Non-Linear Associative Mappings with Positive Bounded Coefficients

Adam Kowalczyk, Greg N. Aumann
Telecom Australia
Research Laboratories
770 Blackburn Road, Clayton Vic 3168

Herman L. Ferrá
Department of Mathematics
Monash University
Clayton, Vic 3168

Jacob L. Cybulski
Amdahl Australian Intelligent Tools Programme
Dept of Comp. Sci. and Comp. Eng., La Trobe Uni
Bundoora Vic 3083

Abstract

The paper presents a concept of simplified non-linear associative mapping with positive integer weights of predetermined size. An algorithm for generation of such neural networks, some simulation results and an architecture for digital VLSI implementation are also given.

1 Introduction

In any practical implementation of neural networks, weights have to be restricted to a finite set of values. Although this limitation is hardly noticeable in digital simulation, it becomes a critical issue when it comes to dedicated hardware implementations, be they digital or analog [5]; as the number of possible weight values has direct impact on the size of neural network implemented on a single chip.

In this paper we discuss a class of modulo perceptrons [1, 2] which are neural networks that explicitly take the limited size of possible weights into account. The main innovation is in the introduction of an efficient training algorithm based on term selection and a modification of the architecture involving taking modulo of the final sum before imposing thresholds or performing some other processing. The latter allows, among other things, unsigned digits of limited size as weights of links, and so eliminates problems in hardware implementation of negative weights [6].

2 Non-linear associative mappings

Following terminology in [6] we define a Non-linear Associative Mapping (NAM, for short) as a family of functions $\eta_\alpha = S \circ P_\alpha(\vec{x})$, $(\alpha = 1, \ldots, m)$ where $S : \Re \to \Re$ is a function and $P_\alpha : \Re^n \to \Re$ are polynomials (similar neural networks were considered for instance in [3, 4, 10, 11]). Three particular cases of NAM defined on a set $\mathcal{X} \subset \{0,1\}^n$ of binary vectors are of special interest to us:

1. A NAM with the step function, $S \stackrel{\text{def}}{=} \theta(t)$ defined as 1 for $t \geq 0.5$, and 0 otherwise, will be called a *mask-perceptron*. We can write in this case:

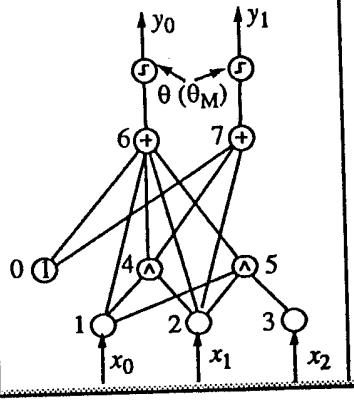
$$\eta_\alpha = \theta\left(\sum_{\vec{\imath} \in \mathcal{I}} w_{\alpha\vec{\imath}} \vec{x}^{\vec{\imath}}\right), \qquad \vec{x} \in \mathcal{X} \subset \{0,1\}^n \tag{1}$$

where $\mathcal{I}$ is a set of n-tuples $\vec{\imath} = (i_1, \ldots, i_n) \in \{0,1\}^n$, $\vec{x}^{\vec{\imath}} \stackrel{\text{def}}{=} x_1^{i_1} x_2^{i_2} \ldots x_n^{i_n}$ and $w_{\alpha\vec{\imath}} \in \Re$.

(a)

input			output	
x_0	x_1	x_2	y_0	y_1
1	1	1	1	0
1	0	1	0	1
0	1	1	0	1
0	0	1	1	1

(b)



(c)

Nodes / to / from	Type of NAM (...-perceptron) mask 6	mask 7	*mod*(8) 6	*mod*(8) 7	binary 6	binary 7
0	1.20	0.80	5	3	1	1
1	-0.90		4		1	
2	-1.10	0.10	4	0	1	0
4	2.13	-0.88	1	4	1	1
5	-0.33		7		1	

Figure 1: Example of three NAMs implementing the same decision table (a); (b) architecture, (c) weights to the outputs. (The mod(8)-perceptron was obtained by rounding weights of the mask perceptron to the nearest 0.25, and then applying the Conversion Theorem.)

