m}]+1)}} \tag{4}$$

where $C_i[a_{i_m}]$ is the content addressed by a_{i_m}, and α has been tested for values, 2.5, 4, 5, 10 and 25.

Clearly, the choice of the number of possible stored values and the output function depends on the problem involved in the task as well as other parameters such as: neuron connectivity,

training set size, number of neurons, desired output code, expected performance, order of presentation, etc. This problem is not simple and may involve a lot of simulation.

A MPLN net is said to converge when all locations in all nodes either have probabilities of outputting 1 equal to 0 or 1, or else are never addressed. After convergence, a MPLN behaves in exactly the same way as the PLN, and MPLNs could be replaced by PLNs preserving the behaviour and saving memory requirements[7]. This idea is not very attractive because a network should not be seen as static in the sense that after the learning phase nothing will change anymore.

4 Goal-Seeking Neurons

The GSN model accepts and generates values from the set $T = \{0, 1, u\}$, where the presence of undefined values on the input terminals identifies a set of possible addresses rather than a single address. The set of possible addresses is the *addressable set* and the set of contents thus addressed is designated the *addressable contents*. The area over which the goal is sought can therefore vary from a single address (in the case where all inputs are binary values) to the entire set of addresses (in the case where all inputs are undefined values). Computation is then accomplished by examining the addressable contents and generating the best value for the sought goal. The addressable set is determined by computing the fixed address given by the defined values on the input terminals and finding all combinations of addresses for the undefined values on the input terminals. A schematic representation of a GSN n_i is shown in Figure 2. In contrast to the RAM model, a GSN has a set of *state terminals* which indicate the state of the neuron and a set of *desired inputs* define the input signals which access the

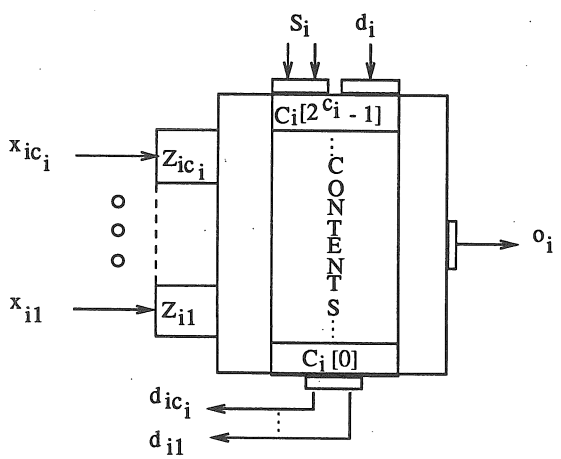

Figure 2: The Goal Seeking Neuron Model

The fixed address a_i^f of a neuron n_i is given by

$$a_i^f = \sum_{j=1}^{j=c_i} x_{ij} z_{ij} \mid x_{ij} \neq u; \tag{5}$$

where x_{ij} is the input value of the j^{Th}-terminal and z_{ij} is the weighted power value of the j^{Th}-terminal. Equation 5 computes the fixed address for inputs with defined values. Let $Z = \{z_{i1}^u, z_{i2}^u, ..., z_{iu_i}^u\}$ be the set of weighted powers with u_i-undefined inputs to n_i, thus the addressable set, A_i, is given by

$$A_i = \{a_{i_m} = a_{i.}^f + \sum_{j=1}^{j=u_i} b_j^m z_{ij}^u, \forall m = 0, 1, \ldots, 2^{u_i} - 1\}; \tag{6}$$

where b_j^m is the j^{th} bit of the binary representation, $b^m = b_1^m b_2^m \ldots b_{u_i}^m$, of m. Equation 6 gives the 2^{u_i} addresses which make up the addressable set. Each address a_{i_m} represents a combination of binary values for the undefined input terminals, and an address a_{im} is computed by clamping the binary representation of m into the input terminals of undefined values, thereby defining all inputs and so calculating the address.

To complete the definition of the GSN, consider a specification of its goals at each state of operation as follows:

4.1 The Validating State

When a neuron is in the validating state the goal is to establish an undefined value, ensuring that the neuron can learn any desired output, otherwise the neuron can produce only a specific output during learning. The output in the validating state of a neuron n_i is given by

$$o_i = \begin{cases} 0 & \text{iff } \forall a_{i_m} \in A_i, C_i[a_{i_m}] = 0; \\ 1 & \text{iff } \forall a_{i_m} \in A_i, C_i[a_{i_m}] = 1; \\ u & \text{iff } \exists a_{i_m} \in A_i \mid C_i[a_{i_m}] = u \text{ or } \exists a_{i_m}, a_{i_l} \in A_i \mid C_i[a_{i_m}] \neq C_i[a_{i_l}]; \end{cases} \tag{7}$$

where $C_i[a_{i_m}]$ $C_i[a_{i_l}]$ are contents of n_i addressed by a_{i_m} and a_{i_l} respectively;

Equation 7 shows that when the addressable contents contain an undefined value, or when there is a mixture of 0 and 1 values then the output is undefined. The output is zero when all addressable contents are zero. The output is one when all addressable contents are one. This mode of operation helps to propagate undefined paths through the network, seeking out unused cells or conflicting values, indicating the possibility of learning a desired output. An unused cell can be used to store the desired output(0 or 1), and if there are conflicting values, then a cell that store the desired output bit is selected to be used in the mapping without disrupting previously stored information.

