

## Quantum Logical Neural Networks

Wilson R. de Oliveira Jr.  
DEInfo-UFRPE, Recife, PE - Brazil  
wrdo@ufrpe.br

Teresa B. Ludermit  
CIn-UFPE, Recife, PE - Brazil  
tbl@cin.ufpe.br

Wilson R. Galindo  
CIn-UFPE, Recife, PE - Brazil  
wrg@cin.ufpe.br

Adenilton J. Silva  
DM-UFRPE, Recife, PE - Brazil  
adenilton.silva@gmail.com

Amanda Leonel  
DSC-UFPE, Recife, PE - Brazil  
aln@dsc.upe.br

Jefferson C. C. Pereira  
DEInfo-UFRPE, Recife, PE - Brazil  
jefferson@jeffersonpereira.com

### Abstract

*Quantum analogues of the (classical) Logical Neural Networks (LNN) models are proposed in [6] (q-LNN for short). We shall here further develop and investigate the q-LNN composed of the quantum analogue of the Probabilistic Logic Node (PLN) and the multiple-valued PLN (MPLN) variations, dubbed q-PLN and q-MPLN respectively. Besides a clearer mathematical description, we present a computationally efficient and simply described quantum learning algorithm in contrast to what has been proposed to the quantum weighted version.*

### 1. Introduction

Classical Artificial Neural Networks (ANN) can be broadly classified as *weightless* or *weighted* accordingly to their connection and neuron (unit) model. Weightless Neural Networks where the nodes compute boolean functions of its inputs are called Logical Neural Networks (LNN) [19]. Quantum Weighted Neural Models (q-WNN) have already been proposed and investigated in a multiplicity of models (see [9] for a review). The quantum weightless counterpart was firstly proposed in [6] in a general form. Here we further develop and investigate two particular models of Quantum Logical Neural Networks (q-LNN), a quantum analogue of the Weightless Neural Networks in which the units compute quantum logical functions of its inputs. An outstanding characteristic of LNN is their direct implementation in hardware and high speed of the learning process [19]. In its turn Quantum Computation has been shown capable of solving some problems dramatically faster than the

classical counterpart, as the Shor's factoring algorithm [28]. We have to remind that there is no proof that a quantum computer can polynomially solve NP-complete problems.

The idea is then to combine Quantum Computation with the Weightless Neural Network model producing Neural Networks that are in principle faster trained and easily implementable in hardware. We show here that the classical learning algorithm for LNN can be naturally adapted to q-LNN and each network can be implemented in a fixed quantum circuit. Once available, a quantum Random Access Memory [11] would be the natural hardware to implement a q-LNN. Contrary to what usually happens with the learning algorithms in the quantum weighted models, our learning algorithm does not directly interfere with system: the learned values are actually parameters of a "universal" (*unitary*) matrix representing the neuron node. In Quantum Physics (and Quantum Computing) certain interference requires measurements which *collapses* a general state to a basis state and so "loosing" information in the process.

Negative results concerning the impossibility of a universal programmable gate array [25] has motivated the study of statistically or approximately programming a quantum computer [15]. Besides, the difficulty of designing purely quantum algorithms has led to the proposal of using alternative techniques for problem solving on quantum computer such as Learning Machine [29] and Genetic Algorithms [12]. Neural Networks is one of the alternatives.

The main contribution of this work are displayed in Sections 5, 6 and 7. In Section 5 we investigate the quantum analogue of the Probabilistic Logic Node (PLN) [19]: the Quantum Probabilistic Logic Node (q-PLN). The zero, the one and the undefined (or unknown) value  $u$  which makes the node randomly fires a value in  $\{0,1\}$  is generalised to

the application of (respectively) the identity, the NOT and the Hadamard quantum gates (or matrices) to the computational base state  $|0\rangle$ .

In Section 6 we investigate an extension of q-PLN model, the Quantum Multi-Valued Probabilistic Logic Neuron (q-MPLN), that has a more general class of matrices, denoted by  $A_p$ , which applied to  $|0\rangle$  gives (after measurement) 1 (more precisely  $|1\rangle$ ) with probability  $p$ . These "small" matrices are combined in sort of universal matrix which reproduces the behavior of a node. The selection of the "small" matrices being parameterised allowing training consistent with both paradigms of quantum computation and neural networks.

