ESANN'2009 proceedings, European Symposium on Artificial Neural Networks - Advances in Computational Intelligence and Learning. Bruges (Belgium), 22-24 April 2009, d-side publi., ISBN 2-930307-09-9.

Quantum RAM Based Neural Networks

Wilson R. de Oliveira [*]

Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática Recife, PE - Brazil
wilson.rosa@gmail.com

Abstract. A mathematical quantisation of a Random Access Memory (RAM) is proposed starting from its matrix representation. This quantum RAM (q-RAM) is employed as the neural unit of q-RAM-based Neural Networks, q-RbNN, which can be seen as the quantisation of the corresponding RAM-based ones. The models proposed here are direct realisable in quantum circuits, have a natural adaptation of the classical learning algorithms and physical feasibility of quantum learning in contrast to what has been proposed in the literature.

1 Quantum computation and Mathematical Quantisation

Quantum computing [14] was originally proposed by Richard Feynman [7] in the 1980s and had its formalisation with David Deutsch which proposed the quantum Turing machine [4]. Quantum Computing has been popularised through the quantum circuit model [5] which is a quantisation [15] of the classical boolean circuit model of computation. Quantum computer also became a potential parallel device for improving the computational efficiency of neural networks [6].

The quantum information unit is the quantum bit or ”qubit”. A very intuitive view of the quantisation procedure is put forward by Nik Weaver in the Preface of his book *Mathematical Quantization* [15] with just a phrase which says it all: ”The fundamental idea of mathematical quantisation is *sets are replaced with Hilbert spaces*”.

The quantisation of the boolean circuit logic starts by simply embedding the the classical bits $\{0, 1\}$ in a convenient Hilbert space. The natural way of doing this is to represent them as (orthonormal) basis of a Complex Hilbert space. In this context these basis elements are called the computational-basis states.Linear combinations (from Linear Algebra [9]) of the basis spans the whole space whose elements, called states, are said to be in *superposition*. Any basis can be used (recall from Linear Algebra [9] that there usually are many!). But in Quantum Computing it is customary to use the most conventional and well known one: $|0\rangle$, $|1\rangle$ are a pair of orthonormal basis vectors representing each classical bit, or ”cbit”, as column vector, $|0\rangle = [1 \;\; 0]^{\mathrm{T}}$ and $|1\rangle = [0 \;\; 1]^{\mathrm{T}}$. A general state of the system (a vector) can be written as: $|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, where α, β are complex coefficients (called probability amplitudes) constrained by the normalization condition: $|\alpha|^2 + |\beta|^2 = 1$. This is the model of one qubit. Multiple qubits are obtained via *tensor products*. By linearity we just need to say how tensor behaves on the basis: $|i\rangle \otimes |j\rangle = |i\rangle\,|j\rangle = |ij\rangle$, where $i, j \in \{0, 1\}$

If that were all the story, quantum computing would be just a trivial extension of classical computing. But Quantum Mechanics Principles [14] restrain the kind of permissible operations. Operations on qubits are carried out only by unitary operators (i.e.

[*] Supported by MCT-CNPq and PRONEX/FACEPE. On sabbatical leave at the School of Computer Science, University of Birmingham

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). Quantum algorithms on n bits are represented by unitary operators U over the 2^n-dimensional complex Hilbert space:$|\psi\rangle \rightarrow U|\psi\rangle$. Thus all operators on qubits are reversible. The sole exception is a special class of operations called *measurement* which is how information are retrieved from a quantum system. In a sense it is a destructive operation that loses the information about the superposition of states. 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.

A toolkit for a *quantum programmer* includes: the identity operator[1], **I**, which does nothing; the flip operator, **X**, which behaves as the classical NOT on the computational basis; and the Hadamard transformation, **H**, which generates superposition of states. Their matrix representation and and corresponding action on the basis [14]:

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

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

Quantum operators are also pictorially represented as quantum circuits with corresponding quantum gates. Figure 1 shows an n-qubit controlled gate U, where U is an arbitrary unitary operator, whose action on the target qubit (bottommost) is applied or not depending on the $n-1$ (topmost) control qubits [14]. The output is checked by measurement gates.

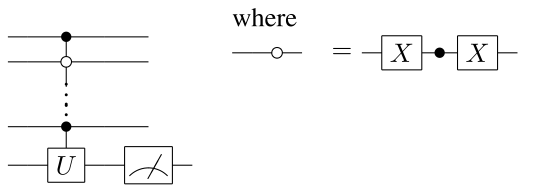

Fig. 1: A quantum circuit

2 RAM as Matrix Circuits

Random Access Memory [10] is an addressable memory device in which information can be stored and retrieved. It consists of an array of memory cells where information is stored as $m-$bits. Each cell location is unique and associated to a unique number (address) which can be directly accessed and thus named "random access". A RAM is composed of the memory array, an input register and an output register. Given an

[1]Very useful when applied in combination (tensor product) with other operators when one needs to leave part of the input unchanged.

address in the input register, the content of the respective memory cell is returned in the output register. If the input register is of size n bits, there are 2^n addressable memory cells. The contents of the memory position $0 \leq k < 2^n$ is denoted here as $C[k]$. $C[k]$ is itself a $m-$bit register.

