Kohonen neural network for mobile robot trajectory planning, and Back-propagation algorithm for character recognition.

Cristian Ruiz 1802566, David Galvis 1802584, Eric Buitrago 1802410

Universidad Militar Nueva Granada

Abstract -- This document implements a neural network using the back-propagation algorithm in order to recognize characters entered through an interface. user graph, for this the network had to be trained with 4 different patterns for each character and at the end of the program it must indicate which character has just been entered. A Kohonen self organized map is developed in order to classify the characters on a work field, on which a mobile robot will move, to carry out this development it was necessary to implement the 3 fundamental stages, competitive phase, cooperative and adaptive phase.

Abstract -- In this document the implementation of a neural network is carried out by means of the back-propagation algorithm in order to recognize characters entered through a graphical user interface, for this the network had to be trained with 4 different patterns for each character and at the end of the program you must indicate which character has just been entered. In addition, a self-organizing map of Kohonen is developed in order to classify the characters on a work field, on which a mobile robot will move. To carry out this development it was necessary to implement the 3 fundamental stages, competitive phase, cooperative phase and adaptive phase.

Keywords: Neural network, learning, adaptation, recognition.

Keywords: Neural network, learning, adaptation, recognition.

INTRODUCTION

Artificial neural networks are massively parallel interconnected networks of simple elements (usually adaptive) and with hierarchical organization, which attempt to interact with each other and with objects in the real world. Due to their constitution and their foundations, they present a large number of characteristics similar to those of the brain. They are able to learn from experience, to generalize from previous cases to new cases, to abstract essential features from inputs that represent

irrelevant information. Among the advantages that these offer are included[1]:

-Adaptive

-Self-organization

-Fault

in real time

Standard modeling of neural networks

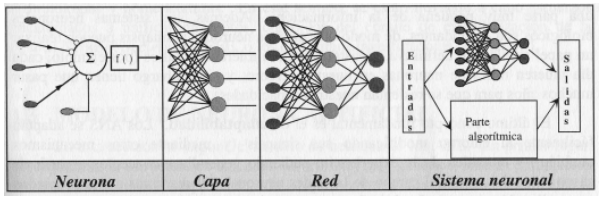


Figure 1: Global processing system of a neural network [2]

Objectives

- General

Design and build a character recognition system based on neural networks.

-Specific

Implement software for the creation, training (Back-propagation algorithm) and use of neural networks.

MATERIALS

- Matlab R2017a

PROCEDURE

Back-Propagation Algorithm

For the implementation of pattern recognition (characters) using a neural network, the algorithm parameters are first defined, the desired minimum error, an initial learning factor (since for this application this will be dynamic) and a moment factor. After these basic parameters, the error parameters are configured and an epoch-zero error is initialized.

4 different patterns are defined for 10 different characters, based on this and by defining a number of layers, it is possible to continue with the initialization of the weight matrix, which at first are random weights that will later be stored in different vectors. Each neuron output is set equal to zero as are the inputs of each layer. Changes in weights (dW delta weights) are also initialized to zero.

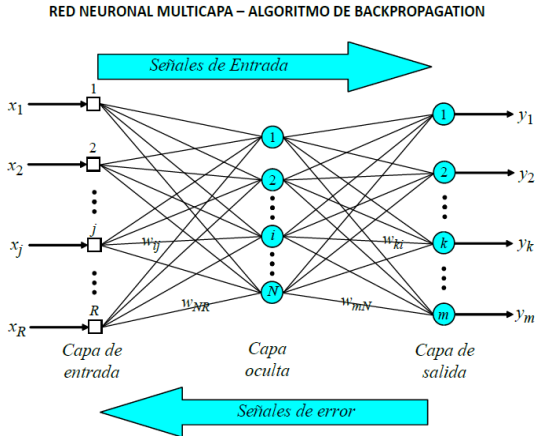


Figure 2: Back-Propagation Algorithm [2]

After initializing the output values to zero, the back-propagation algorithm is carried out first by performing a forward run.

```
%% 2. Recorrido hacia adelante y backpropagation
while (mse > errorMIN) && (epoch <= 5000-1) %Dos parametros de stop (error y epoca)
  e = zeros(Q,P);
  for p = 1:P                %Recorrido forward
    aux{1}(1:end-1) = X(:,p);
    for i = 1:depth-1
      y{i}=W{i}*aux{i};
      if i<depth-1
        aux{i+1}(1:end-1) = tansig(y{i});  %Capas oculta con funcion de activacion tanh
      else
        aux{i+1} = logsig(y{i});       %Capa de salida con funcion de activacion log
      end
    end
```

In the previous image, it can be seen that the hyperbolic tangent activation function is used for the hidden layers, which in Matlab corresponds to the <tansig> command, and the logarithm activation function is used for the output layers. And having this, we proceed to find the sensitivity of the output layer, on which we work when adjusting the weights with the previously defined moment, we go backwards <Backpropagation>. After the adjustment of weights, the previous weights are saved since with these the moment is found again. The expression for updating weights is given as follows.

$$\omega_{ij}^{(l)}(k+1) = \omega_{ij}^{(l)}(k) + \alpha * \delta_i^{(l)}(k) * y_j^{(l-1)}(k)$$

For the iterations, the mean square error is calculated and stored in a variable, to then increase another epoch, it should be noted that a maximum of 5000 epochs is defined for this program. The alpha is dynamic and changes depending on how the epochs increase, defined so that when the error increases, the alpha decreases by 70% and in the opposite case, when the error decreases, the alpha increases its value to 105%. Since the learning factor can only take values between 0.1 and 0.9, these two critical values are saturated so that the system does not fail.

