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.

Probabilistic automata simulation with single layer weightless neural networks

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

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

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

Abstract. Computability of weightless neural networks is the major topic of this paper. In previous works it has been shown that, one can simulate a Turing machine with a weightless neural network (WNN) with an infinite tape. And it has also been shown that one can simulate probabilistic automata with a WNN with two queues. In this paper, we will show that is possible to simulate a probabilistic automata with a single layer WNN with no auxiliary data structures.

1 Introduction

Weightless neural networks [1, 2] (WNN) were proposed by Igor Aleksander in the late sixties [3]. WNNs are a simple and powerful model which has been used in several applications [4, 5, 6]. In this paper, we study the computational capabilities of recurrent WNNs.

The computational capabilities of WNNs have been studied previously in several occasions [7, 8, 9, 10]. In [7] and [9] the authors showed an equivalence between weightless neural networks and probabilistic automata. Probabilistic automata are more powerful [11] than finite state automata [12] and can recognize all regular languages, some context-free, some context-sensitive and some recursive enumerable languages, cutting the Chomsky Hierarchy [12] elliptically.

The advantage of [9] over [7] is that the neural network in [9] has a well-defined architecture. On the other hand, in [9] two queues are used as auxiliary data structure whilst in [7] neurons have an additional parameter storing a probability and the network does not use an auxiliary data structure. In this paper, it will be presented an algorithm to convert a probabilistic automaton into a WNN. The WNN used has a single layer (as in [9]) and it does not need an auxiliary data structure (as in [7]).

The remainder of this paper is divided in 3 sections. Section 2 presents the concepts of probabilistic automata and weightless neural networks. Section 3 presents the main results of this work: the definition of the pGSN Weightless Neural Networks and the algorithm to convert probabilistic automata in WNNs. Finally, Section 4 is the conclusion.

*This work is funding by research grants from CNPq (Edital Universal) and FACEPE-PRONEX.

2 Preliminary concepts

2.1 Weightless neural networks

A RAM neuron with n binary inputs consists of 2^n one bit addressed memory positions C. When an input x is presented to a RAM neuron, its output will be the memory content $C[x]$ addressed by x. The learning procedure in the RAM neuron consist in to changing values stored in the memory C.

A PLN neuron is a generalization of the RAM neuron. In PLN is possible to store a two-bit number in each memory position. Values stored in PLN memory can be 0, 1 and u. If a neuron has input x then the output of the neuron will be 0 if $C[x] = 0$, 1 if $C[1] = 1$, and 0 or 1 with same probability if $C[x] = u$.

The PLN neuron can be further generalised allowing one to store a real number in its memory positions, such neuron is known as the pRAM neuron [13]. The output of a pRAM neuron with input x is probabilistic, and will be 1 with probability p or 0 with probability $1 - p$ if $C[x] = p$. The PLN neuron can also be modified to produce outputs and receives inputs in the set $\{0, 1, u\}$, and in this case is called Goal Seeking Neuron (GSN) [14].

2.2 Probabilistic automata

Probabilistic automata [11] generalize the concept of deterministic finite automata. In the probabilistic automata the transition function determines the probabilities $p_{ij}(a)$ of going with input a from state q_i to state q_j. Definition 1 formalises the concept of probabilistic automata [9].

Definition 1 *A probabilistic automata PA is a 5-tuple* $A_p = (\Sigma, Q, H, q_I, F)$ *where:*

- $\Sigma = \{\sigma_1, \cdots \sigma_{|\Sigma|}\}$ *is a finite set of symbols called the* input alphabet*;*
- $Q = \{q_0, q_1, \cdots, q_{|Q|}\}$ *is a finite set of states;*
- $H = \{H_a\}_{a\in\Sigma}$ *is a set of stochastic* $n \times n$ *state transition matrices (where* $n = |Q|$ *is the number of states in* Q*). The* (i, j)*-entry of* H_a*, is* $H_a[i, j] = p_{ij}(a)$*, where* $p_{ij}(a)$ *is the probability of entering state* q_j *from state* q_i *under input* a*.*
- $q_I \in Q$ *is the initial state in which the machine is found before the first symbol of the input string is processed;*
- $F \subset Q$ *is the set of final states.*

Diagram of a probabilistic automaton A_p is displayed in Figure 1. With input $w = 110$, A_p follows the state sequence ($\{q_0\}$, $\{q_1\}$, $\{q_1(50\%), q_2(50\%)\}$, $\{q_4(25\%), q_3(50\%)\}$). The language of a probabilistic automata is the set of strings that leads the automata to a final state q with probability greater than a given threshold λ. Languages recognized by probabilistic automata include all regular languages, some context-free, some context-sensitive and some recursively enumerable languages [11].

