International Journal of Innovative Computing, Information and Control
Volume **8**, Number **11**, November **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 7561–7578

ENERGY-EFFICIENT REPROGRAMMING IN WSN USING CONSTRUCTIVE NEURAL NETWORKS

Daniel Urda Muñoz, Eduardo Cañete Carmona
José Luis Subirats Contreras, Leonardo Franco
Luis Manuel Llopis Torres and José Manuel Jerez Aragonés

Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
Bulevar Louis Pasteur, No 35, 29071, Málaga, Spain
{ durda; jlsubirats; lfranco; jja; ecc; luisll }@lcc.uma.es

Received August 2011; revised December 2011

Abstract. *In this paper, we propose the use of neural network based technologies to carry out the dynamic reprogramming of wireless sensor networks as an alternative to traditional methodology. An analysis and comparison of the energy costs involved in reprogramming wireless sensor networks was done using rule-based programming (TP), standard feedforward neural networks (FF), and the C-Mantec (CM) algorithm, a novel method based on constructive neural networks. The simulation results, first performed on an array of sensor networks under the COOJA simulator (considering best, medium and worst case scenarios for three benchmark problems) and finally evaluated on a case of study with identical conditions, show that the use of neural network based methodologies (FF & CM) produces a significant saving in resources, measured by the number of packets transmitted, the energy consumed and the time needed to reprogram the sensors.*
Keywords: Wireless sensor networks, Constructive neural networks, Dynamic reprogramming, Feedforward neural networks

1. **Introduction.** *Wireless Sensor Networks* (WSNs) are a new form of distributed computing where sensors (tiny, low-cost and low-power nodes, colloquially referred to as "motes") deployed in the environment communicate wirelessly to gather and report information about physical phenomena [47]. WSNs have been successfully used for various application areas, such as environmental monitoring, object and event detection, military surveillance, precision agriculture [18] where thousands of motes are deployed. One of the most important issues in the context of the WSNs is energy consumption, which must be minimized, in order to increase longevity. The energy consumption centered in the communication process between sensor nodes is one of the most delicate issues when a WSN system is going to be developed [1,22]. A bad management of the energy resources could cause the partitioning of the network, a quicker reduction in the life of the network and a poor performance. Due to the importance of this issue, a large number of articles have been presented by the scientific community dealing with the problematic of energy consumption. Most of the approaches that have been presented focus on protocols related with routing [16,28], duty cycle [9], cluster formation [20,21] and data aggregation [32].

In the context of intelligent systems and innovative computing can be found several texts dealing with different problematics related to WSNs as is amply described in [12]. Specifically, there are several texts where the *Artificial Neural Networks* (ANN) have been used to solve typical problems of WSNs such as path discovering [4,19,33], node clustering [3,7], cluster-head selection [6], data aggregation [25,29,37,46], data association, data classification [25,26,29,48] and data prediciton [25,31,35] focusing on maximizing the

node's battery life. However, there is no reference given in the literature to the use of ANN in order to deal with the reprogramming problem, in terms of energy consumption, of a WSN.

Recent technological advances have permitted the incorporation of dynamic (over-the-air) reprogramming of sensor nodes of a WSN, instead of the manual reprogramming [24,34,45,49] where developers have to interact physically with the sensor nodes. Reprogramming is necessary and desirable in WSNs in order to remove bugs, to add new functionality or to change some conditions on the developed software. Even if a WSN is initially well designed and may appear that it will not need further reprogramming, experience has shown that real performance is different from simulations and testbeds phases and thus reprogramming is indeed necessary in most cases. For example, let us suppose that a set of nodes sense temperatures and they rely the data to the base station when this temperature is higher than a parameter value. If we need to change the parameter value or we want to take into account the humidity level together with the temperature to send the data, then a new image code must be installed in the nodes. In the case of traditional programming, programmers must take the node from the deployment, reinstall the code and set the node again. The dynamic reprogramming allows programmers to transmit the new image through a wireless link to the sensor nodes improving the repogramming time. This process is known as *code dissemination* and its main drawback is the energy consumption required to transmit the entire new or modified code image to the sensor nodes. Moreover, this drawback is accentuated when the code image that is being sent contains minimum changes in comparison to the previous code.

