ELSEVIER

Theoretical Computer Science 168 (1996) 405–416

Theoretical Computer Science

Symmetric discrete universal neural networks [1]

Eric Goles *, Martín Matamala

Departamento de Ingeniería Matemática, Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Casilla 170-3 Correo 3, Santiago, Chile

Abstract

Given the class of symmetric discrete weight neural networks with finite state set $\{0,1\}$, we prove that there exist iteration modes under these networks which allow to simulate in linear space arbitrary neural networks (non-necessarily symmetric). As a particular result we prove that an arbitrary symmetric neural network can be simulated by a symmetric one iterated sequentially, with some negative diagonal weights. Further, considering only the synchronous update we prove that symmetric neural networks with one refractory state are able to simulate arbitrary neural networks.

1. Introduction

Let us consider a real $n \times n$ matrix W, a vector $b \in \mathbb{R}^n$ and a partition $\mathscr{Y} = \{I_1, \ldots, I_p\}$, $p \geqslant 1$, of the set $\{1, \ldots, n\}$. We say that $\mathscr{N} = (W, b, \mathscr{Y}, n)$ is a neural network updated under the partition $\mathscr{Y}$, i.e. given a global state $x \in \{0,1\}^n$ for the network the new global state $F(x)$, is computed as follows:

$$\forall i \in \{1, \ldots, p\} \quad \forall j \in I_i \quad F_j(x) = \mathbb{1}\left(\sum_{k \in I_i^-} w_{jk} F_k(x) + \sum_{k \in I_i^+} w_{jk} x_k - b_j \right),$$

where $\mathbb{1}(u) = 1$ iff $u \geqslant 0$ (0, otherwise), $I_i^- = \bigcup_{l<i} I_l$ and $I_i^+ = \bigcup_{l \geqslant i} I_l$.

It is clear that previous scheme can be formulated as a non-linear iterative system

$$F(x) = \mathbb{1}^n(W^{\mathrm{L}} F(x) + W^{\mathrm{U}} x - b),$$

where

$$w_{jk}^{\mathrm{L}} = \begin{cases} w_{jk}, & j \in I_i,\ k \in I_i^-, \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad w_{jk}^{\mathrm{U}} = \begin{cases} w_{jk}, & j \in I_i,\ k \in I_i^+, \\ 0, & \text{otherwise.} \end{cases}$$

[1] Partially supported by Fondecyt 1940520(E.G) and 1950569(M.M), ECOS (E.G, M.M) and CEE-CII*CT92-0046.

* Corresponding author.

PII S0304-3975(96)00085-0

Moreover, $\mathbb{1}^n$ is the function defined from $\mathbb{R}^n$ into $\{0,1\}^n$ as the function $\mathbb{1}$ in each component.

Previous update mode can be interpreted as follows: one iterates the network block by block in the periodic order $1,2,\ldots,p$. Inside each block the neurons are updated synchronously.

For instance, consider the example given in Fig. 1. There, we have a neural network with four neurons connected with the weight matrix W. Consider the partition $\mathscr{Y} = \{\{1\},\{2,3\},\{4\}\}$. Then, if we set the initial state of the network to $x=(1,0,1,0)$ we get as the new global state $F(x)=(0,1,1,1)$. One global transition is split in three substeps. First, neuron one is updated producing the substate $x'=(0,0,1,0)$. Second, neurons two and three are updated in parallel giving the substate $x''=(0,1,1,0)$. Finally, neuron four is updated alone.

As particular cases of previous schema one finds: the parallel update $\pi=\{\{1,\ldots,n\}\}$ or $W^{\mathrm{L}}=0$ and $W^{\mathrm{U}}=W$, which means that the whole network is updated synchronously. The sequential update corresponds to $\mu=\{\{1\},\ldots,\{n\}\}$, that is to say one updates the network site by site in the periodic order $1,2,\ldots,n$. Partition μ is also called the finest partition. It is easy to see that the sequential update is associated to upper triangular matrices W^{U}.

In previous context, one step of the neural network dynamic consists in the iteration of every neuron under the partition $\mathscr{Y}$.

We will say that a network $\widetilde{\mathscr{N}} = (\tilde{W},\tilde{b},\tilde{\mathscr{Y}},\tilde{n})$ simulates $\mathscr{N}=(W,b,\mathscr{Y},n)$ if given a trajectory $\{x(t)\}_{t\geqslant 0}$ of the neural network $\mathscr{N}$ there exists a trajectory $\{y(t)\}_{t\geqslant 0}$ on $\widetilde{\mathscr{N}}$, where $y(0)$ contains as a subconfiguration a code of $x(0)$, such that $y(tq)$, $t\geqslant 0$, contains a subconfiguration coding $x(t)$, for some $q\in\mathbb{Q}$. It is natural to measure the complexity of the simulation as the product of the size of $\widetilde{\mathscr{N}}$ and the time factor q. So, when this product is cn we say that the simulation is in linear time. When $c=1$ we

