Backpropagation through Time Algorithm for Training Recurrent Neural Networks using Variable Length Instances

Isel Grau[1], Gonzalo Nápoles[1], Isis Bonet[2], and María Matilde García[1]

[1]Centro de Estudios de Informática, Universidad Central “Marta Abreu” de Las Villas, Cuba

[2] Escuela de Ingeniería de Antioquia, Colombia

{igrau, gnapoles, mmgarcia}@uclv.edu.cu

Abstract. Artificial Neural Networks (ANNs) are grouped within connectionist techniques of Artificial Intelligence. In particular, Recurrent Neural Networks are a type of ANN which is widely used in signal reproduction tasks and sequence analysis, where causal relationships in time and space take place. On the other hand, in many problems of science and engineering, signals or sequences under analysis do not always have the same length, making it difficult to select a computational technique for information processing. This article presents a flexible implementation of Recurrent Neural Networks which allows designing the desired topology based on specific application problems. Furthermore, the proposed model is capable of learning to use knowledge bases with instances of variable length in an efficient manner. The performance of the suggested implementation is evaluated through a study case of bioinformatics sequence classification. We also mention its application in obtaining artificial earthquakes from seismic scenarios similar to Cuba.

Keywords. Recurrent neural networks, backpropagation through time, sequence analysis, bioinformatics, artificial earthquakes.

Algoritmo de retropropagación a través de tiempo para el aprendizaje de redes neuronales recurrentes usando instancias de longitud variable

Resumen. Las Redes Neuronales Artificiales (RNAs) se agrupan dentro de las técnicas conexionistas de la Inteligencia Artificial. En particular las Redes Neuronales Recurrentes son un tipo de RNA de amplio uso en tareas de reproducción de señales y análisis de secuencias, donde se reflejan relaciones causales en el tiempo y el espacio respectivamente. Por otra parte, en muchos problemas de la ingeniería y la ciencia, las señales o secuencias analizadas no siempre tienen la misma longitud, dificultando la selección de la técnica computacional a utilizar para su procesamiento. En este artículo se presenta una implementación flexible de Redes Neuronales Recurrentes que permite definir la topología deseada en función del problema específico de aplicación. Además este modelo es capaz de aprender utilizando bases de conocimiento con instancias de longitud variable de una forma eficiente. El rendimiento de la implementación propuesta es evaluado a través de un caso de estudio de clasificación de secuencias bioinformáticas y además se describe su aplicación en la obtención de terremotos sintéticos a partir de información de escenarios sísmicos similares a los de Cuba.

Palabras clave. Redes neuronales recurrentes, retropropagación a través de tiempo, análisis de secuencias, bioinformática, terremotos artificiales.

1 Introduction

Nowadays, most real-world problems are not susceptible to be solved using formal symbolic approaches or well-established algorithms due to the ambiguous or vague character of the problems. For this reason, sub-symbolic methods have been developed within Artificial Intelligence, such as Artificial Neural Networks (ANN), in order to deal with non-structured problems where the best way to solve them is unknown in advance.

ANNs are classified within connectionist techniques of Artificial Intelligence, and they are an extensively studied research field [1]. This mathematical technique for problem modeling allows obtaining intrinsic relations from data in regression, classification, pattern recognition problems, etc. They are also capable to approximate non-linear functions by learning relevant characteristics from data and reproducing them in noisy or incomplete environments [2].

Particularly, Recurrent Neural Networks (RNNs) are a type of ANN which allows feedback connections. This is the main feature of this technique, for this reason it has become a commonly used tool for signal reproduction and sequence analysis tasks. Generally, this approach is applied for solving real-world problems that reflect dynamic and complex structural relations in time or space [3, 4].

For instance, problems related to sequence classification in bioinformatics are characterized by huge knowledge bases with many attributes, where each attribute represents a position in a protein or DNA sequence. Particularly, resistance classification of a HIV protein mutation such as protease or reverse transcriptase (into two classes, i.e., resistant or non-resistant to a certain antiviral drug) is a widely treated problem in literature by using Neural Networks [5, 6], Support Vector Machine regression [7], Decision Tree classification [8], classifier ensembles [9, 10] and more recently Fuzzy Cognitive Maps [11, 12], etc. However, due to deletions and insertions naturally presented in mutations and especially due to the incompleteness of sequences, the instances (cases) of these knowledge bases do not always have the same length (sequence positions). One of the most common solutions to this problem is to work with only a part of a sequence, or to describe it with other chemical and biological descriptors [13, 14, 15], but such techniques may ignore important information or even stop analyzing some positions which are actually determinants with respect to resistance.