2. A *mod(M)-perceptron* ($M \in \{2,3,\ldots\}$), or generically a *modulo-perceptron*, is defined as:

$$\eta_\alpha = \theta_M\left(\sum_{\vec{\imath}\in\mathcal{I}} u_{\alpha\vec{\imath}}\vec{x}^{\vec{\imath}}\right), \qquad \vec{x}\in\mathcal{X}\subset\{0,1\}^n \tag{2}$$

where $u_{\alpha\vec{\imath}} \in \{0,1,\ldots,M-1\}$ and $\theta_M(t) \stackrel{\text{def}}{=} 1$ if $0.25\times M \le t \bmod M \le 0.75\times M$ and 0 otherwise; modulo-perceptrons for $M = 2^n$ are of special interest in this paper.

3. A *binary-perceptron* which is by definition *mod*(2)-perceptron. In this case $u_{\alpha\vec{\imath}} \in \{0,1\}$, and θ_2 becomes addition in the binary Galois field GF(2), i.e. parity, or equivalently multiple XOR.

The use of the term "perceptron" in these working definitions was intended to emphasise the similarity of these neural networks to perceptrons with mask partial predicates investigated extensively in [9]. The main advantage of mask perceptrons is their simplicity: masks $\vec{x}^{\vec{\imath}}$ are logical conjunctions [9], so evaluation of $\sum_{\vec{\imath}\in\mathcal{I}} w_{\alpha\vec{\imath}}\vec{x}^{\vec{\imath}}$ is extremely easy to implement in hardware since it involves no multiplication. The implementation of modulo perceptrons and binary perceptrons, in particular, is even easier: weights are unsigned integers of pre-determined maximal size or single bits, respectively, and to compute the output modulo-sums we need to only to implement registers of the the same size.

The following simple theorems give some relations between mask perceptrons Eq. (1) and modulo perceptrons Eq. (2).

Universality Theorem *Any dichotomy of* $\mathcal{X} \to \{0,1\}$ *can be implemented as a mod(M)-perceptron with prime M (a binary-perceptron in particular).*

Conversion Theorem *If the weights* $w_{\alpha\vec{\imath}}$ *in Eq. (1) are rational numbers of the form* $w_{\alpha\vec{\imath}} = k_{\alpha\vec{\imath}}/M$ *with integers* $k_{\alpha\vec{\imath}}$ *and* $M > 0$*, then the mask-perceptron defined by Eq. (1) and mod(2M)-perceptron of the form given by Eq. (2) with weights* $u_{\alpha\vec{\imath}} \stackrel{\text{def}}{=} k_{\alpha\vec{\imath}} \bmod (2M)$*, provide identical results for every* $\vec{x}$ *such that* $-0.5 < \sum_{\vec{\imath}\in\mathcal{I}} w_{\alpha\vec{\imath}}\vec{x}^{\vec{\imath}} \le 1.5$.

Modulo Prime Theorem *Let M be a prime number (*$M = 2$ *in particular), a and c be real numbers such that* $a + (M-1)c/2 < 0.5 \le a + (M+1)c/2$ *and* $w_{\alpha\vec{\imath}}$ *be real coefficients (weights) such that* $\sum_{\vec{\imath}\in\mathcal{I}} w_{\alpha\vec{\imath}}\vec{x}^{\vec{\imath}} \in \{a, a+c, \ldots a+(M-1)c\}$ *for every* $\vec{x}\in\mathcal{X}$ *and every* $\alpha \in \{1,\ldots,m\}$*. Then there exists a set of integer weights* $u_{\alpha\vec{\imath}} \in \{0,1,\ldots,M-1\}$*, such that a mod(M)-perceptron Eq. (2) has identical output as the mask-perceptron Eq. (1) for every* $\vec{x}\in\mathcal{X}$.