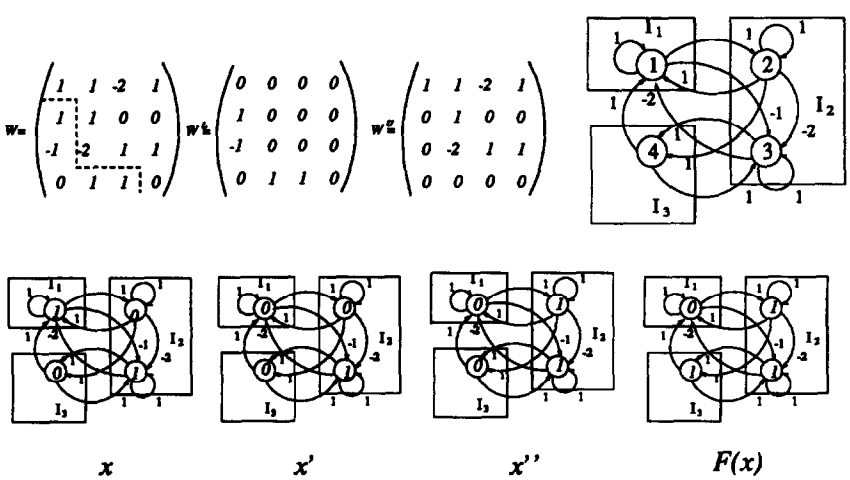

Fig. 1. Block-sequential iteration of a neural network with four neurons.

speak about a real time simulation. We impose that the coding be an injective function and easily computable.

It is well known that without restrictions on the matrix weights, a finite state neural network iterated in parallel can simulate arbitrary algorithms, and, if infinitely many neurons are provided a Universal Turing Machine [3, 6]. In this paper we will focus our attention in internal simulation capabilities of neural networks, i.e. given a class of neural nets defined by some weights and/or partition characteristics, we study the capabilities of such class in order to simulate in polynomial time and space every other neural network.

In previous framework we will study the simulation capabilities of symmetric neural networks iterated under a given partition $\mathscr{Y}$. In fact, we will prove that previous class is universal in the sense that every neural network (non necessarily with symmetric weights) can be simulated in linear time and space by a symmetric one iterated under a partition $\widetilde{\mathscr{Y}}$. Furthermore, we prove that one may always consider only the sequential iteration, so the class of symmetric weights neural nets iterated sequentially is universal. This last result is interesting because, as we will recall in next paragraph, the sequential iteration with symmetric and non-negative diagonal weights corresponds to the Hopfield model and it admits an energy operator which implies the convergence only to fixed points [3, 5]. So, our universality result holds because we relax the non-negativity hypothesis on diagonal weights.

As notation we define the following classes of neural networks. Given a partition $\mathscr{Y}$,

$$N(\mathscr{Y},n) = \{(W,b,\mathscr{Y},n); b\in\mathbb{R}^n, W \text{ being a } n\times n \text{ real matrix}\},$$

$$NS(\mathscr{Y},n) = \{(W,b,\mathscr{Y},n); b\in\mathbb{R}^n, W \text{ being a symmetric } n\times n \text{ matrix}\},$$

$$NS^+(\mathscr{Y},n) = \{(W,b,\mathscr{Y},n); b\in\mathbb{R}^n, W \text{ being symmetric, } diag(W)\geqslant 0\}.$$

Clearly, $NS^+(\mathscr{Y},n)\subseteq NS(\mathscr{Y},n)\subseteq N(\mathscr{Y},n)$.

In the context of partition iteration we have the following result, which will be useful in next sections.

Proposition 1. *Given a neural network $\mathscr{N}=(W,b,\mathscr{Y},n)$, where $\mathscr{Y}$ is not the finest partition, there exists $\widetilde{\mathscr{N}}=(\widetilde{W},\widetilde{b},\mu,2n)$, which simulates $\mathscr{N}$ in real time.*

Proof. The iteration in $\mathscr{N}=(W,b,\mathscr{Y},n)$ is given by $F(x)=\mathbb{1}^n(W^{\mathrm{L}}F(x)+W^{\mathrm{U}}x-b)$. We define $\widetilde{W}$ and $\widetilde{b}$ by

$$\widetilde{W}=\begin{pmatrix} W^{\mathrm{L}} & W^{\mathrm{U}} \\ W^{\mathrm{U}} & W^{\mathrm{L}} \end{pmatrix}, \quad \widetilde{b}=\begin{pmatrix} b \\ b \end{pmatrix}.$$

