Neural Processing Letters **18:** 213–218, 2003.

A Solution for the *N*-bit Parity Problem Using a Single Translated Multiplicative Neuron

EDUARDO MASATO IYODA, HAJIME NOBUHARA and KAORU HIROTA
Department of Computational Intelligence and Systems Science, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama 226-8502, Japan. e-mail: {iyoda, nobuhara, hirota}@hrt.dis.titech.ac.jp

Abstract. A solution to the N-bit parity problem employing a single multiplicative neuron model, called translated multiplicative neuron (π_t-neuron), is proposed. The π_t-neuron presents the following advantages: (a) $\forall N \geqslant 1$, only 1 π_t-neuron is necessary, with a threshold activation function and parameters defined within a specific interval; (b) no learning procedures are required; and (c) the computational cost is the same as the one associated with a simple McCulloch-Pitts neuron. Therefore, the π_t-neuron solution to the N-bit parity problem has the lowest computational cost among the neural solutions presented to date.

Key words. multiplicative neurons, N-bit parity problem, neural networks

1. Introduction

The N-bit parity problem is a challenging benchmark for testing neural network architectures and their learning algorithms. Besides being non-linear, the N-bit parity problem is difficult to solve, because changing only 1 bit in the input causes the output to change.

The N-bit parity problem can be stated as follows. Let $\mathbf{x} = [x_1, \ldots, x_N]^T$ be an N-bit binary vector, i.e., $x_i \in \{0, 1\}$ $(i = 1, \ldots, N)$. The *parity generator function* $p: \{0, 1\}^N \rightarrow \{0, 1\}$ is defined by

$$p(\mathbf{x}) = \begin{cases} 0, & \text{if } \sum_{i=1}^{N} x_i \text{ is even} \\ 1, & \text{otherwise.} \end{cases} \tag{1}$$

The objective is to design a neural network capable of realizing the function (1). Note that, for $N = 1$, we have simply $p(\mathbf{x}) = \mathbf{x}$ and for $N = 2$, the parity problem is equivalent to the XOR problem.

Many neural network approaches have been proposed to solve the N-bit parity problem, e.g., [1, 2, 6–8, 10–12]. Most of these works solve the parity problem using specialized activation functions or specialized network topologies or both. All these previous solutions, however, require the use of at least 1 hidden neuron to solve the parity problem. Moreover, in some of these approaches, the number of hidden neurons required to solve the problem increases with N. In fact, a solution using only 1 neuron has been presented by Arslanov et al. [1], but they themselves reject this solution, because it uses a complicated activation function. Finally, all these previous solutions are based on networks composed of McCulloch-Pitts neurons, which employ additive composition to aggregate their input signals.

Unlike the previous approaches, we propose a solution based on an extended multiplicative neuron model, called translated multiplicative neuron, or π_t-neuron for short [5]. We show that only 1 π_t-neuron, employing threshold activation function and parameters defined in certain intervals, solves the N-bit parity problem, $\forall N \geqslant 1$. Additionally, no learning procedure is necessary to obtain a solution. Furthermore, the computational complexity of a π_t-neuron is the same as the one associated to a McCulloch-Pitts neuron, if both employ the same activation function. Consequently, the proposed approach has the lowest computational complexity among neural solutions presented so far.

In Section 2, π_t-neuron and its properties are presented, and in Section 3, a solution for the N-bit parity problem using a π_t-neuron is proposed.

2. Translated Multiplicative Neuron (π_t-neuron)

Multiplicative neuron models are mainly employed in high-order neural networks [3] and in hybrid neural architectures [4, 13]. Although several multiplicative neurons have already been proposed [9], we initially consider a particular model, called *product* or *multiplicative neuron* (π-neuron) [13]. This model is defined by the following equations

$$v = \prod_{i=1}^{m} w_i x_i, \quad y = f(v), \tag{2}$$

where $x_i \in \mathbb{R}$ $(i = 1, \ldots, m)$ are the neuron's inputs, $w_i \in \mathbb{R}$ $(i = 1, \ldots, m)$ are the adjustable parameters (weights) of the model, v is the *level of internal activity*, $f\colon \mathbb{R} \to \mathbb{R}$ is the neuron's *activation function*, and y is the output of the model.