	Number of terms		% words with errors				
				mod (2^m)- perceptrons			
	order ≥ 2	total	mask	m=6	m=5	m=4	m=3
Experiment 1 (Tests for 1,000 words)	0	270	41.3	40.8	36.8	44.8	95.4
	141	300	21.6	21.3	19.3	42.8	98.2
	236	400	8.7	9.1	9.0	33.3	99.5
	328	500	2.9	2.9	2.7	28.3	98.3
	428	600	0.5	0.6	0.8	36.9	98.9
	523	700	0.0	0.0	0.2	33.6	99.0
Experiment 2 (Tests for 10,000 exemplars)	0	270	9.0	9.9	10.9	21.5	91.1
	66	239	5.3	6.1	6.5	17.7	92.6
	165	338	3.8	4.5	4.7	24.8	84.2
	215	388	4.6	5.3	5.6	18.6	90.5
	264	437	6.5	7.8	7.9	23.2	95.5

Figure 2: Performance of perceptrons for recognition of lexical categories generated: (a) for 1000-words dictionary (Experiment 1), (b) for misspelled 100 words (Experiment 2); for each test exemplar an error was counted as 1 if any one of the 22 categories was incorrect.

3 Practical generation of modulo perceptrons

In this section we describe some experimental results of practical generation of non-trivial modulo perceptrons which arose in relation to the development of a connectionist parser. The particular task considered in this paper was to develop an NAM for determination of 22 lexical categories (e.g. infinite verb, present verb, past verb, etc.) for two dictionaries of English words up to 10 characters long. The words were presented to the network in the form of sparse input of 270 bits (c.f. [1, 7] for further results of this type including experiments with other dictionaries and also a more compact 60 bit input encoding).

Experiment 1 *Recognition of lexical categories for a dictionary of 1000 correctly spelled words.* A binary perceptron for this task with 100% accuracy reported in [1] had 1000 monomial terms including 735 terms of second and third order. These terms composed the linear space basis of all functions on the dictionary over the binary Galois field GF(2) and were found by sequential examination of candidate terms generated as in step 1.b of the algorithm in Fig.3. The whole generation took about 4352 s. on a Sun 3/60.

A series of different neural networks for the same task was reported in [7]. These are equivalent to mod(2000), mod(200) and mod(20)-perceptrons in the terminology introduced here, so we additionally developed another series of mod(8), mod(16), mod(32) and mod(64)-perceptrons for the purpose of this paper, since they are more suitable for digital hardware implementation described in the following Section. We investigated the first order case first and then we used the following training algorithm based on empirical selection of terms to generate higher order networks. (Note that practically even for a second order NAM [3, 6, 10, 11] some selection of terms has to be implemented, since for this application there are a total of $\sim$ 30,000 of non-trivial second order terms.)

The perceptron generation times in our experiments were no more than a few hours on Sun 3/60; different values of ϵ_{orth} were used; the mask perceptrons were developed in series by adding new terms to the terms of the previous network. Results are shown in Figure 2.

Experiment 2 *Recognition of lexical categories for dictionary of 100 words with artificially superimposed misspellings.* In this experiment we created a data base of 10,000 samples, containing 1583 different forms, by introduction of random misspellings to the dictionary of 100 words; the probability of a 10 letter word being spelled correctly was 75%, c.f. [7] for details. The first 2000 samples containing 474 different word forms was used for training (c.f Fig. 4). Results of test

1. *Selection of suitable polynomial terms.* Given constant $\epsilon_{\text{orth}} > 0$:

   (a) *Initialization of residual errors* $e_{\alpha\beta} = \eta_{\alpha\beta}$, $\alpha = 1,\ldots,m$, $\beta = 1,\ldots,N$.

