Neurocomputing 75 (2012) 52–60

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Classical and superposed learning for quantum weightless neural networks

Adenilton J. da Silva [a,*], Wilson R. de Oliveira [b], Teresa B. Ludermir [a]

[a] Centro de Informática Universidade Federal de Pernambuco, Brazil
[b] Departamento de Estatística e Informática Universidade Federal Rural de Pernambuco, Brazil

ARTICLE INFO

Available online 31 July 2011

Keywords:
Quantum neural networks
Weightless neural networks
Quantum-supervised learning algorithm
Superposition-based learning algorithm

ABSTRACT

A supervised learning algorithm for quantum neural networks (QNN) based on a novel quantum neuron node implemented as a very simple quantum circuit is proposed and investigated. In contrast to the QNN published in the literature, the proposed model can perform both quantum learning and simulate the classical models. This is partly due to the neural model used elsewhere which has weights and non-linear activations functions. Here a quantum *weightless* neural network model is proposed as a quantisation of the classical weightless neural networks (WNN). The theoretical and practical results on WNN can be inherited by these quantum weightless neural networks (qWNN). In the quantum learning algorithm proposed here patterns of the training set are presented concurrently in superposition. This superposition-based learning algorithm (SLA) has computational cost polynomial on the number of patterns in the training set.

1. Introduction

Quantum computation was proposed by Richard Feynman in 1982 [1] motivated by the observation that a quantum system cannot be simulated by a classical computer without an exponential computational cost. In quantum theory of computation, a single quantum computer can follow many distinct computational paths in parallel and produce a final output depending on the interference of all of them. This parallelism enables the proposal of algorithms not matched by classical computation regarding computational costs. Amongst those are the Deutsch-Josza and Simon [2] with an exponential speed up. Shor's Algorithm which solves the factoring problem in polynomial time, a problem believed to be classically intractable. Grover's search algorithm [3] which searches an unordered database quadratically faster then any classical one.

There are several works applying quantum computing in artificial intelligence: quantum neural networks [4–11], decision tree [12], pattern recognition [13] and associative memory [14]. This paper investigates the use of quantum computing techniques to design learning algorithms for neural networks. We propose a quantum weightless neural network and a quantum supervised learning algorithm. The study of quantum weightless neural networks (qWNN) was started by de Oliveira et al. in [4,5] where it is investigated quantisations of probabilistic logic node (PLN) and multi-valued probabilistic logic node (MPLN) [15]. Here we propose and investigate a novel neuron model which is a quantum analogue of the RAM node which we call q-RAM node. The q-RAM node is very simple, but despite its simplicity it can simulate any of its classical siblings, PLN, MPLN, goal seeking neuron (GSN) and pRAM, and their quantisations.

The proposed learning algorithm for training weightless neural networks, the superposition-based learning algorithm (SLA), is based on Grover's search algorithm [3]. The SLA is a quantum supervised learning algorithm for neural networks where all patterns of the training set are presented at the same time to the network using a state in superposition. The computational cost of SLA is polynomial in the number of patterns in the training set. The SLA is able to train any model of weightless neural networks that can be quantised i.e. the free parameters, inputs and outputs of the network can be represented as qubits in different registers and the action of the network can be represented by a quantum operator.

This paper is organised as follows: Sections 2–4 present the basic definitions for quantum computation, classical weightless neural networks and quantum neural networks. Section 5 describes a novel quantum weightless neural node, one of the contributions of this work. Section 6 describes the superposition-based learning algorithm, another contribution of this work. Finally, Section 7 summarises our conclusions and presents future works.

2. Quantum computation

The cross-disciplinary nature of quantum computing makes it difficult to present their main definitions and results unbiased. The presentation which follows is biased towards Mathematics and mostly Computer Science.

The fundamental unit of information in classical computation is the bit which can assume one of the two possible abstract

* Corresponding author.
E-mail address: adenilton.silva@gmail.com (A.J. da Silva).

0925-2312/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2011.03.055

values in $\mathbb{B}=\{0,1\}$. More complex data types are encoded as sequences of bits. To represent a bit a classical computer must contain a corresponding physical system which can exist in two unambiguously distinguishable states, associated with its two possible abstract values. For example, one such system could be a switch in an open or a shut state; or a magnet whose magnetisation could be in two different orthogonal directions.

Similarly, the fundamental unit of information in quantum computing is the quantum bit or *qubit*. One qubit is a bi-dimensional vector in a (complex) Hilbert space.[1] A qubit represents a state of (bi-dimensional) quantum mechanical system. In actual physical quantum systems, where "Nobody knows *how* it can be like that" [16], Hilbert spaces can be very efficiently used to tell *what* happens.

In this section we briefly introduce the concepts of quantum computing necessary for the remaining sections. For a more complete introduction on Quantum Computation and Information, see [17] or [2] for a more Computing Science oriented approach.

2.1. Quantum bits

The passage from bits to qubits can be understood via mathematical quantisation. A very intuitive view of the quantisation procedure is put forward by Nik Weaver in the preface of his book *Mathematical Quantisation* [18] which briefly says: "The fundamental idea of mathematical quantisation is *sets are replaced with Hilbert spaces*". So the idea is to represent bits 0 and 1 as pairs of orthonormal (column) vectors, a *basis*[2] for $\mathbb{C}^2$. In spite of fact that 0 and 1 can be represented by any orthogonal base of $\mathbb{C}^2$, the mostly used one is the *canonical* (or *computational*) basis defined as the pair

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

where $|0\rangle$ and $|1\rangle$ are, in a notation commonly used by physicists (and quantum computing scientists), the computational basis states and read "ket zero" and "ket one". They are also called basic or ground states. The ket notation was invented in 1939 by Paul Dirac [19] and is related to inner product. This notation is also used for an arbitrary vector $|\psi\rangle$. The other part of the bracket defining the inner product (of say x and y, $\langle x,y\rangle$) is unsurprisingly called *bra*. The bra of a ket vector $|\psi\rangle$ is its conjugate transpose, and thus a row vector, denoted as $\langle\psi|$. Their matrix product $\langle\psi||\psi\rangle$ is a scalar, their inner product. From the inner product we obtain a norm $\||\psi\rangle\|^2=\langle\psi|\psi\rangle$.

2.2. Superposition and parallelism

While in classical computing a one bit state can be only 0 and 1, in quantum computation we can have a continuum of linear combinations of $|0\rangle$ and $|1\rangle$ by the quantisation procedure. For instance the general qubit state $|\psi\rangle=\alpha_0|0\rangle+\alpha_1|1\rangle$ is a state that can be seen as part $|0\rangle$ and part $|1\rangle$ as a *superposition* of the basic states. $|\psi\rangle$ is at the same time in both state $|0\rangle$ and $|1\rangle$.

One of the main properties of quantum computation is the *quantum parallelism*. If one applies a quantum operator $\mathbf{U}_f$ that implements a function $f(x)$ such that $\mathbf{U}_f|x,0\rangle=|x,f(x)\rangle$ in a state in superposition $|\psi\rangle=\sum_{i=0}^{n-1}\alpha_i|x_i,0\rangle$, the value of $f(x)$ will be computed for all qubits $|x_i\rangle$. The resultant state will be $\sum_{i=0}^{n-1}\alpha_i|x_i,f(x_i)\rangle$.