Even though the model (2) is successfully used in some hybrid neural network architectures [4, 13], it has disadvantages. First, note that v in (2) can be rewritten as

$$v = \prod_{i=1}^{m} w_i \prod_{j=1}^{m} x_j = c \prod_{j=1}^{m} x_j,$$

where $c = \prod_{i=1}^{m} w_i$, i.e., m parameters are used simply to compose a scaling factor for v. Because learning algorithms usually try to adjust all the m parameters, precious computational resources are wasted. Furthermore, the decision surfaces generated by (2) are always centered in the origin of the neuron's input space.

To overcome the drawbacks of the π-neuron, an extended multiplicative neuron model, called π_t-neuron, has been proposed [5]. This model is defined by the following equations

$$v = b\prod_{i=1}^{m}(x_i - t_i), \quad y = f(v), \tag{3}$$

where $b \in \mathbb{R}$ and $t_i \in \mathbb{R}$ $(i = 1, \ldots, m)$ are the adjustable parameters of the neuron. The adjustable parameters of (3) have a clear meaning, i.e., b is a scaling factor for v, and t_i's are the coordinates of the center of the decision surfaces generated by (3). Note that the center of π_t-neuron's decision surfaces can be placed anywhere in the neuron's input space.

In the traditional McCulloch-Pitts neuron, the level of internal activity is usually defined as $v_{\mathrm{mc}} = w_0 + \sum_{i=1}^{m} w_i x_i$, where w_0 is the bias term, and the output is given by $y_{\mathrm{mc}} = f(v_{\mathrm{mc}})$. Comparing the equations that define the McCulloch-Pitts neuron with (3), we see that π_t-neuron has the same number of parameters and performs the same number of arithmetical operations as the McCulloch-Pitts model. In other words, a π_t-neuron and a McCulloch-Pitts neuron have the same computational complexity, if both employ the same activation function.

3. Solving the *N*-bit Parity Problem with a Single π_t-neuron

Consider a π_t-neuron employing the threshold activation function $f_{\mathrm{th}}: \mathbb{R} \to \{0, 1\}$, defined by

$$f_{\mathrm{th}}(v) = \begin{cases} 1, & \text{if } v \geqslant 0 \\ 0, & \text{otherwise.} \end{cases}$$

We prove in the following that this π_t-neuron is capable of solving the N-bit parity problem, $\forall N \geqslant 1$.

LEMMA 1. *Let a π_t-neuron employing threshold activation function have its parameters defined as $0 < t_i < 1$ $(i = 1, \ldots, N)$ and $b < 0$ if N is even or $b > 0$ if N is odd. Then this π_t-neuron solves the N-bit parity problem, $\forall N \geqslant 1$.*

Proof. Let $\mathbf{x} = [x_1, \ldots, x_N]^T$ be a binary vector and a π_t-neuron employing threshold activation function have its parameters defined as stated in the lemma. We shall prove that the output of this π_t-neuron, when excited by $\mathbf{x}$, is $y = p(\mathbf{x})$, $\forall \mathbf{x} \in \{0, 1\}^N$. Define $S_0 = \{i \mid x_i = 0\}$ and $S_1 = \{j \mid x_j = 1\}$. Observe that $N = |S_0| + |S_1|$, where $|\cdot|$ denotes cardinality of a set. The level of internal activity of the π_t-neuron, excited by $\mathbf{x}$, is given by

$$v = b\prod_{i=1}^{N}(x_i - t_i) = b\prod_{i \in S_0}(x_i - t_i)\prod_{j \in S_1}(x_j - t_j) = bK_0K_1,$$

where $K_0 = \prod_{i \in S_0}(x_i - t_i)$ and $K_1 = \prod_{j \in S_1}(x_j - t_j)$. We adopt the following convention: for $k \in \{0, 1\}$, if $S_k = \emptyset$ then $\prod_{l \in S_k}(x_l - t_l) = 1$. The output of the π_t-neuron is

$$y = f_{\text{th}}(v).$$

Note that $K_1 > 0$, because $(x_j - t_j) > 0, \forall j \in S_1$. Since $(x_i - t_i) < 0, \forall i \in S_0$, the sign of K_0 and consequently, the sign of $K_0 K_1$, is determined by $|S_0|$: if $|S_0|$ is even then $K_0 K_1 > 0$; if $|S_0|$ is odd then $K_0 K_1 < 0$. We consider 2 different cases:

