International Journal of Advanced Computer Research
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
Volume-5 Issue-18 March-2015

Linearizing the Characteristics of Gas Sensors using Neural Network

Gowri shankari B[*] and Neethu P S
Assistant Professor, Electronics and instrumentation engineering, New Prince Shri Bhavani College of Engineering and Technology, Chennai, India

Received: 09-March-2015; Revised: 24-March-2015; Accepted: 28-March-2015

Abstract

The paper describes implementing arbitrary connected neural network with more powerful network architecture to be embedded in inexpensive microcontroller. Our objective is to extend linear region of operation of nonlinear sensors. In order to implement more powerful neural network architectures on microcontrollers, the special Neuron by Neuron computing routine was developed in assembly language to allow fastest and shortest code. Embedded neural network requires hyperbolic tangent with great precision was used as a neuron activation function. Implementing neural network in microcontroller makes superior to other systems in faster response, smaller errors, and smoother surfaces. But its efficient implementation on microcontroller with simplified arithmetic was another challenge. This process was then demonstrated on gas sensor problem as they were mainly used accurately in measuring gas leakage in industry.

Keywords

Microcontroller, non-linear sensor compensation, embedded, neural network, gas sensor.

1. Introduction

Nonlinear control problems are suitable applications for artificial neural networks (ANNs) and have resulted in remarkable performance characteristics in the last few years. Sensors are widely used in industrial processes, automobiles, robotics, avionics and other systems to monitor and control the system behaviour [5]. Some include motor drives [15], [18] and power distortion dealing with harmonic problems.

One common cause of nonlinearity or linear control systems is the sensors. Due to the nonlinear nature of neural networks, they have become an integral part of the field of control. This work proposes a solution for implementing neural networks on microcontrollers for many embedded applications. The common myth is that neural networks require relatively significant computation power, and advance computing architectures are needed. Primarily, fuzzy systems are implemented on inexpensive microcontrollers [7], [15], [16]. Due to the use of various simplification methods, such as limited bit resolution or piecewise approximations of activation functions, the obtained results are not encouraging.

The exception is the implementation of neural networks on a microcontroller where it is shown that neural networks can be superior to fuzzy system in almost all aspects: smaller errors, smoother surface [4], [10]. Neural network is created for a fast one while introducing an error that is not greater than the error already introduced by using a neural network to approximate the original function.

In this paper the relatively complex algorithm is implemented using microcontroller. To fully utilize the power of neural network, powerful architecture was used with arbitrarily connected neurons. Assembly language implementation of NBN approach allows faster and shorter code. The Pseudo floating point calculation allows the integer computation complexity for high accuracy computation. The most important part of a neural network is the activation function where a method of approximating the tangent hyperbolic function with great precision is described. Neural network incorporates tanh as its activation function. This particular application method has been implemented using assembly language on an 8-bit microcontroller for Fire Classification of Gas Sensor.

*Author for correspondence

International Journal of Advanced Computer Research
ISSN (Print): 2249-7277 ISSN (Online): 2277-7970
Volume-5 Issue-18 March-2015

2. Arbitrary Neural Networks

As mentioned in the introductory section, the power of neural networks strongly depends on the neural network architectures. Popular MLP architectures are not only one of the least powerful architectures but also have other disadvantages [9], [12]. With an increased number of layers, the training of such networks becomes more difficult because the networks become less transparent for error propagation. In other words, the effect of weight changes on the first layers are disturbed by weight changes of the subsequent layers. The second disadvantage of multilayer networks without connections across layers is that inaccuracies in the first layers are magnified by inaccuracies of subsequent layers. With seven hidden layers, where there are only single neurons in each layer, it would be more desirable to introduce several neurons in each layer and to reduce the number of hidden layers. Such architectures, known as bridged multilayer perceptron (BMLP), are a good compromise between MLP and FCC architectures [13], [14]. In order to implement the neural networks with arbitrary connected architecture, a special method of computation scheme had to be developed. The computation process on microcontroller follows the neuron-by-neuron algorithm [3], [4], [8], [11], [17]. This method requires special modifications due to the fact that assembly language is used with very limited memory resources.

To implement more powerful neural network architectures on microcontrollers, the special NBN computing routine was used. The NBN routine was described in detail in [11], but its efficient implementation on microcontroller with simplified arithmetic was another challenge.

