ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7.
Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

Vector space weightless neural networks

Wilson R. de Oliveira[1] [*] and Adenilton J. da Silva[2] and Teresa B. Ludermir[2] [†]

1- Universidade Federal Rural de Pernambuco
Departamento de Estatíca e Informática
Dois Irmãos- CEP: 52171-900 - Recife/PE - Brazil

2- Universidade Federal de Pernambuco
Centro de Informática
Cidade Universitária - 50740-560 - Recife/PE - Brazil

Abstract. By embedding the boolean space $\mathbb{Z}_2$ as an orthonormal basis in a vector space we can treat the RAM based neuron as a matrix (operator) acting on the vector space. We show how this model (inspired by our research on quantum neural networks) is of sufficient generality as to have classical weighted (perceptron-like), classical weightless (RAM-based, PLN, etc), quantum weighted and quantum weightless neural models as particular cases. It is also indicated how one could use it to polynomially solve 3-SAT and briefly mention how could one train this novel model.

1 Introduction

It is well known in Algebra that for any set S and a field $\mathbb{F}$, the function space, i.e. the set of all functions from S to $\mathbb{F}$, $\mathbb{F}^S$, is a vector space with the pointwise sum inherited from $\mathbb{F}$ and pointwise scalar multiplication. There is a bijection between the elements of S and a basis for $\mathbb{F}^S$ so we can think of a basis element as actually being an element of $s \in S$, we denote the vector associated with the element s using the Dirac (or ket) notation as $|s\rangle$ In the special case where an inner product can be defined we get an inner product spaces and when it induces a metric which is complete, a Hilbert space is obtained. All that collapses to one notion (vector, inner product or Hilbert space) when the set S is finite, the case we are mostly concerned with here. For finite sets this function space is nothing but (isomorphic to) the coordinate vector space $\mathbb{F}^n$, where $n = \mid S \mid$ is the cardinality of S. Linear operators from $\mathbb{F}^n$ to $\mathbb{F}^m$ are $m \times n$ matrices (representable in an appropriate basis).

This passage from a set S to the vector space $\mathbb{F}^S$ is a special construction interesting in itself since it not only takes sets to vector spaces but sends functions $f : S \to T$ to linear operators $A_f : \mathbb{F}^S \to \mathbb{F}^T$, making it a *functor* from the *category* of sets and functions to the category of vector spaces and linear operators (over a fixed field). When restricted to Hilbert spaces and unitary operators, a similar construction has been called as mathematical quantisation by Nick Weaver [1]. We shall call the general construction *mathematical vectorisation*.

The above construction is the inspiration for the proposed weightless neuron model and was also the inspiration for our previous work on quantum weightless neural networks [2, 3, 4, 5, 6]. Besides, this work can be seen as generalization of our previous

[*]Corresponding author: wilson.rosa@gmail.com

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

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

work by removing the requirement of having to stick with unitary operators. We here allow for any linear and even relaxing to nonlinear operators. It can also been seen as generalization of the weighted models based on the McCulloch-Pitts neuron as well as the matrix model such Anderson-Kohonen, BAM, etc.

The weightless neuron era started with the use of n-tuple RAM neuron in pattern recognition problems about 56 years ago in the work of Bledsoe and Browning [7]. A few years later, Prof. Igor Aleksander introduced the Stored Logic Adaptive Microcircuit (SLAM) and n-tuple RAM neuron as basic components for an adaptive learning network [8]. The field has since grown and matured, despite the prejudice and lack of interest of general researchers in the field of artificial neural networks, with a great deal of successful applications and novel RAM-based models (see [9, 10] and the references in there). Amongst those we may cite the probabilistic versions: Probabilistic Logic Neuron (PLN) [11], Multivalued Probabilistic Logic Neuron (MPLN) [12] and pRAM [13]. In the PLN neuron it is possible to store 0, 1 and u in the memory positions. The u value corresponds to the probability of 50% to output 0 or 1. In the MPLN neuron there are a finite number of finite precision probabilities that can be stored in the memory position and in the pRAM neuron one can store a finite number of arbitrary precision probabilities.

As final introductory remark I would like to point out that this paper is on Neurocomputing Theory. More precisely is on models of Neurocomputing. It is about a way of looking at and speaking about the subject through the use of a mathematical language where general results can be obtained in a unified manner. It is about underlying ideas and concepts. No applications and no learning algorithm are envisaged at the present work. These practical issues will be dealt with in a follow up to this paper.