The actual implementation in terms of boolean circuits or even semiconductor will not concern us here (see [10]). The pictorial abstract representation of a RAM as a table in Figure 2 helps understanding (the input terminals s and d are respectively the learning strategy and the desired output to be learned and are used later).

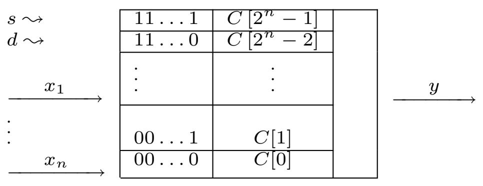

Fig. 2: RAM Node [12]

But view a RAM as circuit composed of matrices acting on cbits (bits represented as basis vectors) is instructive for what comes later. For the sake of simplicity of the exposition and with no loss of generality, let us assume that the output of the RAM is one bit, i.e. $m = 1$. Let the block matrix A be as[2]:

$$A = \begin{pmatrix} I & 0 \\ 0 & X \end{pmatrix} \quad \begin{array}{l} \text{where} \\ A\,|00\rangle = |0\rangle\, I\, |0\rangle \\ A\,|10\rangle = |1\rangle\, X\, |0\rangle \end{array}$$

This matrix behaves a memory cell capable of returning 0 or 1 (actually $|0\rangle$ respect. $|1\rangle$) depending on the corresponding value in the first part of its two-parts input. It may sound redundant, but bear in mind that this is a half way through the quantum model presented below. A $n-$bit input RAM is a collection of 2^n of these A's. The exponential growth is not a concern at the moment, but it can be remedied by similar techniques as in [8]. A RAM with fixed and unalterable memory contents (ROM) has a smaller numbers of matrices. A trained RAM-based neural network has such kind of ROM memory. A circuit composed of these matrices, using a notation borrowed from quantum circuits, representing a two-bit input ROM is shown in Figure 3.

Depending on the value of $|ab\rangle$ the corresponding matrix U_i is selected with the binary to decimal correspondence, e.g. $|10\rangle$ selects U_3, etc. A two-bit RAM with the more general and capable of "storing" cbits matrix A is shown in Figure 4

[2]By abuse of language 0 is used both meaning the number zero and a squared matrix of zeroes or just a symbol devoid of meaning; the context dictates which is the case

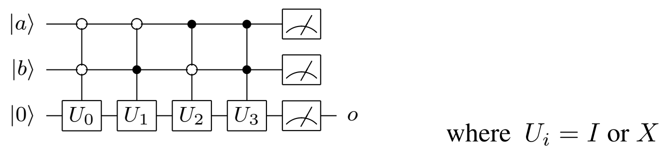

Fig. 3: q-PLN Node in quantum circuit

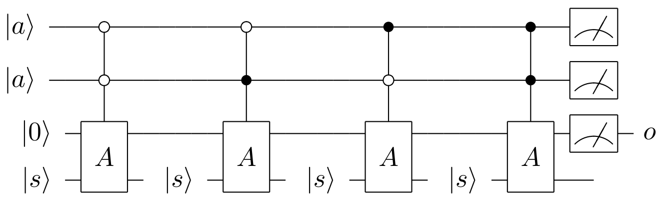

Fig. 4: RAM with selectors

3 RAM-based Neural Networks in a Matrix Form

RAM-based Neural Networks (RbNN) [12] are parallel distributed systems composed of simple processing units realised as RAM memories, in general the unit does generalise but networks of RAMs providing the ability to learn from examples and to generalise. They were introduced by Igor Aleksander in the 1960's.

The building blocks of the our RbNN are the model presented in the last section. Following the classical presentation of RbNN a variety of matrix RbNN can be now be proposed such as WISARD, etc. [12] each with its quantisation.

The weightless neural networks composed of the Probabilistic Logic Node (PLN) need some considerations due to the probabilistic nature of the node. The same Figure 2 can be used to pictorially represent a PLN node. The difference is just that now a 2-bit number is stored at the addressed memory location ($m = 2$). The bitstrings 00, 11 and 01 (or 10) respectively represents $0, 1$ and u. Additionally, one must have a probabilistic output generator. The output of the PLN Node is y, if $C[x] = y$, for $y \in \{0, 1\}$ and randomly 0 or 1 if $C[x] = u$.

