CAINN - Weightless Alpha-Beta Neural Network

Alarcón-Paredes, Antonio; Argüelles-Cruz, Amadeo-José.
Center for Computing Research (CIC) from
National Polytechnic Institute (IPN). México, D.F.
aparedesb07@sagitario.cic.ipn.mx, jamadeo@cic.ipn.mx

Abstract

In this paper, a weightless neural network model is presented, based on the known operations Alpha and Beta. This weightless Alpha-Beta neural network model is called CAINN – Computing Artificial Intelligence Neural Network. The CAINN´s pattern learning and recalling algorithms are created given the Generalized Alpha, Sigma-Alpha, and Sigma-Beta operations, based on the ADAM weightless neural network model. Studies of the CAINN performance are shown. These studies report the reliability of the CAINN model.

1. Introduction

Weightless Neural Networks (WNNs) constitute an important derivation of the Artificial Neural Networks. Due to WNNs provide flexibility, feasibility of implementation, and fast learning algorithms, they represent a paradigm on the field of Neural Networks.

WNNs models base their operation on artificial neurons, which handle only binary values at their inputs and outputs, with no weights between nodes. The functions performed by each of the neurons are stored into look-up tables, which can be shaped by any random access memory (RAM). Since the WNNs have no weights adjusting, they have to change or refresh the contents of the look-up table entries to perform the learning phase [1], taking into account the *No Free Lunch Theorems* [12].

Some of the WNNs models have their origin on the research carried out by Bledsoe & Browning in 1959, where an adaptive classifier called n-tuple was proposed [2]. This method, based on distributed computing, resulted very attractive to Igor Aleksander, who started to work with learning networks using n-tuple sampling machines [3, 4, 5].

The RAM node and the SLAM (Stored Logic Adaptive Microcircuit) were also introduced by Aleksander [1]. Later on, important researches developed WNNs models that were extensions to the n-tuple theory that were applied to a number of image processing tasks such as character recognition, face recognition, and scene analysis; and uses other techniques in the classification of objects [1, 6, 7, 8, 9].

This paper studies a new model for Weightless Neural Networks, which uses the development of Alpha-Beta Associative Memories [10]. Section 2 shows the theory required to describe the new operations needed for the CAINN algorithm. The learning and recalling phase are also presented [11]. The performance of the CAINN model over a database placed in the UCI Machine Learning Repository [14] is presented in Section 3. Last section is devoted to discuss the results of the behavior and performance of CAINN model.

2. Methods

In order to describe the algorithm used by CAINN, three original operations are presented in this section.

2.1. α_g operation

Since this model is based on the Alpha and Beta operations [10], a new operation called Generalized Alpha, denoted by $\boldsymbol{\alpha_g}$, is required to characterize the recalling phase of the CAINN. This new operation is a generalization of the original Alpha binary operation, so it accepts arguments with values in the real numbers, and provides output values in the $\boldsymbol{B} = \{\mathbf{00}, \mathbf{01}, \mathbf{10}\}$ set [11].

Given the operation $\alpha_g = \mathbb{R} \times \mathbb{R} \to B$, we have the following definition:

978-0-7695-3320-9/08 $25.00 © 2008 IEEE
DOI 10.1109/CERMA.2008.94

Let $x \in \mathbb{R}, y \in \mathbb{R}$ be any two real numbers, then:

$$\alpha_g = \begin{cases} 00 & \text{if} \quad x < y \\ 01 & \text{if} \quad x = y \\ 10 & \text{if} \quad x > y \end{cases}$$

2.2. σ_α operation

The Sigma-Alpha operation is defined as follows:

$$\sigma_\alpha : P_{m\times r}\ \sigma_\alpha\ Q_{r\times n} = \left|f_{ij}^{\alpha}\right|_{m\times n}, \quad \text{where}$$

$$f_{ij}^{\alpha} = \sum_{k=1}^{r} \alpha\left(p_{ik}, q_{kj}\right)$$

2.2. σ_β operation

The Sigma-Beta operation is defined as follows:

$$\sigma_\beta : P_{m\times r}\ \sigma_\beta\ Q_{r\times n} = \left|f_{ij}^{\beta}\right|_{m\times n}, \quad \text{where}$$

$$f_{ij}^{\beta} = \sum_{k=1}^{r} \beta\left(p_{ik}, q_{kj}\right)$$

Let $x \in A^n, y \in A^m$ be two given column vectors with $A = \{0,1\}$, and the $m \times n$ dimension matrix denoted by $y\ \sigma_\alpha\ x^t$. The *ij*-component of such matrix would be:

$$[y\ \sigma_\alpha\ x^t]_{i\times j} = \alpha\left(y_i, x_j\right)$$

Suppose a case where we operate a $m \times n$ dimension matrix with a n dimension column vector, using the σ_β operation. We will obtain a m dimension column vector which *i*-component is denoted as follows:

$$\left(\mathbf{P}_{m\times n}\ \sigma_\beta\ \mathbf{x}\right)_i = \sum_{j=1}^{n} \beta\left(p_{ij}, x_j\right)$$