This paper describes the possibility of incorporating neural network based models into WSNs to dynamically reprogram sensor nodes minimising the energy consumption and the reprogramming time. Therefore, our work aims to use feedforward architectures trained by back-propagation algorithm and a novel *Constructive Neural Network* (CNN) algorithm as embedded classifiers into the sensor nodes software. Concretely, the COOJA sensor network simulator [13] has been used to compare the performance of TP reprogramming to neural network based technologies (FF & CM), analyzing the time and energy resources involved in the reprogramming process.

The rest of the paper is structured as follows. Section 2 describes the three benchmark datasets used to carry out the simulations considering three possible real scenarios where WSNs are perfectly used. Moreover, the different methods applied (TP, FF & CM) within the COOJA simulator used to simulate the best, medium and worst cases of the reprogramming task are described at the same time that the results obtained in each benchmark for the different applied methods are presented taking into account the time and energy resources. Section 3 describes in detail a case of study carried out on the problem of falling detection [8,23] to confirm the results obtained from the three benchmark datasets. Finally, in Section 4 we provide some conclusions based on the analysis of the results obtained for the three benchmark datasets and the case of study shown in Section 2.3 and Section 3.2 respectively.

2. Methods and Bechmark Testing.

2.1. Benchmark datasets.

In [2,41], two real WSNs systems are described in order to monitor certain readings, one related to high-quality wine production and the other to congestive heart failure in patients. As there is no available data corresponding to these problems, the data used in this work were obtained from a very well known public database easily accessible from the Internet[1].

[1] http://archive.ics.uci.edu/ml/datasets.html

Three "real-life" problems, which could be modelled through WSNs, were used for the analysis. Each benchmark dataset fits well with the two real WSNs systems presented and models a problem which could be implemented on the nodes of a WSN through a procedure that essentially consists in receiving several parameters and making a decision based on them.

The details of the datasets are as follows:

- Red wine quality case test.
  Red wine quality can be monitored using 11 parameters (alcohol, pH, fixed acidity, residual sugar, density, etc.) present in this dataset. In this case, a WSN can be deployed in a large wine-cellar to control the wine fermentation process. At a given moment, it may be necessary to modify the algorithm in charge of deciding the quality of the wine stored n each of the thousands of barrels allocated in the wine-cellar and, for this purpose, reprogramming the nodes will become necessary.
- Parkinson status test.
  This dataset consists of 197 records from 31 patients for which a set of 22 parameters has been measured with the aim of predicting the advance of Parkinson using speech readings [42]. Most of these parameters could be measured through small wireless sensor networks, known as *body sensor networks*, deployed on the body of a person. The information collected is sent to a main node also fixed to the person's body which is capable of analyzing the information. As not all the parameters might be sensed by the body sensor network, normally the main node sends the information gathered to the control center in charge of further analysis of the information and supervising the entire process. For example, let us imagine an Old Age Pensioner (OAP) Day Care Center with 10 people per floor, as depicted schematically in Figure 1. Each person's body sensors report the person's information periodically via wireless transmission to the control center and reciprocally the control center can