Because of the quantum parallelism, if one have a quantum neural network implemented as a quantum operator, then it will be possible to use states in superposition to evaluate the outputs of all patterns in the training set, all at once in parallel. A drawback is that the individual results of this computation are not direct accessible, due to the properties of the measurement in quantum mechanics.

2.3. Measurement

In Quantum Physics if a system is in a state which is a superposition $|\psi\rangle=\alpha_0|0\rangle+\alpha_1|1\rangle$, upon measurement the system collapses to one of its basis state $|i\rangle, i\in\{0,1\}$ probabilistically:

$$p(|i\rangle)=\frac{|\alpha_i|^2}{\||\psi\rangle\|^2}=\frac{|\alpha_i|^2}{\Sigma_j|\alpha_j|^2}$$

which is the probability that the system will be found in the ground state $|i\rangle$ after a measurement.

After the first measurement of a state $|\psi\rangle$ if one performs another measurements will get the same result. The collapse of the state after measurement says that one cannot see all the results generated by the quantum parallelism. The challenge in quantum computation is how to take advantage of the quantum parallelism before performing a measurement.

2.4. States are rays

Note that if a state is a scalar multiple of another, say $|\phi\rangle=\beta|\psi\rangle$ the chance that the system in state $|\phi\rangle$ will be found, after measurement, in state $|i\rangle, i\in\{0,1\}$ is the same as if the system were in state $|\psi\rangle$

$$p(|i\rangle)=\frac{|\beta\alpha_i|^2}{\||\phi\rangle\|^2}=\frac{\beta^2|\alpha_i|^2}{\beta^2\Sigma_j|\alpha_j|^2}=\frac{|\alpha_i|^2}{\Sigma_j|\alpha_j|^2}$$

and so, the kets $\beta|\psi\rangle$ and $|\psi\rangle$ describe *the same physical system*. The set $\{\alpha|\psi\rangle \mid \alpha\in\mathbb{C}\}$ is the *ray* generated by $|\psi\rangle$ and represents the same state as $|\psi\rangle$. A natural representative of the ray is a normalised vector in the set. As a result, normalising the ket $|\psi\rangle$ i.e. multiplying it by $1/\||\psi\rangle\|$, gives a unit length ket in which the probability of being observed in $|i\rangle$ is

$$p(|i\rangle)=|\alpha_i|^2$$

So the collection of qubits are all the bi-dimensional complex vectors $|\psi\rangle=\alpha_0|0\rangle+\alpha_1|1\rangle$, such that $|\alpha_0|^2+|\alpha_1|^2=1$.

For example, the ket $|\psi\rangle=(1/\sqrt{2})(|0\rangle+|1\rangle)$ represents a system which is $1/2$ equiprobably to be found in any of the two basis states upon measurement.

2.5. Multi-qubits systems

A system with more than one qubit can be represented by the tensor product of their matrix representation. Recall that the tensor product of two bi-dimensional vectors

$$|\psi\rangle=\begin{bmatrix}\psi_0\\\psi_1\end{bmatrix},\quad |\phi\rangle=\begin{bmatrix}\phi_0\\\phi_1\end{bmatrix}$$

is the four-dimensional vector:

$$|\psi\rangle\otimes|\phi\rangle=\begin{bmatrix}\psi_0\begin{bmatrix}\phi_0\\\phi_1\end{bmatrix}\\\psi_1\begin{bmatrix}\phi_0\\\phi_1\end{bmatrix}\end{bmatrix}=\begin{bmatrix}\psi_0\phi_0\\\psi_0\phi_1\\\psi_1\phi_0\\\psi_1\phi_1\end{bmatrix}$$

[1] A complex Hilbert space is a vector space over the complex numbers $\mathbb{C}$ with a *compatible* inner product.

[2] A basis for a vector space $\mathbb{V}$ is any subset of linear independent vectors $B\subseteq\mathbb{V}$ such that any $v\in\mathbb{V}$ is a linear combination of the vectors in B i.e. $v=\Sigma_{x\in B}\alpha_x v$ where $\{\alpha_x\}_{x\in B}$ is a B-indexed set of scalars.

This obviously generalises to any pair of n- and m-dimensional vectors producing a nm-dimensional vector or more generally a (n_0,m_0)-dimensional matrix by a (n_1,m_1)-dimensional one producing a third (n_0m_0,m_0n_0)-dimensional matrix and so on.

Generally we omit the tensor product operator in the notation $|i\rangle \otimes |j\rangle = |i\rangle|j\rangle = |ij\rangle$, where $i,j \in \{0,1\}$. The n-qubits live in the space $\mathbb{C}^{2^n}$.

The construction of $\mathbf{U}_f$ in Section 2.2 for one qubit can be generalised to multi-qubits. Given a Boolean function $f : \mathbb{B}^n \leftrightarrow \mathbb{B}^m$ defines the unitary operator $\mathbf{U}_f : \mathbb{C}^{2^n} \leftrightarrow \mathbb{C}^{2^m}$ where $\mathbf{U}_f|\mathbf{x}\rangle_n|\mathbf{y}\rangle_m = |\mathbf{x}\rangle_n|\mathbf{y} \oplus f(\mathbf{x})\rangle_m$ and $|\mathbf{x}\rangle_n$ are an n qubits base state and $\oplus$ is the bitwise XOR (addition mod 2).

2.6. Linear quantum operators

In Quantum Mechanics observables quantities are *Hermitian operators* whose *eingevectors* form a complete orthonormal basis for the space. The result of a measurement is its *eingenvalue*. The dynamics or time evolution of the system is governed by a *unitary operator* related to the *Halmitonian* of the system. So the next natural step in our quantisation procedure is the representation of the Boolean operators as unitaries.

Four our purposes, a unitary operator $\mathbf{U}$ is a squared matrix over the complex numbers, $\mathbf{U} \in \mathbb{C}^{n\times n}$, such that

$$\mathbf{U}\mathbf{U}^\dagger = \mathbf{U}^\dagger\mathbf{U} = \mathbf{I}_n$$

where $\mathbf{I}_n$ is the identity $n \times n$ matrix and $\mathbf{U}^\dagger$ is the conjugate transpose (Hermitian conjugate) of $\mathbf{U}$. Being invertible, a unitary operator is reversible. They preserve inner product and so they are isometries. In the Bloch sphere representation of qubits, they correspond to rotations or inversions.

Some examples of operators over one qubit in quantum computation are: **I**, identity operator: does nothing; **X**, flip operator: behaves as the classical NOT on the computational basis and **H**, Hadamard transformation: generates superposition of states. Together with the **X** matrix the **Z** and **Y** matrices (see Eq. (2)) form the well-known Pauli matrices which plays an important role in quantum mechanics in general and in quantum computing in particular. Their matrix representations in relation to the computational basis are displayed in Eqs. (2) and (3).

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2}$$

$$\mathbf{Y} = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \mathbf{Z} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Apart from these gates, the Hadamard gate H, the phase gate S, and the $\pi/8$ -gate T are given by the matrices

$$\mathbf{H} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \tag{3}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix}$$

These single qubit gates are important, as they can be used together with the CNOT-gate to give universal sets of discrete quantum gates.