The sequential iteration of $\widetilde{\mathscr{N}}$ is given by

$$G(z)=\mathbb{1}^{2n}(\widetilde{W}^{\mathrm{L}}G(z)+\widetilde{W}^{\mathrm{U}}z-\widetilde{b})=\begin{pmatrix} G_1(z) \\ G_2(z) \end{pmatrix},$$

where

$$\widetilde{W}^{\mathrm{L}} = \begin{pmatrix} W^{\mathrm{L}} & 0 \\ W^{\mathrm{U}} & W^{\mathrm{L}} \end{pmatrix}, \qquad \widetilde{W}^{\mathrm{U}} = \begin{pmatrix} 0 & W^{\mathrm{U}} \\ 0 & 0 \end{pmatrix}$$

and $G_i(z) \in \{0,1\}^n$, $i = 1,2$. Now, consider $z = (z_1, z_2)$ and set $z_1 = z_2 = x$. Then,

$$G_1(z) = \mathbb{1}^n(W^{\mathrm{L}}G_1(z) + W^{\mathrm{U}}z_2 - b) = \mathbb{1}^n(W^{\mathrm{L}}G_1(z) + W^{\mathrm{U}}x - b) = F(x)$$

and

$$G_2(z) = \mathbb{1}^n(W^{\mathrm{U}}G_1(z) + W^{\mathrm{L}}G_2(z) - b) = \mathbb{1}^n(W^{\mathrm{L}}G_2(z) + W^{\mathrm{U}}F(x) - b) = F^2(x).$$

So, $G(x,x) = (F(x), F^2(x))$. It is easy to see that $G(x, F(x)) = (F^2(x), F^3(x))$. Thus, each global step in $\widetilde{\mathcal{N}}$ computes two step of $\mathcal{N}$. Therefore, the simulation is carried out in real time. □

A schema of previous morphism is given in Fig. 2.

As a particular case of previous iteration we have that the parallel iteration can be simulated by the sequential one. In this case the new architecture is shown in Fig. 3.

It is important to point out that between the set of block-sequential updates, the sequential one is universal in the sense that it may simulate any other iteration mode. Since the size of $\widetilde{\mathcal{N}}$ is $2n$ and the time factor is $\frac{1}{2}$, we get a real time simulation ($n = 2n \times \frac{1}{2}$).

Another neural network model was proposed by Shingai [9]. It takes into account the refractory properties of neurons, i.e. after firing, a neuron needs a refractory time to be fired again. Shingai modelized this fact by adding some extra special states. In this context we will prove that the class of refractory neural networks with three states $\{-1,0,1\}$ and with symmetric interconnections is able to simulate arbitrary (non-necessarily symmetric) classical neural networks with states $\{0,1\}$, in linear time and linear space.

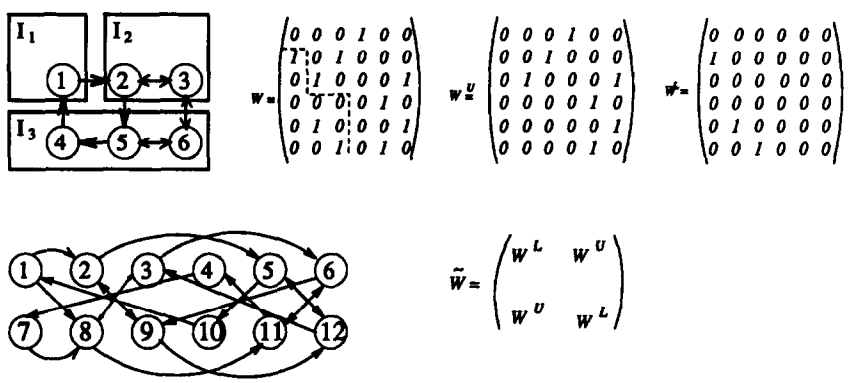

Fig. 2. Neural network $\widetilde{\mathcal{N}}$ for $\mathcal{Y} = \{I_1, I_2, I_3\}$. ⟷: denotes arcs of w_{ij} in both ways (if there exists). →: denotes arcs only in one way.

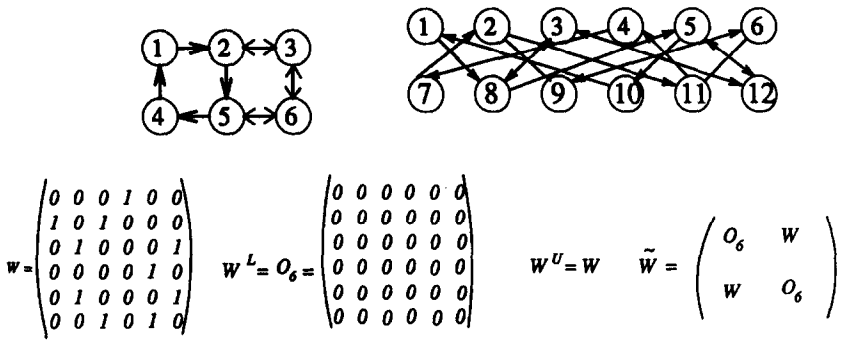

Fig. 3. Neural network $\tilde{\mathcal{N}}$ for the parallel update. Here there are only arcs between the two layers.

2. Symmetric weight neural networks

The motivation to study symmetric neural networks appeared from the application of the Hebb rule to stock pattern as attractors of the neural network dynamics. Since the Hebb rule produces symmetric weights one may associate an energy operator to the network behavior.

