Single Layer Morphological Perceptron Solution to the N-Bit Parity Problem

Gonzalo Urcid[1,⋆], Gerhard X. Ritter[2], and Laurentiu Iancu[2]

[1] Optics Department, INAOE, Tonantzintla, Pue. 72000, Mexico
gurcid@inaoep.mx

[2] CISE Department, University of Florida, Gainesville, FL 32611–6120, USA
ritter,liancu@cise.ufl.edu

Abstract. Morphological perceptrons use a lattice algebra approach to learn and classify a set of patterns. Dendritic structure combined with lattice algebra operations have properties that are completely different than those of traditional perceptron models. In the present paper, we focus our attention in *single layer morphological perceptrons* that classify correctly the parity of all bit strings of length n, as a *one-class* pattern recognition problem. The n-bit parity problem is the n-dimensional extension of the classic XOR problem in the Euclidean plane and is commonly used as a difficult benchmark to test the performance of training algorithms in artificial neural networks. We present results for values of n up to 10, obtained with a training algorithm based on elimination.

1 Introduction

The n-bit *parity problem* is defined as follows, given a binary n-dimensional input vector, $\boldsymbol{x} = (x_1, \ldots, x_n)$, the parity is 1 if the number of 1's in $\boldsymbol{x}$ is *odd*, otherwise the parity is 0. Arithmetically, the parity equals $(x_1 + \cdots + x_n) \mod 2$. The parity problem, categorized as a statistical neutral problem [1], is known to be a "hard" learning benchmark for neural network classifiers and has been the subject of considerable research and experimentation [2–4]. In [4], a single hidden layer feedforward neural network with $(n/2)+1$ hidden units for even n, or $(n+1)/2$ hidden units for odd n, without direct connections, and sigmoids for hidden and output units, correctly classifies all 2^n input patterns. The weights of the network are explicitly computed, e.g., the weights between the hidden and output layers are found by solving a system of $h \times h$ linear equations where h is the number of hidden units. For $n \in \{3, \ldots, 7\}$, training experiments by gradient type algorithms using a single or two hidden layer topology with a variable number of hidden units in each layer are reported in [5–7]. Another network architecture based on the majority algorithm [8], solves the parity problem with a $n : n : 1$ topology for n odd, or a $(n+1) : n : 1$ topology for n even. The network weights equal ± 1 only, it has no direct connections between the input and output layers, and requires n^2 connections from the input layer to the hidden

⋆ Corresponding author. Fax: +52 (222) 247-2940; Tel.: +52 (222) 266-3100 Ext.8205 layer. Stacked vs. cross-validation generalization performance of classifiers have been addressed in [9]; for $n = 9, 13$, generalization accuracy improves rapidly if the number of training exemplars exceeds one third of the entire class, and $\sin(x)$ is used as the activation function for the nodes in the hidden layers. In [10], a reduced number of iterations during the training phase is accomplished by modifying the performance index or the activation function slope used in the Levenberg-Marquardt (LM) optimization technique; a comparison against LM is made for $n = 2, 3, 4$ with $2, \{2, 3\}, 6$ hidden units respectively. A different approach used for 3–8 bit parity problems [11], uses a network that adds a hidden neuron to its hidden layer when several consecutive attempts failed to escape from a local minimum using standard LM. Therefore, neural networks with the least number of hidden units are constructed in agreement with the theoretical results stated in [2, 4].

Recently, the foundation of *morphological perceptrons* (MPs) with dendritic structure was established in [12–16] as a new paradigm in machine learning. It was proved that a single layer morphological perceptron (SLMP) with one output neuron can approximate, to any desired degree of accuracy, any compact set of points in pattern space, whether it is convex or non-convex, connected or not connected, or contains a finite or infinite number of points [13]. Specifically, SLMPs were built to solve the parity problem for $n = 2$ (XOR) and $n = 3$; this paper gives the solution to the general case using SLMPs training by elimination [13, 14].

Our work is organized as follows: Section 2 gives a brief background of single neuron computation based on lattice algebra and describes the basic architecture of an SLMP. Section 3 outlines the SLMP training algorithm based on elimination and Section 4 presents numerical results obtained for values of n up to 10, as well as comments about the neural architecture and its performance. Finally, in Section 5 we give our conclusion to the research presented here.