The Hadamard gate can be used to produce equally weighted superpositions as the following simple example shows

$$\mathbf{H}|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{4}$$

$$\mathbf{H}|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{5}$$

The CNOT-gate is an example of a two-qubit controlled operation. It also goes under the name (quantum) XOR. Its matrix representation in the computational basis is

$$\mathbf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6}$$

The **CNOT** gate performs a **NOT** (i.e. an **X**) operation on the target qubit t conditioned on the control bit c being 1.

2.7. Non-linear quantum operators

A new idea in quantum computing is that the quantum evolution might be slightly non-linear [20]. The non-linearity is useful for overcoming "the difficulties connected with the ordinary linear quantum components" [10]. Following this idea non-linear quantum algorithms have been proposed [21]. The non-linear quantum gates were applied to neural networks in [10,11].

In [20] a non-linear quantum algorithm is proposed that receives a state in superposition $|\psi\rangle = (1/\sqrt{N})\sum_{i=1}^{N}|\psi^i,0\rangle$ and returns the state $|\psi\rangle = (1/\sqrt{N})\sum_{i=1}^{N}|\psi^i,0\rangle$ or $|\psi\rangle = (1/\sqrt{N})\sum_{i=1}^{N}|\psi^i,1\rangle$. The last qubit is changed to $|1\rangle$ if and only if in the superposition all ψ_i are equal to 0. Note that his operator could be used to solve the satisfiability problem.

2.8. Quantum circuits

Quantum operators can be represented as quantum circuits. Fig. 1 shows the quantum circuit for the **CNOT** gate and Fig. 2 shows a quantum circuit where an n-qubit controlled gate U whose action on the target qubit (bottommost) is active or not by $n-1$ (topmost) control qubits [17]. The output is checked by measurement gates.

2.9. Grover's algorithm

Grover's algorithm is a quantum algorithm for searching an unordered data quadratically faster than any classical method [22]. Given a data set with N items the most efficient classical algorithm will need execute an average of 0.5 N classical steps before finding the desired item. In the worst case, N classical steps are necessary. Grover's algorithm outperforms any classical algorithm and realises the task with $O(\sqrt{N})$ quantum steps. The iteration number T of the algorithm is very important and is calculated as in Eq. (7), where M is the number of answers in the search space:

$$T = \left\lfloor \frac{\pi}{4}\sqrt{\frac{N}{M}} \right\rfloor \tag{7}$$

The Algorithm 1 is based on the one shown in [2] and M is a squared 2^n-dimensional matrix whose each entry is $1/2^n$.

Algorithm 1 (*Grover's algorithm*).

1 Initialise the system $|\psi\rangle = |0\rangle_n$
2 Apply the Hadamard Transform $|\psi\rangle = H^{\otimes n}|\psi\rangle$
3 Apply the phase inversion operation: $U_f(\mathrm{I} \otimes \mathrm{H})$
4 Apply the inversion about the mean operation: $-\mathrm{I}+2\mathrm{M}$
5 Repeat steps 3 and 4, $T = O(\sqrt{2^n})$ times.
6 Measure the state $|\psi\rangle$

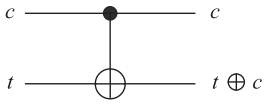


Fig. 1. Controlled NOT gate.

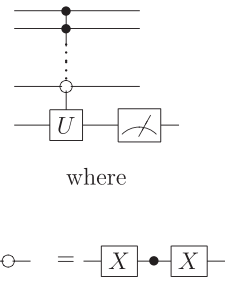


Fig. 2. A quantum circuit for the general controlled **U** gate where **U** is an arbitrary 1-qubit unitary operator. Note that appropriately choosing circles (○) and bullets (•) one could define a controlled operator which would apply **U** to the bottommost qubit if the corresponding binary combination (0′ s for circles and 1′ s for bullets) is present in the remaining qubits from top to bottom.

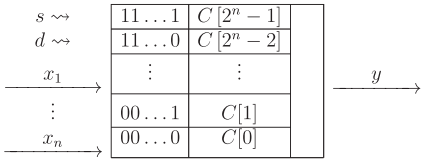


Fig. 3. RAM node.

The iterations 3 and 4 of Algorithm 1 can be represented as a quantum operator **G** called the Grover iteration [22].

3. Weightless neural networks

The RAM-based neural networks were proposed by Igor Aleksander [23] and do not have weights associated in their connections.

A RAM node with n inputs has 2^n memory locations, addressed by the n-bit string $a=(a_1a_2\ldots a_n)$. A binary signal $x=(x_1x_2\ldots x_n)$ on the input lines will access only one of these locations resulting in $y=C[x]$ [15]. In Fig. 3, s and d are respectively the learning strategy and the desired output to be learned.

Learning in weightless neural networks takes place simply by writing into the corresponding look-up table entries. This learning process is much simpler than the adjustment of weights. In spite of the simplicity of the RAM-based nodes, RAM-based networks have good generalisation capabilities [24] and computational power [25].

The PLN is based on the RAM node. The difference between the PLN and RAM nodes is that a 2-bit number (rather than a single bit) is now stored at the addressed memory location. The content of this location is turned into the probability of firing (i.e. generating 1) at the overall output of the node. In other words, a PLN consists of a RAM node, where now a 2-bit number 00, 11 and 01 (or 10) stored at the addressed memory location are to be interpreted respectively as 0, 1 or u. The output of the PLN Node is given by [15]

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

The multi-valued probabilistic logic node (MPLN) differs from PLN by allowing a wider but still finite set of finite precision probabilities $M=\{p_0,\ldots,p_{2^n-1}\}$ to be stored at each memory content. The output of the MPLN Node is given by [15]

$$y=\begin{cases}0 & \text{if} \quad C[x]=0\\ 1 \quad \text{withprobability } \dfrac{1}{p} & \text{if} \quad C[x]=p\end{cases}$$

The goal seeking neuron (GSN) differs from the PLN by allowing that the node receives and generate values 0, 1 and u. If the GSN node receives one value u in its input lines a set of memory positions will be addressed. The set of addressed positions is derived from the original address replacing the u values in the input vector to 0 and 1. For instance, if a three input GSN receives the input $I=0u1$, two addresses of the GSN node will be accessed $a_1=001$ and $a_2=011$ because of the u value in the second position of I. The output of the GSN node is determined by the number of values 0, 1 and u in the addressed memory positions.

4. Quantum weighted neural networks

The concept of quantum neural computation was first introduced by Kak in 1995, creating a new paradigm using neural networks and quantum computation which opened new directions in neural network research [26]. 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.

Since the Kak's work further neural networks models have been proposed [6,7,10,4,9,27,28], but there remains the challenge of direct implementation in quantum circuits, natural adaptation of the classical learning algorithms and quantum learning algorithms respecting the postulates of the quantum mechanics. These are characteristics not altogether found in any of the proposed quantum weighted neural networks models but are found in our new model.

In this section we describe some models of quantum neural networks and their learning algorithms. We verify that the learning algorithms for these models break the postulates of quantum computing by the use of non-linear or non-unitary quantum operators. Use of non-linear quantum operators is an open question, but non-linear quantum computation implies P=NP [20].

