A TOOLBOX FOR ANN LEARNING

Poliana Magalhães Reis[1] and Carlos Alberto Ynoguti[2]

Abstract *— Neural networks are abstract models of the human nervous system, and have multiple practical uses in several fields of knowledge like simulation of biological systems, economics, telecommunications, control engineering, among others. They are also becoming more and more present in our everyday lives, for example in automatic call centers, in the airports (user identification by the iris), etc. As a relatively new area, most of the literature available is quite complicated for undergraduate students. In order to provide some assistance to this audience, some Matlab® scripts were developed, covering basic concepts to train a multilayer perceptron neural network with the Backpropagation algorithm. A simple speech recognition system was constructed using these tools to illustrate its use.*

Index Terms — Artificial neural networks, Multilayer perceptron, Backpropagation.

INTRODUCTION

Intelligent systems are becoming part of our lives: user identification by iris or fingertips inspection, intelligent washing machines that decide the amount of soap and water temperature depending on the clothes conditions, automatic call centers, etc., are just some examples of this new trend.

Many of these systems have in their core an artificial neural network or another kind of intelligent system.

Unfortunately, most of the literature available in this field is directed to graduate students, and is written in a very complicated language. This can discourage undergraduate students from entering this exciting field of knowledge.

In order to help them in their first incursion in this area, some Matlab® scripts were developed. Although there exists several kinds of ANNs (multilayer perceptron, Kohonen, Radial Basis functions, etc.), scripts were developed for a multilayer perceptron only. For the future it's intended to create scripts for full ANNs in various architectures.

The organization of this work is as follows: initially a brief explanation of the main aspects of the multilayer perceptron ANN are presented, with some explanations of it's behavior. Although it's not complete, it tries to present the subject in a didactical form and gives the main idea behind the subject. For further information, see [2] and/or [3]. In the sequel, the Matlab® scripts are presented, together with some graphic outputs generated by these scripts. At the end, some conclusions and future work are presented.

ARTIFICIAL NEURAL NETWORKS

In this section an overview of the main concepts concerning to ANN will be outlined. For more details, the interested reader should consult specialized bibliographies, like [2] and [3].

Definition and structure

Artificial Neural Networks (ANNs) can be viewed as an abstract simulation system that contains a collection of neuron units communicating with each other via axon connections. Such a model bears a strong resemblance to axons and dendrites in a nervous system. In Figure 1, the mathematical model of a neuron, proposed by McCulloch and Pitts [1] is depicted.

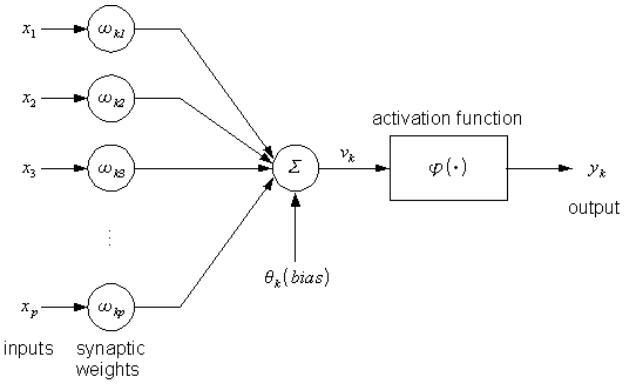

FIGURE 1
MCCULLOCH & PITTS MODEL OF A SINGLE NEURON.

There are several types of ANNs, each one designed for a specific task, but in this work, we will focus on the multilayer perceptron, an ANN that is well suited for classification tasks. Ins this case, the neurons are arranged in layers, with each neuron communicating with all the neurons of the prior and subsequent layers, as shown in Figure 2.

Each oriented arch in the structure of Figure 2 represents a synaptic connection between two neurons or between a neuron and a input stimulus. In an ANN, the knowledge about the relationship between the inputs and outputs is stored in the intensity of these synaptic weights.

Two phases can be identified in neural information processing: the *learning phase* and the *retrieving phase*. In the learning phase, several examples are presented and according to this, the ANN modifies its behavior to

[1] Poliana Magalhães Reis, National Institute of Telecommunications, Av. João de Camargo, 510, Santa Rita do Sapucaí, MG, Brazil, polyana@inatel.br
[2] Carlos Alberto Ynoguti, National Institute of Telecommunications, Av. João de Camargo, 510, Santa Rita do Sapucaí, MG, Brazil, ynoguti@inatel.br

© 2005 GCETE

incorporate the information. In the next section, the learning process will be described with more details.

