Research Journal of Biological Sciences 2 (5): 607-611, 2007

Using an Easy Calculable Complexity Measure to Introduce Complexity in the Artificial Neuron Model

Ana Carolina Sousa Silva, Sérgio Souto, Euvaldo Ferreira Cabral and Ernane José Xavier Costa
LAFAC-ZAB/FZEA Universidade de São Paulo, Rua Duque de Caxias Norte n. 225,
ZIP Code 13635-900, Pirassununga SP, Brazil

Abstract: This study introduces an approach to simulate neural complexity by changing the McCulloch and Pitts neuron model. The new approach was tested by comparing the classification performance of a multilayer perceptron with complexity measurement capability to a traditional multilayer perceptron with McCulloch and Pitts neuron model The results showed that the multilayer perceptron implemented with the complexity measurement approach achieved best classification performance (worst score of 94%) when compared with multilayer perceptron without the complexity approach (best score of 51%) in task of classifier large time series generated by a logistic map with different generator parameter.

Key words: Calculable copmplexit, artificial neurm model, complexity measurement, performance, multilayer

INTRODUCTION

Neural dynamics may be described at several levels of abstraction (Hausser *et al*., 2000). On a microscopic level, ionic flow changes the axon-hillock membrane potential by modifying the membrane's conductance. Conductance change involves selectively opening and closing molecular channels. Sodium (Na^+) and potassium (K^+) ions flow across the membrane through these molecular channels. The sodium and potassium ions change their conductance (their ability to flow) as they flow across the membrane by altering these molecular channels in complex ways. This complex dynamic dictates how the neuron field will work (Magee *et al*., 1998; Rhodes, 1999). Mathematical models use a higher level of abstraction and consider the neuron as a homogeneous unit, which generates spikes if the total excitation is sufficiently large (Colli, 1994). Models based in this idea are the so-called integrate-and-fire models (Jiong *et al*., 2000; Burkitt and Clark, 1999).

Computational models of the neuron are inspirited in the integrate-and-fire models (Polsky *et al*., 2004). The most common of these computational models is the McCulloch and Pitts (1943) model. This model computes the weighted sum of its inputs and produces an output based on the activation function. Therefore, the complexity of neuron dynamics shows that the McCulloch and Pitts model, despite its simplicity, does not have biological evidence. The integrate-and-fire model is not the only possibility to explain the neuron reaction to the inputs (Polsky *et al*., 2004; Chapline, 1997).

A larger variety of techniques and models have been developed in order to understand or predict the neural complexity. These techniques focus on the complexity topology (Lucia *et al*., 2005) and on the measurement of the information complexity that flow through the neural network (Kon and Plaskota, 2000).

This research suggests a modified computational neuron model by modeling the complexity of neuron dynamics using complexity measures acting on all inputs arising to the neuron. This model differs from the classical McCulloch and Pitts model by computing the complexity of the all weighted inputs to the neuron, i.e., the model computes the weighted input complexity and produces an output based on the activation function. The complexity was measured using the method developed by Lempel and Ziv (1976). For testing the new computational model, a multi-layer perceptron (named complex multi-layer perceptron) was implemented using this new neuron configuration. The Complex Multi-Layer Perceptron (CMLP) was compared with a Convectional Multi-Layer Perceptron (MLP) to classify a standard benchmark complex time series produced by a logistic map with different parameters. The only difference between the CMLP and MLP implemented was the neuron model.

MATERIALS AND METHODS

Complexity measurement algorithm: The calculation of complexity was based on the work of Lempel and Ziv (1976), where the measure $c(n)$ is introduced. The complexity $c(n)$ measures the number of distinct patterns

Corresponding Author: Ernane José Xavier Costa, LAFAC-ZAB/FZEA Universidade de Sao Paulo, ZIP Code 13635-900, Pirassununga SP, Brazil

that must be copied to reproduce a given string. In practical application, $c(n)$ is independent of the sequence length and normalized by a random string that is meaningful (Zhang and Roy, 2001). If the length of the sequence is n and the number of different symbols is s, the upper bound of c(n) is given by:

$$c(n) < \frac{n}{(1-\varepsilon)\log_s(n)} \text{ and } \varepsilon_n \to 0 \ (n \to \infty) \quad (1)$$