In Section 7 we propose a learning algorithm for the q-PLN adapted directly from the classical version and give examples of its behaviour. Work is being done on other algorithms based on a (discrete) variation of complex back-propagation and discrete differential geometry [4].

## 2 Artificial Neural Network (Classical)

Artificial Neural Networks are parallel distributed systems composed of simple processing units, providing the ability to learn from examples and to generalise.

The first artificial neuron (the MCP node) was developed by W. McCulloch and W. Pitts [21]. This was an abstraction of what was known in 1930 about the biological neuron. The MCP node has binary inputs  $x_i$ , binary outputs  $y$  and real valued weights  $w_i$  associated with its connections. The node is activated, output say 1, when  $\sum x_i w_i > \theta$ , a given *threshold*. The majority of the works in the literature deals with variations of the MCP node such as allowing real valued input and outputs with more general *activation functions*. In this paper we broadly call networks composed of the MCP-like nodes Weighted Neural Networks (NN). MCP nodes are only capable of computing *linearly separable functions*.

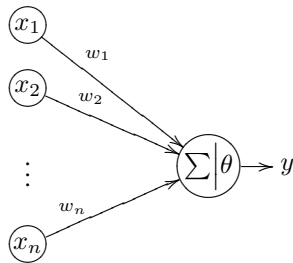


Figure 1. MCP Node

The kind of networks we are most concerned with in this work are the Logical Neural Network (LNN). The LNN's have some advantages when compared with NN. A weight-

less node has also binary inputs and outputs, can compute any boolean function of its input and are easily implementable in hardware. Prof. Igor Aleksander, the pioneer in this field, proposed a digital node known as RAM<sup>1</sup> node back in the late 60's [1]. An  $n$  input RAM node has  $2^n$  memory locations, addressed by the  $n$ -bit string  $a = a_1 a_2 \dots a_n$ . A binary signal  $x = x_1 x_2 \dots x_n$  on the input lines will access only one of these locations resulting in  $y = C[x]$  [19]. In Figure 2 below  $s$  and  $d$  are respectively the learning strategy and the desired output to be learned.

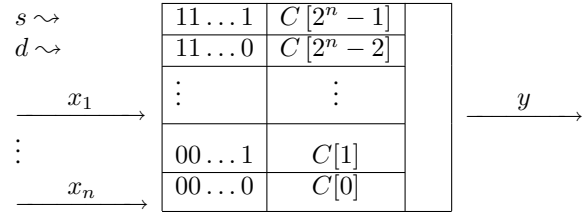


Figure 2. Ram Node depicted in [19]

The Probabilistic Logic Node (PLN) consists of a RAM node, where now a 2-bit number is stored at the addressed memory location (0, 1 or  $u$ ), respectively represented as say 00, 11 and 01 (or 10), with a probabilistic output generator. The output of the PLN Node is given by [19]:

$$y = \begin{cases} 0, & \text{if } C[x] = 0 \\ 1, & \text{if } C[x] = 1 \\ \text{random}(0, 1), & \text{if } C[x] = u \end{cases}$$

The Multi-Valued Probabilistic Logic Node (MPLN) differs from PLN by allowing a wider but still discrete range of probabilities to be stored at each memory content. A  $m$ -bit valued MPLN node can store a value  $k$  in the range  $\{0, \dots, 2^m - 1\}$  which is interpreted as the firing probability  $p = \frac{k}{2^m - 1}$  of the node.