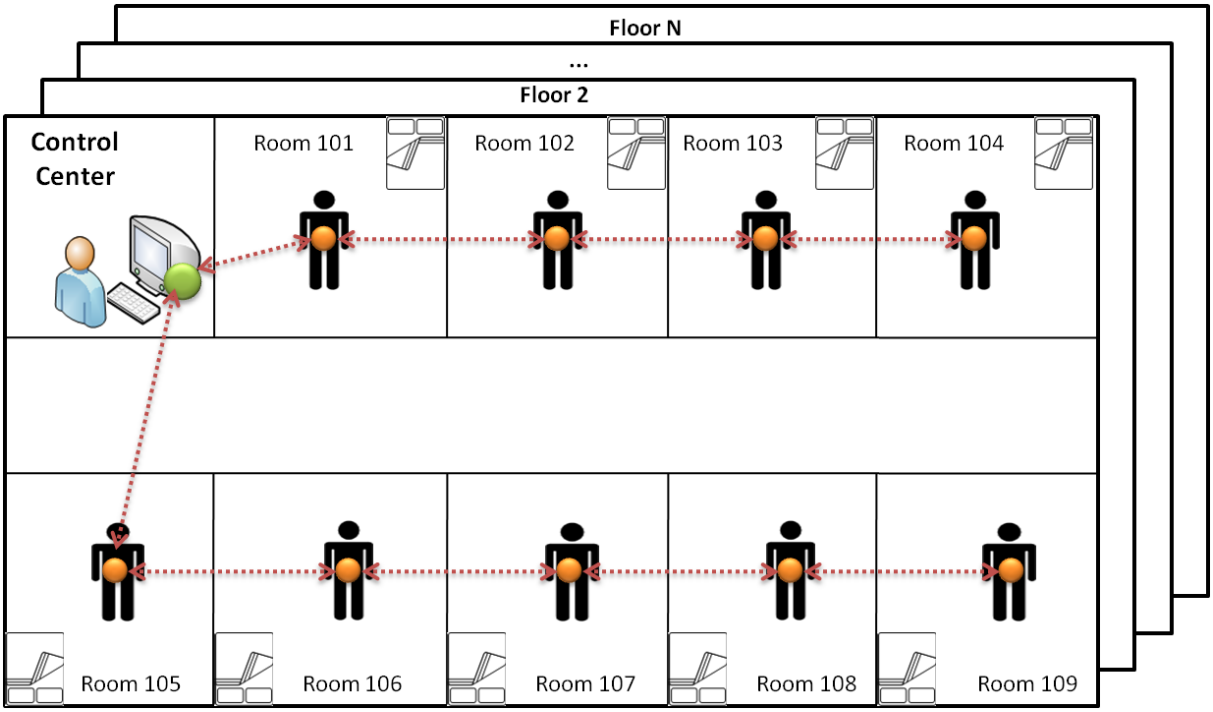

FIGURE 1. Schematic representation of a day care center where a body sensor network might be used to monitor the disease status of the patients

process this information and send complementary data, that cannot be sensed directly, such as age, or height, to the person's sensor. Then the progress of Parkinson of the patient can finally be computed at the main node of the body sensor network.

- Pima indians diabetes data set.

  This well known dataset consists in 768 registers from subjects belonging to the population of Pima Indians. The data can be used to predict whether a person can be considered diabetic or healthy. As in the case of Parkinson dataset, similar body sensors could be used to control the patient's status, that for this case could be, for example, at their home in a determined neighbourhood instead of at a Day Care Center. The information can also be transmitted via a wireless connection to a control node for further analysis.

2.2. **Methods.** Figure 2 describes the two different and paralell approaches that have been followed in this work: Traditional Dynamic Reprogramming (TDR) and Neural Network Dynamic Reprogramming (NNDR) of wireless sensors (Section 2.2.1). COOJA simulator (Section 2.2.3) has been used in order to simulate a real reprogramming environment and carry out all the experiments. On the left hand side of Figure 2, the traditional reprogramming approach that implies modifying the software implementation