In general

$$\frac{n}{\log_s(n)}$$

is the upper limit of c(n), where the base of the logarithm is s, i.e.,

$$\lim_{n\to\infty} c(n) = b(n) \equiv \frac{n}{\log_s(n)} \quad (2)$$

In practical applications b(n) is obtained for a random string of length n with complexity given by

$$b(n) = \frac{hn}{\log_k(n)} \quad (3)$$

where k denotes the number of different characters in the string and h denotes the normalized source entropy given by:

$$h = \frac{-1}{\ln(n)} \sum_{i=1}^{n} p_i \ln(p_i) \quad (4)$$

where p_i is the probability for each state i. The normalized complexity measure C(n) is given by:

$$C(n) = \frac{c(n)}{b(n)} \quad (5)$$

For a string S composed by symbol sequences $s_1s_2\ldots s_n$, i.e, $S = (s_1s_2\ldots s_n)$, the algorithm used for calculation of c(n) is based on the how S can be reconstructed using a given symbol sequence (Szczepanski *et al*., 2003). It is assumed that this symbol sequence has been reconstructed up to the symbol s_r and that s_r has been newly inserted, i.e., $S = s_1s_2\ldots s_r$. will denote the symbol sequence up to s_r, where the dot indicates that s_r is newly inserted. The question is how the rest of S can be reconstructed by simple copying of the previous sequence or whether one has to insert new digits. The algorithm to solve this problem is to take $Q \equiv s_{r+1}$ and check whether this term is contained in the vocabulary of the symbol sequence S so that Q can simply be obtained by copying a symbol of S. This is equivalent to finding whether Q is contained in the symbol vocabulary v(SQπ) of SQπ where SQπ denotes the sequence which is composed of S and Q and π means that the last digit has to be deleted, i.e.,

$$Sq\pi = S \quad (6)$$

To exemplify this, consider that s_{r+1} can indeed be copied from the vocabulary of s. Then, the algorithm asks whether $Q = s_{r+1}s_{r+2}$ is contained in the sequence vocabulary of SQπ and so on until Q becomes so large that it can no longer be obtained by copying a symbol from v(SQπ) and one has to insert a new digit. The number c(n) of production steps to create a sequence symbol, i.e., the number of newly inserted digits (plus one if the last copy step is not followed by insertion of a digit), was used as a measure of the complexity of a given sequence symbol.

For example, consider the symbol sequence 0010. The following steps can determine the complexity c(n) for this sequence.

Step 1: The first digit always has to be inserted _0.
Step 2: S=0, Q=0, SQ=00, SQπ=0, Q is in v(SQπ)_0.0
Step 3: S=0, Q=01, SQ001, SQπ=00, Q isn't in v(SQπ) _0.01.
Step 4: S=001, Q=0, SQ=0010, SQπ=001, Q is in v(SQπ)_0.01.0

For this symbol sequence c(n)=3 i.e., the total number of parts of the symbol sequence that are separated by dots. The algorithm to calculate c(n) is:

Begin
Step 0: cn←1;
Step 1: L←1;
Step 2: I←0;
Step 3: K←1;
Step 4 Kmax←1;
Step 5: if S(I+k)=S(L+k) do
Step 5.1: – if (k>Kmax) do
  Step 5.1.1: – Kmax ← k;
  Step 5.1.2: else do I ← I+1;
Step 5.2: if (I=L) do
  Step 5.2.1: cn ← cn+1;
  Step 5.2.2: L ← L + Kmax;
  Step 5.2.3: if (L+1)>n the Stop
  Step 5.2.3.1: else go to Step 2;
  Step 5.2.4: else k ← 1 ; go to Step 5;
End.