## 3 Quantum computation

Neural computing and quantum computing can be both considered as unconventional models of computing. Quantum computing [26] was originally proposed by Richard Feynman [10] in the 1980s and had its formalisation with David Deutsch which proposed a universal computing model based on the principles of quantum mechanics: the quantum Turing machine [7]. Quantum computer also became a potential parallel device for improving the computational efficiency of neural networks [18]. The quantum information unit is the quantum bit or "qubit". The general state of a qubit is a superposition (linear combinations

<sup>1</sup>Random Access Memory.

from Linear Algebra [16]) of the two computational-basis states[23]:

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$$

- $\alpha, \beta$  are complex coefficients (called probability amplitudes) constrained by the normalization condition:  $|\alpha|^2 + |\beta|^2 = 1$ ; and
- $|0\rangle, |1\rangle$  are a pair of orthonormal basis vectors representing each classical bit, or "cbit", as column vector:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Tensor products are applied in order to represent more than one cbits as follows:  $|i\rangle \otimes |j\rangle = |i\rangle |j\rangle = |ij\rangle$ , where  $i, j \in \{0, 1\}$

Operations on qubits are carried out by unitary operators (i.e. matrices  $U$  such that  $UU^\dagger = U^\dagger U = I_n$ , where  $I_n$  is the identity  $n \times n$  matrix and  $U^\dagger$  is *conjugate transpose* also called the *Hemitian adjoint* of the matrix  $U$ ). So quantum algorithms on  $n$  bits are represented by unitary operators  $U$  over the  $2^n$ -dimensional complex Hilbert space [23]:

$$|\psi\rangle \rightarrow U |\psi\rangle$$

In quantum computation, all operators on qubits are reversible, except for the process called *measurement*, that loses the information about the superposition of states [23]. To measure a general state  $|\psi\rangle$  collapses (projects) it into either the  $|0\rangle$  state or the  $|1\rangle$  state, with probabilities  $|\alpha|^2$  or  $|\beta|^2$  respectively.

Others widely used operators and their corresponding matrix operators are [26]:

- **I**, identity operator: does nothing.

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{array}{l} \mathbf{I}|0\rangle = |0\rangle \\ \mathbf{I}|1\rangle = |1\rangle \end{array}$$

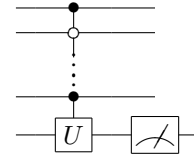
- **X**, flip operator: behaves as the classical NOT on the computational basis.

$$\mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \begin{array}{l} \mathbf{X}|0\rangle = |1\rangle \\ \mathbf{X}|1\rangle = |0\rangle \end{array}$$

- **H**, Hadamard transformation: generates superposition of states.

$$\mathbf{H} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{array}{l} \mathbf{H}|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle) \\ \mathbf{H}|1\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle) \end{array}$$

Quantum operators also may be represented in quantum circuits by corresponding quantum gates. Figure 3 shows an  $n$ -qubit controlled gate  $U$  whose action on the target qubit (bottommost) is active or not by  $n - 1$  (topmost) control qubits [26]. The output is checked by measurement gates.



where

$$\text{---} \circ \text{---} = \text{---} \boxed{X} \bullet \text{---} \boxed{X} \text{---}$$

Figure 3. A quantum circuit

## 4 Quantum Neural Network

The concept of quantum neural computation apparently was first introduced by Kak in 1995, creating a new paradigm using neural networks and quantum computation which opens several new directions in neural network research [17]. It is expected that Quantum Neural Networks are more efficient than Classical Neural Networks, parallel to what is expected from Quantum Computation in relation to Classical Computation.

In that same year, Menneer and Narayanan proposed a Quantum *Inspired* Neural Network applied the multiple universes view from quantum theory to one-layer ANNs [22]. In 1998 Ventura and Martinez introduced a quantum associative memory with a capacity exponential in the number of neurons [30].

The non-linear activation functions used in models of NN delayed further development of their quantum analogue. But in 2001 Altaisky proposed a quantum system where the rules of perceptron learning were proposed [2].

Recently, several Quantum Weighted Neural Networks models have been proposed but there remains the challenge of direct implementation in quantum circuits, natural adaptation of the learning algorithms and physical feasibility of quantum learning. These are characteristics not altogether found in any of the proposed models. For example in the Qubit Neural Network [18] training is realised by adjusting the phase of the qubits in a node, expressing a qubit  $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$  by a phase  $\varphi$  as  $\alpha = |\alpha| \cdot e^{i\varphi}$  and  $\beta = |\beta| \cdot e^{i\varphi}$ . In its turns, the Quantum M-P Neural Network [31] uses a quantum analogue of the MCP node where learning is realized by changing the matrix of weights requiring a measurement.

