On-line learning in an embedded maximum sensibility neural network

Gustavo González Sanmiguel, Luis Lauro Gonzalez, Luis M. Torres-Treviño, César Guerra
Universidad Autónoma de Nuevo León, FIME
Av. Universidad S/N, San Nicolás de los Garza, N.L.
tavo.gzzsm@gmail.com, luislaurogzz@gmail.com, luis.torres.ciidit@gmail.com, cguerratorres@gmail.com

***Abstract*—A maximum sensibility neural networks was implemented in an embedded system to make on-line learning. This neural network has advantages like easy implementation and a quick learning based on manage information in place of a gradient algorithm. The embedded maximum sensibility neural network was used to learn non linear functions on-line using potentiometers and a push button giving the function of activation and learning. The results give us a platform to apply on-line learning using neural networks.**

I. Introduction

It is known that the neural networks offers a very huge number of areas of application, in particular, neural technology offers much potential in the areas of nonlinear modeling, direct model and reference adaptive control [1][2][3][4]. They are mostly used to learn from data examples presented to them in order to capture the subtle functional relationships among the data even if the underlying relationships are unknown or the physical meaning is difficult to explain, this give us the advantage to learn almost every system, but there is a disadvantage, because when there is new data presented for training, it is needed to be done offline,then a process starts with the recalculation of the internal values of the artificial neuronal network (ANN). That is caused because the neuronal network takes much time to learn a new pattern. There are only a few neurons that try to learn online the process [5][6].

The maximum sensibility neuronal network (ANN-MS) [7] represent a possible solution to the problems mentioned, because it can learn the system either offline or online , and also keep updating the behavior of the system online. This advantage is obtained because of the training process inherent by the ANN-MS structure, where the update of the values is only done in the neurons that is needed, optimizing the time needed to times of milliseconds or even less.

Adding this powerful ANN to the advantage of an embedded system that does not need a computer for training, give us a great advantages, one of the biggest advantages is that the embedded system can be adapted to almost every system in the word, that is because of the resistance inherent of the embedded systems to heat, cold, and many other places, also the cost of the system is diminished to only need a micorontroler board with its I/O board. This network also got calculation but these are not calculations that consume a lot of time, giving it the advantage to implement it in a microcontroler and not in an FPGA system, making it to not need much hardware materials and easy to use.

II. Maximum sensibility neural network (ANN-MS)

The ANN-MS is a multilayer network that is made principally of a layer of distribution neurons (or input layer), a layer of Gaussian neuron, a layer that add the effects of the neurons or select the neuron with maximum sensibility and its value to be the output. This behavior is shown in the figure 1.

Each layer has an specif function to the neuron:

- The first layer has two primary works, the first one is to normalize the data input and the second is to distribute the information to all the neurons to the second layer. The way to the second layer has an amplifier (W) that regulates the information, and the information is finally delivered to the second layer.
- The second layer has the function to use the Gaussian equation to the sum of all the inputs of that specif neuron. The way to the third layer also has an amplifier values (AC) for each output in every neuron.
- The third layer has the function to detect if any of the actually has a maximum sensibility and use the output of that neuron specific, and if there is not any neuron with maximum sensibility the approximated value to that input is calculated using the values of all the neurons.

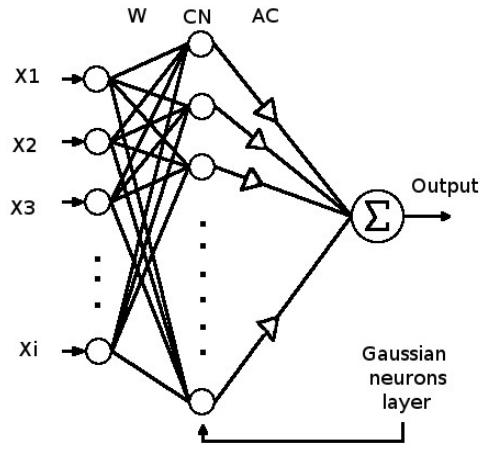

Fig. 1. ANN-MS Diagram

The main pseudocode of the program is based in two states, those two are to stay in "Running the neuron", and the other one is to be in the state of learning, this second state is the main characteristic of this neuron, and the main advantage of

this is that, it can be used any moment, even in the work area, and also that it takes only one or two trials to learn a pattern (One or two clicks in the learn function, in the case of the embedded system).

