International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 09 | Sep-2016 www.irjet.net p-ISSN: 2395-0072

Iris Classification using ANN on 8 bit Microcontroller

Sanket Nartam[1], Anant More[2]

[1]Research Student, Dept. of ECE, R. M. D. Sinhgad School of Engineering Pune, Maharashtra, India
[2]Ph.D Scholar, Dept. of ECU, K L University Guntur, AndraPradesh, India

---

Abstract - *Conventional approach to develop embedded applications are commonly being used today. However recent study has revealed that techniques involving Artificial Neural Network can be used as an alternative to develop embedded applications. A lot of applications have benefited using them, most common example is Google search engine, Stock market prediction etc. ANNs are a kind of machine learning models, inspired on the functioning of the brain can be incorporated in low cost microcontrollers. ANN could be implemented in number of applications related to embedded system, process control wireless sensor networks. Few common applications are dynamic reprogramming of sensor nodes, intelligent sensors, auto calibration of process instruments, weather prediction, free-fall detection etc. In this project work, we have selected C-Mantec which is a constructive neural network algorithm as it generates compact architecture suitable for microcontroller implementation. For the implementation, on embedded side we have selected 8 bit microcontroller and to compare on faster processors python programming language is used.*

Key Words: Artificial Neural Network (ANN), Competitive Majority Network Trained by error correction (C-Mantec), Constructive Neural Network, 8-bit Microcontroller.

1.INTRODUCTION

From the literature survey it is found that to choose a proper neural network architecture is difficult. Many authors have proposed different methods to solve this problem but there is no general rule to select the best architecture for neural network. Trial and error methods are inefficient in terms of computation power but are in use to develop applications of ANN. In recent years, different constructive algorithms and techniques have surfaced. In Conventional neural network, (NN) training algorithms (such as Back propagation) need to define the NN architecture before it can start learning [2].

Constructive algorithm needs a minimum neural network architecture to be defined at the start. It adds layers, connections and nodes during the training, as required by the given problem. Thus, it produces neural network architecture which is compact.

Embedded applications involve the use of 8 bit, 16 bit and 32 bit microcontrollers. This microcontroller has limitations in terms of memory and computational power. Moreover dedicated hardware such as shifter is used to increase the capability of computation. Considering the memory constraints the neural network algorithm should produce compact neural network architecture. We have selected C-Mantec algorithm as it has better generalization ability. In addition, it produces very compact network architecture. C – Mantec algorithms do have a built in filtering method which avoids over fitting problems.

In the project work, we have selected ATMEGA328P which is Atmel 8-bit AVR RISC-based microcontroller. Complete processing of the algorithm is done on 8 bit microcontroller, operating on 16 MHz clock frequency having 2KB of SRAM. The reason for selecting 8 bit microcontroller was basically to populate the capability of C-Mantec algorithm to work on low end devices. To demonstrate we have selected, Fisher's Iris data base. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. One class is linearly separable from the other two; the latter are not linearly separable from each other [2].

The paper is structured as follows, firstly we will briefly discussed about C-Mantec algorithm and its steps. In later sections the methodology and results are discussed.

2. C-Mantec Algorithm

In Competitive Majority Network Trained by Error Correction (C-Mantec) algorithm is a neural network constructive algorithm which works on competition between neurons. It uses stable modified perceptron learning rule. The Thermal perceptron rule is responsible for providing stability to acquired knowledge. The network architecture grows keeping the old information and neurons compete for new incoming information. The effect of competition is such that the new neuron can still learn information, provided the incoming information is quite similar to the information saved. This is one of the major distinguishing factor compared to the existing constructing algorithms .Briefly in this algorithm there is a completion between neurons to learn the new incoming information. As C-Mantec is a constructive neural network algorithm, it avoids the trial and error method of selecting the network architecture. It generates the architecture of network as the learning proceeds [2].

