International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:10, No:12, 2016

Algorithm and Software Based on Multilayer Perceptron Neural Networks for Estimating Channel Use in the Spectral Decision Stage in Cognitive Radio Networks

Danilo López, Johana Hernández, Edwin Rivas

***Abstract*—**The use of the Multilayer Perceptron Neural Networks (MLPNN) technique is presented to estimate the future state of use of a licensed channel by primary users (PUs); this will be useful at the spectral decision stage in cognitive radio networks (CRN) to determine approximately in which time instants of future may secondary users (SUs) opportunistically use the spectral bandwidth to send data through the primary wireless network. To validate the results, sequences of occupancy data of channel were generated by simulation. The results show that the prediction percentage is greater than 60% in some of the tests carried out.

***Keywords*—**Cognitive radio, neural network, prediction, primary user.

I. Introduction.

ONE of the main problems in the insertion of new applications in the transport structure of cognitive radio is associated with the inefficient distribution of available spectrum by the government bodies [1]. That is, there are currently licensed regions where the radio spectrum is underused (bands VHF/UHF) [2], [3] and other spectral regions (cellular bands) where there has been a degradation in the quality of service.

Several researchers, including Mitola [4], have concluded that dynamic spectrum access (DSA) is a good strategy to address the issue of administering electromagnetic bands. In [4], Mitola raises the possibility that his administration is performed dynamically through Cognitive Radio (CR). Such administration includes spectral sensing, decision, sharing and mobility within CR. The spectral decision phase is responsible for selecting the best channel for transmitting the SU from the estimated future use of the band licensed by the PU; in this sense the successful selection of the best bands will depend on how accurate is the future appearance of PU; if the accuracy rate is high, spectral decision making will be optimal; if instead the percentage of prediction is not good enough, collisions between PU and SU will occur, a condition that is not acceptable for telecom operators. Supported by previous appreciations, the article proposes the use of the MLPNN artificial intelligence technique, in order to model and predict the future state of the PU on a channel by generating simulated data and determine whether or not it is capable of being used as a methodology to characterize licensed users in CR.

Danilo López S is with the Universidad Distrital "Francisco José de Caldas", Faculty of Engineering, Cra. # 40-53, Bogotá, Colombia (Corresponding Author; phone: +573108651144; e-mail: dalopezs@udistrital.edu.co).

Edwin Rivas Trujillo is with the Universidad Distrital "Francisco José de Caldas", Faculty of Engineering, Cra. 7 # 40-53, Bogotá, Colombia (e-mail: erivas@udistrital.edu.co).

Johana Hernández is with the Universidad Manuela Beltran, Faculty of Engineering, Avenida Circunvalar 60-00, Bogotá, Colombia (e-mail: ing_joha_h@hotmail.com).

II. Artificial Neural Networks

Artificial Neural Networks (ANN) are computer models that emerged as an attempt to achieve mathematical formalizations about the structure of the brain. These mimic the structure of the nervous system, focusing on the functioning of the human brain, based on learning through experience, thereby extracting knowledge from it. An ANN can be considered a mathematical model of the theories of mental and brain activities, based on the exploitation of the parallel local processing and properties of distributed representation [5]. As for the mathematical model, the ANN emulates the synaptic processes through mathematical functions including functions of propagation, activation and transfer.

MLPNN: It was introduced by Frank Rosenblatt in 1958 based on the model of Mc-Culloch & Pitts and the error correction learning rule. His intention was to illustrate some fundamental properties of intelligent systems in general, without going into further details regarding specific and unknown conditions for specific biological organisms. The first Perceptron model was developed in a biological environment mimicking how the human eye works, hence its name perceptron, [6]. In this type of neural networks, the number of inputs are discrete and the activation function for each neuron corresponds to a step type [7].