What makes this network special is that in the third layer, the detection of the maximum sensibility allows the network to coordinate the new pattern learned and recycle the work that was done by that one, saving it a lot of memory, calculation, energy and time.

A. Run state of the ANN-MS

This state is the normal state of the neuron, and the way it works is to read the input values, and check if the neuron has already learned a pattern that has a similar value and show the output corresponding to already learned input. To know how much similar the network is supposed to work is by the parameter "sensibility margin", which can be selected to be very specific to not recycle a value already learned if it is not almost equal to the new received, or it also can be loose and recycle values, even if they are not alike. There must be mention that the behavior of the network should be selected depending the system that there is needed to learn and the specifications that are wanted. And the other possible behavior is that the input that you have, does not resemble any input already learned by the network. When that happens, what the network does is that it actually approximate the output value with the neuron that most resembles the input. This behavior is shown in the figure 2.

An effect that happens when the values needed to learn are not alike between them, every pattern would need a lot o neurons to remember the full number of patterns, there is normally specified in the neuron a number of maximum neurons to use. The way that the neuron behaves when the maximum number of neurons is used is to select the neuron less used in the moment to optimize or not affect the rest of the patterns already learned.

The function of activation that the second layer has is the Gaussian function:

$$f(x) = e^{\frac{-(x-cm)^2}{\lambda}} \tag{1}$$

where the values cm and λ are parameters defined, also the values of the ms (sensibility margin) which is not specified in this function are the only three parameters that the ANN-MS has in it. These parameters values give actually the behavior of the neuron, and between them the most characteristic is the ms because this is the one in charge of telling that the input has been already learned or if it has already a good approximated value by another of the internal neurons in the system.

B. Training of the ANN-MS

The training process is the principal characteristic of this ANN, the way it works is to first make one execution of the run state, and then read the inputs and check if there is one neuron which has a maximum sensibility value, and if there is one, it adjust the values of that neuron, those values are expected to be changed just a little bit, that is because the

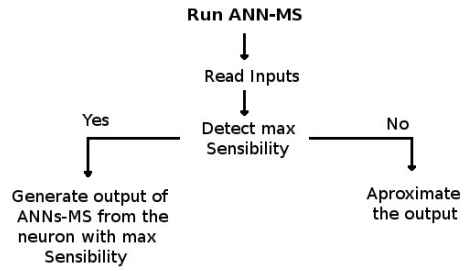

Fig. 2. Diagram of Run state in ANN-MS

neuron got the maximum sensibility. The other case is to add a new neuron to the system, that is done to not affect all the others pattern already learned. In case that there is an upper limit number of neurons expected, the system takes the neuron that in that time is less used by the ANN, and changes it values to match the result of the new pattern learned. The diagram of this behavior is shown in figure 3.

The equation used to update the values of the weights that are between the first and the second layer (values of $\mathbf{W}$) is:

$$\mathbf{W_{cn}} = \frac{\mathbf{W_{cn}} + \mathbf{X_n}}{2} \tag{2}$$

The equation that is used to update the values of the weights that are in the layer between the second and the third layer (values of $\mathbf{AC}$) is:

$$\mathbf{AC_{cn}} = \frac{\mathbf{AC_{cn}} + \mathbf{Y_o}}{2} \tag{3}$$

Depending of the value that is setup in ms (a high value says that every pattern learned should be specialized), there may be the case were the ANN-MS does not learn completely, the solution to this is to resend the pattern a few times more. That is expected because the ms value specifies the ANN-MS that every pattern that is learned should be specialized a lot, and with that any new pattern does not match the already learned, even if they resemble each other a lot.

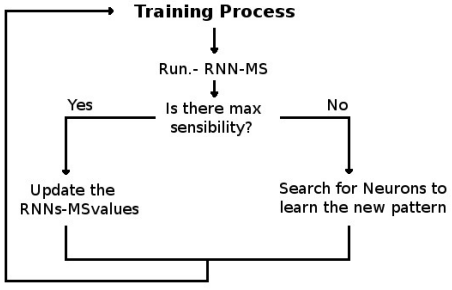

Fig. 3. Learning Diagram of the ANN-MS

III. Implementation in an embedded system

The embedded system was developed using the prototype tool Arduino [8], which works with the language C, also there

Pseudo code of Embedded system:

```
Start Variables
//Main Loop of Microcontroler
While (True)
  Read Input Values
  RunFunctionANN-MS
  if New Pattern is given
    LearnFunctionANN-MS
  endIf
  Display output generated
endWhile
```