Kohonen Network

For this application, a workspace of two meters long by two meters wide is simulated, on which the mobile robot will move, 40 points and a number of neurons that is generally 3 times the number of neurons are randomly located. points, in this case it will then be 120, these 120 can also be located randomly on the field. But for this algorithm they are located on a diagonal line in the field of work. After this, the competitive phase between neurons begins, where the distance of each of the nodes is calculated and the neuron whose weight vector is most similar to the input vector will win. For this, the distance between points on a plane will be used. .

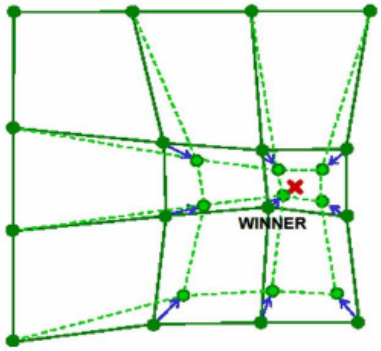
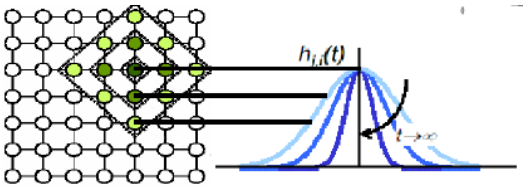


Figure 3: Kohonen Network

At the end of the competitive phase, the cooperative phase continues, in which a neighborhood of cooperation is defined around the winning neuron, where the other neurons "approach" the winner, this function is expressed as follows:

$$h(k) = e^{-\left(\frac{d^2}{2\sigma^2}\right)}$$

Where d is the distance index between the winning neuron and its neighbors.



And finally the adaptive phase in which the weights are now updated based on the winning neuron and with it the entire neighborhood, the

update is given through the following vector expression:

$$\omega_j(k+1) = \omega_j(k) + \mu(k) * h(k) * (x - \omega_j(k))$$

And in turn, in this same phase it happens that the neighborhood size and learning factor decrease as iterations progress, learning factor varies as a function of iterations k.

$$\mu(k) = \mu_0 * e^{\left(-\frac{k}{\tau_2}\right)}$$

RESULTS

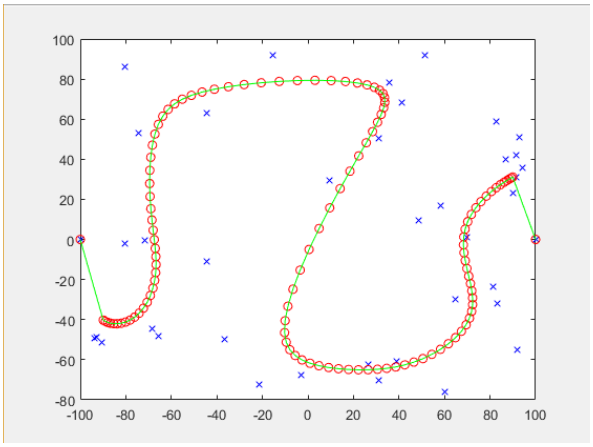


Figure 4: Random position of inputs and neurons

After an elapsed time and “iteration after iteration”, the neurons (circles) adapt to the inputs (shown with blue x's).

This will be implemented for the “Mobile Robots” project where the trajectory planning is needed using a Kohonen self-organizing map.

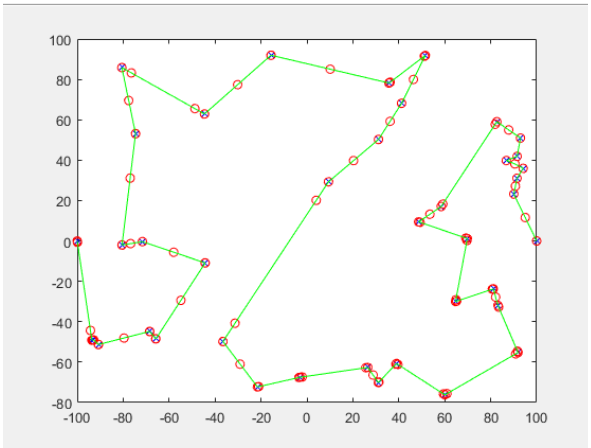


Figure 5: Adapted neurons

CONCLUSIONS

It is possible to create the neural network in such a way that it recognizes and trains different patterns of entered characters, and that with the help of a dynamic alpha and a moment in the weights, the decrease in iterations was achieved compared to a system that does not have these characteristics.

REFERENCES

[1]D. Matich, "Neural networks: basic concepts and applications", Informatics applied to process engineering, National Technological University (Regional Rosario), Argentina, 2001.

[2]P. Larrañaga, I. Inza, A. Moujahid, "Neural Networks", University of the Basque Country - Euskal Herriko Unibertsitatea.

[3] NEURAL NETWORKS, Ricardo Andrés Castillo MSc, Class material. Militar University of New Granada