Another signal analysis problem is within the field of seismology. Particularly in Cuba, there are insufficient earthquakes records (accelerograms), for this reason it is necessary to obtain synthetic earthquakes from data of scenarios similar to those of this country. Such multiple signals could help in structural vulnerability evaluation of different kinds of buildings exposed to seismic hazard.

In this article we propose a flexible implementation of a Recurrent Neural Network which uses the Backpropagation Through Time learning algorithm, allowing learning from variable length cases. The next section discusses some theoretical aspects concerning Recurrent Neural Networks. Section 3 describes the proposed learning algorithm and its modifications for working with variable length instances. Section 4 evaluates the performance of this technique using a resistance classification problem of HIV mutations, and also describes the use of the same technique in obtaining artificial earthquakes from different scenarios. Finally, some conclusions are discussed in Section 5.

2 Recurrent Neural Networks

A neural network can be defined as a set of computational units called neurons which are interconnected by weighted arcs as a directed graph. The aim of such network, which can be seen as a black box, is to calculate an output Y from previously received information X.

Neural networks usually obtain information from outside through a set of input neurons, and they also have another different set of units, called output neurons, to deliver the computed results. The remaining neurons are organized in hidden layers. The overall network calculation is obtained from the information independently processed by each neuron. Each unit receives information from the others and calculates its own output from that input and its current state, changing eventually to a new state. Generally, the calculation flow moves gradually from input neurons to output neurons. In this process, hidden neurons are activated progressively according to the connection scheme of each network [16].

In recent years, a wide variety of neural network architectures has been proposed; however, most of them are classified in two groups: Feed-Forward Neural Networks (FFN) and Recurrent Neural Networks (RNN) [16]. Particularly, the RNNs support backward connections and sometimes form cycles in the graph describing its connections. These backward connections (called feedback) enable the network to store the previous state memory for calculating outputs of the current state and thus maintaining a sort of recurrence to the past processing.

Figure 1 shows a simple example of a two-layer RNN with feedback connections through time. Here, the computed state using input provided to the network at time t is used with the input provided at time $t+1$ for calculating the new state and the output at $t+1$. The shift operators d^{+1} and d^{-1} represented on recurrent connections describe the dependencies through times. A positive exponent means that the network requires the state computed in previous stages, and a negative exponent means that it requires the state of later stages. This topology

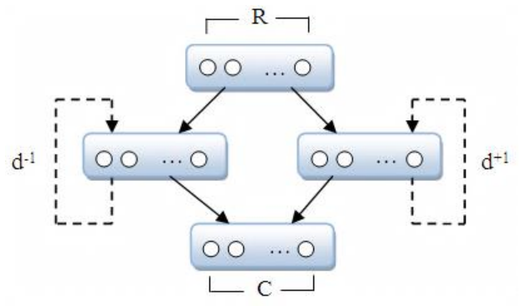

Fig. 1. Recurrent Neural Network with two feedback connections, forward and backward through times

can be used for analyzing protein sequences where influences between initial, middle and ending positions are presented [17].

In general, a RNN can be expressed with the following equations [13]:

$$y_i(t) = \Phi_i(x_i(t), I_i(t)) \quad , \tag{1}$$

$$x_i(t) = \sum_j y_j(t) w_{ij}(t) \quad , \tag{2}$$

where y_i denotes the activation state of neuron i at time t. The activation function Φ_i depends on the network topology (expressed by the net input values x_i) and the external input function $I_i(t)$. The w_{ij} values are the weights to optimize.

The main objective in RNN studies has been focused on understanding the relation between the structure and the dynamic behavior of networks. Researchers has also looked for ways to implement learning algorithms in order to find the best network configuration (topology and weights) according to a given application problem.

3 Backpropagation through Time Learning Algorithm for RNNs

Many exact or approximated algorithms have been proposed in literature for learning weights in RNNs. Most of these methods are based on descending gradient technique and can be grouped (according to [18]) in five general classes:

- Backpropagation Through Time, BPTT;
- Forward Propagation or Real Time;
- Recurrent Learning, RTRL;
- Fast Forward Propagation;
- Block Update.

Particularly, BPTT learning algorithm [19] is a variant of the Backpropagation algorithm for feed-forward neural networks. This is one of the most used algorithms in literature and it is precisely the one implemented for the proposed model.