TABLE I
HARDWARE CHARACTERISTICS

Characteristic	Description
Microcontroler	ATmega168
Operation Voltage	5 V
Recommended alimentation voltage	7 - 12 V
Maximum alimentation voltage	6 - 20 V
Single I/O Digital ports	14 (6 got PWM)
Single I/O Analog ports	6
Current of I/O Digital ports	40 mA
Flash memory	16 KB
SRAM	1 KB
EEPROM	512 bytes
Clock Frequency	16 MHz

were need to create an I/O sensor module which have the next items (also displayed in the figure 4):

- 3 Analogs sensors, implemented with potentiometers (Pot).
- 2 Digital sensors, implemented with buttons (But).
- 1 Pulse Width Modulation (PWM) output actuator, implemented with one LED.

The pseudo code that shows how actually behaves the neuron in the embedded system:

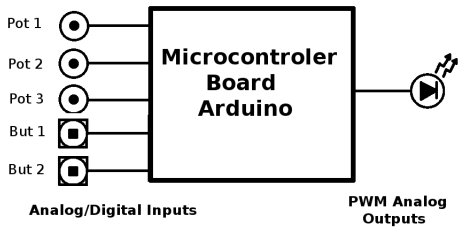

Fig. 4. Simple Schematic

A. Hardware

The specific Arduino used is the diecemila, which has the next technical aspects:

B. Experimental description

The prototype implemented in the microcontroler was tested using oscilloscopes to validate the output and also with a program in a computer to validate the variable that is linked to the pulse width modulation.

The learning process consisted in using one potentiometer as the expected to learn value, and the other two potentiometers to give the pattern to learn. When the inputs were correctly selected, one button was used to tell the microcontroler to learn the pattern, and with that one learning process was run. After that the microcontroler automatically changes back to the running state.

This manual process takes much time if there are a lot of patterns to learn, that was the main cause to create an automated system to teach the microcontroler the values that were needed to learn, this was implemented with a simple program where the learn button and the potentiometers were simulated. After the program was done, the learning process of a non-linear function were done in just a matter of seconds.

IV. On-line learning of non-linear functions

The ANN-MS embedded was tested learning non-linear functions, there were used three nonlinear functions and in every case showed good results, there is need to clarify that every pattern was sent to the neuron only two times. Increasing the input value in an pre selected δ which in this case where the actual resolution of the embedded system $(1/255)$. After the neuron was trained the neuron was tested to show if there were any inconsistency in the running state with the values already learned. The first function used was:

$$f(\mathbf{x}) = Sin(x_1) * Sin(x_2) \tag{4}$$

and the learning process showed the results displayed in the figure 5 and the results of actually testing the already learned patterns are shown in the figure 6. The learning process taken only about 30 seconds to completely learn the function, an as it can be shown, in the testing process the function was learned with a incredible results.

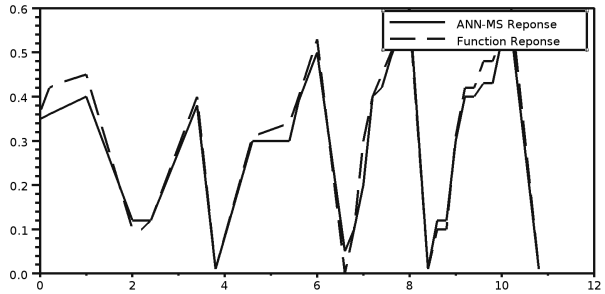

Fig. 5. Function 1 learning process

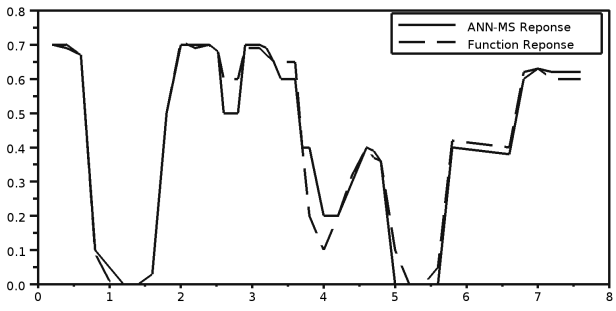

Fig. 6. Function 1 running process

The second function used was:

$$f(\mathbf{x}) = 1 - x_1^2 + x_2^2 \tag{5}$$

The learning process results are shown in the figure 7, these learning results are even more accurate that the first function learned. This shows that the learning process is working fine in the testing face. The results are a little bit more inaccurate than the first function, yet they are accurate. The cause of this behavior is the selection of parameter ms, this parameter tells the neuron that if a parameter learned is almost equal than another one, to consider it like the other one, and learn that pattern by adjusting the value of the neuron that holds that parameter to actually have the information of both of them. The table that shows the running state of this function is figure 8.