2 Morphological Perceptrons with Dendrites

Computation at a neurode M in the classical theory of artificial neural networks (ANNs) is performed within the *ring* of real numbers $(\mathbb{R}, +.\times)$, by adding the products of neural values and connection weights from all input neurons connected to M, followed by the application of a nonlinear activation function. Morphological neurocomputation is performed using the *bounded lattice group* $(\mathbb{R}_{\pm\infty}, \wedge, \vee, +)$ where $\mathbb{R}_{\pm\infty}$ is the extended real number system, $\wedge = \min$, $\vee = \max$, and $+$ is addition. Therefore, the output from neuron M is computed as the minimum or maximum of the sums of neural values and corresponding synaptic weights, and it is nonlinear before application of an activation function. To bear a closer resemblance with biological neurons and their processes, *artificial dendrites* with excitatory and inhibitory responses are incorporated in morphological neurons to perform logical computations as suggested by recent research in the biophysics of real neurons (see References in [13]).

Let $N_1, \ldots, N_n$ denote a set of input neurons with dendrites. Assume these neurons provide synaptic input at an output neuron M also with dendrites. The value of the neuron N_i travels along its axonal tree until it reaches the synaptic knobs that make contact with the dendrites of neuron M. The weight w_{ik}^{λ} is associated to the axonal branch of the neuron N_i terminating on the kth dendrite of M; $\lambda = 0$ represents *inhibitory* input, and $\lambda = 1$ represents *excitatory* input to the dendrite. The kth dendrite of M will respond to the total input received from the neurons $N_1, \ldots, N_n$ and will either reject or pass the input. The value computed at the kth dendrite of M, for input $\boldsymbol{x} \in \mathbb{R}^n$ is given by

$$\tau_k(\boldsymbol{x}) = p_k \bigwedge_{i \in I(k)} \bigwedge_{\lambda \in L(i)} (-1)^{(1-\lambda)}(x_i + w_{ik}^{\lambda}), \tag{1}$$

where x_i is the value of neuron N_i, $I(k) \subseteq \{1, \ldots, n\}$ is the index set of input neurons with terminal knobs that synapse on the kth dendrite of M, $L_i \subseteq \{0, 1\}$ denotes the two only possible types of synapses the input N_i may have on dendrite k of M, and $p_k \in \{-1, 1\}$ signals if the kth dendrite of M, inhibits ($p_k = -1$) or accepts ($p_k = 1$) the received input. In (1), note that, if $\lambda = 0$, then the input $-(x_i + w_{ik}^0)$ is inhibitory, and excitatory for $\lambda = 1$ since in this case we have $(x_i + w_{ik}^1)$.

The total value received by M is computed as the minimum of $\tau_k(\boldsymbol{x})$ for all $k = 1, \ldots, K$ where K denotes the total number of dendrites of M and its next state is determined by a Heaviside type hard limiter f. Before application of the activation function f, neuron M has its own inhibitory ($p = -1$), or excitatory response ($p = 1$). The final output value of the morphological neuron M is therefore given by

$$y(\boldsymbol{x}) = f\left(p \bigwedge_{k=1}^{K} \left[p_k \bigwedge_{i \in I(k)} \bigwedge_{\lambda \in L(i)} (-1)^{(1-\lambda)}(x_i + w_{ik}^{\lambda})\right]\right). \tag{2}$$

A *single layer morphological perceptron* (SLMP) with one output neuron, depicted in Fig. 1, is a morphological neuron endowed with a finite number of dendrites and n input neurons that follows the propagation rule (2). For the n parity problem, we restrict the pattern space to the discrete boolean space $\{0, 1\}^n$ of n-dimensional binary vectors.

3 SLMP Training Algorithm

The architecture of an SLMP is *not* predetermined beforehand. It is during the training phase that the morphological neuron grows new dendrites while the input neurons expand their axonal branches to synapse on the new dendrites to learn the training patterns. The algorithm proposed in [13] is based on elimination of misclassified patterns; basically, an initial hyperbox containing all patterns is reduced through elimination of foreign patterns and smaller regions that enclose them. Training ends when all foreign patterns in the training set have been removed. Removal is performed by computing the intersection of the regions recognized by the grown dendrites, as expressed by the total input value to neuron M, i.e., by $\tau(\boldsymbol{x}) = p_k \bigwedge_{k=1}^{K} \tau_k(\boldsymbol{x})$.