In this algorithm, the update of the activation state of all units occurs in punctual moments of time. The first step involves the unfolding process of the network, which consists in replicating t times (folds) the recurrent network obtaining an equivalent feed-forward network. In this process each replicated connection shares its value w_{ij} in all times. Now, the resultant feed-forward network can be trained using the Backpropagation algorithm. Indeed, this is the main idea of the BPTT algorithm.

In the forward process of Backpropagation, the calculation of output y_i at time t of each neuron can be defined as

$$y_i(t) = f(x_i(t)) \quad , \tag{3}$$

$$x_i(t) = \sum_{j\in H} y_j(t) w_{ij} + \sum_{j\in I} x_j^{in} w_{ij} + \sum_{j\in M} y_j(t-\tau_{ij}) w_{ij} \tag{4}$$

where f refers to the neuron activation function, H denotes hidden layers indexes, I are the input neurons indexes, x_j^{in} is the j^{th} input neuron, whereas M represents the indexes of neurons which store information about the previous network stages and $\tau_{ij} \geq 0$ is an integer value indicating the displacement in recurrent connections through times.

Then, in the error backpropagation process, each neuron j is characterized by an error magnitude δ_j. For the output layer, this value error is calculated as the difference between the expected and the obtained value (see Equation 5), whereas for hidden layers it can be obtained taking into account the error in the successor layers (see Equation 6).

$$\delta_j(t) = (d_j - y_i) y_j (1 - y_j) , \tag{5}$$

$$\delta_j(t) = y_j (1 - y_j) \sum_{i \in Suc(j)} w_{ij} \delta_i . \tag{6}$$

On the other hand, the weights' update is obtained according to Equation 7, where the necessary change in weights is computed according to the learning rate α:

$$\Delta w_{ij}^{e+1} = \alpha \delta_j y_j , \tag{7}$$

where e refers to the order of the weight updates in the learning process.

Once the Backpropagation algorithm is applied to the obtained FFN, the process continues with the second phase of the BPTT algorithm, i.e., the folding process. In this stage the network folds itself obtaining the original recurrent network, where the connection weights are the aggregation of the equivalent connection on each fold.

As a result of the BPTT learning process, a trained RNN is obtained. The weights in the connections represent the knowledge associated with the training instances. These free parameters allow to adjust the network topology to a given application problem.

3.1 Dynamic BPTT Algorithm for Adapting the Network Topology According to the Instance Length

In some real-world problems, the instances that characterize the knowledge involved have a non-homogeneous length. Thus, when these data with variable length are aligned, those segments which have no useful information are often removed, obtaining shorter instances. This procedure guarantees that the instances have equal and compatible length with respect to the neural network topology, but also it may induce a non-optimal behavior in the learning algorithm described above. This means that missing values are interpreted as zeros. Nevertheless, these values cannot be considered as neutral values for the protein representation or the network.

Inspired in these issues, in this section we introduce a modification on the RNN model for adjusting the number of network times through a dynamic approach. This proposal ignores null segments of input sequences, providing a self-adaptive mechanism to adjust the propagation of relevant information according to the features of each problem instance. To achieve this, it is necessary to incorporate alternative considerations into the BPTT algorithm to efficiently fine-tune such parameters that characterize the RNN model.

The first step of developing our self-adaptive technique is to characterize each neuron as valid or non-valid as follows:

- An input neuron is non-valid if the input value associated with the instance (i.e., the value of the descriptor feature) is null.
- A hidden neuron is non-valid if its activation value is only determined by non-valid neurons.
- In a way similar to hidden neurons, an output neuron is non-valid if its activation value is only determined by non-valid neurons.

Then, in the learning process, the dynamic BPTT algorithm proposed here (hereinafter called dBPTT) takes into account only the links among valid neurons, i.e., only weights related to valid neurons will be updated. It is important to notice that dBPTT learns information of the valid sequence only, preserving the knowledge of the previous instances. Moreover, if all input neurons belonging to the same network time are non-valid, then this time is ignored; which automatically adapts the RNN topology (i.e., the number of active times) to the current instance. In a nutshell, for each problem instance the recurrent model and the learning scheme introduced in this section adapt the net behavior depending on the relevant information; which considerably enhances the consistency of RNNs for solving classification problems where cases have variable length.