3. Multiplication

The microcontroller has 8 bit by bit multiplication hardware. A routine was developed as hardware multiplier cannot handle floating point hardware. This allows fastest multiplication of fractional values. 2 8-bit numbers were given to multiply routine as first 8-bit is integer portion and last 8-bit is fractional portion. Finally routine returns 32-bit product.

$$\frac{A.C\ 256^2+256(A.D+B.C)+B.D}{256^2} \quad (1)$$

The hardware does not require any shifts or division. This process allows each neuron to multiply the weights by inputs and then use 32-bit results as an accumulator for inputs of neuron. Once the product is calculated the results are stored in 32-bit net register. It is essential in adding and subtracting stages. If the absolute value of net value is greater than 4, neuron is in saturation and activation function is skipped resulting in positive one or negative one respectively [1].

4. Pseudo Floating Point Implementation

Due to the importance of weight, pseudo floating point is implemented. The first 16 bits are used to represent the weights, nodes, and inputs for the neural network. The nonconventional part of this floating point routine is the way the exponent and mantissa are stored. This allows more significant digits for every weight using less memory. This is tailored directly around the needs of neural network forward calculations. This solution requires the analysis of the weights of each neuron and scales them accordingly and assigns an exponent for the entire neuron.

5. Algorithm

5.1Activation Function

Creating activation function is a challenge because of the limitations of the microcontroller. Several approaches were initially considered such as lookup tables or linear piecewise approximation or Elliot function. But the results were not desirable. As an alternation *tanh* function was considered but accuracy is possible for very small and very large net values. Without hardware division will be too slow for a process for final solution.

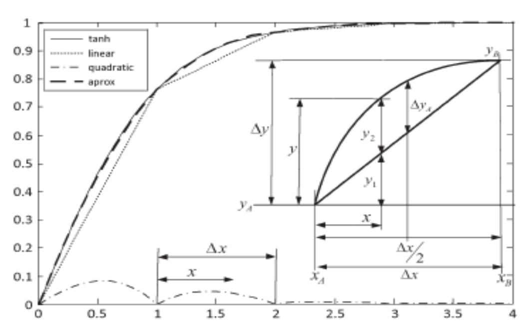

Figure 1: Solid Line Approximation of Tangent Function

A second order approximation of *tanh* was chosen for its accuracy as well as its simple arithmetic calculations. Several features were added to the activation function besides simply calculating a second order approximation of *tanh.* One of these features analyses the inputs to the activation function and converts negative numbers to positive numbers to make the internal calculations faster and reduce the number of values that must be stored in the lookup table. The sign is restored at the end of the activation function. Another feature is a check to see if the neuron is in saturation. In other words, make sure that the net value is within a given range. In this case the second order approximation is skipped and the neuron is put into saturation.

Routine requires 30 values to be stored. In order to obtain acceptable accuracy, *tanh* equivalent of 25 numbers between zero and four are stored. Then a point between each pair from the linear approximation is stored with round off to 16-bits accuracy. These points are the peaks of a second-order polynomial that crosses at the same points as the linear approximations. Then a point between each pair from the linear approximation is stored. Based on the four most significant bits that are input into the activation function, a linear approximation of tangent hyperbolic is selected. The remaining bits of the number are used in the second-order polynomial.

The coefficients for this polynomial were previously indexed by the integer value in the first step.

The approximate value of *tanh*(*net*) is found in several steps [1], [3].

1) Using the higher bits of the *net* value, the proper piecewise segment is selected between x_A and x_B *from fig(1).* Then, the corresponding values of y_A and y_B are found from memory. In our implementation, four most significant bits were used to retrieve data for 16 segments.
2) The Δx value is obtained from lower bits of *net* value.
3) Using x_A, x_B, y_A, and y_B, the first-order linear approximation is computed at first

$$y_1(x) = y_A + \frac{(y_B - y_A)\cdot x}{2\Delta x} \qquad (2)$$

4) Then, the quadratic component is found as

$$y_2(x) = \frac{4.\Delta y_A.x.(\Delta x - x)}{\Delta x^2} \qquad (3)$$

where Δy_A values are read from memory. Divisions in “(2)” and “(3)” can be easily replaced by shift operations.