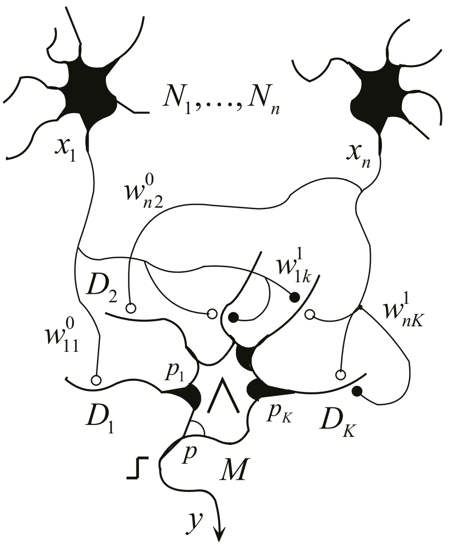

Fig. 1. SLMP with n input neurons N_i and one output neuron M

The SLMP training by elimination algorithm is outlined below and its mathematical description with more detailed steps can be found in [13]. The algorithm builds and trains an SLMP to recognize the training patterns as either belonging to class C_1 (odd parity) or not belonging to it. Hence it solves a *one-class* problem, where the class of interest is denoted by C_1 and the rest of points in pattern space by C_0 (even parity). In the present study, $C_1 \cup C_0 = \{0, 1\}^n$.

Algorithm 3.1. (SLMP training by elimination [17].)

```
STEP 1. Grow a first dendrite, $D_1$, that recognizes the hyperbox enclosing all
        patterns labeled class $C_1$. This dendrite is excitatory. Initialize the
        dendrite counter $K = 1$.
STEP 2. Using the $K$ dendrites grown thus far, $D_1, \ldots, D_k$, use (2) to compute
        the output of the perceptron for each pattern in the training set.
STEP 3. If all training patterns are correctly classified, STOP. Otherwise,
        increment $K$ and grow another dendrite, $D_k$. This dendrite will be
        inhibitory.
STEP 4. Select a pattern in class $C_0$ that is erroneously classified as belonging
        to class $C_1$.
STEP 5. Find a region enclosing the pattern selected in STEP 4 such that this
        region may also contain other patterns from $C_0$ but not from $C_1$.
STEP 6. Assign weights and responses to make dendrite $D_K$ recognize the region
        determined in STEP 5.
STEP 7. Repeat from STEP 2.
```

The separation surfaces drawn in pattern space during training are always closed and the trained SLMP will always correctly recognize 100% of the patterns in the training set. Thus, the SLMP procedure based on lattice computation with dendrites provides a competitive learning alternative compared with other multilayer perceptron architectures mentioned in the Introduction [4, 8, 11].

4 Numerical Results and Performance

Each training pattern set for the n parity problem was formed as an augmented matrix T with 2^n rows and $n + 1$ columns, by adjoining the class vector C to the binary matrix B of patterns using the following expressions

$$\begin{aligned} B_{ij} &= \text{mod}\,(\lfloor (i-1)2^{j-n} \rfloor, 2), & (3) \\ C_i &= \text{mod}\,(B_{i1} + \cdots + B_{in}, 2), & (4) \end{aligned}$$

where $i = 1, \ldots, 2^n$ and $j = 1, \ldots, n$. The training algorithm described in 3.1 was applied to each matrix T to generate the corresponding dendritic structure as well as compute the weights and responses of the SLMP that solves the parity problem for $n = 1$ to 10. For example, Table 1 shows the network parameters for $n = 4$.

Table 1. Weights and responses of the SLMP that solves the 4-parity problem

D_k	w^1_{1k}	w^1_{2k}	w^1_{3k}	w^1_{4k}	w^0_{1k}	w^0_{2k}	w^0_{3k}	w^0_{4k}	p_k
1	0	0	0	0	-1	-1	-1	-1	$+1$
2	∞	∞	∞	∞	-1	-1	-1	-1	-1
3	∞	∞	0	0	-1	-1	$-\infty$	$-\infty$	-1
4	∞	0	∞	0	-1	$-\infty$	-1	$-\infty$	-1
5	∞	0	0	∞	-1	$-\infty$	$-\infty$	-1	-1
6	0	∞	∞	0	$-\infty$	-1	-1	$-\infty$	-1
7	0	∞	0	∞	$-\infty$	-1	$-\infty$	-1	-1
8	0	0	∞	∞	$-\infty$	$-\infty$	-1	-1	-1
9	0	0	0	0	$-\infty$	$-\infty$	$-\infty$	$-\infty$	-1