Fig. 2 shows a representative example of an unfolded RNN having a topology with three times and two context blocks, representing one block for recurrence to the past and another one for recurrence to the future. The example also shows the adaptability capabilities of the proposed RNN based model for processing a new instance which has segments of irrelevant information. Here, the instance is described by 12 descriptors which are not sequenced in the first four positions (their values are null). It is important to notice that the model uses the whole sequence three times for processing, which means that each time it processes only four descriptors. So, following the above mentioned criteria, all neurons corresponding to the first input layer are considered as non-valid neurons since their values are null, and thus the first time ($t = 1$) is not taken into account when the final output of the model is computed.

3.2 Output Aggregation Functions

Once the basic algorithm steps for processing a problem instance have been defined, it is necessary to introduce a procedure for combining the results. The model output could be represented as a one-dimensional vector, where each position contains the membership probability for each class, taking into account network times (see Figure 3 for a better understanding). In literature, several approaches for aggregating

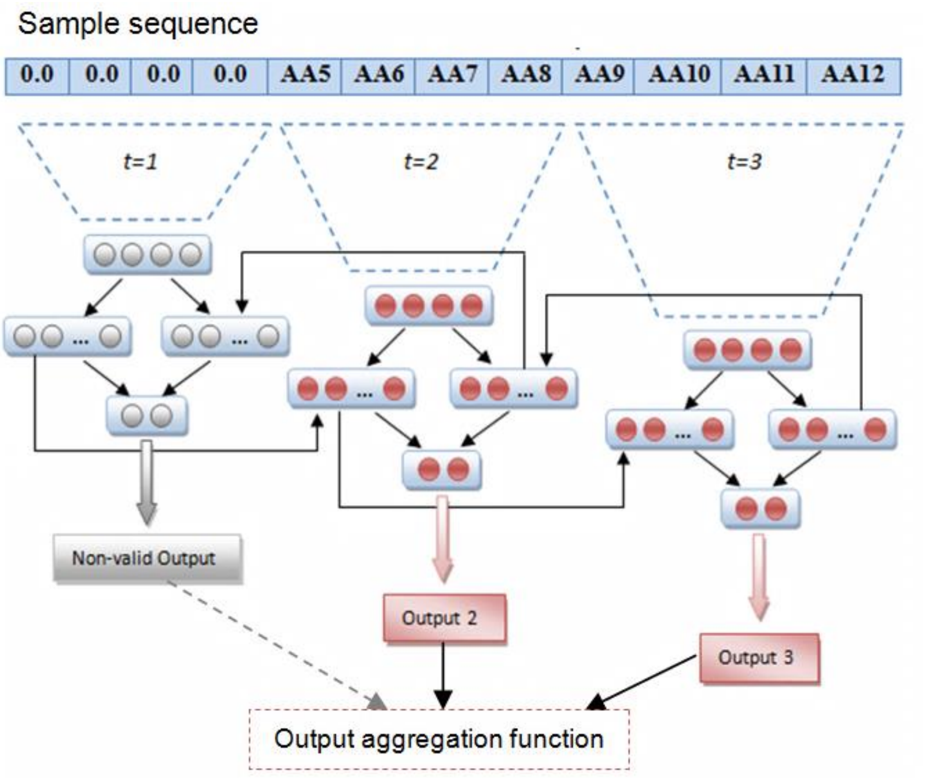

Fig. 2. Unfolded RNN for adapting the network topology according to the instance length

these values into a single output have been proposed and discussed. In this paper, we use four variants as follows:

1. **Average function:** (Fig. 3) Calculates the average of the probabilistic values associated with the class membership of the network outputs.
2. **Weighted average function:** Calculates the weighted average by assigning a weight to each network output. Here output weights are automatically determined taking into account the number of active neurons in the input layer. So, the more active neurons an input has, the higher is the weight associated to the input sequence and, thus, this sequence provides more information in the averaged output.
3. **Mode function:** Calculates the mode of times involved in the recurrent model. First, all values are grouped in similarity classes. Next, the average of the most popular similarity class (i.e., membership value) is returned.
4. **Median function:** This function calculates the median of all output values, i.e., the output value associated to the time located in the center of the network. Here, it is important to mention that this approach may be useful for RNNs with small number of times, since the center of the network generally concentrates a better information consistency.