For instance, if $\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$, then

$$\mathbf{P}\ \sigma_\alpha\ \mathbf{x} = \begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix} \quad \text{and} \quad \mathbf{P}\ \sigma_\beta\ \mathbf{x} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

Similar results for the $\boldsymbol{P}_{m\times n}\ \sigma_\beta\ x$ operation are obtained [11].

2.3 CAINN algorithm

The algorithm for this new Weightless Alpha-Beta Neural Network uses the set $A = \{0,1\}$ where input and output patterns are taken, having the following fundamental set $\{(x^\mu, y^\mu) | x^\mu \in A^n, y^\mu \in A^m\}$ where $\mu = 1,2,\dots,\rho$.

A class code for every input and output pattern at both learning and recalling phase is required. This class code is represented as the pattern $z^\mu \in A^k$, where k is the dimension of a one-hot vector corresponding to the decimal value for the binary pattern x^μ.

2.3.1. Learning Phase. This is carried out as follows:

- Two null matrices named $\boldsymbol{P} = \left[p_{ij}\right]_{k\times n}$ and $\boldsymbol{Q} = \left[q_{ij}\right]_{m\times k}$ are created.
- The *ij*-component of **P** and **Q** are denoted as $p_{ij}(0)$ and $q_{ij}(0)$ respectively.
- Each of the components p_{ij} and q_{ij} are modified according to the following rules:
  - The *k*-dimensional class code z^μ is proposed.
  - The matrix **P** is upgraded as follows: $p_{ij}(\mu) = \beta\left(\alpha\left[p_{ij}(\mu - 1), 0\right], \beta\left(z_i^\mu, x_j^\mu\right)\right)$.
  - The matrix **Q** is upgraded as follows: $q_{ij}(\mu) = \beta\left(\alpha\left[q_{ij}(\mu - 1), 0\right], \beta\left(y_i^\mu, z_j^\mu\right)\right)$.
- An additional column vector $s \in A^n$ is generated. This column vector contains at their *i*-component, the sum of all positive values presented at the *i*-row of the matrix **P**, i.e.:

$$s_i = \sum_{j=1}^{k} p_{ij} \quad such\ that\ \ p_{ij} > 0$$

2.3.1. Recalling Phase. Consider an input pattern $\tilde{\boldsymbol{x}}^\omega \in A^n$, where $\omega \in \{1,2,\dots,\rho\}$ (if $\tilde{\boldsymbol{x}}^\omega = \boldsymbol{x}^\omega$, then the pattern belongs to the input fundamental set). The following actions are taken:

- The operation $\boldsymbol{P}\ \sigma_\beta\ \tilde{\boldsymbol{x}}^\omega$ is made, i.e., $\left(\boldsymbol{P}\ \sigma_\beta\ \tilde{\boldsymbol{x}}^\omega\right)_i = \sum_{j=1}^{n} \beta\left(p_{ij}, \tilde{x}_j^\omega\right), \forall i \in \{1,2,\dots,k\}$
- A correspondent transition vector t^ω from class code z^ω is computed as follows:

$$t_i^\omega = \alpha_g\left(\sum_{j=1}^{n} \beta\left(p_{ij}\tilde{x}_j^\omega\right), \bigvee_{h=1}^{k}\left[\sum_{j=1}^{n} \beta\left(p_{hj}\tilde{x}_j^\omega\right)\right]\right), \forall i \in \{1,2,\dots,k\}$$

- Then, an H set is defined:

$$H = \left\{ h \left| \sum_{j=1}^{n} p_{hj} \tilde{x}_j^{\omega} = \bigvee_{i=1}^{k} \left(\sum_{j=1}^{n} p_{ij} \tilde{x}_j^{\omega} \right) \right. \right\}$$

- Now the class code z^{ω} is computed:

$$z_i^{\omega} = \beta \left\{ \alpha_g(t_i^{\omega}, 1), \alpha_g \left[\bigwedge_{h \in H} \left(\sum_{j=1}^{n} p_{hj} \right), \sum_{j=1}^{n} p_{ij} \right] \right\}, \forall i \in \{1,2,\dots,k\}$$

- Thus, we can compute the following operation:

$$\left(\boldsymbol{Q} \, \sigma_{\beta} \, \boldsymbol{z}^{\omega} \right)_i = \sum_{j=1}^{k} \beta \left(q_{ij}, z_j \right), \forall i \in \{1,2,\dots,m\}$$

- Finally, the output vector y^{ω} would be:

$$y_i^{\omega} = \alpha_g \left(\sum_{j=1}^{k} \beta(q_{ij}, z_j), \bigvee_{h=1}^{m} \left[\sum_{j=1}^{k} \beta(q_{hj}, z_j) \right] \right), \forall i \in \{1,2,\dots,m\}$$

3. Results

In this section, an application of the CAINN algorithm to the Iris Plant data base from the UCI Repository [14] is presented.