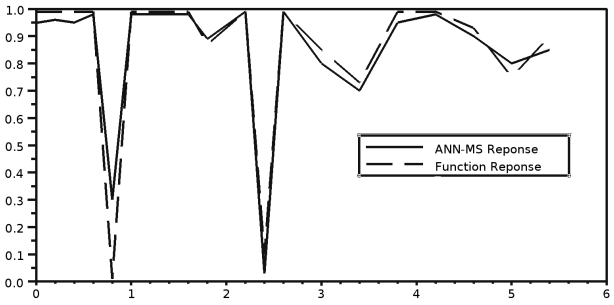

Fig. 7. Function 2 learning process

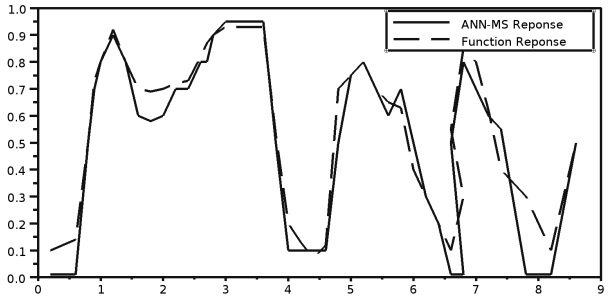

Fig. 8. Function 2 running process

And the third function used was:

$$f(\mathbf{x}) = x_1^2 - 0.35x_2 \tag{6}$$

This last equation has almost equal results in the learning state (as shown in the figure 9) as the equation given. The results acquired in the test are shown in the figure 10. This equation was almost completely learned, as in the tables can be seen. It only took this function to learn the series of patterns about 25 seconds.

All the three equations were retested several times to validate that the results were accurate and repetitive. That was to make sure that the ANN-MS can learn a consecutive series of patterns even if it is already working in an specific activity.

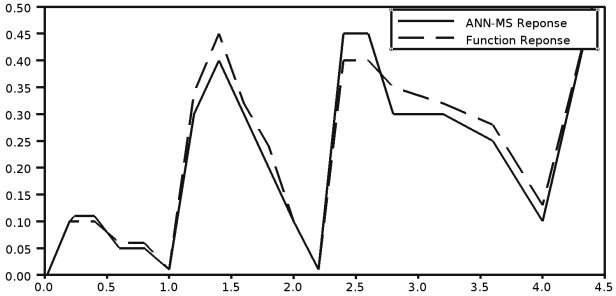

Fig. 9. Function 3 learning process

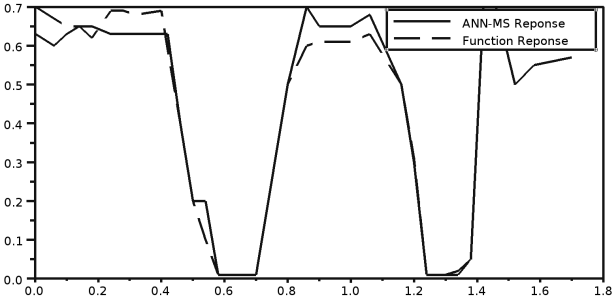

Fig. 10. Function 3 running process

V. Conclusions and future work

In this paper was shown an ANN that has the capacity learn online and easy, that not have difficulties of use. Also there must be mentioned that the learning process of the ANN-MS is not an iterative process, which gives it a short learning process time.

As future work, this is going to be added:

- Learn a pattern that is used by a real system (robot arms, electrical systems, prototype cars, etc..)
- Add the capacity to have a value λ and ms specific for each neuron.
- Add the capacity to automatically update the values of λ and ms with the use of an evolutive algorithm.
- Develop a software in Java to check the current status of embedded system

Acknowledgments

We thank CONACYT by the support provided in this project (grand number 258684 and grant number 258615) also thanks to the support provided by PROMEP/103.5/10/6687.