2 Linear Algebra in Dirac Notation

For lack of space I must assume the basic notions of fields, vector spaces, basis, span, tensor and inner and outer products, eingenvalues and eigenvector, etc all of these notions could be easily accessed in classics such as Halmos [14] and Young [15] or modern approaches such as Weaver's [1]. We shall only explicit mention the scalar field when strictly necessary assuming our remarks are true for any field but having the field of complex numbers $\mathbb{C}$ as our default field. In particular our canonical vector space is $\mathcal{Q} = \mathbb{C}^2$ or its tensor product powers, $\mathcal{Q}^{\otimes n} = \mathbb{C}^{2^n}$.

It is appropriate for our purpose to adopt the Dirac notation for vectors mostly used by quantum physicists and quantum computer scientists. Usually n-dimensional vectors in Linear Algebra are represented as line vectors, $1 \times n$, but here we use column vectors $n \times 1$. In Linear Algebra, we use variables, x, sometimes in boldface, $\mathbf{x}$, or with overline, $\overline{x}$ or $\overrightarrow{x}$, but here we use the bra-ket notation where a name of a vector representing say a state ψ of the system (or space) $\mathcal{A}$ is denoted as $|\psi\rangle$ or $|\psi\rangle_{\mathcal{A}}$. The application of an operator (or matrix) A to a vector ψ is $A|\psi\rangle$, which in the case of matrices is simply the product of the $n \times n$ matrix by the $n \times 1$ column matrix. For a n-dimensional vector space, the canonical basis is here called the *computational basis* also called *cbits*. So, e.g. for the 3-dimensional vector space the computational basis is the set of three vectors $|0\rangle = (1, 0, 0)^T$, $|1\rangle = (0, 1, 0)^T$ and $|2\rangle = (0, 0, 1)^T$. If $|\psi\rangle \in \mathcal{V}_0$

and $|\phi\rangle \in \mathcal{V}_1$, their tensor product is equally denoted as $|\psi\rangle \otimes |\phi\rangle$, $|\psi\rangle|\phi\rangle$ or $|\psi\phi\rangle$. In analogy with the classical bits, where we we represent say a 3-bit state as say 101, we also represent a basis 3-qubit state in $\mathcal{Q}^{\otimes 3}$ as $|101\rangle$ but also as $|1\rangle \otimes |0\rangle \otimes |1\rangle$, $|1\rangle|0\rangle|1\rangle$. Notice that n-qubit state lives n 2^n-dimensional complex vector space. Recalling that the conjugate transpose of a matrix (operator) is the transpose of the conjugate of its entries $A^\dagger = (A^*)^T$, a bra is then the conjugate transpose of a ket $\langle\psi| = |\psi\rangle^\dagger$. With this notation the inner and outer product of a vector named ψ are respectively $\langle\psi||\psi\rangle$ and $|\psi\rangle\langle\psi|$, respectively a number and a matrix. It is usual and useful to consider basis which are orthonormal having $\langle e_i||e_j\rangle = 1$ if, and only if, $i = j$ and zero otherwise. Our preferred and canonical computational basis is orthonormal and the operator

$$R = \sum_{x \in \mathbb{Z}_2^n} |\alpha_x\rangle\langle x|$$

where $\mathbb{Z}_2 = \{0, 1\}$, the field of integers modulus 2, has a very interesting interpretation which relates it to a a sort of generalised look up table or RAM memory: the action of R on a basis element $|x\rangle$ returns only $|\alpha_x\rangle$ which can be interpreted as the content of the memory location addressed by x (or $|x\rangle$). This is start point of our general model below.

3 A Brief Review of RAM-based Weightless Neural Networks

Here we intend to give a most general definition of a RAM and the related RAM-based neuron. Random Access Memory [16] 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 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 $C[k]$ which is itself a $m-$bits register. Our model allow for a internal *activation function* $a : \mathbb{Z}_2^m \to \mathbb{Z}_2^p$ which transform the internal m-bits number into an output p-bits number. In case the activation function is the identity function, we have the usual RAM.

In a first level of abstraction a RAM is just a special kind of function with finite domain and codomain (or simply a look-up table). The domain are $n-$bits and the codomain $p-$bits, respectively represented as $\mathbb{Z}_2^n$ and $\mathbb{Z}_2^p$ and $\mathbb{Z}_2^n$ can be seen as either a string of bits or the number $0 \leq k < 2^n$ it represents. But this is not enough. A RAM has the ability of store and recover data. A table or an array indexed by $\{0, 1, \ldots, 2^n - 1\}$ is a more appropriate level of abstraction but the function notation will be kept. The actual implementation in terms of boolean circuits or even semiconductor will not concern us here (see [16]). The pictorial abstract representation of a RAM as a table in Figure 3 helps the understanding. The input terminals s and d are respectively the learning strategy and the desired output to be learned but are not considered here once learning is not an issue at the moment. It is useful to adopt the notation of calling the RAM defined above a RAM(n,m,p) or when the activation function is emphasised RAM(n,m,p,a)