First we recall some classical properties about symmetric networks.

Proposition 2. *Suppose that W is a $n \times n$ symmetric weight matrix. Then:*

(i) *The parallel iteration on $\mathcal{N} = (W, b, \pi, n)$ converges to fixed points or two cycles* [1, 5].

(ii) *The transient time of the parallel iteration could be exponential on n* [3].

(iii) *With the additional hypothesis, $diag(W) \geqslant 0$, the sequential iteration converges to fixed points* [4, 5].

(iv) *The transient time of the sequential update could be exponential on n* [3].

(v) *Given a partition $\mathcal{P} = \{I_1, \ldots, I_k\}$, if the submatrices corresponding to each block I_s are positive definite the neural network converges to fixed points* [2].

Previous results (i), (iii), (v) have been obtained by the construction of algebraic and energy operators which drives the network dynamics [1, 3]. It is interesting to point out that (i) can be obtained from (iii) by applying Proposition 1. In fact, it suffices to simulate the parallel update of $\mathcal{N}$ by a neural neuron with $2n$ sites as in Fig. 4. Clearly, the new weight matrix is symmetric with null diagonal, so by (ii) it converges to fixed points of the form $(x(q), x(q+1))$ which represents fixed points or two cycles for the parallel trajectory.

It is also important to point out that, by using the construction given in (ii), Orponen [8] proved that non-necessarily symmetric neural network whose convergence time to an attractor is polynomial, can by simulated by a symmetric one iterated in parallel.

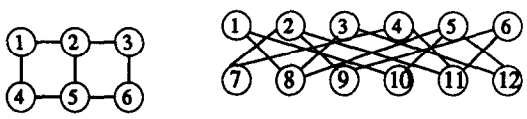

Fig. 4. Simulation of the parallel iteration by the sequential sequential one in the symmetric case.

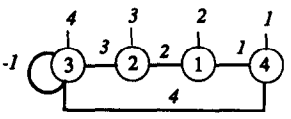

Fig. 5. A symmetric neural network.

Table 1
(i) Parallel iteration; cycle of period 2. (ii) Block-sequential iteration for $\mathscr{Y}=\{\{1\},\{2,3\},\{4\}\}$; cycle of period $T>2$. (iii) Sequential iteration; cycle of period $T>2$

(i) 3	2	1	4
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
0	1	0	1
1	0	1	0
0	1	0	1

(ii) 3	2	1	4
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1
1	0	0	1
1	0	0	1
1	0	0	1
0	1	0	1
0	1	0	0

(iii) 3	2	1	4
0	1	0	0
0	1	1	0
0	0	1	0
0	0	1	1
0	0	0	1
0	0	0	1
1	0	0	1
1	0	0	1
1	0	0	1
1	1	0	1
0	1	0	1
0	1	0	0

Further, previous results (i) and (iii) show that, in our sense, the symmetric and diagonal constraints on the neural networks implies that this kind of networks, iterated sequentially or parallel are not universal.

Suppose, for instance, the symmetric network given in Fig. 5. Clearly, its associated matrix is symmetric, so, from Proposition 2, the parallel iteration converges to fixed points or two cycles for any initial configuration. But since there exists a negative diagonal weight it is not the case for the sequential update $\mu=\{\{1\},\{2\},\{3\},\{4\}\}$ and neither for the partition $\mathscr{Y}=\{\{1\},\{2,3\},\{4\}\}$. In both cases there exists a cycle of period $T>2$, as we exhibit in Table 1.

The next result establishes that an arbitrary neural network can be simulated in real time and linear space by a symmetric one.

Proposition 3. *Let* $(W,b,\mathscr{Y},n)\in N(\mathscr{Y},n)$; *then there exists* $(W',b',\mathscr{Y}',3n)\in NS(\mathscr{Y}',3n)$ *such that* $(W',b',\mathscr{Y}',3n)$ *simulates* $(W,b,\mathscr{Y},n)$. *The simulation is carried out in* $3n$ *time.*

Proof. Let us take the interconnection scheme of two neurons 1,2 in $(W,b,\mathscr{Y},n)$ as shown in Fig. 6.

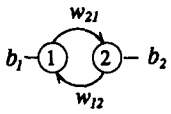

Fig. 6. Directed neural network.

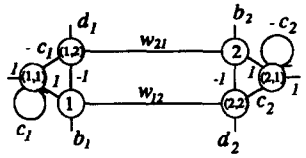

Fig. 7. Symmetric neural network.

We replace the previous scheme by the one given in Fig. 7, where $d(i)=\sum_{w_{ji}>0} w_{ji}+1$, $c(i)=2d(i)$.

So the new network corresponds to the previous one plus $2n$ neurons $\{(i,1),(i,2)\}_{i=1}^{n}$ connected as above.