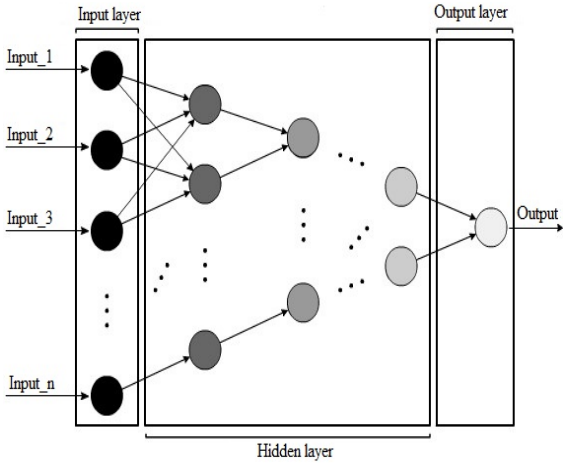

Fig. 1 MLPNN Neural System

A simple perceptron generates a decision region which ranks a set of points by separating them. The decision region divides the space into two halves by a certain hyperplane defined by the synaptic weights obtained during the training process. The synaptic weights are calculated by the error correction algorithm, which calculates the difference between the value obtained after training and the expected value; and multiplies it by the gradient of the hyperplane updating the decision region with each training example. A set of simple perceptrons generate a multi-layer perceptron, which allows to divide the region into more than two halves generating a hyperplane able to classify several linearly independent points [8].

In Fig. 1, the general structure of a pyramidal multilayer perceptron is shown by a directed graph, nodes represent neurons and edges correspond to connections between neurons. Neurons are separated into groups known as layers and each neuron on a layer is connected to all the neurons on the next layer. The layers between the input and output nodes are known as hidden layers, and they contain neurons that inhibit or excite the components of the next layer to give a result to an input entered through the input layer and direct the result to the output layer.

The number of hidden layers depends on the selected topology (which in our case, geometric pyramidal), but if there is no topology available, it must be remembered that for every aggregate neuron a hyperplane is created capable of separating a set of linearly independent points. When a hyperplane is created for each point it is said that the neural network is specialized, i.e., it is only capable of classifying the points of its training and not another set of points. Moreover, the lack of neurons generates an inability to fully differentiate the set of points. Determining the number of neurons to solve a problem is not an easy task, knowledge engineers often vary the number of neurons and evaluate the performance for each variation to find a topology that proves to solve the problem in the best way possible.

III. MLPNN Model for Characterization of a PU

The algorithm constructs a pyramidal neural network based on the channel occupancy history of a **PU**. Subsequently, the neural network is trained with the history presented and generates metrics for performance evaluation of the neural network. Fig. 2 shows the flowchart used to obtain the results of characterization of PUs, followed by the developed pseudo-code.

Algorithm MLPNN

Data: The existence of a W arrangement representing ANN.
Result: Neural network trained with the data from the training examples;

```
neurons = W.size () //The size of the arrangement that represents the neural network is obtained;
for i = 0; i < neurons; i++ do
    W[i] = random(−1,1) //Each neuron is covered and is started with a random number beetween -1 and 1;
end i < neurons; i++ do
bias = 0.5 //Approximation to the obtained output;
inputs = readInputs () //Reading of the inputs used for the training;
outputs = readOutputs () //Storing of the size of training examples;
size = inputs.size () //Storing of the size of training examples;
for i = 0; i < size; i++ do
    sum = 0;
    for j = 0; j < neurons; j++ do
        sum = sum + W[j] * inputs[i][j] //The ANN output is calculated for each one of the outputs;
    end
    output = hardlims(sum + bias) //The output is approximated using hardlims;
    if output! = outputs[i] then
        error = outputs[i] − output //The ANN output error is calculated with respect to the expected output, in case they are different;
        for j = 0; j < neurons; j++ do
            W[j] = W[j] + inputs[i][j] * e //each neuron is corrected and its weight is corrected with respect to calcuted error;
        end
        bias = 0.5 + error //Correction of approximation;
    end
end
```

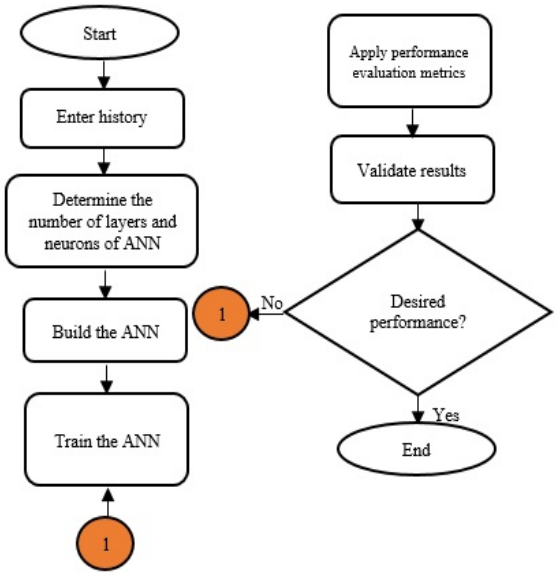

Fig. 2 Diagram of Step Sequence included in the MLPNN

Representation of channel occupancy history: For the algorithm to operate, the history must be represented as a character string consisting of 1 and 0, where each string position (Fig. 3) represents a unit of time, 1 means the channel was being occupied by the **PU** and 0 the opposite, as can be evidenced.