VI. Appendix.- Pseudo codes and Variables and normal values of parameters

TABLE II
Run State pseudo code implemented in the embedded system

Run state Pseudo code:

```
RunFunctionANN-MS(X,W,AC,ms)

  for n=1,Number of Neurons
    S=0;
    for i=1,Total of inputs
      S=S+(x(i)-W(n,i)^2;
    end
    //fa = activation function
    SN(n)=fa(sqrt(S),alpha);
  end
  Maximum Value and Position in SN;
  if MaxValue > Maximum Sensibility Value
    for o=1, Total of Outputs
      y(o)=AC(pos,o)*SN(pos)
    end
  else
    for o=1, Total of outputs
      s1=0;
      s2=0;
      for n=1, CN
         s1=SN(n)*AC(n,o)+s1;
         s2=SN(n)+s2;
      end
      y(o)=s1/s2;
    end
  end
Output Value of the ANN-MS = y
```

TABLE III
Learn State pseudo code implemented in the embedded system

Learning state Pseudo code:

```
RunFunctionANN-MS one time

LearningFunctionANN-MS(X,Ye,W,AC,ms,Lambda,U)
  Maximum Value and Position in SN;
  If Maximum Value is >= ms
    for pr=1, Total of inputs
      W(Position,pr) = (W(Position,pr)+X(pr))/2;
    end
    for o=1, Total of outputs
      AC(Position,o)=(AC(Position,o)+yo(o))/2;
    end
  else
    if (CN+1) > Maximum value of neurons avaliable to use
      Minimum Value and Position of the U
      for pr=1, Total inputs
        (Position, pr) = X(pr)
      end
      for o=1, Total of outputs
        AC(Position,o)=yo(o);
      end
    elseif
      New neuron = Number of neurons actualy used + 1;
      for pr=1, Total inputs
        W(NewNeuron, pr) = X(pr)
      end
      for o=1, Total of outputs
        AC(NewNeuron,o)=yo(o);
      end
    end
  end
```

Parameters of the neuronal network:

Parameter	Description	Normal Values
α	Activation function parameter	[0.05 - 0.2]
ms	Maximum sensibility value	[0.88 - 0.98]
NTN	Maximum numbers of neurons	[1-MaxMemory]

Variables of the neuronal network:

Variables	Description	Normal Values
X	Input pattern normalized	[0 -1]
Ye	Output expected to learn	[0 - 1]
W	Weights of the first Layer	Initial Values [1]
SN	Second Layer output	[0 -1]
AC	Output neuron value	[0 -1]

References

[1] A. Cheang-Martinez, J. Acevedo-Davila, L. Torres-Trevino, F. Reyes Valdes, and A. Saldivar-Garcia, "Corrosion prediction and annual maintenance improvement of concrete structural components using neural networks," in *Electronics, Robotics and Automotive Mechanics Conference, 2007. CERMA 2007*, sept. 2007, pp. 202 –206.

[2] J. Acevedo-Davila, L. Torres-Trevino, and L. Gomez, "Tezontle aggregate substitute optimization in building blocks mixture." in *Electronics, Robotics and Automotive Mechanics Conference, 2007. CERMA 2007*, sept. 2007, pp. 307 –311.

[3] I. Escamilla, P. Perez, L. Torres, P. Zambrano, and B. Gonzalez, "Optimization using neural network modeling and swarm intelligence in the machining of titanium (ti 6al 4v) alloy," in *Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on*, nov. 2009, pp. 33 –38.

[4] L. Torres-Trevino and A. Reyes-Valdes, "Conceptual design of carbon steels to support heavy crude refinement using neural network modeling and evolutionary optimization," in *Electronics, Robotics and Automotive Mechanics Conference, 2008. CERMA '08*, 30 2008-oct. 3 2008, pp. 439 –442.

[5] L. H. C. H. J. Zhao, Weiguo; Wang, "Cascade-correlation neural network for sensor fault detection and data recovery with on-line learning," *American Scientific Publishers*, vol. 9, pp. 2034–2037, 2011.

[6] N. Takehiko, "Theoretical analysis of batch and on-line training for gradient descent learning in neural networks," *Neurocomputing*, vol. 73, pp. 151–159, 2009.

[7] I. Escamilla, L. Torres, P. Perez, and P. Zambrano, "A comparison between back propagation and the maximum sensibility neural network to surface roughness prediction in machining of titanium (ti 6al 4v) alloy," *Lecture Notes of Artificial Intelligence 2008*, vol. 5317, pp. 1009–1019, 2008.

[8] C. Massimo, Banzi; David, "Arduino proyect," 2002, www.arduino.cc.