   (b) *Generation of a candidate term* $\vec{x}^{\vec{j}} = \xi_1^{j_1}\xi_2^{j_2}\cdots\xi_n^{j_n}$, $\vec{j} \in \{0,1\}^n$, starting with Stage 1 below, and passing to the next Stage only if all possibilities for a given Stage were exhausted.

      i. Stage 1. The constant term $\equiv 1$.

      ii. Stage 2. First order terms (in natural order).

      iii. Stage *k+1*. Multiplication of selected terms of order k by selected terms of order 1.

   (c) *Checking "average correlation" of candidate term with residual errors*: if

$$\left(\sum_{\alpha=1}^{m}\sum_{\beta=1}^{N} e_{\alpha\beta}\xi_1^{j_1}\xi_2^{j_2}\cdots\xi_n^{j_n}\right)^2 \left(m\sum_{\beta=1}^{N}\xi_1^{j_1}\xi_2^{j_2}\cdots\xi_n^{j_n}\right)^{-1} \left(\sum_{\alpha=1}^{m}\sum_{\beta=1}^{N} e_{\alpha\beta}^2\right)^{-1} < \epsilon_{\text{orth}},$$

   then reject the term $\vec{x}^{\vec{j}}$ and return to step (b), else

   (d) *Addition of the term* $\vec{x}^{\vec{j}}$ *to the terms already selected* and update of residual errors to the values for the optimal approximation of desired output vectors $(\vec{y}_\beta)_{\beta=1}^N$ in the sense of least mean square error. This was accomplished in practise with the use of Gram-Schmidt orthogonalization with respect to terms expanded to N-dimensional vectors.

   (e) *Exit to the step 2 below* if either no new candidate term can be generated, or the residual least mean square error $\sum_{\alpha=1}^{m}\sum_{\beta=1}^{N} e_{\alpha\beta}^2$ is below the required minimum, or other requirements.

2. *Computation of mask-perceptron weights* (real values with full available precision) for selected terms using Penrose-Moore pseudoinverse [3, 6, 10]

3. *Rounding of weights*, in our case to fractions with denominators 32, 16, 8 and 4, respectively.

4. *Application of Conversion Theorem* of the previous Section.

Figure 3: Algorithm for development of modulo perceptron with n inputs and m outputs for a given training set: $T = (\vec{x}_\beta, \vec{y}_\beta)_{\beta=1}^N = ((\xi_{1\beta},\ldots,\xi_{n\beta}),(\eta_{1\beta},\ldots,\eta_{m\beta}))_{\beta=1}^N$

against the whole set are shown in Figure 2.

4 An implementation architecture

In this section we concentrate on a simple architecture for VLSI implementation of mod(2^m)-perceptrons. We will initially assume that the weights are stored in on chip memory allowing the network to be quickly retrained. Some alternative ways of storing the weights and the benefits of these approaches will also be examined, later.

The architecture of a *binary* perceptron chip consists of two rectangular matrices of the basic cells, called the input and the output matrix. The first matrix takes the inputs and calculates the activations of polynomial terms ($\equiv$ multiple AND). The activations are subsequently taken by the second matrix which calculates the outputs ($\equiv$ multiple XOR); c.f. Fig. 4a.

The binary-perceptron architecture can be easily extended for mod(2^m)-perceptron by grouping output bits together into a single integer value. To do this we add the facility to selectively pass carry bits across output bits. This is done with the addition of an AND gate. This approach allows the efficient use of different output precisions in the one network.