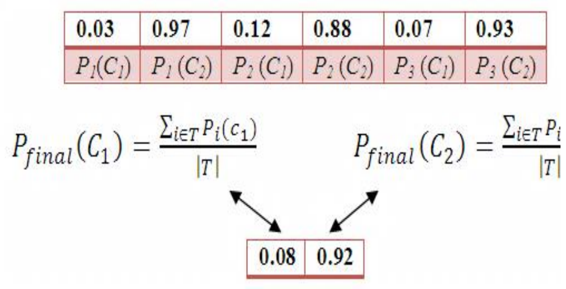
0.03
0.97
0.12
0.88
0.07
0.93
$P_1(C_1)$
$P_1(C_2)$
$P_2(C_1)$
$P_2(C_2)$
$P_3(C_1)$
$P_3(C_2)$
$P_{final}(C_1) = \frac{\sum_{i \in T} P_i(c_1)}{|T|}$
$P_{final}(C_2) = \frac{\sum_{i \in T} P_i}{|T|}$
0.08
0.92

Fig. 3. Average aggregation function

According to the logic of the learning scheme proposed in Section 3.1, the aggregation functions use only the network times which are partially or fully activated, that is, those network times which have all input neurons or at least one of them as active neuron. Clearly the appropriate choice of the output aggregation function may be a relevant aspect which will be explored and assessed in future works, in order to maximize the performance of the proposed learning algorithm.

4 Experimental Results and Real-World Applications

In this section, we evaluate the performance of the proposal through a classification problem with instances (cases) of variable length, in a bioinformatics context. We also describe another application in the field of seismology.

4.1 Sequence Classification for HIV Drug Resistance Problems

The Human Immunodeficiency Virus (HIV) is a complex and dynamical *Lentivirus* plaguing humanity harshly and causing millions of deaths yearly. Most HIV antiviral drugs are designed to inhibit the function of three essential proteins in the virus replication process: protease, reverse transcriptase and integrase. However, due to a high mutation rate, this virus is capable to develop resistance to existing drugs, eventually causing treatment failure. Thus, modeling of resistance mechanisms requires a study of viral genome for designing more effective therapies using existing drugs [20].

Resistance testing can be performed either by measuring viral activity in the presence and absence of a drug (phenotypic resistance testing), or by sequencing the viral genes coding for the drug targets (genotypic resistance testing). Genotypic assays are much faster and cheaper, but sequence data provide only indirect evidence of resistance [21], giving a limited knowledge of the HIV behavior. A combination of both tests in genotype-phenotype pairs is available in well-known international databases such as Stanford HIV Resistance Database [22]. These knowledge bases have been used for predicting the resistance of a mutation to a certain antiviral drug from its primary structure, i.e., the genotypic information.

Particularly, reverse transcriptase mutations are characterized by frequent insertions and

Table 1. Average classification accuracy over 20 runs of a 10-fold cross-validation process for each algorithm configuration. Best performances are emphasized in boldface

Inhibitor	*BPTT+Av*	*BPTT+Wav*	*BPTT+Mo*	*BPTT+Mi*	*dBPTT+Av*	*dBPTT+Wav*	*dBPTT+Mo*	*dBPTT+Mi*
3TC	57.3408	56.3655	57.4465	57.5449	**70.1357**	65.0374	67.0926	67.8464
ABC	53.1759	55.2475	52.1246	52.9841	59.3190	**61.7147**	57.3658	58.5934
AZT	49.9559	48.2569	47.9985	47.2565	**56.9686**	54.6425	52.7522	53.1586
D4T	88.5523	87.2698	87.1459	85.2476	**89.4904**	88.7333	88.4619	86.8904
DDC	67.4348	68.7548	66.8976	64.6541	77.6452	**78.8376**	77.4799	74.8196
DDI	67.2434	64.2596	66.4169	65.2547	**81.3526**	77.3775	81.3153	79.2164
DLV	52.6293	50.2586	52.3874	53.1247	**58.8922**	52.5344	57.9353	57.6379
EFV	52.3094	51.3659	51.4785	51.6459	**56.9358**	55.9735	55.1132	55.5584
FTC	72.3375	72.1124	70.2146	71.4569	**86.9945**	86.5342	86.0649	84.8285
NPV	56.1741	52.4796	53.5469	53.2469	**65.0650**	60.6459	61.5932	61.4667
TDF	71.8446	71.2456	70.8654	70.3297	82.5589	**83.2663**	81.5603	79.7850

deletions; also, they are not completely sequenced in the available literature. For these reasons, instances of knowledge bases associated to this protein are highly variable in length.

4.2 Experimental Results

A statistical experiment was designed involving 11 knowledge bases corresponding to drug resistance in the following reverse transcriptase inhibitors: lamivudina (3TC), abacavir (ABC), zidovudina (AZT), estavudina (d4T), zalcitabina (ddC), didanosina (ddI), delavirdina (DLV), efavirenz (EFV), emtricibina (FTC), nevirapina (NPV) and tenofovir (TDF), extracted from [22]. All sequences were previously aligned, and the non-sequenced fragments of the mutations were represented with null values. The contact energy [23] of an amino acid was used as a numerical descriptor that corresponds to the 3D structure of proteins, and it can be calculated statistically from a large number of diverse sequences.