3.1 Pattern Recalling

In order to recall any pattern of the fundamental set, first we have to carry out the learning phase.

Fist, it was made by learning all the dataset and trying to recover the entire fundamental set. The entire fundamental set was retrieved successfully.

Then, we apply a process consisting in learning the first pattern and trying to recover it. Then, learn the second pattern and try to retrieve the two learned, and so on for the entire fundamental set.

The Iris Plant database used for this means, is composed by a dataset of 150 patterns. This experiment is shown in *Fig1*.

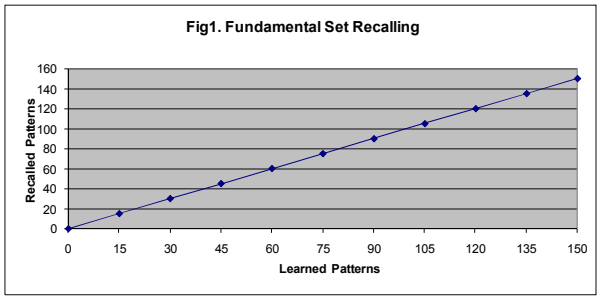

Fig1. Rate between learned and recalled patterns, both from fundamental set.

For achieving a reliable measurement of the recalling rate reached in the dataset, the process above was performed 10 times using a random altered order of the patterns in the fundamental set. The results are shown in *Table1* and in *Fig2* as well.

Table1. Recalling attempts and rate.

Recalling attempts	Patterns recalled correctly	Recalling rate
1500	1500	100%

Table1. The performance of the model presented in this paper is clearly reliable since it always recalls the entire fundamental set.

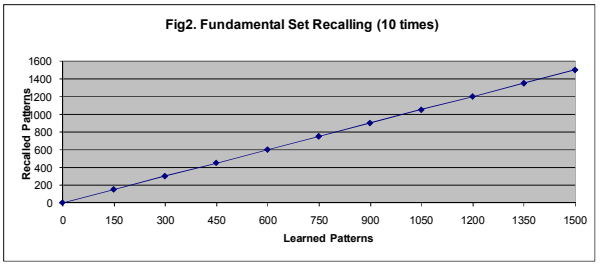

Fig2. Rate between learned and recalled patterns, both from fundamental set, performed 10 times.

It is clear that the algorithm of the Alpha-Beta Weightless Neural Network model is highly reliable since it always recalls the entire fundamental set.

3.1 Pattern Recovery

Besides the recalling of the fundamental set, it was made an estimation of performance for the CAINN model in terms of recovering a partition of the fundamental set.

This estimation of the CAINN performance was carried out by learning 70% of fundamental set. Then we try to recover other 30%. The *Fig3* provides the results of this experiment.

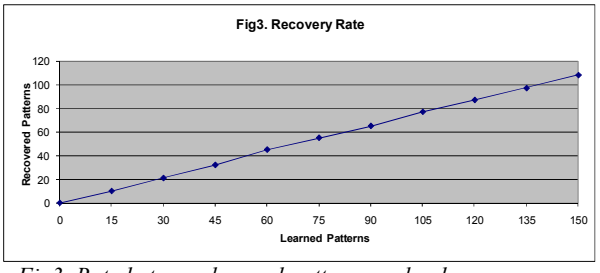

Fig3. Rate between learned patterns and unknown recovered patterns.

For a reliable estimation of recovering rate, this process was performed 10 times taking randomly the dataset patterns for each partition. *Table2* shows the results of this set of experiments.

Table2. Recovering unknown patterns (10 times).

Experiment number	Recovering attempts	Patterns recovered	Recalling rate
1	150	108	72.00%
2	150	115	76.67%
3	150	112	74.67%
4	150	109	72.67%
5	150	109	72.67%
6	150	114	76.00%
7	150	111	74.00%
8	150	108	72.00%
9	150	111	74.00%
10	150	112	74.67%
TOTAL	**1500**	**1306**	**73.93%**

The representation of the results above is shown in both *Fig4* and *Fig5*. These results represent the effectiveness of the Computer Artificial Intelligence Neural Network model.