1. N even: in this case, $|S_0|$ is even iff $|S_1|$ is even. Since $b < 0$, if $|S_0|$ is even then $v < 0$; if $|S_0|$ is odd then $v > 0$. Consequently, the π_t-neuron's output is given by

   $$y = \begin{cases} 0, & \text{if } |S_1| \text{ is even} \\ 1, & \text{otherwise,} \end{cases}$$

   i.e., for N even, $N \geqslant 2, y = p(\mathbf{x}), \forall \mathbf{x} \in \{0, 1\}^N$.

2. N odd: here, $|S_0|$ is even iff $|S_1|$ is odd. Because $b > 0$, if $|S_0|$ is odd then $v < 0$; if $|S_0|$ is even then $v > 0$. Then the π_t-neuron's output is given by

   $$y = \begin{cases} 0, & \text{if } |S_1| \text{ is even} \\ 1, & \text{otherwise,} \end{cases}$$

   i.e., for N odd, $N \geqslant 1, y = p(\mathbf{x}), \forall \mathbf{x} \in \{0, 1\}^N$.

Hence, the considered π_t-neuron solves the N-bit parity problem, $\forall N \geqslant 1$.

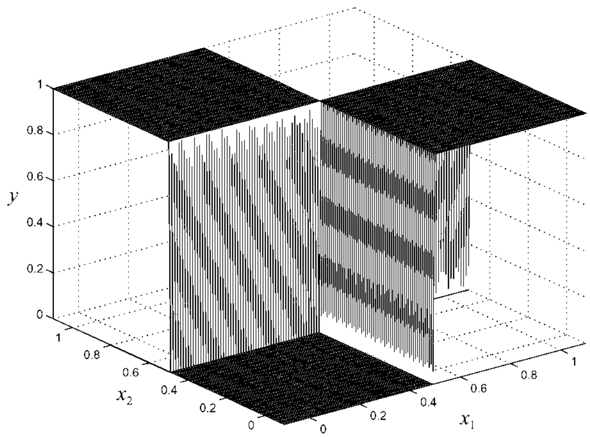
Note that the conditions (parameter values) stated in Lemma 1 are sufficient conditions only. Moreover, there are infinite parameter values that can be used to solve the N-bit parity problem with a π_t-neuron. The choice of appropriate values for the parameters depends on the restrictions imposed by the particular implementation architecture used. If some performance measure related to the implementation architecture must be optimized, the parameter intervals suggested in Lemma 1 can be used as constraints for the corresponding optimization problem. See Figure 1 for a solution for the 2-bit parity (XOR) problem, with $b = -1$ and $t_1 = t_2 = 0.5$.

Table I presents a comparison among neural architectures proposed to solve parity problems. In this table, $[\cdot]$ stands for truncation to the nearest integer, $\lceil \cdot \rceil$ stands for rounding toward $+\infty$, and $\lfloor \cdot \rfloor$ stands for rounding toward $-\infty$. Observe that the solution using π_t-neuron has the lowest computational complexity, since it does not require hidden neurons and, again, the π_t-neuron solution has the same computational complexity as that of a simple McCulloch-Pitts neuron.

As pointed out before, Arslanov et al. [1] have presented a solution (different from that showed in Table I) that uses only 1 additive neuron with a complicated activation function. Since Arslanov et al. themselves reject that solution, it is not included in Table I. Anyway, because the π_t-neuron solution employs a simple threshold activation function, it is still less computationally complex than Arslanov et al.'s rejected one.

x_1	x_2	$p(x)$
0	0	0
0	1	1
1	0	1
1	1	0

(a)



(b)

Figure 1. Solution obtained by a π_t-neuron for the 2-bit parity (XOR) problem: (a) XOR truth table; (b) decision surface generated by π_t-neuron.

Table I. Comparison between neural architectures for the N-bit parity problem.