In this experiment we compared the performance in terms of classification accuracy between the classic implementation of BPTT and the proposed one (dBPTT), using the four output aggregation functions described in Subsection 3.2. The best network topology for each knowledge base was determined by successive approximations. Table 1 shows the average accuracy over 20 runs of a 10-fold cross-validation process for each algorithm configuration. In general, the best results were achieved by the dynamic BPTT using the average as an output aggregation function.

In order to prove this, we computed the Friedman test (Friedman two-way analysis of variances by ranks) [24, 25]. This test is a multiple comparison procedure capable to detect significant differences between the behaviors of two or more algorithms; i.e., it can be used for discovering whether at least two of the samples represent populations with different median values or not, in a set of n samples ($n \geq 2$).

Table 2 shows the mean rank and the p-value associated with this test for each algorithm configuration. Using a significance level of 0.05, corresponding to the 95% confidence interval, the Friedman test suggests rejecting the null hypothesis (p-value < 0.05), thus, there exist highly significant differences between at least two methods across inhibitors. Also, it can be observed that dBPTT+Av and dBPTT+Wav are best ranked; however, this information cannot be used to conclude that our proposals are involved in this differences; that is why we also applied the Wilcoxon signed ranks test [26]. This is a pairwise procedure that aims to detect significant differences between two sample means, that is, the behavior of two algorithms. Using a significance level of 0.05, corresponding to the 95% confidence interval, the Wilcoxon test suggests to reject the null hypothesis (p-value < 0.05) for the following pairwise comparisons: BPTT+Av vs. dBPTT+Av, BPTT+Wav vs. dBPTT+Wav, BPTT+Mo vs. dBPTT+Mo, BPTT+Mi vs. dBPTT+Mi. These results statistically confirm that the proposed dynamic BPTT is capable to handle instances of variable length in a more efficient manner than the classic approach. Specially, the average aggregation function seems to be the best variant for this application problem, followed by the weighted average function.

Table 2. Mean Rank achieved by the Friedman test

Evaluated Algorithms	***Mean Rank*** [a]
dBPTT+Av	7.73
dBPTT+Wav	6.36
dBPTT+Mo	5.91
dBPTT+Mi	5.45
BPTT+Av	3.82
BPTT+Wav	2.36
BPTT+Mi	2.18
BPTT+Mo	2.18

[a] Monte Carlo signification (p-value) = 0.000

In this way we have explored the algorithm behavior in a bioinformatics classification problem which involves instances of variable length. In future work this proposal will be deeper assessed by including more statistical experimentation using other HIV inhibitors and descriptors. Due to flexibility of the proposed implementation, future work could be also focused on using this model in other science or engineering problems. For example, it could be used for obtaining multiple earthquake accelerograms from a given scenario. These scenarios are signals with variable length, and the resultant multiple signals could help in the structural vulnerability evaluation of different kinds of buildings exposed to seismic hazard.

5 Conclusions

In this paper, we presented a novel modification of the Backpropagation Through Time algorithm for training RNNs where problem instances have variable length. The proposed model was evaluated through a bioinformatics study case for classifying several HIV mutations which are resistant or not resistant to a given antiviral drug.

Particularly, we studied HIV resistance mechanisms when different mutations of reverse transcriptase are detected, taking into account 11 well-known drug inhibitors. In this context, the statistical tests show that the modified algorithm significantly outperforms the classic Backpropagation through Time algorithm, using the prediction (classification) accuracy as quality measure.

Based on the proposal's flexibility, we also plan to develop an application for solving problems in the synthetic earthquakes prediction field, where the presented algorithm will be extensively evaluated.

