Engineering Applications of Artificial Intelligence 30 (2014) 179–188

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Smart sensor/actuator node reprogramming in changing environments using a neural network model

Francisco Ortega-Zamorano [a,*], José M. Jerez [a], José L. Subirats [a], Ignacio Molina [b], Leonardo Franco [a]

[a] Department of Computer Science, E.T.S.I. Informatica, University of Malaga, Bulevar Louis Pasteur, 35, 29071 Malaga, Spain
[b] Max Planck Institute, Munich, Germany

ARTICLE INFO

Article history:
Received 17 September 2013
Received in revised form
23 December 2013
Accepted 9 January 2014
Available online 1 February 2014

Keywords:
Constructive Neural Networks
Microcontroller
Arduino

ABSTRACT

The techniques currently developed for updating software in sensor nodes located in changing environments require usually the use of reprogramming procedures, which clearly increments the costs in terms of time and energy consumption. This work presents an alternative to the traditional reprogramming approach based on an on-chip learning scheme in order to adapt the node behaviour to the environment conditions. The proposed learning scheme is based on C-Mantec, a novel constructive neural network algorithm especially suitable for microcontroller implementations as it generates very compact size architectures. The Arduino UNO board was selected to implement this learning algorithm as it is a popular, economic and efficient open source single-board microcontroller. C-Mantec has been successfully implemented in a microcontroller board by adapting it in order to overcome the limitations imposed by the limited resources of memory and computing speed of the hardware device. Also, this work brings an in-depth analysis of the solutions adopted to overcome hardware resource limitations in the learning algorithm implementation (e.g., data type), together with an efficiency assessment of this approach when the algorithm is tested on a set of circuit design benchmark functions. Finally, the utility, efficiency and versatility of the system is tested in three different-nature case studies in which the environmental conditions change its behaviour over time.

1. Introduction

Sensors (or detectors) are devices that permit the measurement of chemical or physical variables, transforming them into electrical signals, in order to interpret the signals from various sensors and send the sensed data or make a decision according to them. Microcontroller boards are an economic, small and flexible solution, and thus are the most common controller used in a sensor node, also known as a Mote, commonly used in Wireless sensor network (Yick et al., 2008; Sengupta et al., 2013), but also used in other important technologies such as Embedded systems (Marwedel, 2006; Mamdoohi et al., 2012) and Real-time systems (Kopetz, 1997; Wang et al., 2010). Motes are nowadays widely employed in all kind of industrial applications, in several of them, the problem requires an action upon the environmental conditions, in this case a sensor/actuator node is required.

In cases when the environmental conditions evolve over time, the original sensor/actuator programming can lead to incorrect decisions, and thus it is necessary to change or adapt the decision-making process to the new conditions (Sayed-Mouchaweh and Lughofer, 2012). The traditional option to solve this problem has been to send the sensed data to a central unit, where a person interprets the data and reprogram the microcontroller with the new set of rules (Han et al., 2005; Wang et al., 2006; Shaikh et al., 2010). Different reprogramming techniques have been proposed as a way of dynamically changing the behaviour of the sensors without having to manually reprogram them, because traditional reprogramming requires in most cases the interruption of the process for loading the new binary code, with the consequent loss of time and energy, involved in the communication process to the central unit (Rassam et al., 2013; Aiello et al., 2011). A first step towards reducing the previous effects has been to incorporate machine learning systems in the decision-making process, automating the response of the microcontroller without interrupting its execution and sending just a small fraction of code to the microcontroller (Urda et al., 2012; Canete et al., 2012; Farooq et al., 2010). However, recent advances in the computing power of sensors permit the inclusion of learning systems in the microcontroller ("on-chip" learning), adapting the sensor/actuator behaviour dynamically according to the sensed data (Aleksendrić et al., 2012; Mahmoud et al., 2013).

Artificial Neural Networks (ANNs) (Haykin, 1994) are a kind of machine learning models, inspired on the functioning of the brain,

* Corresponding author. Tel.: +34 952 13 28 47; fax: +34 952 131 397.
E-mail addresses: fortega@lcc.uma.es, FOZamorano@gmail.com (F. Ortega-Zamorano).

0952-1976/$ - see front matter © 2014 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2014.01.006

that can be utilised in clustering and classification problems, having been applied successfully in several fields, including pattern recognition (Dhanalakshmi et al., 2011), stock market prediction (Park and Shin, 2013), control tasks (Zhai and Yu, 2009), medical diagnosis and prognosis (Kodogiannis et al., 2007), and so on. Despite years of research in the field of ANN, selecting a proper architecture for a given problem remains a difficult task (Gómez et al., 2009; Hunter et al., 2012; Lakshmi and Subadra, 2013), and several strategies have been proposed for solving or alleviating this issue. In particular, Constructive Neural Networks (CoNNs) offer the possibility of generating networks that grows as input information is received, matching the complexity of the data (Franco et al., 2009). Moreover, the training procedure in CoNN, considered a computationally expensive problem in standard feedforward neural networks, can be done on-line and relatively fast. C-Mantec is a recently introduced CoNN algorithm that implements competition between neurons, also incorporating a built-in filtering scheme to avoid overfitting problems. These two characteristics permit the algorithm to generate compact neural architectures with very good generalisation capabilities, making the algorithm suitable for its application to devices with limited resources like microcontrollers. The main limitations of these devices are memory size and computing speed, and thus an efficient implementation of the algorithm is needed. Despite the existence of alternative evolving models (Lughofer, 2011; Angelov, 2010; Huang et al., 2005), C-Mantec has been selected based mainly on the three following features: dynamic generation of compact architectures, good prediction ability and robustness to parameter setting.