Solution	Hidden neurons	Activation function
Stork and Allen [11]	2	Specialized
Brown [2]	1	Specialized
Minor [8]	$[N/2]$	Sigmoid
Setiono [10]	$\lceil (N+1)/2 \rceil$	Sigmoid
Lavretsky [6]	$N-1$	Sigmoid/Threshold
Liu et al. [7]	$\lfloor N/2 \rfloor$	Threshold
Arslanov et al. [1]	$\lceil \log_2(N+1) \rceil$	Threshold
Torres-Moreno et al. [12]	N	Sigmoid
π_t-neuron	0	Threshold

4. Conclusion

The N-bit parity problem has been widely used to evaluate neural networks, because it is nonlinear and considered hard to solve. A variety of neural architectures have been proposed to solve the parity problem, but all of them require the use of at least one hidden neuron. We propose a solution to the N-bit parity problem based on an extended multiplicative neuron model, called translated multiplicative neuron (π_t-neuron). The π_t-neuron solution does not require learning and, $\forall N \geqslant 1$, only 1 π_t-neuron with threshold activation function and parameters defined within a specific interval solves the N-bit parity problem. Furthermore, since 1 π_t-neuron has the same computational complexity as a single McCulloch-Pitts neuron (if both use the same activation function), the proposed solution has the lowest computational cost among the neural solutions reported to date.

The simplicity of the proposed solution makes it suitable for applications demanding fast computation of parity bits. The π_t-neuron solution may also be attractive for those interested in hardware implementations, because compact analog parity generator circuits may be developed based on it.

The applicability of π_t-neuron is not limited to the N-bit parity problem. Actually, neural networks composed of hidden π_t-neurons and trained by supervised learning techniques have been applied in function approximation problems, producing encouraging results [5]. A future research direction is to investigate theoretical properties and limitations of neural networks using π_t-neurons. Another research direction is to insert π_t-neuron in the context of hybrid and automatic generated neural network architectures [4, 13].

Acknowledgements

The authors thank Prof. Fernando J. Von Zuben, from State University of Campinas, Brazil, for the valuable comments.

References

1. Arslanov, M. Z., Ashigaliev, D. U. and Ismail, E. E.: *N*-bit parity ordered neural networks, *Neurocomputing* **48** (2002), 1053–1056.
2. Brown, D. A.: *N*-bit parity networks, *Neural Networks* **6** (1993), 607–608.
3. Giles, C. L. and Maxwell, T.: Learning, invariance, and generalization in high-order neural networks, *Applied Optics* **26**(23) (1987), 4972–4978.
4. Iyoda, E. M., Hirota, K. and Von Zuben, F. J.: Sigma-Pi cascade extended hybrid neural networks, *Journal of Advanced Computational Intelligence* **6**(3) (2002), 126–134.
5. Iyoda, E. M., Nobuhara, H. and Hirota, K.: Translated multiplicative neuron: An extended multiplicative neuron that can translate decision surfaces, In: *Proceedings of the IEEE International Conference on Computational Cybernetics*, Siófok, Hungary, 2003.
6. Lavretsky, E.: On the exact solution of the Parity-*N* problem using ordered neural networks, *Neural Networks* **13**(6) (2000), 643–649.
7. Liu, D., Hohil, M. E. and Smith, S. H.: *N*-bit parity neural networks: new solutions based on linear programming, *Neurocomputing* **48** (2002), 477–488.
8. Minor, J. M.: Parity with two layer feedforward nets, *Neural Networks* **6**(5) (1993), 705–707.
9. Schmitt, M.: On the complexity of computing and learning with multiplicative neurons, *Neural Computating* **14**(2) (2002), 241–301.
10. Setiono, R.: On the solution of the parity problem by a single hidden layer feedforward neural network, *Neurocomputing* **16**(3) (1997), 225–235.
11. Stork, D. G. and Allen, J. D.: How to solve the *N*-bit parity problem with two hidden units, *Neural Networks* **5**(2) (1992), 923–926.
12. Torres-Moreno, J. M., Aguilar, J. C. and Gordon, M. B.: The minimum number of errors in the *N*-parity and its solution with an incremental neural network, *Neural Processing Letters* **16**(3) (2002), 201–210.
13. Zhang, B.-T.: A Bayesian evolutionary approach to the design and learning of heterogeneous neural trees, *Integrated Computer-Aided Engineering* **9**(1) (2002), 73–86.