0	1	1	0	1	0	0	0	1	1

Fig. 3 Representation of Use Occupancy history of the spectrum of a PU

In order to facilitate pattern recognition, time units are represented in binary system. Therefore, the highest position of the history must be taken and convert it to binary system for determining the length of the strings to be introduced into the neural network, this length represents the number of neurons in the input layer of the neural network. Taking the example shown in Fig. 3, the chain length corresponding to 10 therefore in binary representation is 1010 and the number of neurons in the input layer corresponds to 4.

Training and prediction of the neural network: To train the neural network used 70% of the PU behavioral history. After this process and in order to evaluate system performance the following variables were measured (among others):

Error and training. It indicates the degree of correction to be performed on learning using cross entropy.

$$E_e = \sum_{x=0}^{n} -(t(x) * \log(o(x)) + (1 - t(x)) * \log(o(x))) \quad (1)$$

where $t(x)$ and $o(x)$ is the historical value and the expected output at time x respectively; n is the size of the chain.

Validation error. Metrics that gives the likelihood of success of the outputs of the neural network in the modeling phase.

$$E_v = \frac{\sum_{x=0}^{n} t(x) - \sum_{x=0}^{n} o(x)}{n} * 100 \quad (2)$$

To validate the level of prediction the rest of the string (30%) was used and the metric called prediction error was evaluated, which yields the probability of success on a prediction made by the neural network.

IV. Software Implementation

With the _n to test the presented model, a program was developed in the programming language C # and whose use case diagram is shown in Fig. 4.

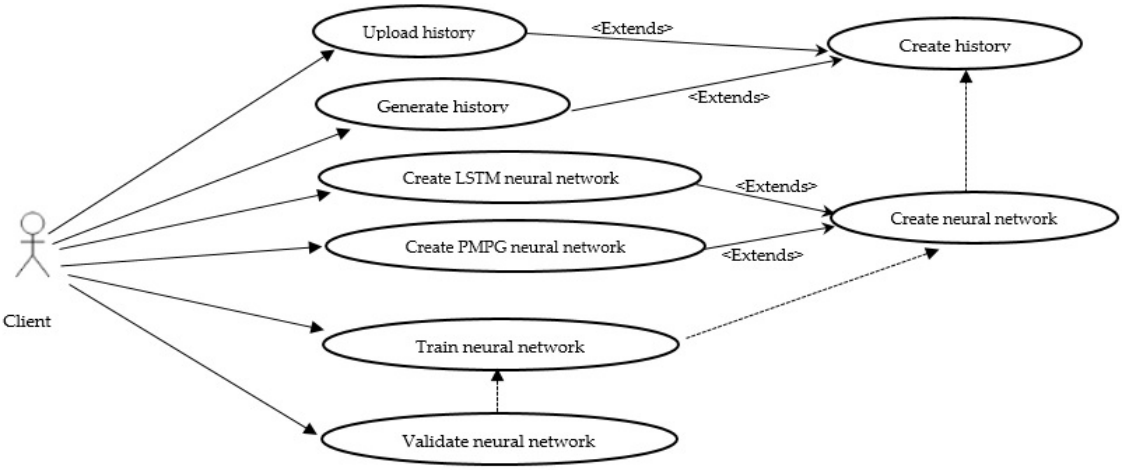

Fig. 4 Software characterization of PUs

In Fig. 4, the dependence between each of the steps of the algorithm is evidenced, it is not possible to train a neural network without having created it first. Use cases "Upload history" and "Generate history" are an extension of "Create history" and only one of the two may be used at the same time. Fig. 5 shows the create history phase of the application when the user clicks the "Upload file" button it may upload a "CSV" file that represents the channel occupancy history. In case of not

having a user history you may create one by entering a pattern for the distribution of 1 and 0, along with the size thereof.