A brief summary of the main quantum NN found in the literature is listed below [14]:

**Implementation by Optical Devices:** Proposed by [2], is a simple substitution of the classical bits for quantum bits, and utilises the perceptron as base model.

**Implementation by Quantum Circuits:** Proposed by Gupta and Zia [13], is based on model of Deutsch [8], and basically builds a net composed of quantum gates and the input/output are quantum bits. The differential of this model is the use of D-gate that allow the use of bias.

**Implementation by Quantum Dot Molecules:** Created by Behrman [3], the operation of net is to take a Quantum Dot Molecule, i.e. small areas or lines in a semiconductor composed of a number of electrons occupying well defined quantum states. This model can be represented by a quantum circuit model and uses the D-gate.

**Based on Double-slit experiment:** The Narayanan approach [24] is based on the Double-slit architecture. The slits are the input neurons, the connections between two initial layers are waves created by training pattern in superposition. The output units are the detector that contemplate the arrival pattern.

**Implementation by C-Not Gates:** Shafee [27] proposes a Quantum Artificial Neural Network where the neurons are qubits with deterministic states and quantum operators replacing the activation functions. In short, this model presents integrated-and-fire neurons as qubits connected with the near neighbors through C-Not gates.

## 5. Quantum Probabilistic Logic Node: q-PLN

The values stored in a PLN Node 0, 1 and  $u$  are, respectively, represented by the qubits  $|0\rangle$ ,  $|1\rangle$  and  $H|0\rangle = |u\rangle$ . The probabilistic output generator of the PLN are represented as measurement of the corresponding qubit. We obtain a node q-PLN with the following output:

$$y = \begin{cases} 0, & \text{if } C[x] = |0\rangle \\ 1, & \text{if } C[x] = |1\rangle \\ \text{random}(0,1), & \text{if } C[x] = |u\rangle \end{cases}$$

There is an obvious relationship between outputs of a PLN and that of a q-PLN that associates  $i$  to  $|i\rangle$ , where  $i = 0, 1$  or  $u$ . The random properties of the PLN are guaranteed to be implemented by measurement of  $H|0\rangle = |u\rangle$  from the quantum mechanics principles (see e.g. Section 2.2 and Chapter 8 of [26] for a lengthier discussion).

Figure 5 shows a sketch of a q-PLN where qubits are "stored" in "memory locations" and qubits are in the input lines of a node. In order to implement this model we must first tackle three problems: (1) when a memory position contains a quantum state that is not in the computational base, information will be lost after measurement; (2) how to address a memory position with quantum states in the input line of node; and (3) how to train this network.

Without loss of generality let us suppose a node with two inputs. Figure 5 shows a node that solves problems (1) and (2) above, where the  $|a\rangle$ ,  $|b\rangle$  are the input register and  $|0\rangle$  is in the output register. The control port will be the address

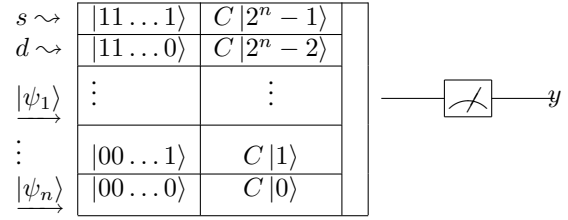


Figure 4. q-PLN Node

and the action of  $I$ ,  $H$  or  $X$  in the  $|0\rangle$  act as the output values of the PLN.