The proposed learning algorithms for quantum neural networks [6,10,28,9] can be classified as iterative [28,9] (in each iteration only one pattern is presented to the network) or superposition based [10,6] (all patterns are presented to the networks concurrently in superposition). Here we analyse one iterative algorithm [28] and one based on the superposition [10] and we show that these algorithms for weighted neural networks are non-unitary and non-linear.

The one qubit output $|y\rangle$ of a quantum perceptron [28] with inputs $|x_1\rangle,\ldots,|x_n\rangle$ is defined in Eq. (8) where $\hat{w}_j$ are 2×2 matrices representing the weights of neuron and $\hat{F}$ is a quantum operator. The quantum iterative learning rule for this model is presented with $F=I$ in Eq. (9), where t is the iteration number, $|d\rangle$ is the desired output and the output of the node in time t is described by Eq. (10):

$$|y\rangle=\hat{F}\sum_{j=1}^{n}\hat{w}_j|x_j\rangle \tag{8}$$

$$|y(t)\rangle=\sum_{j=1}^{n}\hat{w}_j(t)|x_j\rangle \tag{9}$$

$$\hat{w}_j(t+1)=\hat{w}_j(t)+\eta(|d\rangle-|y(t)\rangle)\langle x_j| \tag{10}$$

The learning rule of quantum perceptron drives the quantum perceptron into the desired output $|d\rangle$ [28], but the learning rule (10) does not preserve unitary operators.

Theorem 4.1. *The learning rule described in Eq.* (10) *does not preserve unitary operators.*

Proof. We construct a counterexample and execute one iteration of the iterative quantum learning rule. In Eq. (10) set $j=1$, the weight $\hat{w}_1(t)=I$, the desired output $|d\rangle=|1\rangle$, the network output is $|0\rangle$, the input of $\hat{w}_1(t)$ is $|x_j\rangle=|1\rangle$ and $\eta=0.5$. Then

$$\hat{w}_j(t+1)=I+0{,}5(|1\rangle-|0\rangle)\langle 1|=I+0{,}5(|1\rangle\langle 1|-|0\rangle\langle 1|)$$
$$=\begin{pmatrix}1&0\\0&1\end{pmatrix}+\begin{pmatrix}0&-0.5\\0&0.5\end{pmatrix}=\begin{pmatrix}1&-0.5\\0&1.5\end{pmatrix} \quad (11)$$

where we see that $w_1(t+1)$ is non-unitary. □

One may think that the trouble lies in the choice of the learning rate but let η be arbitrary. Then

$$\hat{w}_1(t+1)=\begin{pmatrix}1&-\eta\\0&1+\eta\end{pmatrix}\rightarrow\hat{w}_1(t+1)\hat{w}_1(t+1)^{\dagger}$$
$$=\begin{pmatrix}1+\eta^2&-\eta-\eta^2\\-\eta-\eta^2&(1+\eta)^2\end{pmatrix} \quad (12)$$

and since $\hat{w}_1(t+1)$ is unitary

$$\begin{pmatrix}1+\eta^2&-\eta-\eta^2\\-\eta-\eta^2&(1+\eta)^2\end{pmatrix}=\begin{pmatrix}1&0\\0&1\end{pmatrix}$$

and we must have $\eta=0$. In this case there is no non-zero learning rate that allows the use of rule (10) without violating the quantum computing postulates.

The learning algorithms that use states in superposition [10,6] also violates the quantum computation postulates. In [10,11] it is used as a non-linear quantum operator proposed in [20] and in [6] it is necessary for the use of a non-linear quantum oracle. One difficulty here is that non-linear quantum mechanics implies $P=NP$ [20].

5. Quantum weightless neural networks

In [6] we define a quantisation of the RAM node, the qRAM node. Quantum weightless neural networks do not use non-linear activation function, like sigmoid or tangent hyperbolic. This is important because non-linear activation functions will hardly have an exact quantum analogous [28].

Usually the RAM node stores one bit at each addressable location. We follow this approach by investigating first the qRAM node which store just one qubit. This qubit will not be directly stored in a quantum register as in the classical case. We will rather use a selector (parameter) and an operator which applied to the selector will produce the desired qubit. This form of quantisation of weightless neural networks was proposed in [4,5] using different matrices from the matrix **A** used here

$$\mathbf{A}=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix} \quad (13)$$

The matrix **A** defines a quantum operator over two qubits that simply flip the second qubit if the first is in the state $|1\rangle$ and it does nothing if the first state is in the state $|0\rangle$. The reader familiar with quantum computing terminology will notice that our matrix **A** is well known as the *controlled not* or c-NOT gate. We keep with the alternative notation since it gives rooms for further generalisations as in [4,5] where more complex matrices are used.

Definition 5.1. A qRAM node with n inputs is represented by the operator **N** described in Eq. (14). The inputs, selectors and outputs of **N** are organised in three quantum registers $|i\rangle$ with n qubits, $|s\rangle$ with 2^n qubits and $|o\rangle$ with 1 qubit. The quantum state $|i\rangle$ describes qRAM input, and quantum state $|s\rangle|o\rangle$ describes qRAM state.

$$\mathbf{N}=\sum_{i=0}^{2^n-1}|i\rangle_n\langle i|_n A_{s_i,o} \quad (14)$$

Fig. 4 describes the quantum circuit of a qRAM node with two inputs $|\psi\rangle$ and $|\varphi\rangle$. Four selectors $|s_i\rangle$ and four operators $\mathbf{A}_{s_i,o}$ each one equals to the **A** operator above, where the first qubit is the selector $|s_i\rangle$ and the second is the state in the output register $|o\rangle$. We can now see that learning is achieved by adapting the value of the selectors to the training set. The main advantage of qRAM over classical weightless neural networks is its capacity to receive input and selectors in superposition. When the selectors are in the computational basis the qRAM node acts exactly as a RAM node. But it behaves in a much more interesting way when is fed with a superposition of basis state.

Let us see the operation of qRAM node **N** when given a value in the form $|\psi\rangle=(1/\sqrt{2})|0\rangle+(1/\sqrt{2})|1\rangle$ in one of its input lines. Further assume that $|\varphi\rangle=0$, $|s_1s_2s_3s_4\rangle=|0111\rangle$ and $|o\rangle=0$. Eq. (15) shows the operation of the qRAM. In this example the two matrices A_0 and A_2 are simultaneously addressed and the output register is in state $|o\rangle=(1/\sqrt{2})(|0\rangle+|1\rangle)$ i.e. outputs of 00 and 10 are calculated simultaneously.

$$\mathbf{N}\left[\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)|0\rangle|0111\rangle|0\rangle\right]$$
$$=\frac{1}{\sqrt{2}}[\mathbf{N}(|0\rangle|0\rangle|0111\rangle|0\rangle)+\mathbf{N}(|1\rangle|0\rangle|0111\rangle|0\rangle)]$$
$$=\frac{1}{\sqrt{2}}[(|0\rangle|0\rangle|0111\rangle|0\rangle)+(|1\rangle|0\rangle|0111\rangle|1\rangle)] \quad (15)$$

One example of qRAM network is presented in Fig. 5, qRAM networks have pyramidal architecture and low connectivity. The configuration of a given qRAM network is represented by a string of qubits $(s_{11},s_{12},\ldots,s_{1n};\ldots;s_{m1},s_{m1},s_{m2}\ldots,s_{mn})$ representing network selectors, where m is the number of neurons and n is the number of inputs of neurons.