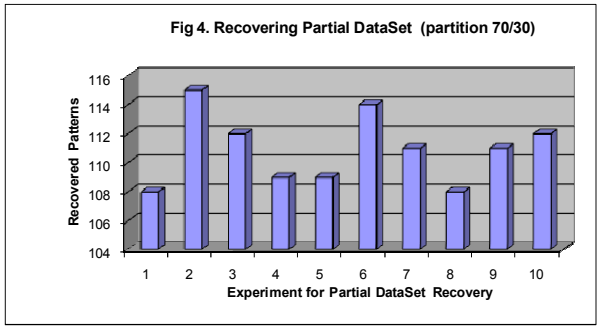

Fig4. The figure shows the results for individual set of experiments.

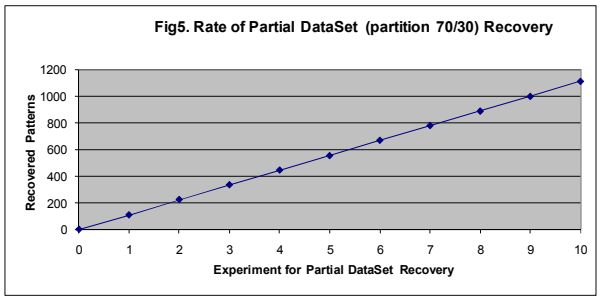

Fig5. Results of the accumulative rate for effectiveness at recovering the 10 experiments carried out.

4. Discussion

Taking into account the experiments made over the Iris Plant data base from the UCI Repository [14], we can say that the results are quite satisfactory since the CAINN model reached 73.93% recovery index.

It is also worth to mention that, although the model achieved a good recovery index at the experiments carried out in this paper, there would be some other datasets in which would reach a lower rate than some other neural networks.

This is, of course, very normal since the *No Free Lunch Theorems* [12] tell us that when an algorithm or classifier is very good with some problems, might be not so good with another kind of problem.

A variety of comparative studies of the CAINN over the ADAM model and some other models included in the literature are shown in [11].

We must remember that most of the neural network models are iterative and thus have more complexity. The CAINN algorithm is *one-shot*, which constitute an advantage over some other models. We can say this model is a simple and reliable way to classify and recover patterns.

References

[1] Ludermir, T. B. et al. (1999). Weightless Neural Models: A Review of Current and Past Works. Neural Computing Surveys, 1, 2, 41-61.

[2] Bledsoe, W. & Browning, I. (1959). Pattern Recognition and Reading by machine. In Proceedings of Eastern Joint Computer Conference, Boston, pp. 225-232.

[3] Aleksander, I. (1996). Self-adaptive Universal Logic Circuits. IEEE Electronics Letters.

[4] Aleksander, I. (1989). Canonical neural nets based on logic nodes. In First IEE International Conference on Artificial Neural Networks, London: IEE, pp. 110-114.

[5] Myers, C. E. & Aleksander, I. (1989). Output functions for probabilistic logic nodes. In First IEE International Conference on Artificial Neural Networks, pp. 310-314.

[6] Howells, G., Fairhurst, M.C. & Rahman, F. (2000). An Exploration of a new paradigm for weightless RAM-based neural networks, Connection Science, vol. 12, no.1, pp. 65-90.

[7] Ullamnn, J. R. (1969). Experiments with the n-tuple Method of Pattern Recognition. Trans. IEEE Computers C-18 (12), pp. 1135-1137.

[8] Aleksander, I., Thomas, W. V., Bowden, P. A. WISARD, a Radical Step Forward in Image Recognition. Sensor Review, vol.4, no.3. pp. 120-124.

[9] Austin, J. (1986). The Design and Application of Associative Memories for Scene Analysis. PhD. Department of Electrical Engineering. Brunei University.

[10] Yáñez-Márquez, C. (2002). Memorias Asociativas basadas en Relaciones de Orden y Operadores Binarios. Tesis de Doctorado, Centro de Investigación en Computación, Mexico City.

[11] Argüelles-Cruz, A (2007). Redes Neuronales Alfa-Beta sin pesos: Teoría y Factibilidad de Implementación. Tesis de Doctorado, Centro de Investigación en Computación, Mexico City.

[12] Wolpert, D., Macready, W. (1997). No Free Lunch Theorems for Optimization. IEEE Transactions on Evolutionary Computation, Vol. 1, No. 1.

[13] Tran, Q.-., Toh, K.-., Srinivasan, D.-., Wong, K.-. & Low, S. Q.-. (2005). An empirical comparison of nine pattern classifiers, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics Vol. 35, No. 5, pp. 1079-1091.

[14] Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository, found at [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.