The binary activation state of the neuron depends on the different factors which are the actual value of the N synaptic weights, bias, N input signals

$$S = 1\ (ON) if \Phi \geq 0;\ 0\ (OFF) otherwise \quad (1)$$

Where, φ is the synaptic potential of the neuron defined as:

$$\Phi = \sum_{i=1}^{N} \omega i \ \psi i - b \quad (2)$$

The modification of the synaptic weights in thermal perceptron learning rule, Δωi is done runtime according to the following equation

$$\Delta wi = (t - S)\psi i \ Tfac \quad (3)$$

Where, ψ represents the value of input unit I connected to the output by weight ωi and t is the target value of the presented input.

$$Tfac = \frac{T}{T0} e^{\left\{-\frac{Mod\ \Phi}{T}\right\}} \quad (4)$$

Table -1: Results on MCNC function

Functions	Results on MCNC functions	
	Neurons Theoretical	Neurons Practical
XOR2	2±0	2±0
XOR3	3±0	3±0
Cm82f	3±0	3±0
Cm82ah	3±0	3±0
9symml	3±0	3±0
Z4ml24	1±0	1±0

3. STEPS OF ALGORITHM

Step 1: Create Initial Network by adding one hidden layer neural network with a single neuron and an output neuron [2].

Step 2: Initialize Parameters and Set parameter Values and Initialize Counters [2].

gfac {0.05 to 0.5}

Imax {1000- 100000}

T0= N

T =T0

Step 3: Check the output of the Network by giving a random input pattern [2].

Step 4: Check for correct classification

Compute the value of Tfac for all existing hidden neuron which has wrongly classified input presented. Modify the weight of neuron, with the largest value of Tfac, such that this value should be larger than the value of the parameter gfac. Reduce the internal temperature of the modified neuron. Check for no neuron with the value of Tfac larger than gfac. After that add a new neuron to the hidden layer that will learn the actual example. Next step is to, reset T to T0 and I to Imax [2].

Step 5: Loop back until all patterns are classified loop back to step 3[2].

4. METHODOLOGY

We have implemented C-Mantec algorithm in a microcontroller suitable for different embedded applications. Network is trained by providing training patterns. The neural network architecture generated is tested with test patterns and checked if the outputs are classified properly [2]. To verify the correctness of the implemented the algorithm we tested the output, in terms of number of neurons generated. For that we have compared the results of the original published [2].

5. IMPLEMENTATION

The data base contains the following attributes: Sepal length in cm- It is Float number in 1.0 to 9.9 which requires 4 Bytes. Sepal width in cm- It is Float number in 1.0 to 9.9 which requires 4 Bytes. Petal length in cm- It is Float number in 1.0 to 9.9 which requires 4 Bytes. Petal width in cm- It is Float number in 1.0 to 9.9 which requires 4 Bytes .Class: - Require 1 Byte.17 bytes for 1 pattern is required.17x150 patterns of dataset requires = 2550 bytes. Max pattern that can be accommodated=Size of Eeprom-1/No of bytes needed for 1 pattern (1024-1)/17bytes= 60.176 rounded to 60 patterns. So 20 samples of each class can be taken.

In order to accommodate all the patterns we have proposed two solutions so that all 150 patterns can be used for implantation in microcontroller. In first instead of taking input parameters as a float we have taken it is integer value. For example, Sepal length 4.5 is converted to 45 and the stored in the Eeprom memory. So instead of taking up 4 bytes now only 1 byte can be sufficient.

Steps:-

1) All Input data is multiplied by 10 and stored in Eeprom.

2) During Calculation: Calculate by dividing with 10 to compensate.

3) 5 bytes required to store 1 pattern: 150x10= 750 bytes are needed.

This gives us a 75% saving in the Eeprom memory.

In Solution 2 we have proposed a shuffling mechanism where instead of taking a larger array in a microcontroller only 10 patterns of inputs are read into an array. Training of the algorithm is done on those patterns and the then the next set of patterns are taken into array. This helps in reduction of the number of variables in the RAM.