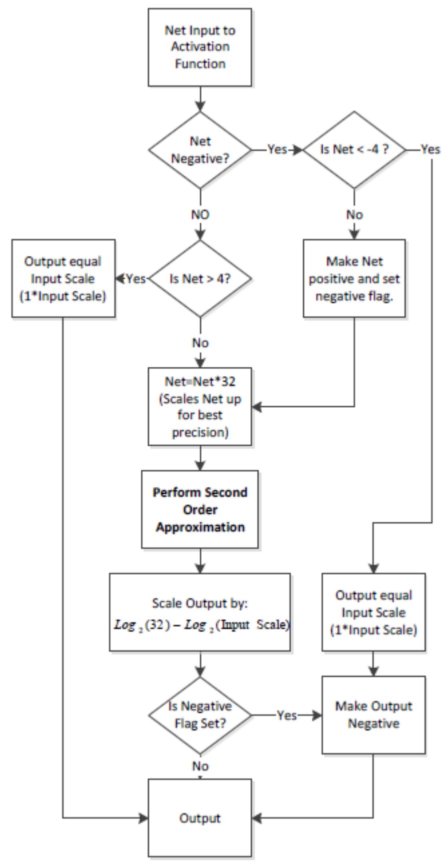

Figure 2: Logic Block Diagram of the Activation Function

In order to utilize 8-bit hardware multiplication, the size of Δx was selected as 128. This way the division operation in both equations can be replaced by the right shift 54 operation. Calculation of y_1 requires one subtraction, one 8-bit multiplication, one shift right by 7 bits, and one addition. Calculation of y_2 requires one 8-bit subtraction, two 8-bit multiplications and shift right by 14-bits.

6. Application

In order to demonstrate that the microcontroller neural network is performing correctly, several example control problems were tested. Fire classification using gas sensor was used as a practical application for this embedded neural network [2]. The particular application is shown for the level of fire to be classified as Very Low, Low, Medium, High and Very high. The process is tested with the microcontroller hardware in the loop for MLP and Neuron by Neuron process. The sensor data is transmitted via the serial port from MATLAB to the microcontroller. The microcontroller then calculates the results and transmits this data via the serial port back to MATLAB. The reason for this simulation is to isolate the errors in the system to those produced by the microcontroller calculations. In this test the system having any inaccuracy of the sensors can be avoided. The possibility of errors entering the system from external measurement tools can also be removed.

7. Functional Block Diagram

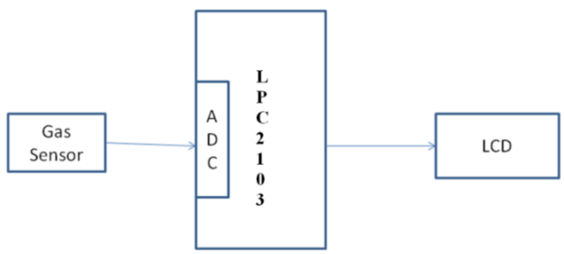

Figure 3: Linearizing Nonlinear Gas Sensor using Neural Network

A MQ-2 Gas sensor with 300 – 10000ppm is given as input to the ARM7 processor. MQ-2 is used for detecting gases like LPG, i-butane, propane, methane, alcohol, hydrogen and smoke. LPC2103 is used as ARM7 processor. LPC2101/02/03 microcontrollers are based on a 16-bit/32-bit ARM7TDMI-S CPU with real time emulation that combines the microcontroller with 8 KB, 16 KB or 32 KB of embedded high flash memory. A 128-bit wide memory interface and unique accelerator architecture enables 32-bit code execution at the maximum clock rate. Due to their tiny size and low power consumption, LPC2101/02/03 are ideal for applications where miniaturization is a key requirement, such as access control and point-of-sale. The 10-bit ADC provides eight analogue inputs, with conversion time as low as 2.44µs per channel. Various 32-bit, 16-bit timers and improved PWM features through output, match on all timers and suitable for industrial control. Low power Real Time Clock (RTC) with independent power and dedicated 32 KHz clock input. NBN algorithm is implemented in Embedded C for individual neuron calculation. The final Net value is calculated and then the output is displayed on the LCD screen (2x16 Liquid crystal Display).

8. Experimental Results

8.1 Comparison result

8.1.1MLP Training data

The problem consists of one input variable X and one target variable T with data generated by sampling X at equal intervals and then generating target data by computing SIN(2*PI*X) and adding Gaussian noise. A 2-layer network with linear outputs is trained by minimizing a sum-of-squares error function using the scaled conjugate gradient optimizer.

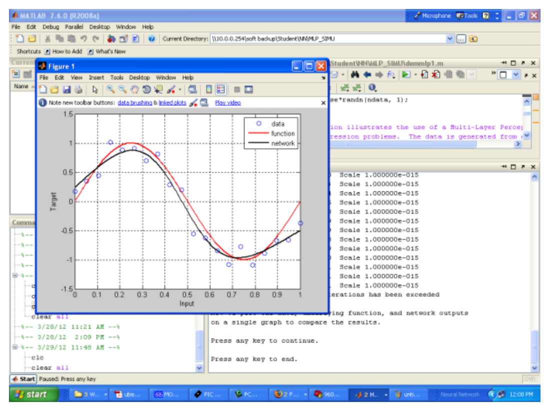

Figure 4: MATLAB View of MLP

8.1.2 EBP Training Data

Table 1: Normalized Input Readings used for ENN Training for fire classification using array of Gas Sensor.