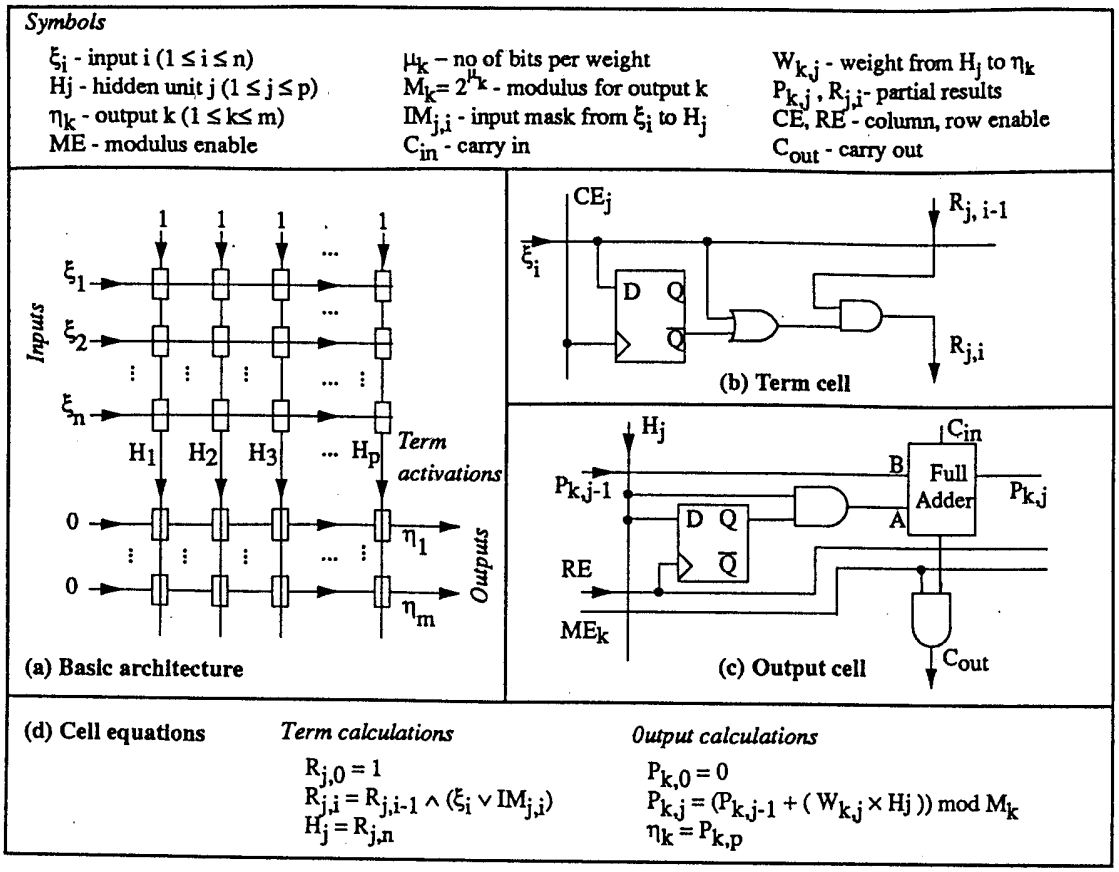


Figure 4: Basic architecture for digital VLSI implementation of modulo perceptron

The term cell shown in Fig. 4b is the basic cell which is repeated to form the input matrix for a mod(2^m)-perceptron. Similarly the output cell in Fig 4c is repeated to form the output matrix. This repetitive structure is shown in Fig. 4a. The term and output cells perform the calculations in Fig. 4d.

Loading the weights into the network is achieved by adding "column enables" on the input matrix and using the inputs to carry the weights. For the output matrix we added "row enables", breaking the vertical term unit lines in between the two matrices and using them to carry the weights. An extra line is also added to the output units to indicate whether to pass the carry bit to the next output cell in the same column or not (0 for the most significant bit of each output and 1 otherwise, thus for binary outputs they would all be 0).

So far we have assumed that the weights are stored in on chip memory. However the basic cells can be simplified considerably if less flexibility in reconfiguring the network is required. For instance, the weights could be stored in EEPROM cells (with some "off line" reprogramming capabilities) or they could be mask programmed and thus fixed at manufacture. The use of EEPROM storage of weights should allow smaller cells and so more cells per chip, while the use of mask programming allows the simplification of the input cells to a single transistor which will only be present if the input weight is 1. (This structure is very similar to that of the AND plane in a Programmable Logic Array [8], although simpler because the complement of the input is not needed.) The output cells in the mask programed case cannot be simplified as much as the input cells. They will still need to contain an adder but the flip flop and the AND gates can be eliminated.