$$\mathbf{N}=|00\rangle\langle 00|\otimes A_{s_1,o}+|01\rangle\langle 01|\otimes A_{s_2,o}$$
$$+|10\rangle\langle 10|\otimes A_{s_3,o}+|11\rangle\langle 11|\otimes A_{s_4,o} \quad (16)$$

Exponential cost in simulation of quantum computers makes it prohibitive to simulate qRAM networks. Here we present a simple example to show qRAM network operation. We can use the qRAM network in Fig. 5 to solve the four bit parity problem. The

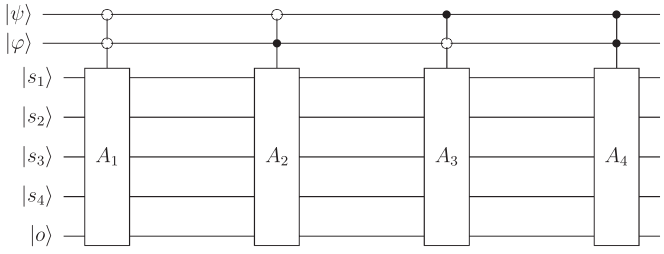


Fig. 4. qRAM node.

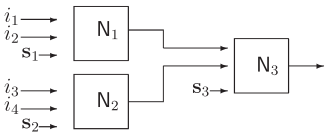


Fig. 5. Two-layer qRAM network.

configuration of the network is $\mathbf{s}_1 = |0110\rangle$, $\mathbf{s}_2 = |0110\rangle$ and $\mathbf{s}_3 = |0110\rangle$. One can present inputs simultaneously as in Eq. (17) to qRAM network

$$|i\rangle = \frac{1}{2}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle) \tag{17}$$

The neuron $\mathbf{N}_1$ receives as input the two first qubits of state $|i\rangle$, $|i_1 i_2\rangle = |00\rangle$. The action of neurons of network is described by the operator in Eq. (16). The action of the neuron $\mathbf{N}_1$ and $\mathbf{N}_2$ are described in Eqs. (18) and (19)

$$\begin{aligned}\mathbf{N}_1|i_1 i_2\rangle|\mathbf{s}_1\rangle|o_1\rangle &= (|00\rangle\langle 00| \otimes A_{s_1,o} + |01\rangle\langle 01| \otimes A_{s_2,o} \\ &\quad + |10\rangle\langle 10| \otimes A_{s_3,o} + |11\rangle\langle 11| \otimes A_{s_4,o})|00\rangle|0110\rangle|0\rangle \\ &= |00\rangle\langle 00| \otimes A_{s_1,o}(|00\rangle|0110\rangle|0\rangle) = |00\rangle|0110\rangle|0\rangle\end{aligned} \tag{18}$$

$$\begin{aligned}\mathbf{N}_2|i_3 i_4\rangle|\mathbf{s}_2\rangle|o_2\rangle &= (|00\rangle\langle 00| \otimes A_{s_1,o} + |01\rangle\langle 01| \otimes A_{s_2,o} \\ &\quad + |10\rangle\langle 10| \otimes A_{s_3,o} + |11\rangle\langle 11| \otimes A_{s_4,o})\frac{1}{2}(|00\rangle \\ &\quad + |01\rangle + |10\rangle + |11\rangle)|00\rangle|0110\rangle|0\rangle \\ &= \frac{1}{2}(|00\rangle\langle 00| \otimes A_{s_1,o}(|00\rangle|0110\rangle|0\rangle) \\ &\quad + |01\rangle\langle 01| \otimes A_{s_2,o}(|00\rangle|0110\rangle|0\rangle) \\ &\quad + |10\rangle\langle 10| \otimes A_{s_3,o}(|00\rangle|0110\rangle|0\rangle) \\ &\quad + |11\rangle\langle 11| \otimes A_{s_4,o}(|00\rangle|0110\rangle|0\rangle)) = |00\rangle|0110\rangle|0\rangle \\ &\quad + |01\rangle|0110\rangle|1\rangle + |10\rangle|0110\rangle|1\rangle + |11\rangle|0110\rangle|0\rangle\end{aligned} \tag{19}$$

The outputs of neurons $\mathbf{N}_1$ and $\mathbf{N}_2$ are $|o_1\rangle = |0\rangle$ and $|o_2\rangle = (1/\sqrt{2})(|0\rangle + |1\rangle)$. These outputs will be used as inputs of neuron $\mathbf{N}_3$. Eq. (20) shows the action of $\mathbf{N}_3$ and the network calculates the outputs of all the inputs in superposition

$$\begin{aligned}\mathbf{N}_3|o_1 o_2\rangle|\mathbf{s}_3\rangle|o_3\rangle &= \frac{1}{\sqrt{2}}(|00\rangle\langle 00|A_{s_1 o_3}|00\rangle|0110\rangle|0\rangle \\ &\quad + |01\rangle\langle 01|A_{s_2 o_3}|01\rangle|0110\rangle|0\rangle) \\ &= \frac{1}{\sqrt{2}}(|00\rangle|0110\rangle|0\rangle + |01\rangle|0110\rangle|1\rangle)\end{aligned} \tag{20}$$

The action of the qRAM network **Net** can be summarised as in Eq. (21). The network calculates the output of each input in superposition

$$\begin{aligned}&\mathbf{Net}\left(\frac{1}{2}(|0000\rangle + |0001\rangle + |0010\rangle + |0011\rangle)\right)|0\rangle \\ &= \frac{1}{2}(\mathbf{Net}|0000\rangle|0\rangle + \mathbf{Net}|0001\rangle|0\rangle \\ &\quad + \mathbf{Net}|0010\rangle|0\rangle + \mathbf{Net}|0011\rangle|0\rangle) \\ &= \frac{1}{2}(|0000\rangle|0\rangle + |0001\rangle|1\rangle + |0010\rangle|1\rangle + |0011\rangle|0\rangle)\end{aligned} \tag{21}$$

In Section 6 the capacity to process states in superposition will be explored to train a neural network.

5.1. Simulation of classical weightless models

Interestingly, if we perform a measurement at the output wire of the qRAM and allow for superposition of the selectors the qRAM node can simulate any of its siblings PLN, MPLN and pRAM and the training algorithm of the classical weightless neural networks can be adapted to the qRAM.

Let $|u\rangle = H|0\rangle$ be an undefined state as in Section 3, in Eq. (15) the qRAM node behaves as a GSN node. The node can receive and produce $|0\rangle$, $|1\rangle$ and $|u\rangle$ then the qRAM node can be viewed as a qGSN node, but the qRAM node can receive and produce others quantum states behaving as a sort of *continuum* valued qGSN node.

