International Journal of Systems Signal Control and Engineering Application 10 (1-6): 1-6, 2017
ISSN: 1997-5422

Algorithm and Software Based on MLPNN for Estimating Channel Use in the pectral Decision Stage in Cognitive Radio Networks

[1]Johana Hernandez Viveros and [2]Danilo Lopez Sarmiento
[1]Faculty of Economics and Administration, International University La Rioja,
Av. de la Paz, 137, 26006 Logrono, La Rioja, Espana, Spain
[2]Faculty of Engineering, Distrital University "Francisco Jose de Caldas" Cra. no 40B-53,
Bogota, Colombia

Abstract: The use of the Multilayer Perceptron Neural Networks (MLPNN) technique is proposed 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 they were generated by simulation, channel occupancy data sequences. The results show that the prediction percentage is >60% in some of the tests carried out.

Key words: Cognitive radio, neural network, prediction, primary user, secondary user

INTRODUCTION

Some studies show that 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 (Shukla *et al*., 2006). That is there are currently licensed regions where the radio spectrum is underused (bands VHF/UHF) (FCC, 2010; Taher *et al*., 2011) and other spectral regions (cellular bands) where there has been a degradation in the quality of service.

Several researchers including mitola and akyildiz have concluded that Dynamic Spectrum Access (DSA) is a good strategy to address the issue of administering electromagnetic bands. Mitola (1993) they raise 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 Secondary User (SU) from the estimated future use of the band licensed by the Primary User (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 study 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.

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, there by 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 (Lopez and Fernandez, 2008). 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 and 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

Corresponding Author: Johana Hernandez Viveros, Faculty of Economics and Administration, International University La Rioja, (Av. de la Paz, 137, 26006 Logrono, La Rioja, Espana, Spain

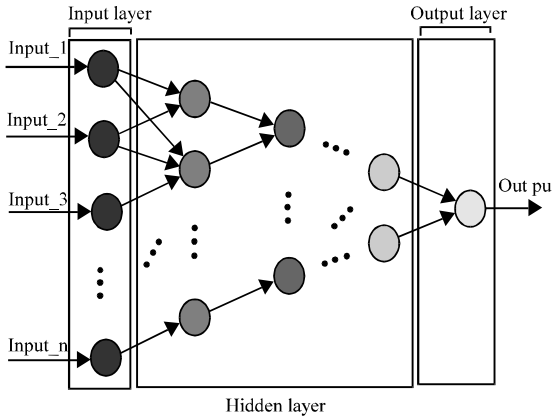

Fig. 1: MLPNN neural system

organisms. The first perceptron model was developed in a biological environment mimicking how the human eye research, hence its name perceptron (Zerpa, 2001). In this type of neural networks the number of inputs are discrete and the activation function for each neuron corresponds to a step type (Levine, 2009).

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 >2 halves generating a hyperplane able to classify several linearly independent points (Shmulevich and Zhang, 2007).

Figure 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 is 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.