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 $\{0, ..., 2^m - 1\}$ which is interpreted as the firing probability $p = \frac{k}{2^m-1}$ of the node.

A similar analysis to the one in Section 2 using the Hadamard matrix $\mathbf{H}$ in addition to the matrices $\mathbf{I}$ and $\mathbf{X}$ interpreting the output of $\mathbf{H}|0\rangle$ as u leads to a matrix formulation of the PLN node. For lack of space this not done here and we rather skip to the presentation of the quantum version of the PLN node which subsumes the matrix formulation of the PLN.

4 Quantisation of the RAM-based Neural Network

The concept of quantum neural computation apparently was first introduced by Kak in 1995 [11]. Kak's model is an attempt to the quantisation of the weighted neural model but actually is just a complexification - changing the real valued weights by complex valued ones - since learning is proposed by directly interfering with the system, violating quantum principles. Various other models and implementation has since been proposed. Good reviews are [6] and the web page by Li Weigang[3]. They are all weighted and there is 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 weighted models but are all found in our model [3].

Another line of research has recently been proposed which closer to this work in the sense which does not use weighted model and proposes learning algorithms directly in the quantum circuit model but differs considerably from ours since they do not propose a (weightless) neural network model but rather uses techniques similar to Machine Learning [2, 1]

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.

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 $\mathbf{H}|0\rangle = |u\rangle$ from the quantum mechanics principles (see e.g. Section 2.2 and Chapter 8 of [14] for a lengthier discussion). The pictorial representation of a q-PLN fully trained (q-ROM) is similar to the Figure 3 but adding the possibility of U_i be $\mathbf{H}$.

Learning is to change the matrix. Universal deterministic Programmable Quantum Gate Arrays are not possible in general [13] but the q-PLN requires only three possible types of matrix. We can define a matrix A similar to the one previously defined

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

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 3 each matrix U_i by the matrix A with the two added input registers we obtain the q-PLN circuit similar to the Figure 4 but know the A circuits has two selectors. The selectors determine which the U_i is *selected* for computation. Learning is adjust the selectors [3].

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

[3]http://www.cic.unb.br/ weigang/qc/aci.html

output 1 (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 $[0, 1]$ which is useful for the quantisation of the pRAM networks. It is a simple exercise to check that all matrices on this form are unitary [9]. The *universal* matrix A for q-MPLN has similar form as for the q-PLN but with the diagonal blocks being the U_p.

5 Conclusion and Future Works

The q-RAM proposed above are directly realisable in quantum circuits (quantum hardware) and can be used for the quantisation of all the (classical) RAM-based neural neural networks and their learning algorithms. Quantisation of the WISARD, GSN, pRAM and GRAMT are being investigated. Encouraging results on the natural adaptation of the learning algorithms and physical feasibility of quantum learning for the q-MPLN models are reported in [3]. Another line of research being pursued is the relationship between quantum neural networks and the quantum automata.

References

[1] J. Bang *et.al.* Quantum learning machine, 2008. arXiv:0803.2976v2 [quant-ph].

[2] E. C. Behrman *et.al.* Quantum algorithm design using dynamic learning. *Quan.t Inform. and Comp.*, 8(1-2):12–29, 2008.

[3] W. R. de Oliveira *et.al.* Quantum logical neural networks. *SBRN '08. 10th Brazilian Symposium on Neural Networks, 2008.*, pages 147–152, Oct. 2008.

[4] D. Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. of the Royal Soc. of Lond. Ser. A*, A400:97–117, 1985.

[5] D. Deutsch. Quantum computational networks. *Proc. of the Royal Soc. of Lond.*, A 425:73–90, 1989.

[6] J. Faber and G. Giraldi. Quantum models of artificial neural networks. Electronically available: http://arquivosweb.lncc.br/pdfs/QNN-Review.pdf.

[7] R. Feynman. Simulating physics with computers. *Int. J. of Theor. Phy.*, 21:467, 1982.

[8] V. Giovannetti *et.al.* Quantum random access memory. *PRL*, 100(16):160501, 2008.

[9] K. Hoffman and R. Kunze. *Linear Algebra.* Prentic-Hall, 1971.

[10] R. C. Jaeger and T. N. Blalock. *Microelectronic Circuit Design.* McGraw-Hill, 2003.

[11] S. C. Kak. On quantum neural computing. *Info. Sci.*, 83(3&4):143–160, 1995.

[12] 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.

[13] M. A. Nielsen and I. L. Chuang. Programmable quantum gate arrays. *PRL*, 79(2):321–324, Jul 1997.

[14] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.

[15] N. Weaver. *Mathematical Quantization.* Studies in Advanced Mathematics. Chapman & Hall/CRC, Boca Raton, Florida, 2001.