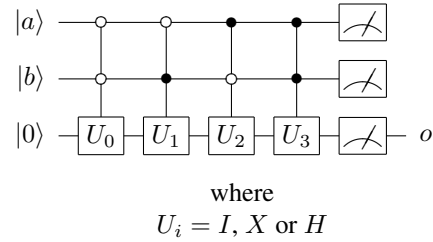


Figure 5. q-PLN Node in quantum circuit

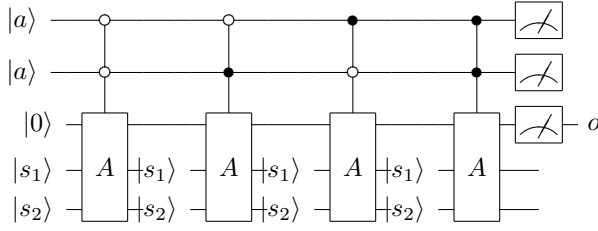
There still remains the third problem since training is a change of matrix. Universal deterministic Programmable Quantum Gate Arrays are not possible in general [25] but the q-PLN requires only three possible types of matrix. We can then easily define a matrix  $A$

$$A = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & X & 0 & 0 \\ 0 & 0 & H & 0 \\ 0 & 0 & 0 & U \end{pmatrix} \quad \text{where} \quad \begin{aligned} A|000\rangle &= |00\rangle I|0\rangle \\ A|010\rangle &= |01\rangle X|0\rangle \\ A|100\rangle &= |10\rangle H|0\rangle \end{aligned}$$

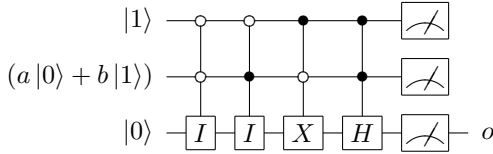
that operates on two additional qubit (which we call *selectors*) and produces the output of any  $U_i$  matrix, only by adjusting the values of these selectors. By substituting in the q-PLN in Figure 5 each matrix  $U_i$  by the matrix  $A$  with the two added input registers we obtain the q-PLN in Figure 6. The selectors determine which the  $U_i$  is *selected* for computation. We can easily see that learning is achieved by adapting the value of the selectors to the training set (see Section 7 below).

Let us see the operation of q-PLN node when given a value in the form  $a|0\rangle + b|1\rangle$  in its input lines. Let  $\mathcal{N}$  be the operator corresponding to the node. Rewriting the state we have:

$$|1\rangle (a|0\rangle + b|1\rangle) |0\rangle = a|100\rangle + b|110\rangle$$



**Figure 6. q-PLN Node with selectors**



Therefore for the node above:

$$\begin{aligned}
 \mathcal{N}(a|100\rangle + b|110\rangle) &= \mathcal{N}(a|100\rangle) + \mathcal{N}(b|110\rangle) \\
 &= a|10\rangle X|0\rangle + b|11\rangle H|0\rangle \\
 &= a|101\rangle + b|11\rangle \left( \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \right) \\
 &= a|101\rangle + b/\sqrt{2}|110\rangle + b/\sqrt{2}|111\rangle
 \end{aligned}$$

And node output is 0 with probability  $b^2/2$  and 1 with probability  $a^2 + b^2/2$

## 6. Quantum Multi-valued PLN: q-MPLN

In the classical case the difference between PLN and MPLN is that each MPLN node stores the probability of the node having the output 1 [5] (out of a discrete set of values in general bigger than just  $\{0, 1, u\}$ ). We do the same for the q-MPLN. In order to accomplish that we simply define the matrices  $U_p$ :

$$U_p = \begin{pmatrix} \sqrt{1-p} & -\sqrt{p} \\ \sqrt{p} & \sqrt{1-p} \end{pmatrix}$$

With this notation it is routine to check that  $U_0|0\rangle = |0\rangle$ ,  $U_1|0\rangle = |1\rangle$  and  $U_{\frac{1}{2}}|0\rangle = |u\rangle$  are the corresponding values in a q-PLN node. A general q-MPLN node has  $p$  varying in the (complex) interval  $[0, 1]$ . For example, if the probability to get 1 on a given position of a MPLN node is 0.25, then

$$U_{0.25} = \begin{pmatrix} \sqrt{0.75} & -\sqrt{0.25} \\ \sqrt{0.25} & \sqrt{0.75} \end{pmatrix}$$