Entries in Table 1 marked with $\pm\infty$ mean that no excitatory/inhibitory connection exists between the input neurons $N_1, \ldots, N_4$, and the dendrites $D_1, \ldots,$ D_9 of the output neuron M. Inspection of the parameters for each trained SLMP using Algorithm 3.1 reveals that the *number of dendrites* $K(n)$, and the *number of weights* $\omega(n)$ (both excitatory and inhibitory) necessary to solve the n-bit parity problem are given by

$$K(n) = 2^{n-1} + 1, \tag{5}$$

$$\omega(n) = n(K(n) + 1). \tag{6}$$

In addition, we observe that the SLMP weight assignments for n-bit parity contains all the weights for $(n-m)$-bit parity for $m = 1, \ldots, n-2$. For example, the weights for $n = 3$ are embedded in Table 1 and can be extracted by considering the subtable formed by rows $D_1, \ldots, D_5$ and columns $w_{2k}^{\lambda}, w_{3k}^{\lambda}, w_{4k}^{\lambda}$. Similarly, the weights for $n = 2$ (XOR) are obtained as the submatrix from Table 1 with rows D_1, D_2, D_3 and columns $w_{3k}^{\lambda}, w_{4k}^{\lambda}$. Therefore, once an SLMP is trained for n-bit parity, the SLMP for $(n - m)$-bit parity can readily be obtained with *no* training; in this case, the weights correspond to the submatrix formed with rows $D_1, \ldots, D_{2^{(n-m)}+1}$ and columns $w_{(1+m)k}^{\lambda}, \ldots, w_{nk}^{\lambda}$. The diagram shown in Fig. 2 illustrates the *morphological neural structure* that corresponds to Table 1.

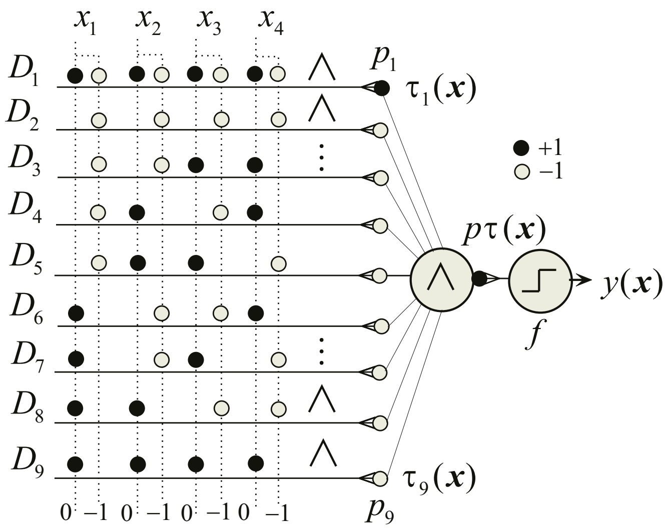

Fig. 2. SLMP structure for 4-parity; 0,-1 at the bottom are weight values

Table 2 displays the number of patterns, the number of dendrites, the number of weights, the *learning time* (LT) needed to find the network parameters (including I/O file operations on data and result sets), and the *recognition time* (RT) spent to classify correctly all patterns in boolean space for $n = 1, \ldots, 10$. We remark that the training phase for each SLMP takes only *one* iteration to complete without any convergence problems. The computer used was a Pentium 4 processor running at 1.2 GHz with 512 Mb main memory.

It is important to remark that direct comparison of the SLMP performance against known *numerical* solutions to the n-bit parity problem, proposed by several researchers, would be difficult since in each reported study different goals and performance measures have been used. For example, in [7], the goal was to compare the average number of epochs between a single hidden layer and a two hidden layer network topology trained by backpropagation. On the other hand, the results presented in [18, 19], were focused to find the minimum number of hidden units in a single hidden layer topology together with the number

Table 2. SLMP performance parameters for the n-bit parity problem

n	2^n	$K(n)$	$\omega(n)$	LT	RT
1	2	1	2	200 ms	20 ms
2	4	3	8	310 ms	20 ms
3	8	5	18	420 ms	20 ms
4	16	9	40	550 ms	20 ms
5	32	17	90	760 ms	30 ms
6	64	33	204	1.71 sec	70 ms
7	128	65	462	8.75 sec	240 ms
8	256	129	1,040	1.13 min	1.01 sec
9	512	257	2,322	9.54 min	4.38 sec
10	1,024	513	5,140	1.4 hrs	19.05 sec

of iterations needed to converge, respectively, by a dynamic node creation or a feedforward neural network construction algorithm. Interested readers with the n-bit parity problem as a pattern recognition challenge, will find specific examples related to computer experiments and numerical results using a wide variety of learning algorithms in [5, 7, 10, 11, 18, 19].