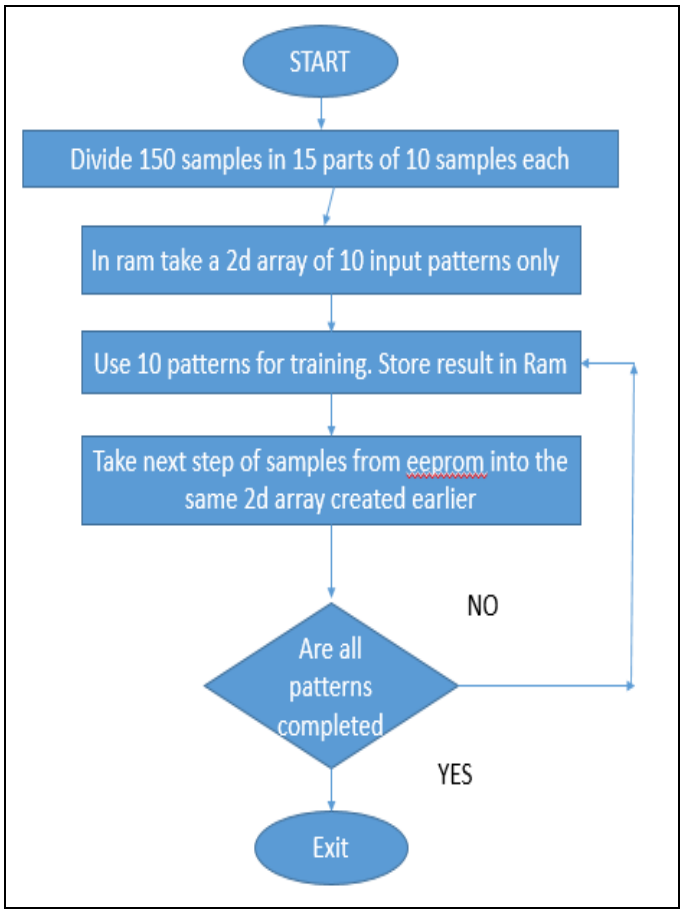

Fig -1: Flowchart for solution 2

6. RESULTS

We verified the results published of the original paper as seen in table I in terms of generated neurons index from the functions as given in the MCNC benchmark database. The results of classification of IRIS plant database are as shown in Table III. We verified the training time required for the algorithm to get started but the training time was more as per the expected results. The reason could be data conversion techniques used and refreshing the array parameters frequently to accommodate in 2KB of SRAM.

Table -2: Classification Results

Parameter	Classification	
	Patterns Provided	Patterns Classified Correctly
Iris Setosa	50	48
Iris Versicolour	50	49
Iris Virginica	50	47

Table -3: Average Time Analysis

Parameter	Platform	
	Implementation on I3 processor	Implementation on Atmega 8
Number of Neuron	5	5
Training Time	14.154 s	3.6 min

7. Conclusion

C-Mantec algorithm when compared with other algorithms gives very compact architectures with good prediction capabilities. In addition, algorithm is constructive there is no trial and error method involved in selecting a proper neural network architecture. Due to which the programming part from the microcontroller point of view can be simplified easily. The algorithm suits completely for the low end microcontroller which has constraints in terms of processing ability and memory. Given the existence of devices with much more powerful computing resources than the considered board, the results shows the potential of the proposed algorithm. The training time required for the microcontroller implementation can be reduced by using more efficient logic. A dedicated hardware block for multiplication and division could speed up the training in embedded application. A generic stack for this algorithm could be developed so that it could be easy to port the algorithm in different applications.

REFERENCES

[1] FranciscoOrtega-Zamoranoa,*,JoséM.Jereza, JoséL.Subirats a, IgnacioMolinab, LeonardoFranco a, "Smart sensor/actuator node reprogramming in changing environments using a neural network model"Journal on Engineering Applications of Artificial Intelligence Elsevier ,Volume 30 April 2014, Pages 179–188

[2] Sanket Nartam and Prof. Mr Anant More, "Neural Network based alternate approach for embedded applications", e-PG Project Exhibition 2016 for M.E. E&Tc Students under Savitribai Phule University of Pune, June2016.

[3] José L. Subirats, Leonardo Franco ∗, José M. Jerez," C-Mantec: A novel constructive neural network algorithm incorporating competition between neurons" Journal on Neural Networks, Elsevier 26 (2012) 130–140.

[4] Francisco Ortega-Zamorano, José M. Jerez, and Leonardo Franco, Senior Member, IEEE. FPGA Implementation of the C-Mantec Neural Network Constructive Algorithm". IEEE Transactions on Industrial Informatics, vol. 10, no. 2, May 2014.