Sample case	**Temp sensor**	**TGS 880**	**TGS 822**	**TGS 2600**	**TGS 2611**	**TGS 2610**	**TGS 2602**
Paper	0.72	0.14	0.72	0.63	0.52	0.20	0.12
Paper	0.67	0.06	0.66	0.60	0.55	0.20	0.30
Petrol	0.69	0.10	0.69	0.52	0.69	0.18	0.09
Petrol	0.56	0.24	0.85	0.72	0.74	0.40	0.07

Plastic	0.67	0.09	0.38	0.39	0.35	0.09	0.13
Plastic	0.67	0.01	0.43	0.56	0.50	0.13	0.09
Kerosene	0.60	0.13	0.74	0.67	0.70	0.13	0.09
Kerosene	0.63	0.19	0.78	0.66	0.63	0.14	0.08
Normal	0.58	0.02	0.10	0.12	0.09	0.05	0.08
Wood	0.69	0.13	0.55	0.70	0.73	0.28	0.39

Looking at the training data it is assumed that most of the neurons in the network are not actually contributing to the correct classification of the fires. It is very difficult to detect whether the entire network is working or a small number of neurons are carrying the load. Error Back Propagation (EBP) training algorithm is used for training the data [6].

8.1.3 NBN Training data for n number of samples

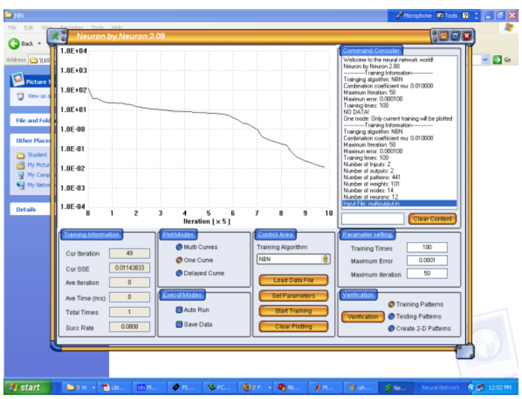

Figure 5: MATLAB View of NBN

After comparing the result of different implementation methods of neural network, it was obvious that Neuron by Neuron method starts settling initially for n number of neurons whereas MLP were at peak increasing for set of samples and activation function. Due to increase in variable size in microcontroller, it becomes necessary to store all weights and nodes in program memory. So the accuracy and the speed are increased. Looking at Figure (5) which is the error analysis of the ideal neural network it can be seen that about half of the large outlier errors are produced by the neural network itself.

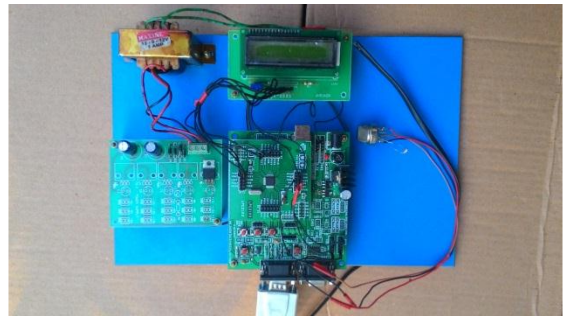

Figure 6: Schematic of the Experimental Setup

9. Conclusion

This paper presents a solution for embedded neural networks across many types of hardware and for many applications. The application shows the method linearizing nonlinear sensor data for nonlinear control problems using neural networks at the embedded level. With the correct neural network architectures, the very difficult problems can be solved with just few neurons. The operational goal was to create neural network that was as fast as possible while introducing an error that is not greater than the error already introduced by using a neural network to approximate the original function. When using NBN training method, these networks can be easily trained. Then by using NBN forward calculation method, networks with any architecture can be used at embedded level. The second order approximation of *tanh* in conjunction with pseudo floating-point routines allows almost any neural network to be embedded in a simple low-cost microcontroller. This process balances speed with accuracy for systems that do not having floating-point hardware. The neuron by neuron approach using the arrays for weights and nodes can be taken to any platform. The limitations on the architecture embedded in this microcontroller are limited only by the number of weights.

References

[1] Cotton N. J., "A neural network implementation on embedded systems," Ph.D. dissertation, Dept. Elect. Eng., Auburn Univ., Auburn, AL, 2010.

[2] Jasmine M., "Neural networks for compensation of nonlinearities," 'International Journal of P2P Network Trends and Technology (IJPTT), vol2, issue4.No 1, July 2012. pp 24-27.