Algorithm 2 (*PLN net learning*).

```
1  All memory contents are set to u;
2  while some stopping criterion is not met do
3  |    One of the N training patterns p
   |    is presented to the net;
4  |  learn ← FALSE
5  |  for t = 1 to η do
6  |  |    The net is allowed to produce the output for the
   |  |    pattern p
7  |  |  if s is equal to the desired output for the
   |  |  pattern p then
8  |  |  |  learn ← TRUE
9  |  |  |  break
10 |  |  end
11 |  end
12 |  if learn then
13 |  |  all the addressed memory content are made 7
   |  |    to assume their
   |  |  current output values, making those with  u
   |  |   become definitely 0
   |  |  or 1, accordingly
14 |  else
15 |  |  all the addressed memory content are made to assume
   |  |  the u value
16 |  end
17 end
```

The classical algorithm of PLN network is defined in Algorithm 2. In order to simulated a PLN the values stored in a PLN node 0, 1 and u, will be represented respectively by the qubits $|0\rangle$, $|1\rangle$ and $|u\rangle = H|0\rangle$. The probabilistic output of the PLN is obtained with a measurement in the output of the qRAM node. The results of this measurement is described in Eq. (22)

$$y = \begin{cases} 0 & \text{if} \quad |o\rangle = |0\rangle \\ 1 & \text{if} \quad |o\rangle = |1\rangle \\ random(0,1) & \text{if} \quad |o\rangle = |u\rangle \end{cases} \tag{22}$$

With a small modification of Algorithm 2 one can train the qRAM node. It is necessary to measure the output of each node and define $|u\rangle = \mathbf{H}|0\rangle$, replace the line 1 of Algorithm 2 with *all selectors are set to* $|u\rangle$ and replace lines 13 and 15 respectively with *all the addressed A gates have their selectors changed to produce the current output of their nodes, making those with* $|u\rangle$ *become definitely* $|0\rangle$ *or* $|1\rangle$, *accordingly* and *all the addressed A gates have their selectors set to* $|u\rangle$. We can see the PLN node as a particular case of the qRAM where a measurement is realised in the output of each node i.e. we are working only with classical data.

The Fig. 6 shows the quantisation of lines 5–11 in Algorithm 2. Operator *Net* represents a qRAM network with input $|p\rangle$, selectors $|s\rangle$ and output $|o\rangle$. There are η *Net* operators to perform

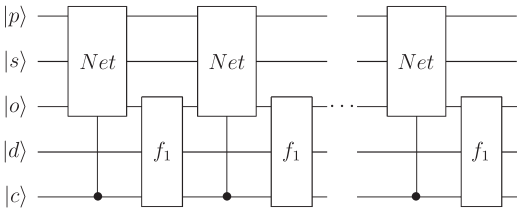


Fig. 6. PLN learning iteration.

the for loop in Algorithm 2. The operator f_1 change the control register $|c\rangle$ to $|0\rangle$ if the desired output $|d\rangle$ is equal to network output $|o\rangle$ and stop the execution of the networks. Operator f_1 also changes $|o\rangle$ to $|0\rangle$ if $|d\rangle$ is different from $|o\rangle$ preparing the output register to the next *Net* operator.

Algorithm 3 shows the quantisation of Algorithm 2. In this algorithm all neuron outputs are measured, data in registers $|p\rangle$, $|o\rangle$, $|d\rangle$ and $|c\rangle$ are in computational basis. Only the register $|s\rangle$ receives states in superposition.

Algorithm 3 (*Naive qRAM net learning*).

```
All selectors |s⟩ are set to |u⟩;
while some stopping criterion is not met do
    One of the N training patterns |p⟩ is
    presented to the net;
    Set |c⟩ = |1⟩ and |o⟩ = 0
    Let d(p) be the desired output of
    pattern p. Set |d⟩ = |d(p)⟩
    Apply the quantum circuit described in Fig.6
    if |c⟩ = |0⟩
        all the addressed memory content are made
        to assume their
        current output values, making those with |u⟩
        become definitely
        |0⟩ or |1⟩, accordingly
    else
        all the addressed memory content are made
        to assume the |u⟩
        value
    end
end
```

In [4,5] the size of matrices depends on the number of stored probabilities which make the quantisation of the pRAMs impossible. In contrast our novel model has constant size matrices independent of the number of stored probabilities. The probability is stored as amplitude of the input state and/or the selectors which can be tuned during the learning phase through phase rotations.

One can easily simulate the other weightless neural nodes with the qRAM node. Then any computation done by a weightless neural node can be realised by a qRAM. This implies, for instance, that the qRAM maintain the generalisations capabilities of the weightless models. In the next section we present a quantum learning algorithm to train the qRAM node.

Algorithm 3 is called naive because it does not explore quantum mechanics properties. In next section we present a quantum algorithm to train qRAM networks exploring its capacity to receive states in superposition.

6. Superposition-based learning algorithm

Let us imagine a classically improbable learning algorithm. For a fixed architecture the number of possible WNN is finite, say n. Supposing we have all the required computational resource available, we could present all the p patterns in the learning set at once in parallel to each individual network; perhaps making p copies of each one of the n networks. Then we could search amongst the pn combinations networks and input patterns that network which recognises the training set mostly accordingly to some fixed criterion. The quantisation of this absurdly computational expensive algorithm is what we propose next and show that in a quantum computer it is a polynomial time algorithm!

The superposition-based learning algorithm (SLA) is a supervised algorithm that takes advantage of states in superposition to approximate the idea in the previous paragraph. SLA can be used to train qRAM networks where the selectors and inputs are a quantum state $|\psi\rangle$ composed of four quantum registers. The first register **s** will store the selectors, the second register **p** will store the training pattern, the third register **o**, with one qubit, will store the output of the node and the fourth register **d**, used only in the learning phase of the algorithm, will store the desired output.

Algorithm 4 (*Superposition-based learning*).

1. Initialise all the qubits in register **s** with the quantum state $\mathbf{H}|0\rangle$.
2. Initialise the register **p**, **o**, **d** with the quantum state $|p\rangle = \sum_{i=1}^{p} |p_i,0,d_i\rangle$.
3. $|\psi\rangle = \mathbf{N}|\psi\rangle$, where **N** is a quantum operator representing the action of the neuron.
4. Use a quantum oracle to change the phase of the states where the registers **p**, **o**, **d** $= \sum_{i=1}^{p} |p_i,d_i,d_i\rangle$
5. Apply the operator inverse to the neuron in the state $|\psi\rangle$, $|\psi\rangle = \mathbf{N}^{-1}|\psi\rangle$, to disentangle the state in the register s
6. Apply the inversion about the mean operation (see Algorithm 1) in the register **s**
7. Repeat steps 3–6, $T = (\pi/4)\sqrt{n}$ times, where n is the number of selectors of the networks.
8. Measure the register **s** to obtain the desired parameters.

In step 1 of Algorithm 4 a superposition of all possible configurations for the network is created. This step is realised setting the quantum register **s** to $|0,\ldots,0\rangle$ and applying the Hadamard operator in all qubits of the register **s**, at this moment **s** will hold the quantum state $|s\rangle$ described in Eq. (23), where m is the number of selectors, $a = 2^m - 1$ and $s_j = j$

$$|s\rangle = \frac{1}{\sqrt{a+1}} \sum_{j=0}^{a} |s_j\rangle \tag{23}$$

Let p_i be a pattern of the training set and d_i be its desired output. In step 2 a superposed quantum state holding all p_i and d_i is set to the quantum registers **p** and **d** respectively. The output register **o** is set to the basis state $|0\rangle$.