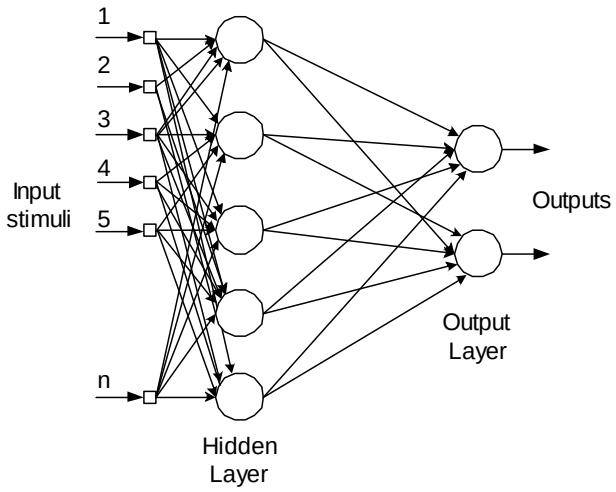

FIGURE 2
STRUCTURE OF A MULTILAYER PERPEPTRON.

The learning process

In the learning phase, a training data set is used to determine the synaptic weights of each connection. This trained neural model will be used later in the retrieving phase to process real test patterns and yield classification results.

For this particular architecture, the supervised paradigm is used: the process consists of presenting many pairs of input/output training patterns and, therefore, the learning will benefit from the assistance of a teacher, as shown in Figure 2.

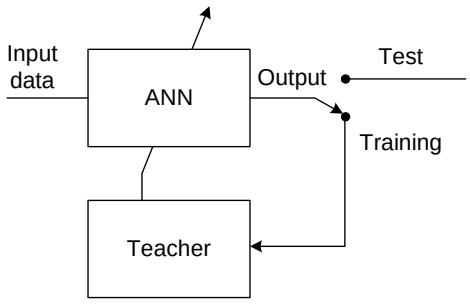

FIGURE 3
SUPERVISED TRAINING SCHEME.

Mathematically, this procedure can be summarized by the following formula:

$$\omega_{ij}^{(m+1)} = \omega_{ij}^{(m)} + \Delta\omega_{ij}^{(m)} \quad (1)$$

where $\omega_{ij}^{(m+1)}$ is the new synaptic weight for the connection between neurons *i* and *j*, $\omega_{ij}^{(m)}$ is the old synaptic weight, and $\Delta\omega_{ij}^{(m)}$ is the correction applied by the "teacher".

In words, for each training input, the output of the ANN is compared with the corresponding "expected output". The difference between these two quantities can be viewed as an error, and based on this information, the synaptic weights are modified by the learning algorithm, in order to minimize this error. This is the base for the Backpropagation algorithm that was used in this work.

A learning epoch consists in the presentation of all the training samples to the ANN. After each training epoch, a cross validation test is carried out using samples that were not presented during the learning phase (more on this later).

After this cross validation phase, another training epoch is performed. Why? Think about ourselves: some things we do not learn in the first time. Difficult things must be studied several times until we reach a full comprehension. The same principle applies for the ANNs. After this second learning phase, another cross validation is performed and so on, until convergence is reached.

Now the question that was left unanswered: why the cross validation phase must be performed over samples that were not presented in the training phase? The answer is a concept called "overfitting": after each learning epoch, the ANN learns a little more about the training samples, then theoretically, the higher is the number of training epochs, better is the final result. This is true only for the training samples. When new data are presented to this ANN, the performance will probably be poor, because it is "addicted" to the training samples, and lost the *generalization capacity*. Overfitting is diagnosed when a system have good performance on training data, but poor performance on new data.

Two strategies are used to avoid the overfitting effect: a) the convergence is tested using the cross validation samples and, b) the presentation order of the training samples for each learning epoch is altered in a random fashion.

In the next section, the developed set of Matlab scripts will be described.

THE TOOLBOX

Two scripts were developed to implement a multilayer perceptron ANN with a single hidden layer. They are simply called *train* and *test.*

The *train* script creates an ANN and trains it with the given training samples. The usage of this function is depicted in Figure 4:

```
train(samples,dTrain,nTrain,crossValid,dCross
Valid,nCrossValid,nInputs,nHidden,nOutputs,
eta1,eta2,nEpochs,'outputFile')
```

FIGURE 4
THE ANN TRAINING SCRIPT.

where:

- *samples*: a matrix with one training sample per column;
- *dTrain*: a matrix with the desired outputs of the ANN for each training sample. As the previous matrix, each desired output must be stored in a column of the matrix; also, the sequence of the desired outputs must be the same as the training samples;
- *nTrain*: number of training samples;
- *crossValid*: a matrix with one cross-validation sample per column;
- *dCrossValid*: a matrix with the desired outputs of the ANN for each cross-validation sample. The organization for this matrix is the same as the *dTrain* matrix;
- *nCrossValid*: number of cross-validation samples;
- *nInputs*: number of inputs;
- *nHidden*: number of neurons in the hidden layer;
- *nOutputs*: number of neurons in the output layer;
- *eta1* and *eta2*: in the learning phase, the teacher indicates in which *direction* the synaptic weights must be corrected. The *amount* of change in the direction pointed by the teacher is determined by the learning rate. In this script, a different learning rate for each layer is allowed;
- *nEpochs:* maximum number of training epochs allowed for training;
- *outputFile*: indicates where the trained ANN will be stored.