[3] Cotton N. J., “Compensation of Non-linearities Using Neural Networks Implemented on Inexpensive Microcontroller,” IEEE Transactions on Industrial Electronics, vol.58, No.3, March 2011, pp. 733-740.
[4] Cotton N. J., Wilamowski B. M., and Dundar G., “A neural network implementation on an inexpensive eight bit microcontroller,” in Proc. 12th Int. Conf. INES, Miami, FL, Feb. 25–29, 2008, pp. 109–114.
[5] Dempsey, G.L.; Alt, N.L.; Olson, B.A.; Alig, J.S., "Control sensor linearization using a microcontroller-based neural network," Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation', IEEE International Conference on 1997, vol.4, pp.3078-3083.
[6] Bashyal.S, Venayagamoorthy G.K and Paudel. B, "Embedded neural network for fire classification using an array of gas sensors” in proc., IEEE SAS, 2008, pp.146-148.
[7] CottonN. J. and Wilamowski B. M., “Compensation of sensors non-linearity with neural networks,” in Proc. 24th IEEE Int. Conf. AINA, Apr. 20–23, 2010, pp. 1210–1217.
[8] Jian-Xun Peng; Kang Li; Irwin, G.W., "A New Jacobian Matrix for Optimal Learning of Single-Layer Neural Networks," Neural Networks, IEEE Transactions on, vol.19, no.1, Jan. 2008 pp.119-129.
[9] Wilamowski, B.M., "Neural network architectures and learning algorithms," Industrial Electronics Magazine, IEEE, vol.3, no.4, Dec. 2009, pp.56-63.
[10] Wilamowski, B. M.; Cotton, N.; Hewlett, J.; Kaynak, O., "Neural Network Trainer with Second Order Learning Algorithms," Intelligent Engineering Systems, 11th International Conference, vol., June 29 2007-July 1 2007 pp.127-132.
[11] Wilamowski, B. M.; Cotton, N. J.; Kaynak, O.; Dundar, G., "Method of computing gradient vector and Jacobean matrix in arbitrarily connected neural networks," Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on, vol.55, no.10. oct.2008, pp.3784-3790.
[12] Wilamowski, B.M.; Iplikci, S.; Kaynak, O.; Efe, M.O., "An algorithm for fast convergence in training neural networks," Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on, Vol.3, no., 2001,pp. 1778-1782.
[13] Wilamowski, B.M, “Challenges in applications of computational intelligence in industrial electronics”, in Proc. IEEE ISIE, Bari, Italy, Jul. 5-7, 2010, pp. 15-22.
[14] Wilamowski, B.M and Yu H, “Neural network learning without back propagation,” IEEE Trans. Neural Netw., vol. 21, no. 11, nov 2010, pp. 1793-1803.
[15] Betin.F, Pinchon.D and Capolino, “Fuzzy logic applied to speed control of a stepping motor drive,” IEEE Transaction Ind. Conf. AINA. Apr.20-23,2010, pp.1210-1217.
[16] Medrano-Marques N.J and Martin-del-Brio, “Sensor linearization with neural networks,” IEEE Trans. Ind. Electron., Vol.48, no.6, Dec.2001, pp.1288-1290.
[17] Wilamowski, B.M and Yu H, “Improved computation of Levenberg-Marquard training,” IEEE Trans. Neural Netw., Vol.21, no.6, Jun. 2010, pp.930-937.
[18] Betin.F, Silvert.A, Yazidi. A and Capolino. G.A, “Determination of scaling factors for fuzzy logic control using the sliding-mode approach: Application to control of a DC machine drive,” IEEE Trans. Ind. Electron., vol.54, no.1, feb, 2007, pp. 296-309.

B. GowriShankari Current Place is Chennai and born on 04-04-1982.
She completed her ME post graduation from Bharath University, Chennai, in Embedded System Design. B.E graduation in Electronics and Instrumentation from Annamalai University. She is currently working as Assistant Professor in New Prince Shri Bhavani College of Engineering & Technology; with more than 4 years of experience in teaching profession she has published various national and international journals. Her research area includes embedded system, Microprocessor.
Email: gowrishankari2506@gmail.com

Neethu PS Current place is Chennai and born on 30-10-1986.
She completed her ME post graduation from Anna University, Chennai, in Applied Electronics. B.Tech graduation in Electronics and Instrumentation from MG University, Kerala. Currently working as Assistant Professor in New Prince Shri Bhavani College of Engineering & Technology, with more than 5 years of experience in the teaching profession. She has published various national and international journals. Her research area includes Image processing, Process control.