5 Discussion of results and conclusions

Empirically, we show that a NAM with real weights can be converted to a modulo perceptron (mod(M)-perceptron) with bounded, positive integer values of link weights without a significant drop in performance.

It was also demonstrated that mod(2^m)-perceptrons are especially easy to implement in digital VLSI with currently available technologies. The presented simple architecture for modulo-perceptron implementation allows dynamic re-programming including adjustment of weight size (i.e. allocation of the number of bits per weight by changing the "row enable" values). As expected, the experimental results prove that in general the network accuracy improves with the increase in allocated weight size. For the experiments presented in this paper, we found that five-bit weights give performance close to that of the mask perceptron with real weights; the performance of modulo-perceptrons with 3- and 4-bit weights was significantly worse (which is partially due to the severity of our error counting criteria).

By example we showed that in a case of a significant number of inputs causing an explosion of higher-order terms (~30,000 second-order terms in our case) the selection algorithm presented in our paper is a practical way to overcome the limitations of a first-order NAM by addition of a small number of "especially useful" higher-order terms although a large number of such additions may cause a drop in generalisation ability (c.f. results for Expt. 2)

Experimental results show that the training algorithm presented in the paper is a robust tool for the generation of higher order NAMs for problems with thousands of training exemplars and hundreds of inputs. Direct training of modulo perceptrons (without the mask-perceptron step) is possible and algorithms for such purpose should be investigated. The efficiency of the algorithm could also be improved by its parallelization.

Acknowledgements. We would like to express our thanks to Mr. Chris Rowles for his helpful comments during preparation of this paper. The permission of Executive General Manager of Telecom Australia Research Laboratories to publish this paper is also acknowledged.

References

[1] J.L. Cybulski, H.L. Ferrá, A. Kowalczyk, and J. Szymański. Determining word lexical categories with a multi-layer binary perceptron. In M. H. Hanza, editor, *Proc. of the 1 IASTED Inter. Symp. ...*, pp. 45–49, Anaheim, 1989. Acta Press.

[2] –. Experiments with multi-layer perceptrons. In R. Jarvis, editor, *AI'89, Proc. of the Austral. Joint AI Conf.*, pp. 398–412, 1989.

[3] R.O. Duda and P.E. Hart. *Pattern Recognition and Scene Analysis.* J. Wiley & Sons, NY, 1973.

[4] C.L. Giles, R.D. Griffin, and T. Maxwel. Encoding geometric invariances in higher-order neural networks. In *Proc. Neur. Inf. Proc. Sys., Denver 1987*, pp. 301–9, NY, 1988. Amer. Ins. Phys.

[5] P.H. Graf and L.D. Jackel. Analog electronic neural network circuits. *IEEE Circuits and Devices Magazine*, July :44–55, 1989.

[6] T. Kohonen. *Self-Organisation and Associative Memory.* Springer-Verlag, Berlin, 1989.

[7] A. Kowalczyk and H.L. Ferrá. Experiments in lexical classification by multi-layer perceptrons with simplified interconnection weights. In C.P. Tsang, editor, *AI'90, Proc. ...*, pp. 209–23, Singapore, 1990. World Scientific.

[8] J. Millman. *Microelectronics: Digital and Analog Circuits and Systems.* McGraw Hill, 1979.

[9] M.L. Minsky and S.A. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, Cambridge, Massachusetts, 1969. (1988 edition).

[10] D. Psaltis, C.H. Park, and J. Hong. Higher order associative memories and their optical implementations. *Neural Networks*, 1:149–163, (1988).

[11] D.F. Specht. Probabilistic neural networks and polynomial adaline as complementary techniques for classification. *IEEE Trans. Neural Networks*, 1:111–121, 1990.