Neurons $(i,1)$ and $(i,2)$ must be updated sequentially and they copy the state of neuron i. Suppose that $x_{(i,1)}=x_{(i,2)}$ and consider the argument of neuron $(i,1)$, $\sum(i,1)$ (the first one to be updated), so

$$\sum(i,1)=x_i-c(i)x_{(i,1)}+c(i)x_{(i,2)}-1=x_i-1,$$

hence $x'_{(i,1)}=\mathbb{1}(x_i-1)=x_i$. Now, one updates $(i,2)$:

$$\sum(i,2)=-x_i+c(i)x_{(i,1)}+\sum_{j\neq i} w_{ij}x_j-d(i).$$

Suppose first $x_i=0$, so, from definition of $d(i)$,

$$\sum(i,2)=\sum w_{ij}x_j-d(i)<0, \text{ hence } x'_{(i,2)}=\mathbb{1}(\sum(i,2))=0=x_i$$

If $x_i=1$, we get

$$\begin{aligned}\sum(i,2)&=c(i)-d(i)-1+\sum_{j\neq i} w_{ij}x_j\\&=d(i)-1+\sum_{j\neq i} w_{ji}x_j=\sum_{j\neq i}|w_{ij}|+1-1+\sum_{j\neq i} w_{ji}x_j\geqslant 0\end{aligned}$$

so, $x'_{i,2}=\mathbb{1}(\sum(i,2))=x_i=1$.

Hence, neurons $(i,1)$ and $(i,2)$ copy the current state of neuron i when we update them sequentially. Clearly, in the next update for i, states $x_{(i,1)}$ and $x_{(i,2)}$ will be not significant. Now to simulate $(W,b,\mathscr{Y},n)$; let x be the initial configuration of $(W,b,\mathscr{Y},n)$. We define the initial configuration, y, of $(W',b',\mathscr{Y}',3n)$ as follows: $y_i=x_i\ \forall i=1,\ldots,n$, $y_{(i,1)}==y_{(i,2)}(0)=x_i$. On the other hand, if $\mathscr{Y}=\{I_1,\ldots,I_p\}$, the new partition is the following:

$$\mathscr{Y}'=\{I_1,\{(i,1)\}_{i\in I_1},\{(i,2)\}_{i\in I_1},I_2,\ldots,I_p,\{(i,1)\}_{i\in I_p},\{(i,2)\}_{i\in I_p}\}.$$

It is direct that each application of the update on $(W,b,\mathscr{Y},n)$ is simulated by one update of $(W',b',\mathscr{Y}',3n)$. □

Corollary 1. *Consider* $\mathscr{N}=(W,b,\mu,n)\in N(\mu,n)$, *then there exists* $(W,b,\mu,3n)\in NS(\mu,3n)$ *with negative diagonal, i.e.* $(W,b,\mu,3n)\notin NS^{+}(\mu,3n)$, *which simulates* $\mathscr{N}$.

Proof. Direct from the previous theorem. It suffices to remark that in the construction of Theorem 1, the weights $w_{(i,1)(i,1)}$ are negatives. Thus, the new partition is given by $\{\{1\},\{(1,1)\},\{(1,2)\},\ldots,\{n\},\{(n,1)\},\{(n,2)\}\}$ □

The previous corollary proves that the sequential iteration on symmetric matrices with some negative diagonal entries is universal, i.e. it may simulate arbitrary neural networks. This fact gives an explanation to the non-negativity diagonal hypothesis assumed to determine the fixed point behavior of Hopfield networks [1, 5]. Furthermore, one may eliminate the negative diagonal entries by changing the iteration mode as in the following result.

Proposition 4. *Let $\mathcal{N}=(W,b,\mathscr{Y},n)\in NS(\mathscr{Y},n)\setminus NS^{+}(\mathscr{Y},n)$, then $\mathcal{N}$ can be simulated by a neural network $(W',b',\mathscr{Y}',2n)$ which belongs to $NS^{+}(\mathscr{Y}',2n)$.*

Proof. It suffices to change in the initial network $(W,b,\mathscr{Y},n)$, each neuron i, for two neurons $(i,1)$, $(i,2)$ such that

$$\forall i,j\in\{1,\ldots,n\}: w_{(i,1)(j,1)}=w_{(i,1)(j,2)}=\tfrac{w_{ij}}{2}w_{(i,2)(j,2)}=w_{(i,2)(j,1)}=\tfrac{w_{ij}}{2}b_{(i,1)}=b_{(i,2)}=b_i.$$

So, for any $x_{(i,1)},x_{(i,2)}$ with $x_{(i,1)}=x_{(i,2)}=x_i$, one gets

$$\begin{aligned}\textstyle\sum(i,1)&=\textstyle\sum_j w_{(i,1)(j,1)}x_{(j,1)}+\sum_j w_{(i,1)(j,2)}x_{(j,2)}-b_i\\&=\textstyle\sum_j(w_{(i,1)(j,1)}+w_{(i,1)(j,2)})x_j-b_i=\sum_j w_{ij}x_j-b_i.\end{aligned}$$