In the present work, we have fully implemented the C-Mantec (Subirats et al., 2012) constructive neural network model in an Arduino UNO board, including the whole learning process to implement the automatic reprogramming process for decision-making into the sensor/actuator in changing environments, avoiding communication to other devices.

The Arduino UNO board was used (Oxer and Blemings, 2009) as it is a popular, economic and efficient open source single-board microcontroller that allows easy project development (Lian et al., 2013; Cela et al., 2013; Ortega-Zamorano et al., 2013; Kornuta et al., 2013). We have also proposed three case studies of different nature which require reprogramming to the decision-making process, demonstrating that the time involved in the sensor reprogramming is significantly lower than in the traditional case, without the need to send any information to another device (as a control unit), saving a large fraction of the required energy resources.

The paper is structured as follows: first, we briefly describe in Section 2 the C-Mantec constructive neural network algorithm used, followed by a description of the Arduino UNO microcontroller board in Section 3. Section 4 includes the details of the implementation with the results obtained shown in Section 5. Thereafter, three case studies are evaluated in Section 6, checking the efficiency of the system in them, to finally extract the conclusions in Section 7.

2. C-Mantec, constructive neural network algorithm

C-Mantec (Subirats et al., 2012) (Competitive Majority Network Trained by Error Correction) is a novel neural network constructive algorithm that utilises competition between neurons and a modified perceptron learning rule (thermal perceptron Frean, 1990) to build single hidden layer compact architectures with good prediction capabilities for supervised classification problems. As a CoNN algorithm, C-Mantec generates the network topology on-line during the learning phase, avoiding the complex problem of selecting an adequate neural architecture. The novelty of C-Mantec in comparison to previous proposed constructive algorithms is that the neurons in the single hidden layer compete for learning the incoming data, and this process permits the creation of very compact neural architectures. The binary activation state (S) of the neurons in the hidden layer depends on N input signals, ψ_i, and on the actual value of the N synaptic weights (ω_i) and bias (b) as follows:

$$S = \begin{cases} 1(ON) & \text{if } h \geq 0 \\ 0(OFF) & \text{otherwise} \end{cases} \quad (1)$$

where h is the synaptic potential of the neuron defined as

$$h = \sum_{i=0}^{N} \omega_i \psi_i \quad (2)$$

In the thermal perceptron rule, the modification of the synaptic weights, $\Delta\omega_i$, is done on-line (after the presentation of a single input pattern) according to the following equation:

$$\Delta\omega_i = (t - S)\psi_i T_{fac}, \quad (3)$$

where t is the target value of the presented input, and ψ represents the value of input unit i connected to the output by weight ω_i. The difference in the standard perceptron learning rule is that the thermal perceptron incorporates the T_{fac} factor. This factor, whose value is computed as shown in Eq. (4), depends on the value of the synaptic potential and on an artificially introduced temperature (T).

$$T_{fac} = \frac{T}{T_0} e^{-|h|/T}, \quad (4)$$

The value of T decreases as the learning process advances according to Eq. (5), similar to a simulated annealing process.

$$T = T_0 \cdot \left(1 - \frac{I}{I_{max}}\right), \quad (5)$$

where I is a cycle counter that defines an iteration of the algorithm on one learning cycle, and I_{max} is the maximum number of iterations allowed. One learning cycle of the algorithm is the process that starts when a random chosen pattern is presented to the network and finishes after checking that the output of the network is equal to the target for this pattern, or when a chosen neuron (the neuron with largest T_{fac} value or a new added neuron) modifies its synaptic weights to learn the actual presented pattern.

The C-Mantec algorithm has three parameters to be set at the time of starting the learning procedure, and several experiments have shown the robustness of the algorithm that operates fairly well in a wide range of parameter values. The algorithm has the following three parameters:

- I_{max}: Maximum number of learning iterations allowed for each neuron in one learning cycle.
- g_{fac}: Growing factor that determines when to stop a learning cycle and includes a new neuron in the hidden layer.
- ϕ: Determines in which case an input example is considered as noise and removed from the training dataset according to the following condition:

$$\text{delete}(x_i) | N_{LT} \geq (\mu + \phi\sigma), \quad (6)$$

where x_i represents an input pattern, N is the total number of patterns in the dataset, N_{LT} is the number of times that pattern x_i has been presented to the network on the current learning cycle, and where μ and σ correspond to the mean and variance of the distribution for all patterns on the number of times that the algorithm has tried to learn each pattern in a learning cycle. The learning procedure starts with one neuron present in the single hidden layer of the architecture and an output neuron that computes the majority function of the responses of the hidden