One can execute step 2 as in the case of quantum associative memory proposed in [14,29]. At this moment the register **p** holds all training patterns p_i and the register **d** holds all desired outputs d_i. The state of the registers **p**, **o** and **d** is described in Eq. (24)

$$|\mathbf{p},\mathbf{o},\mathbf{d}\rangle = \frac{1}{\sqrt{p}} \sum_{i=0}^{p-1} |p_i,0,d_i\rangle \tag{24}$$

Let **N** be the operator associated to a quantum neural network. The action of **N** may be summarised as sending the state $|s,p_i,0,d_i\rangle$ into the state $|s,p_i,y_i,d_i\rangle$, where y_i is the calculated output of the node.

In step 3 of Algorithm 4 the operator **N** acts on the state $|\psi_0\rangle$ and, by linearity of the quantum operators, the desired output is produced in all terms of the superposition. Eq. (25) describes the state $|\psi_1\rangle$, where m is the cardinality of the training set and $a = 2^m - 1$

$$|\psi\rangle = \frac{1}{\sqrt{a+1}} \frac{1}{\sqrt{p}} \sum_{i=0}^{p-1} \sum_{j=0}^{a} |s_j,p_i,y_{ij},d_i\rangle \tag{25}$$

Let **O** be the quantum oracle that sends the state $\sum_{i=0}^{p-1} |p_i,d_i,d_i\rangle$ into the state $-\sum_{i=0}^{p-1} |p_i,d_i,d_i\rangle$ and that acts as the identity for the other states. In step 4 of the SLA the oracle **O** receives the state $|\psi_1\rangle$ and will mark only the states where the desired outputs are equals to

the calculated output $y_{ij}=d_i$. The action of the oracle will permit to use the Grover's search algorithm in the register **s**.

In the third step the neuron **N** entangled the quantum register **s** with the quantum registers, **p**, **o** and **d**, then the quantum search algorithm applied in the register *s* will change the registers **p**, **o** and **d**. To avoid this change in the registers **p**, **o** and **d** the operator $\mathbf{N}^{-1}$ is applied to disentangle the states in the registers in the fifth step of the algorithm. Then the inversion about the mean operation is applied in the register **s**. The steps 3–6 are repeated *T* times and a measurement will return the desired parameters.

6.1. Complexity of SLA

In this section we analyse the complexity of the SLA algorithm. To simplify the analysis we suppose that all possible (Boolean) patterns are in the training set, the number of patterns in training set is $N_p=2^n$ and each q-RAM node has two inputs. In this situation patterns can be represented with $n=\log N_p$ bits. A pyramidal qRAM network with $\log N_p$ input terminals will have $2^{log(\log N_p)}-1$ neurons. Then the number of qubits in the quantum register **s** will be $4(\log N_p-1)$. The search occurs in the quantum register *s* with all $2^{4(\log N_p-1)}$ possible selectors and this search will have cost $\pi/4\sqrt{2^{4(\log N_p-1)}}=(\pi/4)(2^{2(\log N_p-1)})=(\pi/4)(2^{-2}\cdot N_p^2)=(\pi/16)(N_p^2)$. Then the SLA algorithm has polynomial time in the number of patterns N_p in the training set. This result is similar to the cost of the learning algorithm proposed in [10].

6.2. Training a q-RAM node with the SLA

Let us look at the concrete example of the SLA training the q-RAM node to solve the XOR problem. We shall see that a q-RAM node with two inputs learn the function with only one iteration. The training is set to $S=\{(00,0),(01,1),(10,1),(11,1)\}$.

A q-RAM node with two inputs has four selectors, then in step 1 the state $|s\rangle$ is initialised as

$$|s\rangle=H^{\otimes 4}|0000\rangle=\frac{1}{4}\sum_{i=0}^{16}|i\rangle_4 \tag{26}$$

In step 2 the registers **p**, **o** and **d** are prepared as in Eq. (27)

$$|p,o,d\rangle=\frac{1}{2}(|00,0,0\rangle+|01,0,1\rangle+|10,0,1\rangle+|11,0,1\rangle) \tag{27}$$

At this moment the state $|\psi\rangle$ can be described as in Eq. (28)

$$|\psi\rangle=\frac{1}{8}\sum_{i=0}^{16}|i\rangle_4(|00,0,0\rangle+|01,0,1\rangle+|10,0,1\rangle+|11,0,1\rangle) \tag{28}$$

In step 3 the operator **N** is applied to $|\psi\rangle$ which calculates the output y_p^i of the node for each selector *i* and for each pattern *p* as in Eq. (29)

$$|\psi\rangle=\frac{1}{8}\sum_{i=0}^{16}|i\rangle_4(|00,y_{00}^i,0\rangle+|01,y_{01}^i,1\rangle+|10,y_{10}^i,1\rangle+|11,y_{11}^i,1\rangle) \tag{29}$$

In step 4 an oracle **O** which inverts the phase of state $|e_1\rangle=(|00,0,0\rangle+|01,1,1\rangle+|10,1,1\rangle+|11,0,0\rangle)$ is applied to $|\psi\rangle$ and we obtain the state in Eq. (30), where δ is the Kronecker's delta and $x(i)=1$ if and only if $|p_i,o_i,d_i\rangle=|e_1\rangle$. The oracle marks the state with the desired parameters

$$|\psi\rangle=\frac{1}{8}\sum_{i=0}^{16}(-1)^{\delta_{1x(i)}}|i\rangle_4(|00,y_{00}^i,0\rangle+|01,y_{01}^i,1\rangle+|10,y_{10}^i,1\rangle+|11,y_{11}^i,1\rangle) \tag{30}$$

In step 5 the operators $\mathbf{N}^{-1}$ and **G** are applied to $|\psi\rangle$ and the amplitude of the marked state goes to one and the amplitude of the others state go to zero. The training finishes in three iterations because $T=\lfloor(\pi/4)\sqrt{16}\rfloor=3$, then one can measure the **s** register and use the result as the parameters of the network.

7. Conclusion

A quantum weightless neural node based on the quantisation of the RAM node, the qRAM node, was proved. Its ability to simulate the classical weightless neural networks was demonstrated, a very important result since all the theoretical and practical results for WNN are inherited by our model. The main property of the qRAM node explored in this paper is the ability to receive all patterns from the training set in superposition.

The size of the matrices *A* in other models of quantum weightless neural networks is exponential on the number of stored probabilities [4,5]. Our model can store probabilities using matrices *A* with constant size. The probabilities are stored in the amplitudes of the selectors not in the matrices *A*. This reduction in space complexity may allow classical simulations of networks composed by qRAM nodes for small sized problems.

We also propose a quantum learning algorithm for neural networks, the superposition-based learning algorithm (SLA). The SLA is a supervised learning algorithm. It explores the capacity of the qRAM to receive qubits in superposition and apply a retrieval algorithm of a quantum probabilistic memory to choose the best configuration of the network. The SLA receives several neural network configurations in superposition and returns a trained neural network.

The SLA has polynomial computational cost in the number of patterns in the training set. This cost is mainly associated with the use of the quantum search algorithm. Because of the superposition capacity one can argue that it may be possible to design an algorithm with a smaller computational cost exploring superposition of neural networks.