Figure 1 illustrates the complexity neuron diagram and Fig. 2 shows the directed graph representation of the complexity neuron. Figure 1 illustrates the neuron

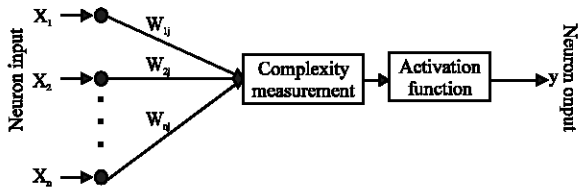

Fig. 1: Diagram representing the Maculloch's neuron model with a complexity measurement block

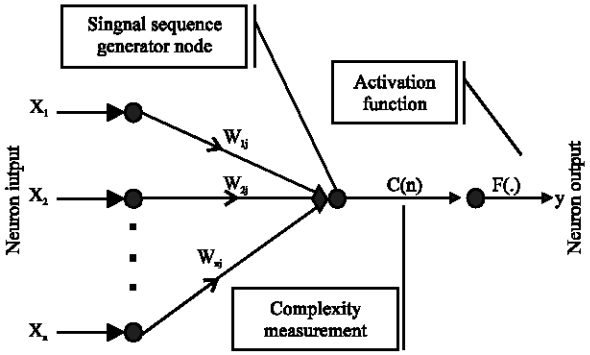

Fig. 2: Graph representation of neuron model with complexity measurement

structure where the main difference to the McCulloch and Pitts model is the complexity measurement block.

Figure 2 shows details of the actions performed in each graph node. The inputs nodes propagate our signal weighted by w's operators to the symbol sequence generator node. The symbol sequence generator node propagates our signal to the output node by means of the action of complexity measurements operator C and finally the neuron output is the neuron function operator f(.) acting in the output node signal.

In this model all the neuron inputs was treated like a symbol sequence weight dependent. This sequence is transformed into a sequence containing zeros and ones. In order to reduce the $x_i w_{ij}$ to two values, it is took s(i) = 1 if $x_i w_{ij}$ was above the spatial average

$$xa = \frac{1}{n}\left(\sum_{k=0}^{n-1} x_i w_{ij}(k)\right) \quad (7)$$

and s(i) = 0 otherwise. For this symbol sequence a complexity measurement c(n) was calculated inside the neuron structure. For one neuron the complexity measurement is used as follows:

Step 1: Transform all neuron inputs into symbol sequence
Step 2: Calculate the complexity of the neuron
Step 2.1: Calculate c(n)
Step 2.2: Calculate h and b(n)
Step 2.3: Calculate C(n) = c(n)/b(n)
Step3: Calculate the neuron output applying the activation function as follows
Neuron_output = F(complexity of neuron C(n))
Step4-propagate the neuron output to other neurons.

ANN experiments: In this researchwork two ANN were implemented. The Multi Layer Perceptron (MLP) trained with the error back propagation algorithm using a conventional neuron model and the Complex multi layer perceptron CMLP also trained with error back propagation algorithm, but using a complex neuron model. The sigmoid was used as the neuron function.

To test the CMLP and to compare to the traditional MLP, standard benchmark datasets of time series generated by a logistic map (Kaspar and Schuster, 1987) with different control parameters r were used to train the network. The ANN task was to associate time series with the control parameter r. The time series was generated by the following equation:

$$x_{i+1} = rx_i(1-x_i)\ x_i \in [0,1] \quad (8)$$

Different control parameters r were used and the CMLP and MLP task was to learn the dependence of time series with the r parameter. The time series length generated was 300 points long and for each time series, a same initial condition x_i (i = 0) was used. So, the MLP and CMLP were implemented with 300 input neurons, 300 hidden neurons and one output neuron. The r-values used were over the interval $3 \le r < 3.5$ that is periodic and not caotic (Faigenbaum point) and over the interval $3.5 \le r \le 4$ where r = 4 represents the extreme case of chaos for the logistic map. Since the amplitude of the logistic series in driven indirectly by r value the time series generated were normalized in respect of amplitude before making the comparisons. The total data set used in training was 100 time series for each r regions.

RESULTS AND DISCUSSION

Figure 3a, b and c show the time series for r = 3 (periodic range), r = 3.5 (quasi-periodic) and r =4 (extreme chaos) respectively.