Fig. 5 PU history creation phase

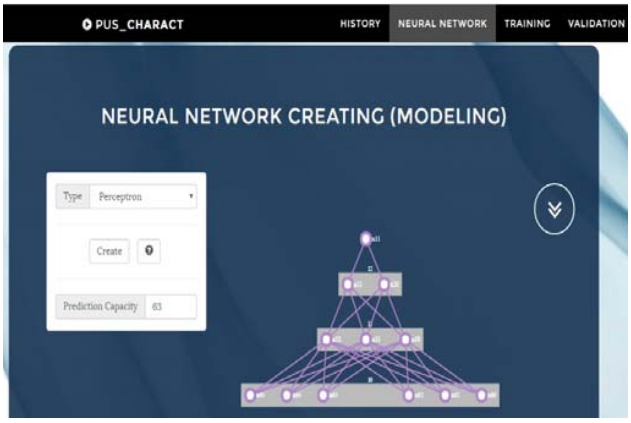

Fig. 6 ANN Creation phase

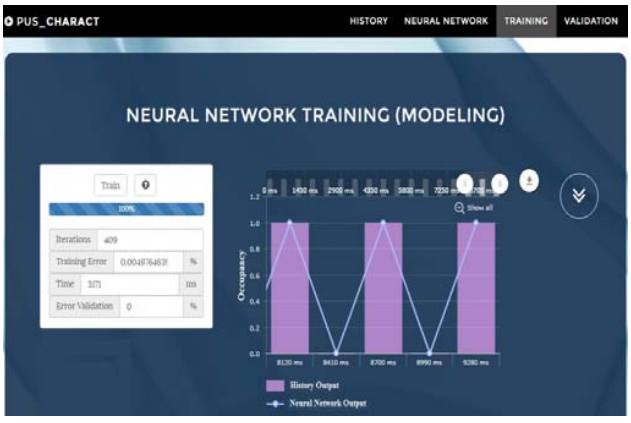

Fig. 7 ANN Training phase

In Fig. 6, a screenshot of the second phase of the algorithm is shown, in which a ANN is created in accordance with the input sequence (note that the res topology is not static but dynamic and adapts to the input matrix).

The third stage of the algorithm can be seen in Fig. 7. Here the user trains the network and the training progress is shown in the left bar.

After the process, the program (Fig. 7) shows the number of iterations, training error, runtime and validation error. On the right side of the screen the history is displayed through means of bars and the output of the neural network represented by points connected by a line.

The last phase of the algorithm corresponds to the prediction (Fig. 8). The software will display a graph with the prediction generated if a pattern and size of history has been generated, or 30% of the entered data in case of having uploaded a file. The graph shows the history through bars and the output of the neural network represented by points connected by a line.

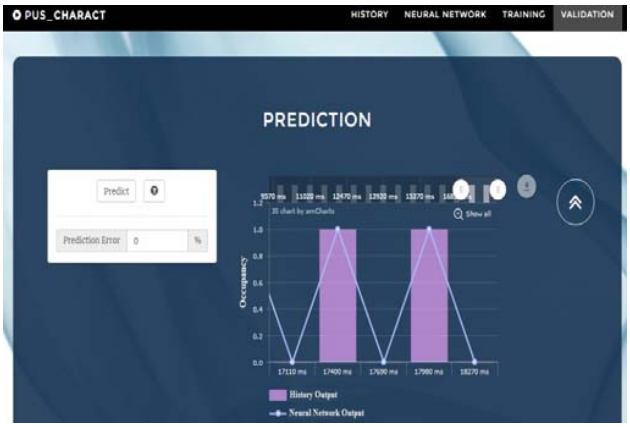

Fig. 8 ANN Prediction phase

Fig. 9 shows the class diagram for the "Predictor class" used for developing the algorithm and implementing all the steps shown in Fig. 2. The first row of the graph corresponds to the class name, the second describes uts attributes and finally the name of the methods used by the class.