After the training phase, the user can evaluate the performance of the ANN using the *test* script. Its usage is shown in Figure 5:

```
error=test(testSamples,dTest,nTest,w1,w2)
```

FIGURE 5
THE ANN TESTING SCRIPT.

where:

- *testSamples*: a matrix with one test sample per column;
- *dTest*: a matrix with the desired outputs of the ANN for each test sample. Similar to *dTrain* matrix.
- *nTest*: number of test samples;
- *w1* and *w2*: synaptic weights of the trained ANN (stored in the '*outputFile*').
- *error:* is the difference between the ANN outputs and the corresponding column of the *dTest* vector.

It should be noted that the scripts were designed with versatility in mind. It means that they should be incorporated to any system without modification. Because of this, the users to implement their systems must do some work, and this is the desired approach for an educative software. In the next section, an application using this toolbox is reported.

APPLICATION: A SPEECH RECOGNITION SYSTEM

The task of speech recognition can be viewed as a conversion of a sequence of acoustic events representing a speech into its graphical representation. It's a very fascinating area, and practical applications of this technology always instigate the students.

In this section a simple isolated word, speaker dependent speech recognizer will be described. Interested readers can refer to specific literature, such as [4] and [5]. In the speech jargon, isolated word means that the speaker must make a short pause between the words, and speaker dependent means that the system works well only for a given speaker: the one that trained the system.

The task selected for this system is the recognition of two words: *open* and *close*. These two words were pronounced 12 times each, by a single male speaker, and therefore, the working database consists of 24 utterances. For each word, 9 utterances were selected for the system training and the 3 remainder ones for cross-validation and testing. In this case, the cross-validation and testing subsets were the same. Although it's not theoretically correct, the results achieved are still valid.

Each utterance was converted into a sequence of mel-cepstral coefficients, and as a result, each utterance was converted into a vector of 720 positions. Details about this can be found in [6].

For this task, the architecture of the ANN is as follows:

- 720 inputs;
- 5 neurons in the hidden layer;
- 2 neurons in the output layer (one for each word).

The desired outputs are [1 0]' when the input is the word *open*, and [0 1]' when the input is the word *close*. For this problem, the learning rates were set to 0.7 for the hidden layer and 0.5 for the output layer.

After each training epoch, a cross-validation test was performed, and it was assumed that the ANN recognized the word open if the output 1 was greater than output 2, and vice-versa. A recognition error was computed each time the relation between the outputs was wrong. In Figure 6, the evolution of recognition errors versus training epochs is shown.

It can be seen that after 48 training epochs, the system correctly recognized all the cross-validation/test utterances. For real word systems, with several users and vocabularies with more words, of course the problem is not so simple. One can expect higher error rates, and much research is

being done to improve these systems. However, the principle is the same, and after all, it's just an illustration example.

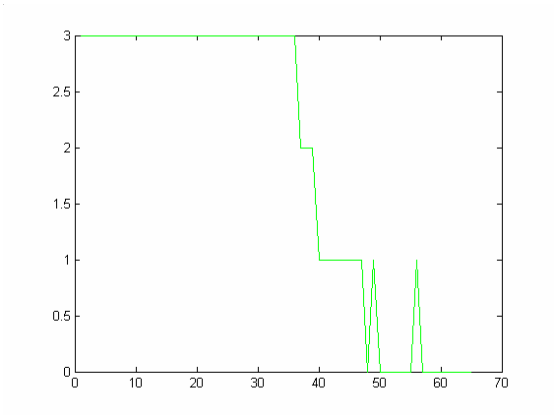

FIGURE 6
RECOGNITION ERRORS VS. TRAINING EPOCHS.

CONCLUSIONS AND FUTURE WORK

In this work a toolbox that implements a multilayer perceptron with a single hidden layer was presented. It was designed to be incorporated in any kind of application without modification, and is expected to be useful not only for teaching, but for research and development too.

A simple, isolated word, speaker dependent speech recognition system was built using the toolbox, in order to illustrate its use.

For the future, its expected to translate these tools to Java and C++ languages, and to implement some other architectures of ANNs.

ACKNOWLEDGMENT

The authors wish to thank Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) for partial funding.

REFERENCES

[1] McCulloch, W., and W. Pitts. "A logical calculus of the ideas imminent in nervous activity". *Bulletin of Mathematical Biophysics* 5: 115-33. 1943.

[2] Haykin, Simon, *Redes Neurais – Princípios e Prática 2ed.*, Bookman, 2001.

[3] Braga, A. P., Carvalho, A. C. P. L, & Ludermir, T.B., *Redes neurais artificiais - Teoria e Aplicações.* LTC, 2000.

[4] RABINER, L. *Fundamentals of speech recognition*. Prentice Hall Press. 1993.

[5] DELLER Jr., J. R., PROAKIS, J. G., HANSEN, J.H.L. *Discrete time processing of speech signals*. MacMillan Publishing Company. New York. 1993.

[6] DAVIS, S. & MELMERTSTEIN, P. "Comparison of parametric representations for monossylabic word recognition in continously spoken sentences". *IEEE Transactions on Acoustics, Speech and Signal Processing*, **ASP-28**(4):357-366. August, 1980.