By taking the $x_{(i,1)}=x_{(i,2)}=x_i$ and $I_i'=\{(i,1),(i,2)\mid i\in I_i\}$, the element of the partition $\mathscr{Y}'$, we simulate the initial network by using $2n$ neurons and $\mathrm{diag}(W')=0$. □

Corollary 2. *The network $(W,b,\mathscr{Y},n)\in N(\mathscr{Y},n)$ can be simulated by $(W',b',\mathscr{Y}',6n)\in NS^{+}(\mathscr{Y}',6n)$.*

Proof. From Proposition 3, we first determine $(\tilde{W},\tilde{b},\tilde{\mathscr{Y}},3n)\in NS(\tilde{\mathscr{Y}},3n)$. From Proposition 4, we determine $(W',b',\mathscr{Y}',6n)\in NS^{+}(\mathscr{Y}',6n)$. □

In particular, the previous corollary shows that any neural network $(W,b,\pi,n)\in N(\pi,n)$ can be simulated by $(W',b',\mathscr{Y}',6n)\in NS^{+}(\mathscr{Y}',6n)$. It is important to point out that $\mathscr{Y}'$ is not a trivial partition, i.e. $\mathscr{Y}'\neq\phi$ and $\mathscr{Y}'\neq\mu$.

3. Neural networks with a refractory state

One of the alternative models of neural networks was proposed in [7]. It takes into account the fact that when a neuron fires there exists a time delay such that the neuron is inhibited to fire. In this section we study the simulation capabilities of such model for the parallel update, symmetric interconnections and only one refractory state.

A refractory neural network is defined by the tuple $\mathcal{N} = (W, b, Q, n)$ where the set of states Q is given by $Q = \{-1, 0, 1\}$. The dynamics of such network is synchronous and given by: for $y \in \{-1, 0, 1\}^n$,

$$G_i(y) = \begin{cases} 0 & \text{if } y_i = -1, \\ -1 & \text{if } y_i = 1, \\ \mathbb{1}\left(\sum_{j=1}^{n} w_{ij}\overline{y}_j - b_i\right) & \text{if } y_i = 0, \end{cases}$$

where $\overline{u} = 1$ iff $u = 1$ (0 otherwise).

We define the class of symmetric refractory neural networks of size $n \in \mathbb{N}$ as follows:

$$\mathrm{RNS}(n) = \{\mathcal{N} = (W, b, \{-1, 0, 1\}, n); W \text{ being symmetric}, n \in \mathbb{N}\}.$$

Our result in this section establishes that each neural network $\mathcal{A} \in N(\{\{1, 2, \ldots, n\}\}, n)$ (in the sequel denoted by $N(n)$) can be simulated by a neural network $\mathcal{B} \in RNS(3n)$. Recall that a classic neural network works only, as in previous section, with two states, $\{0, 1\}$.

Before proving the principal result, we show how asymmetric interactions in $\mathcal{A}$ can be simulated by symmetric ones in $\mathcal{B}$. Consider two neurons i and j in $\mathcal{A}$ as shown in Fig. 8(a).

The interactions between these neurons are simulated in network $\mathcal{B}$ by nine neurons: $i, n+i, 2n+i, j, n+j, 2n+j$ and a, b and c, connected in the way shown in Fig. 8(b).

Previous scheme can be generalized to any element in $N(n)$. In fact, given $\mathcal{A} \in N(n)$ we build $\mathcal{B} \in RNS(3n+3)$ as follows:

- Neuron $i \in \{1, \ldots, n\}$ in $\mathcal{B}$ play the role of the neurons in $\mathcal{A}$.
- Neuron $n+i$, $i \in \{1, \ldots, n\}$ will be used to copy the state of neurons i and to delay the transmission between i and j.
- Neuron $2n+i$, $i \in \{1, \ldots, n\}$ produce a copy of the state of neuron i which is sent to all neurons j, $j \in \{1, \ldots, n\}$ with weight w_{ji}.
- Neurons a, b and c belong to a cycle sending every three steps the value $1 - b_i$ to neuron i. This cycle allows to neuron i to reproduce its threshold each three steps.

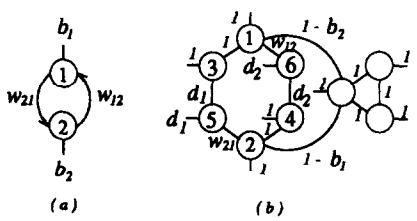

Fig. 8. Schema of how a neural network $\mathcal{A} \in N(2)$ is simulated by a neural network $\mathcal{B} \in RNS(9)$. (a) neural network $\mathcal{A}$ without loops and non-necessarily symmetric; (b) refractory symmetric neural network $\mathcal{B}$ simulating to $\mathcal{A}$.