MATERIALS AND METHODS

MLPNN Model for characterization of a PU: The algorithm constructs a pyramidal neural network based on the channel occupancy history of a primary user. Subsequently, the neural network is trained with the history presented and generates metrics for performance evaluation of the neural network. Figure 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 beet-
                            ween -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 = read output () //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 out-put is calcula-ted
                                                for each one of the outputs;
end
Output = hardlims (sum+bias)//The output is approximated using hardlims;
if out put! = outputs [i] then
Error = outputs [i]-putputs//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 calcu-ted error;
    end
    Bias = 0.5+error //correction of approximation;
  end
end
```

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

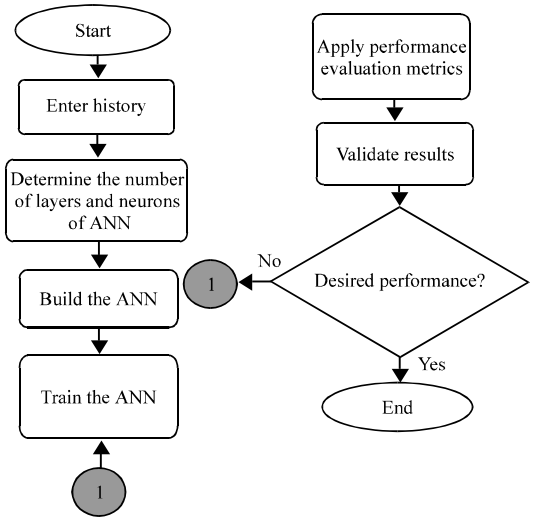

Fig. 2: Diagram of step sequence included in the MLPNN

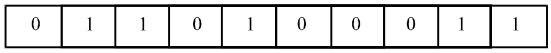

Fig. 3: Representation of use occupancy history of the spectrum of a PU

position (Fig. 3) represents a unit of time, 1 means the channel was being occupied by the primary user and 0 the opposite as can be evidenced.

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:

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

Where:

t(x) and 0(x) = The historical value the expected output at time x respectively

n = The size of chain

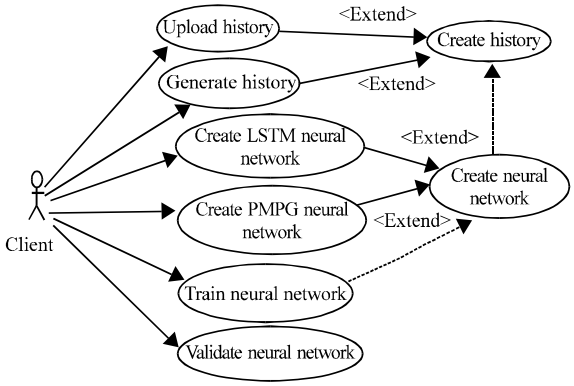

Fig. 4: Software characterization of PUs

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

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

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.

Software implementation: With the _n to test the proposed model, a program was developed in the programming language C no and whose use case diagram is shown in Fig. 4.

Figure 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. Figure 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.

Figure 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.

Fig. 5: PU history creation phase

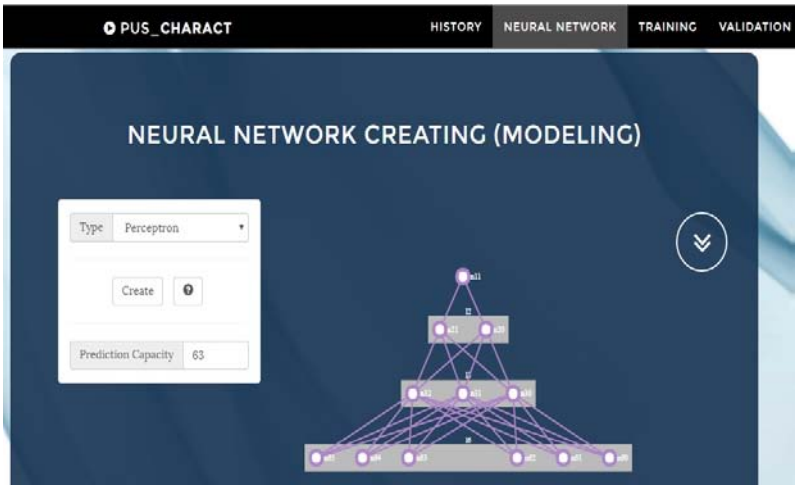

Fig. 6 ANN Creation phase

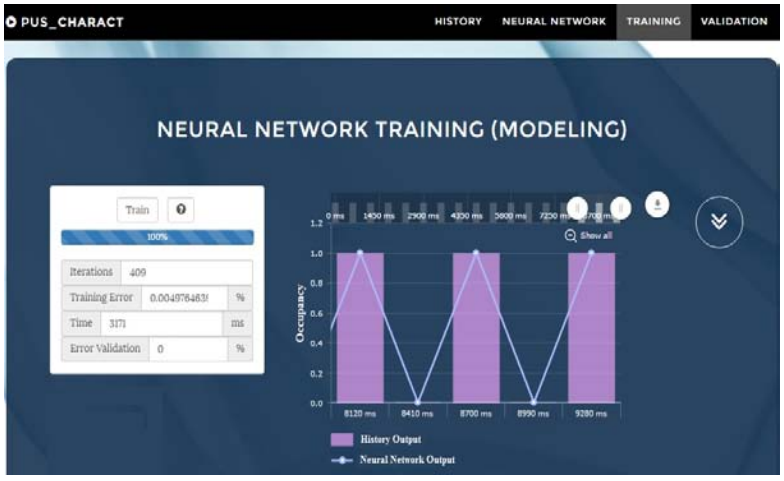

Fig. 7 ANN Training phase

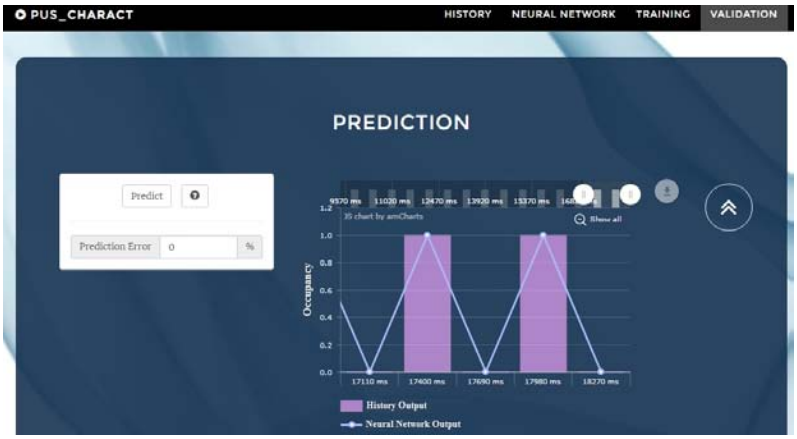

Fig. 8: ANN prediction phase

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.

Algoritham 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.

Algoritham (Predictor):

-historical: {Array|Boolean}
-Type: {String}
-NN: {Architect}
+Predictor (historical: {Arry|Boolean}, type: {String})
+get Time (): {Interger}
+Integer To Base 2 (number: {Ineger}): {String}
+get Hidden Layers (Inputs: {Integer}): {Array|Integer}
+get Neurons By Hidden Layer (inputs: {Integer}): {Array|Integer}
+generate Training (historical: {Array|Boolean}): {ArrayW
Object}
+generate Training (historical: {Array|Boolean}): {Architect}
+train NN (Callback: {Object}, max_iterations: {Integer}, file: {Boolean})