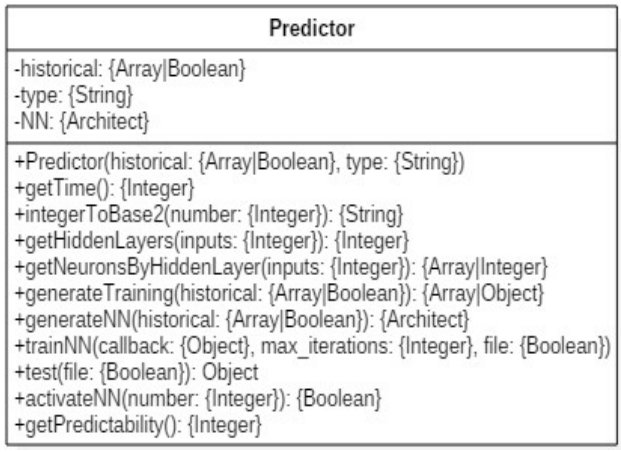

Fig. 9 "Predictor" class Diagrams

V. Results

For the performance evaluation of the application, case

studies were created with multiple data stream sizes which are described with their respective results in Tables I and II. Note that the behavior shown by the neural system is given quantitatively for lack of space.

TABLE I
QUANTITATIVE RESULT FOR THE NEURAL SYSTEM (CASE STUDY 1)

Size of sample	Number of iterations	Processing time (ms)	Validation error	Prediction error
5	1594.5	78.9	0.004977128	0.633333333
9	5314.2	244.1	0.00498651	0.514285714
17	4491.8	517.2	0.0051283	0.513333333
33	4702.4	943.1	0.005075479	0.480645161
65	5139.3	2263.5	0.005051068	0.492063492
Average	4248.44	809.36	0.005043697	0.526732207

TABLE II
QUANTITATIVE RESULT FOR THE NEURAL SYSTEM (CASE STUDY 2)

Size of sample	Number of iterations	Processing time (ms)	Validation error	Prediction error
5	7335	222.7	0.006090868	0.333333333
9	42757	1481.6	0.247637751	0.428571429
17	16712	1559.1	0.005008472	0.36
33	39373	6754.4	0.020299447	0.364516129
65	35776	13838.8	0.248377332	0.393650794
Average	28390.6	4771.32	0.105482774	0.376014337

VI. CONCLUSIONS

The inclusion of artificial intelligence techniques such as MLPNN, for characterizing the use of the licensed channel from the point of view of PU, are a good complement in CRN because they provide some learning capability that can be exploited by cognitive nodes (preferably in distributed networks) to decrease the likelihood of generating interference or collisions with PUs, without having to resort to the development of complex mathematical processes for implementation.

While predicting a percentage of 60% in the characterization with MLPNN is a good result; for this important step within CR spectral decision to work properly, it is necessary to find additional strategies or methodologies that increase future approximation to higher values, in order to make the system even more efficient. Paradigms as metaheuristics and evolutionary algorithms, Ant Colony Optimization (ACO), neural diffusion systems, Long Short-Term Evolution (LSTM) will be tested in order to determine their efficiency in describing the behavior of Pus in CRN.

REFERENCES

[1] A. Shukla; et al. QinetiQ Ltd; Cognitive Radio Technology - A study for OFCOM. (Online), retrieved June 10 of 2013, http://enstakeholders.ofcom.org.uk/binaries/research/technologyresearch/cograd_main.pdf.

[2] Federal Communications Commission, Spectrum policy task force report, Tech. Rep. ET Docket 02-155, 2010.

[3] T. Taher; R. Bacchus; K. Zdunek; D. Roberson. Long-term spectral occupancy findings in Chicago, in Proc. IEEE International Symposium Dynamic Spectrum Access Networks (DySPAN), Aachen, Germany, 2011.

[4] J. Mitola. Software radios: survey, critical evaluation and future directions," IEEE Aerospace Electronic Systems Magazine, Vol. 8, pp. 25–31, 1993.

[5] R. López; J. Fernández. Las redes neuronales artificiales: Fundamentos teóricos y aplicaciones prácticas. Ed. Netbiblo, ISBN: 978-84-9745-246-5, Spain, 2008.

[6] L. Zerpa. Fundamentos lógicos de las Redes Neuronales Artificiales: Reconstrucción estructuralista del perceptron de una y dos capas, Ed: CEP-FHE, ISBN: 980-00-1926-X, 2001.

[7] D. Levine. Introduction to neural and cognitive modeling, Ed: Taylor & Francis, ISBN-13: 978-0805820058, 2009.

[8] I. Shmulevich; W. Zhang. Computational and Statistical Approaches to Genomics, Ed: Springer, ISBN: 978-0-387-26288-8, 2007.