A possible future work is to create an ensemble of neural networks in a single quantum neural network with a polynomial cost in the size of the training set. This task may be realised with an algorithm that marks a parameter (one qubit) of the networks in superposition. Instead of performing a measurement projecting to the basis state one can realise a projective measurement to collapse the network to the marked networks.

Another possible future work is to develop a probabilistic version of the SLA where we must run the neural network only twice, one forward and other backward. This can be obtained changing the Grover algorithm for a retrieval algorithm of a quantum probabilistic memory as in [13].

Acknowledgements

This work is supported by research grants from CNPq, CAPES and FACEPE (Brazilian research agencies).

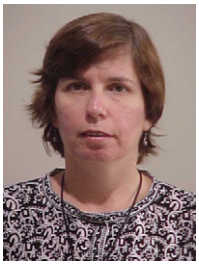
References

[1] R. Feynman, Simulating physics with computers, International Journal of Theoretical Physics 21 (1982) 467–488.
[2] N.S. Yanofsky, M.A. Mannucci, Quantum Computing for Computer Scientists, Cambridge University Press, New York, NY, USA, 2008.
[3] L.K. Grover, A fast quantum mechanical algorithm for database search, in: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, STOC '96, ACM, 1996, pp. 212–219.
[4] W. de Oliveira, A.J. Silva, T.B. Ludermir, A. Leonel, W. Galindo, J. Pereira, Quantum logical neural networks, in: Neural Networks, 2008. SBRN '08. 10th Brazilian Symposium on 2008, pp. 147–152.

[5] W. de Oliveira, Quantum RAM based neural networks, in: European Symposium on Artificial Neural Networks 2009. Proceedings 2009, pp. 331–336.
[6] A.J. Silva, W. de Oliveira, T.B. Ludermir, A weightless neural node based on a probabilistic quantum memory, in: Neural Networks, 2010. SBRN '10. 11th Brazilian Symposium on 2010, pp. 259–264.
[7] A.J. Silva, W. de Oliveira, T.B. Ludermir, Superposition based learning algorithm, in: Neural Networks, 2010. SBRN '10. 11th Brazilian Symposium on 2010, pp. 1–6.
[8] B. Ricks, D. Ventura, Training a quantum neural network, in: S. Thrun, L. Saul, B. Schölkopf (Eds.), Advances in Neural Information Processing Systems 16, MIT Press, Cambridge, MA, 2004, pp. 1–6.
[9] R. Zhou, Q. Ding, Quantum M-P neural network, International Journal of Theoretical Physics 46 (2007) 3209–3215.
[10] M. Panella, G. Martinelli, Neural networks with quantum architecture and quantum learning, International Journal of Circuit Theory and Applications 39 (1) (2011) 61–77, doi:10.1002/cta.619.
[11] M. Panella, G. Martinelli, Neurofuzzy networks with nonlinear quantum learning, IEEE Transactions on Fuzzy Systems 17 (3) (2009) 698–710.
[12] E. Farhi, S. Gutmann, Quantum computation and decision trees, Physical Review A 58 (2) (1998) 915–928.
[13] C.A. Trugenberger, Quantum pattern recognition, Quantum Information Processing 1 (2002) 471–493.
[14] D. Ventura, T. Martinez, Quantum associative memory, Information Sciences 124 (1–4) (2000) 273–296.
[15] T.B. Ludermir, A. Carvalho, A.P. Braga, M.C.P. Souto, Weightless neural models: a review of current and past works, Neural Computing Surveys 2 (1999) 41–61.
[16] R. Feynman, The Character of Physical Law, The MIT Press, 2001.
[17] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.
[18] N. Weaver, Mathematical Quantization, Studies in Advanced Mathematics, Chapman & Hall/CRC, Boca Raton, FL, 2001.
[19] P. Dirac, The Principles of Quantum Mechanics, Clarendon Press, 2004.
[20] D.S. Abrams, S. Lloyd, Nonlinear quantum mechanics implies polynomial-time solution for *NP*-complete and *#P* problems, Physical Review Letters 81 (18) (1998) 3992–3995.
[21] M. Czachor, Notes on Nonlinear Quantum Algorithms, ArXiv:quant-ph/9802051v2.
[22] L.K. Grover, Quantum mechanics helps in searching for a needle in a haystack, Physical Review Letters 79 (2) (1997) 325–328.
[23] I. Aleksander, Self-adaptive universal logic circuits, Electronics Letters 2 (8) (1966) 321–322.
[24] A.F. Souza, F. Pedroni, E. Oliveira, P.M. Ciarelli, W.F. Henrique, L. Veronese, C. Badue, Automated multi-label text categorization with VG-RAM weightless neural networks, Neurocomputing 72 (10–12) (2009) 2209–2217.
[25] M. de Souto, T. Ludermir, W. de Oliveira, Equivalence between RAM-based neural networks and probabilistic automata, IEEE Transactions on Neural Networks 16 (4) (2005) 996–999.
[26] S.C. Kak, On quantum neural computing, Information Sciences 83 (3) (1995) 143–160.
[27] Z. Righi, J. Nan, D. Qiulin, Model and training of QNN with weight, Neural Processing Letters 24 (3) (2006) 261–269.
[28] M.V. Altaisky, Quantum Neural Network, Type Technical Report, Institution Joint Institute for Nuclear Research, Russia, available online at the Quantum Physics repository arXiv:quant-ph/0107012v2, 2001.
[29] C. Trugenberger, Probabilistic quantum memories, Phys. Rev. Lett. 87 (6) (2001) 067901.

Adenilton da Silva received the Licentiate degree in mathematics and M.Sc degree in Computer Science from Universidade Federal Rural de Pernambuco, Brazil, in 2009, and Universidade Federal de Pernambuco, Brazil, in 2011, respectively. He is currently working towards the Ph.D degree in Computer Science at the Universidade Federal de Pernambuco, Brazil. His current research interests include quantum computation, weightless neural networks and hybrid neural systems.

Wilson R. de Oliveira received the B.Sc (1982), M.Sc (1985) and Ph.D (2004) in Computer Science at Universidade Federal de Pernambuco (Brazil) specialising in Computing Theory of Artificial Neural Networks and Automata Theory. Sabbatical leave at the School of Computer Science, University of Birmingham, UK (2008), working on Matroid Theory and Discrete Differential Geometry. Has been working recently on Quantum Weightless Neural Networks, Combinatorial Hopf Algebras, Matroid Representations and Non-linearity and Chaos in Natural Temporal Series. Has published over a 50 articles in scientific journals and conferences. Joined the "Departmento de Estatística e Informática" at the "Universidade Federal Rural de Pernambuco" in 2000 where is now Associate Professor.



Teresa B. Ludermir received the Ph.D degree in Artificial Neural Networks in 1990 from Imperial College, University of London, UK. From 1991 to 1992, she was a lecturer at Kings College London. She joined the Center of Informatics at Federal University of Pernambuco, Brazil, in September 1992, where she is currently a Professor and head of the Computational Intelligence Group. She has published over a 200 articles in scientific journals and conferences, three books in Neural Networks and organised two of the Brazilian Symposium on Neural Networks. She is one of the editors-in-Chief of the International Journal of Computation Intelligence and Applications. Her research interests include weightless Neural Networks, hybrid neural systems and applications of Neural Networks.