hence the measurement of  $U_{0.25}|0\rangle$  return  $|1\rangle$  with probability 25%, as expected. We can easily see that the measurement of  $A_p|0\rangle$  is  $|1\rangle$  with probability  $p$  or  $|0\rangle$  with probability  $p-1$ ; and all  $U_p$  matrix is unitary, since if  $p$  in the

interval  $[0, 1]$  then there is an angle  $\theta$  such that  $\sin\theta = \sqrt{p}$  e  $\cos\theta = \sqrt{1-p}$ , hence

$$U_p = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$$

and all matrices on this form are unitary [16]. The *universal* matrix  $A$  for q-MPLN has the form

$$A = \begin{pmatrix} U_{p_1} & 0 & 0 & 0 \\ 0 & U_{p_2} & 0 & 0 \\ 0 & 0 & U_{p_3} & 0 \\ 0 & 0 & 0 & U_{p_4} \end{pmatrix}$$

Moreover, the  $A$  matrix can have larger dimensions to permit a finer calibration of the probability.

## 7. q-PLN Training

The original PLN algorithm network is recalled below:

1. All memory contents are set to  $u$
2. One of the  $N$  training patterns is presented to the net
3. The net is allowed to run until the output either:
  - (a) matches the desired output or
  - (b) mismatches the desired output  $\eta$  times
4. If 3.a is true all the addressed memory content are made to assume their current output values. Return to step 2. (Reward)
5. If 3.b is true all the addressed memory content are made to assume the  $u$  value. Return to step 2. (Penalty)

A small modification of the original training algorithm of PLN we can train the q-PLN. For this simply change items 1, 4 and 5: all selectors are set to produce  $|u\rangle$ ; if 3.a is true all the addressed  $A$  gate have their selectors changed to produce their current output values; if 3.b is true all the addressed  $A$  gate are made to produce  $|u\rangle$  value. Return to step 2. Observe that the Step 3 (substep b) above can be made in superposition thus dramatically speeding up the learning process.

An example of 4-bit parity check follows. We use a pyramidal network, with two layers that has two nodes(1,2) in the input layer and one node (3) in output layer.

1. Firstly select the training set  $T = \{(|0000\rangle, 1), (|1001\rangle, 1), (|0100\rangle, 0), (|1110\rangle, 0), (|0011\rangle, 1)\}$
2. We define  $\bar{0} = |00\rangle$ ,  $\bar{1} = |01\rangle$  and  $\bar{u} = |10\rangle$ . Set all selectors to  $\bar{u}$ ;
3. Run the network with the input  $|0000\rangle$  until the network has the desired output. Supposing that the output of node 1 is 0, node 2 is 0 and node 3 is 1, then make the Reward for each node
4. Reward. For each node make  $|s_1^a s_2^a\rangle = \bar{0}$ , where

$a = 2 \cdot m_1 + m_2$ , see Figure 6

5. Repeat the steps 3 and 4 for the other elements of set  $T$  we obtain the following outputs:

pattern	node 1	node 2	node 3
$ 1001\rangle$	1	1	1
$ 0100\rangle$	1	0	0
$ 1110\rangle$	0	1	0
$ 0011\rangle$	0	0	1

## 8. Conclusion and Future Works

This paper presented the first results of learning in a model of weightless quantum neural networks. There is much work to be done. The idea that the matrices  $A_p$  can be seen as a complex number thus opening up the possibility of using a complex version of backpropagation learning algorithm. Another line of research being pursued is the relationship between quantum neural networks and the quantum automata. A differential calculus based on matrices and non-Commutative Differential Geometry [20] leading to more general learning algorithm based on (general) gradient descent optimisation is being investigated.

## 9. Acknowledgements

This work is supported by research grants from MCT/CNPq, PRONEX/FACEPE, PIBIC/UFPE and PIBIC/Poli.