Figure 4 shows the network training error for MLP for time series generated for r = 4 with the following sigmoid function parameters:

$$F(x) = \frac{A}{(1+e^{-BX})} - C \text{ with A=1, B=1 and C=1} \quad (9)$$

Figure 5 shows the network training error for CMLP for r = 4 with the sigmoid function in Eq. 5.

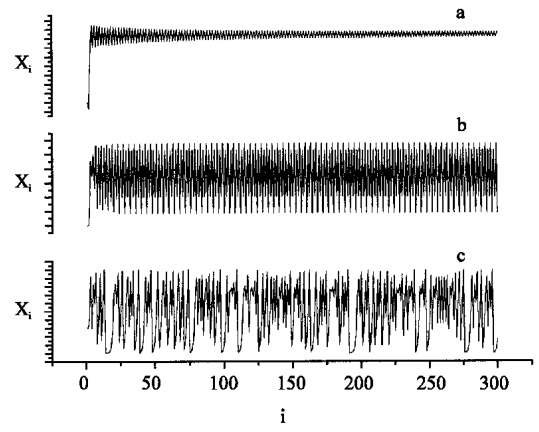

Fig. 3: Three different time series generated by logistic map. Figures a, b and c was generated by r= 3, 3.5 and 4 respectively

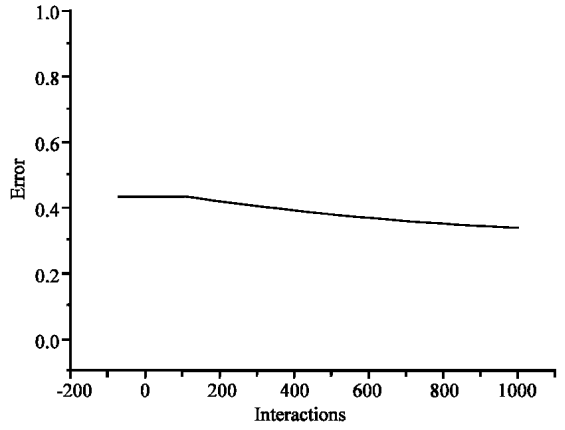

Fig. 4: Training error of MLP using MacChuloch's neuron model

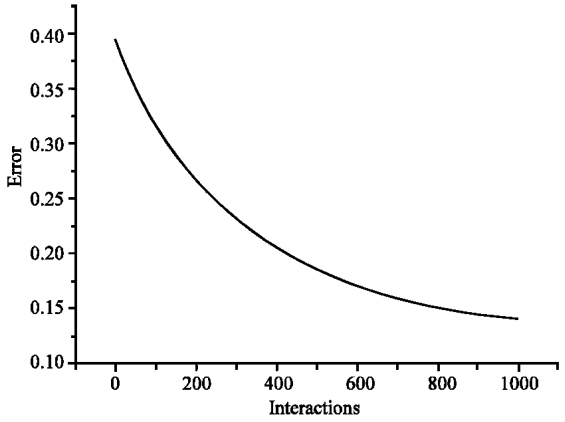

Fig. 5: Training error of CMLP

Table 1 shows the classification score for the CMLP and Table 2 shows the classification score for MLP both in different *r* intervals.