The non-zero elements in the matrix connection U of $\mathscr{B}$ are

$$\begin{aligned} &U_{i,n+i} = U_{n+i,i} = 1, \qquad U_{n+i,2n+i} = U_{2n+i,n+i} = d_i, \\ &U_{j,2n+i} = U_{2n+i,j} = w_{ji}, \qquad U_{i,a} = U_{a,i} = 1 - b_i, \\ &U_{a,b} = U_{b,a} = U_{b,c} = U_{c,b} = U_{a,c} = U_{c,a} = 1, \end{aligned} \tag{1}$$

where $d_i = \sum_{w_{ji}>0} w_{ji} + 1$. The threshold vector v of $\mathscr{B}$ is given by

$$v_i = v_{n+i} = v_a = v_b = v_c = 1, \quad i = 1,\ldots,n \quad \text{and} \quad v_{2n+i} = d_i \text{ for } i = 1,\ldots,n.$$

Notice that the neighborhoods in $\mathscr{B}$ are given by

$$\begin{aligned} &V_i = \{n+i\} \cup \{2n+j : \ j = 1,\ldots,n\}, \qquad V_{n+i} = \{n+i, i\}, \\ &V_{2n+i} = \{n+i\} \cup \{j : \ j = 1,\ldots,n\}, \\ &V_a = \{b,c\} \cup \{i : \ i = 1,\ldots,n\}, \qquad V_b = \{a,c\} \quad V_c = \{a,b\}. \end{aligned}$$

Now we describe how the evolution in $\mathscr{A}$ is simulated in $\mathscr{B}$. We define the map $\eta : \{0,1\}^n \to \{-1,0,1\}^{3n+3}$ by

$$\eta_k(x) = \begin{cases} \phi(x_i), & k = i, \\ x_i, & k = n+i, \\ 0, & k = 2n+i, \end{cases} \qquad \eta_u(x) = \begin{cases} 0, & u = a, \\ 1, & u = b, \\ -1, & u = c. \end{cases}$$

where

$$\phi(u) = \begin{cases} -1, & u = 1, \\ 0, & u = 0. \end{cases}$$

Then, η maps global configurations of $\mathscr{A}$ into global configurations in $\mathscr{B}$.

In Fig. 9 we describe the dynamical evolution of $\mathscr{A}$ together with those of $\mathscr{B}$, when we take $x = (1,0)$ as the global state of $\mathscr{A}$. Then, the global state of $\mathscr{B}$ is given by $(-1,0,1,0,0,0,0,1,-1)$. Now, we prove our result.

Theorem 1. *For any $A \in N(n)$ there exists $\mathscr{B} \in RNS(3n+3)$ built as above such that*

$$\forall x \in \{0,1\}^n \quad \forall t \geqslant 0 \quad \eta(F^t(x)) = G^{3t}(\eta(x)).$$

Proof. We only need to prove it for $t = 1$.

Let $y = \eta(x)$ and $\delta_i(y) = \sum_{l \in V_i} u_{il} y_l - d_i$. Then, from definition of U given in Eq. (1), we get

$$\delta_{n+k}(y) = \bar{y}_k + 2\rho_k \bar{y}_{2n+k} - 1, \quad \delta_{2n+k}(y) = 2\rho_k \bar{y}_{n+k} + \sum_{l=1}^{n} w_{lk} \bar{y}_l - \rho_k,$$

$$\delta_k(y) = \bar{y}_{n+k} + \sum_{l=1}^{n} w_{kl} \bar{y}_{2n+k} + (1 - b_k)\bar{y}_a - 1,$$

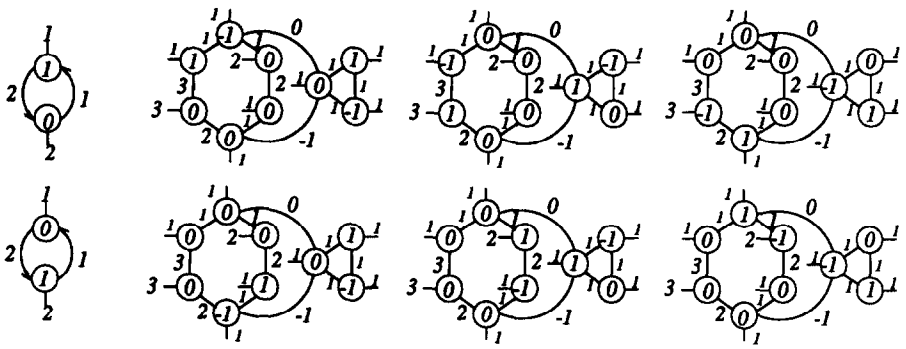

Fig. 9. Refractory symmetric neural network simulating a neural network. Each step in $\mathscr{A}$ is simulated in three steps in $\mathscr{B}$.

$$\delta_a(y) = \sum_{l=1}^{n}(1 - b_l)\bar{y}_l + \bar{y}_b + \bar{y}_c - 1,$$

$$\delta_b(y) = \bar{y}_a + \bar{y}_c - 1, \quad \delta_c(y) = \bar{y}_a + \bar{y}_b - 1.$$

Computation of $G(\eta(x))$.