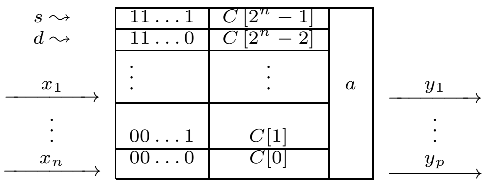

Fig. 1: A general RAM Node

Weightless or RAM-based Neural Networks (RbNN) [9] are parallel distributed systems composed of simple processing units realised as RAM memories, in general the unit does generalise but networks of RAMs provide the ability to learn from examples and to generalise. They were introduced by Igor Aleksander in the 1960's [8] and were mostly restricted to $m = 1$ with a being the identity. A very readable introduction is [17], while [9] is a more complete and updated review.

The building blocks of the our RbNN are the RAM neuron model presented above and depicted in Figure 1. Following the classical presentation of RbNN a variety of matrix RbNN can be now be proposed such as WISARD, etc. [9].

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 1 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 activation function of the PLN Node returns y, if $C[x] = y$, for $y \in \{0, 1\}$ and uniformly random 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 output 1.

The pRAM [13] is also an extension of the PLN like the MPLN, but in which continuous probabilities can be stored, that is, value in the range $[0, 1]$, with infinite precision, $m = \infty$. The usual activation function employed is $\hat{y} = \frac{1}{T}\sum_{t=1}^{T} r(t)$.

4 Vector Space Weightless Neural Networks

The definition of RAM neuron in the last section is quite general and, we conjecture, covers most if not all weightless model which takes binary inputs. Some others models such GSN [18] and a variation of the pRAM, the i-pRAM [19] allow for non binary inputs as well the quantum model in [3]. The weighted models, mostly using the McCulloch-Pitts neuron, use non binary real valued inputs. The definition which follows aims at to be so general as to cover all these cases as well as the models in the

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

previous section. The ultimate general weightless neuron acts over a vector (or Hilbert) space. A *Vector Space Neuron* (VSN) is the mathematical vectorisation of the RAM neuron. In this way, binaries inputs are converted to cbits and the action of recovering the stored valued is performed by the R operator above which can be further generalised to take the activation function into account:

$$R = \sum_{x \in \mathbb{Z}_2^n} A|\alpha_x\rangle\langle x|$$

where A is $p \times m$ matrix (operator) to stay in tune with our general RAM(n,m,p) neuron. If a general state, not necessarily a cbit but a general qubit, say $|\psi\rangle = \sum_{x \in S \subseteq \mathbb{Z}_2^n} c_x|x\rangle$, $c_x \in \mathbb{F}$ are scalars, is given as input, all "cells" with address in S will be recovered. Reader familiar with the GSN model will recognise this. A VSN with just one entry in a linear combination $\sum_{i<n} x_i|i\rangle$ and having as activation $F \circ \sum_{i<n} w_i|i\rangle\langle i|$, outputs the action of F on weighted sum of the inputs x_i thus simulating a McCulloch-Pitts Neuron, if F is one of the nonlinear sigmoid functions usually employed. For multiple inputs to a VSN we take their tensor product. This gives a higher order interaction if the inputs are each linear combinations of the weighted neuron being simulated.

5 A few remarks on polynomially solving 3-SAT

Any boolean function is computed by a RAM neuron and consequently by our VSN. SAT is the problem of determining weather a conjunction of disjunctions of literals (a boolean variable or its negation) is true for assignment of truth values to the variables involved i.e if a boolean function of the variables ever return 1. 3-SAT limits the number of literals in a disjunction (clause) to three. Thus a RAM-based WNN with two layers in which the first layer has k three input neurons, where k is the number of clauses, and the second layer has one k-input neuron performing the conjunction, evaluates a 3-SAT instance. A VSN, using superposition (linear combination) evaluates in one step ALL instances of a given 3-SAT formulae. By using the tensor powers of the operator $O = 2^n|1\rangle\langle 1|$ as activation function in the neuron in the second layer we are able to solve 3-SAT in polynomial steps. It has to be said that the space required (size of the vectors and matrices) of an actual (classical) implementation are exponential on the number of variables.