## References

- [1] I. Aleksander. Self-adaptive universal logic circuits. *Electronics Letters*, 2(8):321–322, 1966.
- [2] M. V. Altaisky. Quantum neural network. Technical report, Joint Institute for Nuclear Research, Russia, 2001.
- [3] E. C. Behrman, L. R. Nash, J. E. Steck, V. G. Chandrashekar, and S. R. Skinner. Simulations of quantum neural networks. *Information Sciences*, 128(3-4):257–269, 2000.
- [4] A. I. Bobenko, P. Schröder, and J. M. Sullivan. *Discrete Differential Geometry (Oberwolfach Seminars)*. Birkhäuser Basel, March 2008.
- [5] A. P. Braga, A. P. L. F. Carvalho, and T. B. Ludermir. *Redes Neurais Artificiais: Teoria e Aplicaes*. LTC, 2000.
- [6] W. R. de Oliveira, W. Galindo, A. Leonel, J. Pereira, and A. J. Silva. Redes neurais quânticas sem peso. In *2º Workshop-Escola em Computação e Informação Quântica, WECIQ 2007*, Campina Grande, Pb, Outubro 2007.
- [7] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.
- [8] D. Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London*, A 425:73–90, 1989.
- [9] J. Faber and G. Giraldi. Quantum models of artificial neural networks. Electronically available: <http://arquivosweb.lncc.br/pdfs/QNN-Review.pdf>.
- [10] R. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467, 1982.
- [11] V. Giovannetti, S. Lloyd, and L. Maccone. Quantum random access memory. *Physical Review Letters*, 100(16):160501, 2008.
- [12] G. Giraldi, R. Portugal, and R. Thess. Genetic algorithms and quantum computation. arXiv:cs/0403003v1 [cs.NE], 2004.
- [13] S. Gupta and R. K. P. Zia. Quantum neural networks. *Journal of Computer and System Sciences*, 63:355–383, 2001.
- [14] R. F. Herbster, W. L. Andrade, H. M. Gomes, and P. D. L. Machado. O estado da arte em redes neurais artificiais quânticas. *Rev. Eletr. de Inic. Cient.*, IV:1–8, 2004.
- [15] M. Hillery, M. Ziman, and V. Bužek. Approximate programmable quantum processors. *Physical Review A*, 73(2):022345, 2006.
- [16] K. Hoffman and R. Kunze. *Linear Algebra*. Prentic-Hall, 1971.
- [17] S. C. Kak. On quantum neural computing. *Information Sciences*, 83(3&4):143–160, 1995.
- [18] N. Kouda, N. Matsui, H. Nishimura, and F. Peper. Qubit neural network and its learning efficiency. *Neural Comput & Applic (2005) 14:114121*, 2005.
- [19] T. B. Ludermir, A. Carvalho, A. P. Braga, and M. C. P. Souto. Weightless neural models: A review of current and past works. *Neural Computing Surveys* 2, 41-61, 1999.
- [20] J. Madore. *An Introduction to Noncommutative Differential Geometry & Its Physical Applications*. CUP, 2000.
- [21] W. S. McCulloch and W. Pitts. A logical calculus of the ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115-133, 1930.
- [22] T. Menneer and A. Narayanan. Quantum-inspired neural networks. Technical Report R329, Department of Computer Science, University of Exeter, Exeter, UK, 1995.
- [23] N. D. Mermin. From cbits to qbits: Teaching computer scientists quantum mechanics. *American Journal of Physics*, 71:23, 2003.
- [24] A. Narayanan and T. Menneer. Quantum artificial neural networks architectures and components. *Information Sciences*, 128(3-4):231–255, 2000.
- [25] M. A. Nielsen and I. L. Chuang. Programmable quantum gate arrays. *Phys. Rev. Lett.*, 79(2):321–324, Jul 1997.
- [26] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [27] F. Shafee. Neural networks with c-NOT gated nodes. Technical report, Department of Physics, Princeton University, Princeton, USA, 2004.
- [28] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J.SCI.STATIST.COMPUT.*, 26:1484, 1997.
- [29] N. Toronto and D. Ventura. Learning quantum operators from quantum state pairs. In *IEEE World Congress on Computational Intelligence*, pages 2607–2612, July 2006.
- [30] D. Ventura and T. Martinez. Quantum associative memory. Available at arXiv.org:quant-ph/9807053, 1998.
- [31] R. Zhou and Q. Ding. Quantum m-p neural network. *Int J Theor Phys*, 2007.