Let $k \in \{1,\dots,n\}$. we must split our computation according to $x_k = 0$ or $x_k \neq 0$. Assume $x_k = 0$. Then, $y_k = 0 = y_{n+k}$. Therefore,

$$G_k(y) = 0, G_{n+k}(y) = -1, G_{2n+k}(y) = \mathbb{1}\left(\sum_{l=1}^{n} w_{kl}\bar{y}_l - \rho_k\right) = 0.$$

Now, suppose $x_k = 1$. Then $y_k = -1$ and $y_{n+k} = 1$. Thus,

$$G_k(y) = 0, G_{n+k}(y) = -1, G_{2n+k}(y) = \mathbb{1}\left(\rho_k + \sum_{l=1}^{n} w_{kl}\bar{y}_l\right) = 1.$$

Moreover, since $\overline{\phi(x_l)} = 0$ we obtain $G_a(y) = 1$. Finally, $G_b(y) = -1$ and $G_c(y) = 0$. So,

$$G_k(y) = \begin{cases} 0, & k = i, \\ \phi(x_i), & k = n + i, \\ x_i, & k = 2n + i, \end{cases} \qquad G_u(y) = \begin{cases} 1, & k = a, \\ -1, & k = b, \\ 0, & k = c. \end{cases}$$

To compute $G^2(y)$ we use the previous formula for $z = G(y)$. Since $G_k(y) = z_k = 0$ we have

$$G_k(z) = \mathbb{1}\left(\overline{\phi(x_k)} + \sum_{l=1}^{n} w_{kl}x_l + 1 - b_l - 1\right) = \mathbb{1}\left(\sum_{l=1}^{n} w_{kl}x_l - b_l\right) = F_k(x).$$

Assume $x_k = 0$. Then $z_k = z_{n+k} = z_{2n+k} = 0$ which implies $G_{n+k}(z) = 0$ and

$$G_{2n+k}(z) = \mathbb{1}\left(2\rho\phi(\bar{x}_k) + \sum_{l=1}^{n} w_{kl} \cdot 0 - \rho_k\right) = 0.$$

Now, suppose $x_k = 1$. Then $z_{n+k} = -1$ which implies $G_{n+k}(z) = 0$ and $G_{2n+k} = -1$. Clearly, $G_a(z) = -1$, $G_b(z) = 0$ and $G_c(z) = 1$. Thus,

$$G_k(z) = \begin{cases} F_k(x), & k = i, \\ 0, & k = n + i, \\ \phi(x_i), & k = 2n + i, \end{cases} \qquad G_u(z) = \begin{cases} -1, & k = a, \\ 0, & k = b, \\ 1, & k = c. \end{cases}$$

Finally, $G^3(y)$ is given by

$$G_k^3(y) = \begin{cases} \phi(F_k(x)), & k = i, \\ F_k(x), & k = n + i, \\ 0, & k = 2n + i, \end{cases} \qquad G_u(z) = \begin{cases} 0, & k = a, \\ 1, & k = b, \\ -1, & k = c. \end{cases} \quad \square$$

4. Conclusions

We have proved that arbitrary neural networks can be simulated in linear space ($0(n)$ neurons) by symmetric neural nets, iterated in a block-sequential mode. This implies that symmetry is not enough to insure simple dynamics of neural network. The dynamics depends strongly on the iteration mode.

Also, our result proved that when the non-negative diagonal hypothesis for sequential iteration does not hold, the class is universal, i.e. symmetric neural networks with negative diagonal entries iterated sequentially may simulate arbitrary neural nets. Similarly, the class of block-sequential iterations on symmetric neural nets which does not verify the hypothesis of non-negative definite submatrices, is also universal. Finally, we proved that symmetric refractory neural nets with only three states are universal in the sense they simulate any arbitrary classical two-states neural net.

References

[1] E. Goles, Fixed point behavior of threshold functions on a finite set, *SIAM J. Algebra Discrete Methods* **3** (1982) 529–531.

[2] E. Goles, F. Fogelman-Soulié and D. Pellegrin, The energy as a tool for the study of threshold networks, *Discrete Appl. Math.* **12** (1985) 261–277.

[3] E. Goles and S. Martínez, Neural and Automata Networks (Kluwer, Dortrecht, 1990).

[4] E. Goles and J. Olivos, Periodic behavior of generalized threshold functions, *Discrete Math.* **30** (1980) 187–189.

[5] J.J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, *Proc. Natl. Acad. Sci. USA* **79** (1982) 2554–2558.

[6] W. McCulloch and W. Pitts, A logical calculus of the ideas immanent in nervous activity, *Bull. Math. Biophys.* **5** (1943) 115–133.

[7] M. Matamala, Complejidad y dinámica de redes de autómatas, Ph.D. Thesis, Universidad de Chile, 1994.

[8] P. Orponen, Complexity classes in neural networks, in: *Proc. 20th Internat. Coll. on Automata, Language, and Programming*, Lecture Notes in Computer Science, Vol. 700 (Springer, Berlin, 19xx) 215–226.

[9] S. Shingai, Maximum period of 2-dimensional uniform neural network, *Inform and Control* **41** (1979) 324–341.