Table 1: Classification score for CMLP

r	Score (%)
3.0	94
3.5	93
4	97

Table 2: Classification scores for MLP

r	Score (%)
3.0	47
3.5	46
4	52

Comparing Fig. 4 with Fig. 5 it is evident that the error backpropagation algorithm implemented in the CMLP achieved a minor error values in the training stage than the implemented in the MLP for time series chaotic. This result shows that the neuron complexity model improved the learning process in the network. Table 1 shows the ANN classification rates for each time series generated by respective *r* parameters. The scores related in Table 1 and 2 confirm the results showed in Fig. 4 and 5, i.e., when the time series complexity increase the backpropagation algorithm implemented with complexity measurement improve ANN performance. Many authors have discussed a complexity theory for neural networks under issues related to information complexity of neural nets (Girosi and Poggio, 1990) or under issues from a topological approach to neural complexity (Sporns *et al.*, 2000). These issues concern the global network structure and operation. These works do not focus on the complexity of the simple neuron. By modeling the neuron activity as complex units the results in this paper show that the global network formed with these neurons are able to learn complex time series generated by a logistic map. The biological plausibility of this new neuron model was related to investigations of Polsky *et al.* (2004) that showed the importance of non-linear and complex behavior of pyramidal neurons, suggesting that the strong spatial compartmentalization effect observed in these neurons is incompatible with a global summation rule. Another characteristic of the ANN with complexity neurons is the ability to deal with large time series implying in large number of input neurons. Future works will explore the other features in this new neuron model; for example, the effect of hidden neuron numbers into the ANN performance and using hybrid MLP with complex neurons and McCulloch and Pitts neurons.

CONCLUSION

The main objective of this research was to test a complexity neuron model based on the Lempel and Ziv complexity measurement in an MLP network trained with the error back propagation and to compare the ANN performance with the MLP using the McCulloch and

Pitts neuron model. All results show that the MLP with complexity neurons is able to deal with large time series generated by complex systems like a logistic map. The results show also that the MLP with the McCulloch and Pitts neuron model failed in this same task.

REFERENCES

Burkitt, A.N. and G.M. Clark, 1999. New technique for analyzing integrate and fire neurons. Neurocomputing, pp: 26-27, 93-99.

Chapline, G., 1997. Spontaneous origin of topological complexity in self-organizing neural networks. Network: Computation in Neural Sys., 8: 185-194.

Colli, P., 1994. Mathematical study of a nonlinear neuron multi-dendritic model. Quarterly of Applied Mathematics, 52: 689-706.

De Lucia, M., M. Bottaccio, M. Montuori and L. Pietronero, 2005. Topological approach to neural complexity. Phys. Rev. E ., 71: 1-6.

Girosi, F. and T. Poggio, 1990. Networks and the best approximation property. Biological Cybernetics, 63: 169-176.

Hausser, M., N. Spruston and G.J. Stuart, 2000. Diversity and dynamics of dentritic signaling. Science, 290: 739-744.

Jiong, R., C. Zhijie, G. Fanji, X. Shixiong and L. Wei, 2000. Chaotic phenomenon in both an integrate-and-fire circuit with periodic pulse-train input and a discrete neural network model. Proc. Int. Joint Conf. Neural Networks, 1: 761-766.

Kaspar, J. and H.G. Schuster, 1987. Easily calculable measure for the complexity of spatiotemporal patterns, 36: 842-848.

Kon, M.A. and L. Plaskota, 2000. Information complexity of neural networks. Neural Networks, 13: 365-375.

Lempel, A. and J. Ziv, 1976. On the complexity of finite sequences, IEEE. Trans. Info. Theory, 22: 75-81.

Magee, J. D. Hoffman, C. Colbert and D. Johnston, 1998. Electrical and calcium signaling in dentrites of hippocampal neurons. Ann. Rev. Phys. 60: 327-346.

McCulloch, W.S. and W. Pitts, 1943. A logical calculus of the ideas imminent in nervous activity. Bull. Math. Biophys., 5: 115-133.

Polsky, A., B. Mel and J. Schiller, 2004. Computational subunits in thin dentrites of pyramidal cells. Nature Neur., 7: 621-627.

Rhodes, PA., 1999. Functional implications of active currents in the dentrites of pyramidal neurons. Cerebral Cortex , 13: 139-200.

Sporns, O., G. Tononi and G.M. Edelman, 2000. Connectivity and complexity: The relationship between neuroanatomy and brain dynamics. Neur. Networks, 13: 909-922.

Szczepañski, J., J.M. Amigó, E. Wajnryb and MV. Sanchez-Vives, 2003. Application of Lempel-Ziv complexity to the analysis of neural discharges. Network: Computation in Neural Sys., 14: 335-350.

Zhang, X.S. and R.J. Roy, 2001. EEG complexity as a measure of depth of anesthesia for patients. IEEE. Trans. Biomed. Eng., 48: 1424-1433.