References

1. **Garcia, M.M., Bello, R., Diaz, A. & Reynoso, A. (2003).** *Redes Neuronales Artificiales*. Mexico: Prometeo Editores. Retrieved from http://148.202.105.23:8991/F/8TGDD3KRGKKHD2X8L9M8EHYA9HKBS7CSJHT9FE5Y4BKDGM2UUT-01381?func=item-global&doc_library=UDG01&doc_number=000219225&year=&volume=&sub_library=EIC.
2. **Hammer, B. & Villmann, T. (2003).** Mathematical Aspects of Neural Networks. *11th European Symposium on Artificial Neural Networks* (*ESANN 2003)*, Bruges, Belgium, 59–72.
3. **Pearlmutter, B.A. (1990).** *Dynamic Recurrent Neural Networks (CMU-CS-90-196)*. Pittsburgh, PA: Carnegie Mellon University.
4. **Baldi, P., Pollastri, G., Frasconi, P., & Vullo, A. (2003).** New Machine Learning Methods for the Prediction of Protein Topologies. In Paolo Frasconi & Ron Shamir (Eds.). *Artificial Intelligence and Heuristic Methods in*

Bioinformatics (51-74). Amsterdam; Washington, DC: IOS Press.

5. **Draghici, S. & Potter, R.B. (2003).** Predicting HIV drug resistance with neural networks. *Bioinformatics*, 19(1), 98–107.
6. **Bonet, I., Garcia, M.M., Saeys, Y., Peer, Y., & Grau, R. (2007).** Predicting Human Immunodeficiency Virus (HIV) Drug Resistance Using Recurrent Neural Networks. *2nd International Work-Conference on The Interplay Between Natural and Artificial Computation (IWINAC'07)*. *Lecture Notes in Computer Science*, 4527, 234–243.
7. **Beerenwinkel, N., Lengauer, T., Selbig, J., Schmidt, B., Walter, H., Korn, K., Kaiser, R., & Hoffmann, D. (2001).** Geno2pheno: interpreting genotypic HIV drug resistance tests. *IEEE Intelligence System*, 16(6), 35–41.
8. **Beerenwinkel, N., Schmidt, B., Walter, H., Kaiser, R., Lengauer, T., Hoffmann, D., Korn, K., & Selbig, J. (2002).** Diversity and complexity of HIV-1 drug resistance: a bioinformatics approach to predicting phenotype from genotype. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12), 8271–8276.
9. **Bonet, I., Rodríguez, A., Grau, R., García, M.M., & Izquierdo, Y. (2008).** Predicting HIV protease drugs resistance with multiclassifier systems. *XVII International AIDS Conference, Mexico City.*
10. **Bonet, I., Rodríguez, A., García, M.M., & Grau, R. (2012).** Combinación de clasificadores para bioinformática. *Computación y Sistemas*, 16(2), 191–201.
11. **Grau, I., Nápoles, G., León, M., & Grau, R. (2012).** Fuzzy Cognitive maps for modelling, predicting and interpreting HIV drug resistance. *Lecture Notes in Computer Science*, 7637, 31–40.
12. **Nápoles, G., Grau, I., León, M. & Grau, R. (2012).** Modelling, aggregation and simulation of a dynamic biological system through Fuzzy Cognitive Maps. *11th Mexican International Conference on Artificial Intelligence, (MICAI 2012) Part II. Lecture Notes in Artificial Intelligence*, 7630, 187–197.
13. **Rhee, S., Taylor, J., Wadhera, G., Ben-Hur, A., Brutlag, D.L., & Shafer, R.W. (2006).** Genotypic predictors of human immunodeficiency virus type 1 drug resistance. *Proceedings of the National Academic of Sciences of the United States of America*, 103(46), 17355–17360.
14. **Kierczak, M., Ginalski, K., Dramiński, M., Koronacki, J., Rudnicki, W., & Komorowski, J. (2009).** A Rough Set-Based Model of HIV-1 Reverse Transcriptase Resistome. *Bioinformatics and Biology Insights*, 5(3), 109–127.
15. **Vermeiren, H., Van den Bulcke, T., Van Marck, H., Lecocq, P., Van Houtte, M., & Bacheler, L. (2004).** Application of multiple linear regression modelling to the quantitative prediction of HIV-1 drug susceptibility phenotype from viral genotype. *XIII International HIV Drug Resistance Workshop,* Costa Adeje, Canary Islands, Spain.
16. **Hilera, J.R. & Martínez, V.J. (1995).** *Redes Neuronales Artificiales: Fundamentos, Modelos y Aplicaciones.*, Madrid: RA-MA.
17. **Bonet, I., Salazar, S., Rodríguez, A., Grau, R., & García, M.M. (2007).** Redes Neuronales Recurrentes para Análisis de Secuencias. *Revista Cubana de Ciencias Informáticas*, 1(4), 48–57.
18. **Atiya, A.F. & Parlos, A.G. (2000).** New Results on Recurrent Network Training: Unifying the Algorithms and Accelerating Convergence. *IEEE Transactions on Neural Networks,* 11(3), 697–709.
19. **Werbos, P.J. (1990).** Backpropagation Through Time: What it does and How to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.
20. **Prosperi, M. & Ulivi, G. (2008).** Evolutionary Fuzzy Modelling for Drug Resistant HIV-1 Treatment Optimization. *Studies in Computational Intelligence*, 82, 251–287.
21. **Beerenwinkel, N., Sing, T., Lengauer, T., Rahnenfuhrer, J., Roomp, K., Savenkov, I., Fischer, R., Hoffmann, D., Selbig, J., Korn, K., Walter, H., Berg, T., Braun, P., Fatkenheuer, G., Oette, M., Rockstroh, J., Kupfer, B., Kaiser, R., & Daume, M. (2005).** Computational methods for the design of effective therapies against drug resistant HIV strains. *Bioinformatics,* 21(21), 3943–3950.
22. **22.** HIV Drug Resistance Database (2012). Retrieved from http://hivdb.stanford.edu//cgi-bin/GenoPhenoDS.cgi
23. **Miyazawa, S. & Jernigan, R.L. (1999).** Self-Consistent Estimation of Inter-Residue Protein Contact Energies Based on an Equilibrium Mixture Approximation of Residues. *Proteins: Structure, Function, and* Bioinformatics, 34(1), 49–68.
24. **Friedman, M. (1937).** The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200), 675–701.
25. **Friedman, M. (1940).** A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics*, 11(1), 86–92.