5 Conclusion

The n-bit parity problem will remain a tough benchmark used to test the learning performance in artificial neural networks, as well as an interesting pattern recognition problem by itself. The lattice algebra approach, coupled with the novel idea of introducing dendrite computation in neurodes has conducted our research in different directions to tackle non-trivial classification problems. As demonstrated in this paper, trained SLMPs, configured with specific excitatory and inhibitory weights and responses in dendrites, imitate biological neurons more closely than their traditional artificial models. The SLMP learning algorithm “grows” dendrites as needed and the n-bit parity data set is an extreme case corresponding to a *worst* case situation. However, SLMPs offer complete recognition capability as well as competitive computational performance in comparison to other artificial neural network architectures and training algorithms. Future research with the SLMP training algorithm used here, will consider its computational complexity and its application to other benchmark problems.

References

1. Thornton, C.: Parity: the problem that won’t go away. Proc. of Artificial Intelligence, Toronto, Canada (1996) 362–374.
2. Sontag, E. D.: Feedforward nets for interpolation and classification. Journal of Computer and System Sciences **45** (1992) 20–48.
3. Duch W.: Scaling properties of neural classifiers. Proc. 3rd Conf. on Neural Networks and their Applications, Kule, Poland (1997) 663–670.

4. Setiono, R.: On the solution of the parity problem by using a single hidden layer feedforward neural network. Neurocomputing **16**(3) (1997) 1059–1065.
5. Looney, C. G.: Pattern Recognition using Neural Networks. Theory and Algorithms for Engineers and Scientists. Oxford University Press, New York, USA (1997).
6. Setiono, R: Algorithmic techniques and their applications, in Neural Networks Systems, Techniques, and Applications, C. T. Leondes, Ed. Academic Press, San Diego, California, USA **5** (1998) 296–301.
7. Hjelmås, E.: A comment on the parity problem. Technical Report (7), Gjøvik University College, Norway (1999).
8. Park, C-Y., Nakajima K.: Majority algorithm: a formation for neural networks with the quantized connection weights. IEICE Trans. Fundamentals **E83-A**(6) (2000) 225–235.
9. Ghorbani, A. A., Owrangh K.: Stacked generalization in neural networks: generalization on statistically neutral problems. IEEE Proc. IJCNN, Washington, DC, USA (2001) 1715–1720.
10. Wilamowski, B. M. et al.: An algorithm for fast convergence in training neural networks. IEEE Proc. IJCNN, Washington, D.C., USA (2001) 1778–1782.
11. Liu, D., Chang T-S., Zhang Y.: A constructive algorithm for feedforward neural networks with incremental training. IEEE Trans. on Circuits and Systems **49**(12) (2002) 1876–1879.
12. Ritter, G. X., Urcid, G., Selfridge R.:Minimax dendrite computation. ASME Proc. ANNIE, St. Louis Missouri, USA **12**(2002) 75–80.
13. Ritter, G. X., Urcid G.: Lattice algebra approach to single neuron computation. IEEE Trans. on Neural Networks **14**(2) (2003) 282–295.
14. Ritter, G. X., Iancu, L., Urcid G.: Morphological perceptrons with dendritic structure. Proc. FUZZ-IEEE, St. Louis, Missouri, USA (2003) 1296–1301.
15. Ritter, G. X., Iancu, L.: Lattice algebra approach to neural networks and pattern classification. Proc. 6th Open German-Russian Workshop on Pattern Recognition and Image Understanding, Katun Village, Altai Region, Russian Federation (2003) 18–21.
16. Ritter, G. X., Iancu, L., Urcid G.: Neurons, dendrites, and pattern recognition. Proc. 8th Iberoamerican Congress on Pattern Recognition, Havana, Cuba (2003) 1296–1301.
17. Ritter, G. X., Iancu, L.: Lattice algebra, dendritic computing, and pattern recognition. Invited tutorial, 8th Iberoamerican Congress on Pattern Recognition, Havana, Cuba (2003) 16–24.
18. Ash, T.: Dynamic node creation in backpropagation networks. Connectionist Science **1** (1989) 365–375.
19. Setiono, R., Hui, L. C. K.: Use of quasi-Newton method in a feedforward neural network construction algorithm. IEEE Transactions on Neural Networks **6** (1995) 273–277.