6 Conclusions and further work

We have given the firsts steps towards what may become an interesting generalization of neural networks where various weightless and weighted models are special cases. A natural follow up is to catalogue what is in the literature, if that is possible and make improvements or further generalisations where is not. Generalise learning algorithms in order to apply to a different class of models sounds promising - this has already been done for the special case of MPLN [12, 20]. A promising future works is to investigate the use of nonlinearity allowed in our model and the know computing power which it brings into play [21, 22, 23]. Relations with geometry of information retrieval [24] must be investigated with a view to applications.

ESANN 2014 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 23-25 April 2014, i6doc.com publ., ISBN 978-287419095-7. Available from http://www.i6doc.com/fr/livre/?GCOI=28001100432440.

References

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

[2] Wilson R. de Oliveira, Adenilton J. Silva, Teresa B. Ludermir, Amanda Leonel, Wilson R. Galindo, and Jefferson C.C. Pereira. Quantum logical neural networks. *SBRN '08. 10th Brazilian Symposium on Neural Networks, 2008.*, pages 147–152, Oct. 2008.

[3] W. R. de Oliveira. Quantum RAM based neural netoworks. In M. Verleysen, editor, *ESANN'09: Advances in Computational Intelligence and Learning*, pages 331–336. ISBN 2-930307-09-9, 2009.

[4] A.J. Silva, T.B. Ludermir, and W.R de Oliveira. A weightless neural node based on a probabilistics quantum memory. In *Proceedings of the Brazilian Symposium no Artificial Neural Networks, SBRN 2010*, 2010.

[5] Adenilton J. Silva, Wilson R. de Oliveira, and Teresa B. Ludermir. Classical and superposed learning for quantum weightless neural networks. *Neurocomput.*, 75:52–60, January 2012.

[6] A.J. Silva, W.R. de Oliveira, and T.B. Ludermir. Single-shot learning algorithm for quantum weightless neural networks. In *Congresso Brasileiro de Inteligência Computacional*, 2013.

[7] W. W. Bledsoe and I Browning. Pattern recognition and reading by machine, 1959.

[8] I. Aleksander. Self-adaptive universal logic circuits. *Electronics Letters*, 2(8):321–322, 1966.

[9] 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.

[10] I. Aleksander, M. de Gregorio, F.M.G. França, P.M.V. Lima, and H. Morton. A brief introduction to weightless neural systems. In *Proceedings of the 17th European Symposium on Artificial Neural Networks (ESANN 2009)*, page 299–305, 2009.

[11] W. K Kan and I. Aleksander. A probabilistic logic neuron network for associative learning. *IEEE First International Conference on Neural Networks (ICNN87), volume II, pages 541-548*, 2:541–548, 1987.

[12] C. Myers and I. Aleksander. Learning algorithms for probabilistic logic nodes. In *Abstracts of the I Annual International Neural Networks Society Meeting (INNS88)*, page 205.

[13] J.G. Taylor. Spontaneous behaviour in neural networks. *Journal of Theoretical Biology*, 36:513– 528, 1972.

[14] P.R. Halmos. *Finite-Dimensional Vector Spaces*. Undergraduate Texts in Mathematics. Springer London, Limited, 2011.

[15] N. Young. *An Introduction to Hilbert Space*. Cambridge mathematical textbooks. Cambridge University Press, 1988.

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

[17] J. Austin. A review of ram based neural networks. *Microelectronics for Neural Networks and Fuzzy Systems, 1994., Proceedings of the Fourth International Conference on*, pages 58–66, Sep 1994.

[18] E.C.D.B.C. Filho, M.C. Fairhist, and D. L. Bisset. Adaptive pattern recognition using the goal seaking neuron. *Pattern Recognition Letters*, 12:131–138, 1991.

[19] D. Gorse and J. G. Taylor. A continuous input RAM-based stochastic neural model. *Neural Networks*, 4:657–665, 1991.

[20] C. Myers and I. Aleksander. Output functions for probabilistic logic nodes. In *Proceedings of the IEE International Conference on Neural Networks*, pages 310–314.

[21] Daniel S. Abrams and Seth Lloyd. Nonlinear quantum mechanics implies polynomial-time solution for np-complete and $\#p$ problems. *Phys. Rev. Lett.*, 81:3992–3995, Nov 1998.

[22] Masanori Ohya and Igor V. Volovich. Quantum computing, np-complete problems and chaotic dynamics. *CoRR*, quant-ph/9912100, 1999.

[23] Masanori Ohya and Igor V Volovich. Quantum computing and the chaotic amplifier. *Journal of Optics B: Quantum and Semiclassical Optics*, 5(6):S639, 2003.

[24] C.J. van Rijsbergen. *The Geometry of Information Retrieval*. Cambridge University Press, 2004.