4.2 The Learning State

When the neuron is in the learning state the goal is to learn the desired output with an address which already stores an appropriate value, otherwise an undefined content is used to store the desired output. The sought address of a neuron n_i with desired output d_i is given by

$$a_{i_m} = \begin{cases} Ran(A_{i/_{d_i}}) & \text{iff } \|A_{i/_{d_i}}\| > 0; \\ Ran(A_{i/_u}) & \text{iff } \|A_{i/_{d_i}}\| = 0; \end{cases} \tag{8}$$

where $A_{i/_s}$ is the addressable set of n_i that stores the value s, and is given by $A_{i/_s} = \{a_{i_k} \in A_i \mid C_i[a_{i_k}] = s\}$, $Ran(A_{i/_s})$ is a random element of $A_{i/_s}$ and $\|A_{i/_s}\|$ is the number of elements in $A_{i/_s}$;

The neuron tries to associate the desired output with an existing cell in the addressable set. If the neuron fails, it chooses an undefined cell and sets its value to the desired output. If there are a number of possible choices a random decision is made. It is necessary to choose one particular cell to represent the output value because the address of this cell is passed back down the input connections as the desired output value for the previous layers. The desired inputs d_{ij}, which indicate the input required to recall the sought address is given by

$$d_{ij} = x_j \mid \sum_{j=1}^{j=c_i} x_j z_{ij} = a_{i_m};$$

Thus each neuron n_j receives its desired output, learn and provide the desired outputs to the previous layers.

4.3 The recall state

When the neuron lies in the recall state the goal is to produce the binary content value with the highest occurrence in the addressable set, and the output in the recall state is given by

$$o_i = \begin{cases} 0 & \text{iff } \|A_{i/_0}\| > \|A_{i/_1}\|; \\ 1 & \text{iff } \|A_{i/_1}\| > \|A_{i/_0}\|; \\ u & \text{iff } \|A_{i/_0}\| = \|A_{i/_1}\|; \end{cases} \tag{9}$$

If the number of ones in the addressable contents is greater than the number of zeros, then the neuron outputs a 1 value. If the number of zeros is greater then the neuron will output a 0 value. If the numbers of ones and zeros is equal then the neuron outputs an undefined value. Thus, even if the addressable contents contain only a single 1 value, and the remainder are undefined, the neuron will output a 1 value. The reason for adopting this scheme is to minimise the propagation of undefined values.

Naturally, the best goal for the GSN is problem-dependent, and the goals previously specified can be redefined appropriately, hereby creating a family of GSNs. Three different GSNs have been identified, as follows: the GSN for the feed-forward architecture previously defined, the GSN^S for self-organisation architectures, and the GSN^+ for feed-forward architectures. This is not to say that the GSN is only applied in a feed-forward architecture, but it is clear that by changing the goals it is possible to get more suitable behaviour for different architectures.

5 Conclusions

The basic processing mechanisms of weightless neurons has been investigated where each neuron model was completely defined by the activation function and operation modes. Clearly, there is an evolution in the complexity of the models which comes to tackle real world problems.

References

[1] I Aleksander. Adaptive systems of logic networks and binary memories. *Proc. Spring Joint Computer Conference*, 1967.

[2] I Aleksander. Adaptive pattern recognition systems and boltzmann machines: A rapprochement. *Pattern Recognition Letters*, 6:113–120, 1987.

[3] E C D B C Filho, D L Bisset, and M C Fairhurst. A goal seeking neuron for Boolean neural networks. In *Proc. International Neural Networks Conference*, volume 2, pages 894–897, Paris, France, July 1990. IEEE.

[4] E C D B C Filho, M C Fairhurst, and D L Bisset. Adaptive pattern recognition using goal-seeking neurons. *Pattern Recognition Letters*, 12:131–138, March 1991.

[5] W K Kan and I Aleksander. A probabilistic logic neuron network for associative learning. In *Proc. IEEE First International Conference on Neural Networks*, volume II, pages 541–548, San Diego, California, June 1987. IEEE.

[6] W S McCulloch and W H Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull Math Biophys*, 5:115–133, 1943. formal neuron.

[7] C E Myers. Output functions for probabilistic logic nodes. In *First IEE International Conference on Artificial Neural Networks*, pages 310–314, London, UK, October 1989. IEE.