+test (file: {Boolean}): Objeact
+activate NN (number: {Integer}): {Boolean}
+get Predictability (): {Integer}

RESULTS AND DISCUSSION

For the performance evaluation of the application, case studies were created with multiple data stream sizes which are described with their respective results in Table 1 and 2. Note that the behavior shown by the neural system is given quantitatively for lack of space.

The results suggest that the model developed presents a better performance when the algorithm is able to find a channel usage behavior pattern predictable by the licensed user in the spectral band. Coming to have a pattern of success in the average prediction of 60% more or less.

Table: 1 Quantitative result for the neural system (case study 1)

Size of sample	Number of iterations	Processing time (msec)	Validation error	Prediction error
5	1594.50	0078.900	0.004977128	0.633333333
9	5314.20	0244.100	0.004986510	0.514285714
17	4491.80	0517.200	0.005128300	0.513333333
33	4702.40	0943.100	0.005075479	0.480645161
65	5139.30	2263.500	0.005051068	0.492063492
Average	4248.44	0809.360	0.005043697	0.526732207

Table 2: Quantitative result for the neural system (case study 2)

Size of sample	Number of iterations	Processing time (msec)	Validation error	Prediction error
5	7335.0	222.70	0.006090868	0.333333333
9	42757.0	1481.60	0.247637751	0.428571429
17	16712.0	1559.10	0.005008472	0.3600
33	39373.0	6754.40	0.020299447	0.364516129
65	35776.0	13838.80	0.248377332	0.393650794
Average	28390.6	4771.32	0.105482774	0.376014337

CONCLUSION

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 cognitive radio networks 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 Evoluction (LSTM) will be tested in order to determine their efficiency in describing the behavior of Pus in Cognitive Radio Networks (CRN).

REFERENCES

FCC., 2010. Spectrum policy task force report. Federal Communications Commission, Washington, DC., USA.

Levine, D., 2009. Introduction to Neural and Cognitive Modeling. Taylor & Francis, Abingdon, UK., ISBN-13:978-0805820058.

Lopez, R. and J. Fernandez, 2008. Artificial Neural Networks: Theoretical Basics and Practical Applications. Netbiblo Publisher, Spain, ISBN: 978-84-9745-246-5.

Mitola, J., 1993. Software radios: Survey, critical evaluation and future directions. IEEE. Aerosp. Electron. Syst. Mag., 8: 25-36.

Shmulevich, I. and W. Zhang, 2007. Computational and Statistical Approaches to Genomics. Kluwer Academic Publishers, New York, USA., Pages: 299.

Shukla, A., A. Alptekin, J. Bradford, E. Burbidge and D. Chandler *et al*., 2006. Cognitive radio technology a study for Ofcom. QinetiQ Company, Farnborough, UK.

Taher, T.M., R.B. Bacchus, K.J. Zdunek and D.A. Roberson, 2011. Long-term spectral occupancy findings in Chicago. Proceedings of the IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks, May 3-6, 2011, Aachen, Germany, pp: 100-107.

Zerpa, L., 2001. Logical Foundations of Artificial Neural Networks: Structuralist Reconstruction of the Perceptron of one and two Layers. CEP, Bayreuth, Germany, ISBN:980-00-1926-X.