26. **Wilcoxon, F. (1945).** Individual comparisons by ranking methods. *Biometrics Bulletin,* 1(6), 80–83.

Isel Grau received the B.Sc. degree (with honors) in Computer Science from the Universidad Central "Marta Abreu" de Las Villas (UCLV), Santa Clara, Cuba, in 2011. She is with the Bioinformatics Lab of the Center of Studies on Informatics, UCLV, where she currently works toward the M.Sc. degree. She has authored/coauthored 7 international conference papers. She earned the Cuban National Award for Computer Science Students in 2010, the Cuban National Award for Best Diploma Thesis in 2011, the Cuban Academy of Sciences Award in 2011 and the Best Paper Award Third Place at MICAI 2012 conference. Her research interests include Soft Computing, Bioinformatics, Neural Networks, Machine Learning, Statistics and Knowledge Discovery.

Gonzalo Nápoles received the B.Sc. degree (with honors) in Computer Science from the Universidad Central "Marta Abreu" de Las Villas (UCLV), Santa Clara, Cuba, in 2011. He is with the Artificial Intelligence Lab of the Center of Studies on Informatics, UCLV, where he is currently works toward the M.Sc. degree. He has authored/coauthored a monograph and 9 international conference papers. He earned the Cuban National Award for Computer Science Students twice (2010 and 2011) and the Best Paper Award Third Place at MICAI 2012 conference. His research interests include Soft Computing, Metaheuristics, Evolutionary Computation, Knowledge Engineering, Machine Learning and Decision Making.

Isis Bonet received the B.Sc. degree in Computer Science from the Universidad Central "Marta Abreu" de Las Villas (UCLV), Santa Clara, Cuba, in 2001, her M.Sc. degree in Computer Science at UCLV in 2005 and her Ph.D. in Technical Sciences at UCLV in 2009. She is currently with the Escuela de Ingeniería de Antioquia, Colombia. She has authored/coauthored 42 papers in conference proceedings and scientific journals and earned several awards including the Cuban Academy of Sciences Award in 2011. Her research interests include Neural Networks, Classification Problems and Bioinformatics.

Maria Matilde Garcia received the B.Sc. degree in Mathematical Cybernetics from the Universidad Central "Marta Abreu" de Las Villas (UCLV), Santa Clara, Cuba, in 1985 and her Ph.D. in Technical Sciences at UCLV in 1997. She is a full professor at the Artificial Intelligence Lab of the Center of Studies on Informatics, UCLV, where she is also the General Coordinator of the Master in Computer Science Program (AUIP awarded). Also, she exhibits a long record of academic exchange with many universities in Latin America and Europe. Dr. Garcia has authored 6 books, published over 150 papers in conference proceedings and scientific journals and supervised several Bachelor, Master and Ph.D. thesis. She has deserved numerous prestigious awards from the Cuban Academy of Sciences and other renowned scientific societies. Her research interests include Neural Networks, Soft Computing, Machine Learning and Pattern Recognition.

Article received on 05/10/2012; accepted on 08/12/2012.