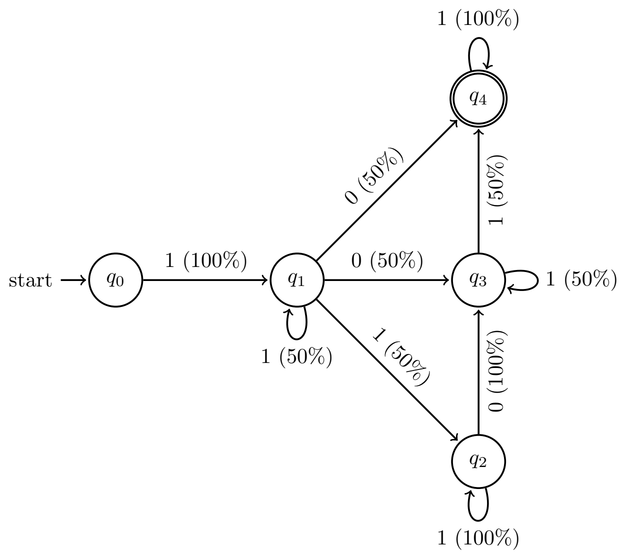

Fig. 1: State diagram of a probabilistic automata A_p.

3 Probabilistic automata simulation by WNN

The neuron used in this section is an evolution of the GSN neuron that can be considered as a contribution of the present work: a novel GSN neuron. Recall that GSN neuron can receives inputs 0, 1 or u (undefined). With input u the neuron will enter in more than one memory position. For instance, if a neuron with two inputs receives the signal $u0$, the neuron will access memory positions 00 and 10. Here we generalize the GSN neuron inspired with concepts from quantum computation.

A quantum bit can be in a linear combination of 0 and 1 with a given probability. We use this idea to define a probabilistic GSN (pGSN) neuron that receives real inputs. For instance, let x and y be real numbers, with input (x, y), a two input neuron will access all its memory positions and its output will be the linear combination $(1-x)(1-y)\cdot C[00]+(1-x)y\cdot C[01]+x(1-y)\cdot C[10]+xy\cdot C[11]$. Each memory content C of the GSN with n inputs stores a n-dimensional vector of real numbers.

Now we simulate a probabilistic automata with a WNN composed of pGSN neurons. Similar result was previously obtained in [7, 9] but with others weightless neural models. Here we perform the simulation without auxiliary data structures and with a single layer neural network composed of pGSN neurons, which we call SpGSN.

For lack of space we only present the Algorithm 1, without a proof of correctness, using the pGSN neuron to simulate a given probabilistic automata. If an automaton has n states, we will impose an order on these states and represent the i-th state by a vector of n bits where the i-th bit is set to 1 and all the

Algorithm 1: From probabilistic automata to WNN

```
1  Arbitrarily order the states.
2  Represent the $ith$ state $q_i$ by a vector with $n_s$ memory positions where
   the $ith$ memory position is set to 1 and the others bits are set to 0.
3  Arbitrarily order the the elements of $\Sigma$
4  Represent the $jth$ input symbol $a_j$ by a vector with $n_i$ memory positions
   where the $jth$ memory position is set to 1 and the others positions are set
   to 0.
5  Create $|\Sigma|$ single layer WNN with $|Q|$ neurons, where each neuron $i$ in
   network $j$ receives as input the $ith$ memory position of state
   representation and the $jth$ memory position of next input representation.
6  foreach Neuron $i$ in Network $j$ do
7      foreach $xy \in \{0,1\}^2$ do
8          if $x \wedge y = 1$ then
9              foreach state $k$ do
10                 Set $k$-th coordinate of memory position $xy$ of neuron $i$ in
                   network $j$ to the probability that $\delta(q_i, a_j) = a_k$
11             end
12         else
13             Set memory position $xy$ of neuron $i$ in network $j$ to $0^{|Q|}$
14         end
15     end
16 end
17 end
```

others are set to 0. This representation is knowed as 1-of-n. We follow the same strategy to represent the elements of Σ. These operations are represented in Steps 1 to 4 of Algorithm 1.

Let n_s be the number of states and n_i the number of possible inputs. Step 5 of Algorithm 1 creates n_i networks, with n_s neurons by network. All neurons in the networks has 2 inputs and 4 memory positions, and each memory position stores n_s bits. The total number of memory positions used in the WNN representation is $4 \cdot n_i \cdot n_s^2 = O(n_i \cdot n_s^2)$.

Neurons memories are initialized in the loop starting at Step 6. The k-th coordinate of memory position $C[11]$ of neuron i in network j is set to the probability of going to state q_k from the state q_i with input a_j. The others memory positions are set to the binary string 0^{n_s}.

Figure 1 is the state diagrama of a probabilistic automata. The WNN representation of this probabilistic automata following Algorithm 1 is presented in Figure 2.

The network constructed by Algorithm 1 process strings as follows. Registers $q_0, \cdots, q_{n_s}$ are initialized with the vector representation of the initial state q_0. Vector representation of the next input is fed to the network, and the network is

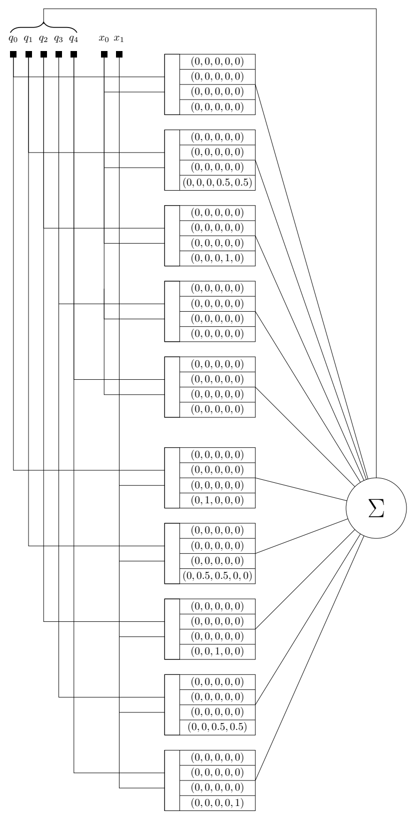

Fig. 2: State diagram of a probabilistic automata A_p.

allowed to produce its output. In the next execution, the network output is fed to the registers $q_0, \cdots, q_{n_s}$ and the vector representation of the next input symbol is fed to the registers $x_1, \cdots, x_{n_i}$. When all the input symbols are read, the network stops and accepts if q_j is a final state and the jth coordinate of the state vector is greater than a threshold λ. One can easily verify with an induction over the input string size that the proposed WNN simulates the probabilistic automata.

4 Conclusion

The computability of WNN networks have been study in several others occasions in the literature [10, 9, 7]. Equivalence between probabilistic automata and WNN is shown in [9, 7]. In these papers it is used an auxiliary data structure or the network has unconventional architecture. Here we show a construction to simulate probabilistic automata with WNN without auxiliary data structures and with a single layer neural network. We proposed a new weightless neuron used in this construction. It is an evolution of the GSN neuron, by allowing real number as inputs and vectors as output.

The SpGSN network can be trained with learning algorithms previously proposed in the weightless literature. One possible future work is to verify if one can train a SpGSN with a learning algorithm that uses continuous optimization such as gradient descent or backpropagation. Another possible future work is to extend the studies in this paper to verify if it is possible to simulate Turing machines with recurrent SpGSN networks, which we conjecture to be true.

References

[1] 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.

[2] 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.

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

[4] S. Yong, W. Lai, and G. Goghill. Weightless neural networks for typing biometrics authentication. In M. Negoita, R. J. Howlett, and L. C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 3214 of *Lecture Notes in Computer Science*, pages 284–293. Springer Berlin Heidelberg, 2004.

[5] D. O. Cardoso, P. M. V. Lima, M. de Gregorio, J. Gama, and F. M.G. França. Clustering data streams with weightless neural networks. In *ESANN*, 2011.

[6] B. Grieco, P. Lima, M. de Gregorio, and F. M. G. França. Producing pattern examples from "mental" images. *Neurocomputing*, 73(7):1057–1064, 2010.

[7] T.B. Ludermir. Computability of logical neural networks. *Journal of Intelligent Systems*, 2(1):261–290, 1992.

[8] M. C. P. de Souto, J. C. M. Oliveira, and T. B. Ludermir. A tool to implement probabilistic automata in ram-based neural networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1054–1060. IEEE, 2011.

[9] M. C. P. de Souto, T.B. Ludermir, and W.R. de Oliveira. Equivalence between ram-based neural networks and probabilistic automata. *Neural Networks, IEEE Transactions on*, 16(4):996–999, 2005.

[10] W.R. de Oliveira, M. C. P. de Souto, and T.B. Ludermir. Turing's analysis of computation and artificial neural networks. *Journal of Intelligent & Fuzzy Systems*, 13(2-4):85–98, 2003.

[11] M. O. Rabin. Probabilistic automata. *Information and control*, 6(3):230–245, 1963.

[12] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Pearson Addison-Wesley, Upper Saddle River, NJ, 3 edition, 2007.

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

[14] 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.