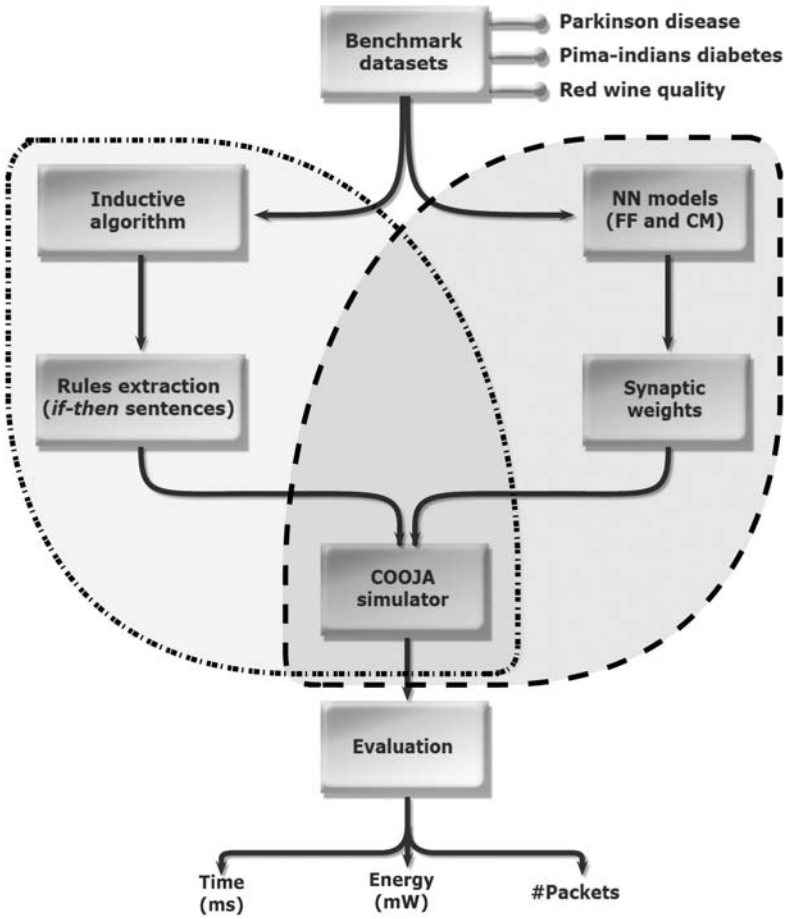

FIGURE 2. Workflow representation of the traditional programming approach (left side) and our proposal using a neural network based methodology (right side) in the reprogramming process of WSNs. Results are obtained in terms of time, energy costs and number of packets sent in the reprogramming process using the COOJA simulator.

is described. On the right hand side, our proposal using a neural network based methodology is described showing how it reduces the energy costs involved in the reprogramming process of wireless sensor networks and, therefore, increases the network life. We do not give explicit details about standard feedforward networks here because they are a well known methodology for the design of classifiers.

2.2.1. *TDR and NNDR of wireless sensors.* The design of classifiers is usually implemented in WSNs through a set of IF-THEN conditional sentences. For each dataset tested, a set of conditional sentences was extracted from the rules obtained by the application of an inductive algorithm. In TDR, a change in the classifier rules, even if the modifications are minimal, involves sending the new entire code image again, with the corresponding extra energy expenditure.

The use of a neural network based classifier involves creating and training the architectures to obtain a set of synaptic weights. This set of weights, together with the initial code needed to compute the output of the classifier, have to be sent to the sensor nodes the first time the nodes are programmed. In NNDR, further changes in the classifier only involves sending the new values for the synaptic weights, reducing the amount of bytes to be transmitted with the corresponding saving in energy consumption. However, it has to be noted that it would be possible to figure out a way of avoiding the process of sending the whole code in TDR. Nevertheless, first, the standard way of reprogramming the sensors nowadays consists in transmitting the whole code; and second, a rules based classifier is more sensitive to changes in the set of input patterns than neural network models.

To exemplify the difference between TDR and NNDR, imagine the scenario of one of the benchmark dataset described in Section 2.1, where patients in a Day Care Center are physically distributed in different rooms. In Table 1, an example of the code needed to reprogram the nodes' software is shown for the case of the Pima Diabetes database. To the left of the table, the case of TP is illustrated using a set of conditional sentences, extracted from a rule generator algorithm, highlighting that normally all the code has to be transmitted every time a reprogramming of the sensor is required. The case of using neural network based programming (FF & CM) is illustrated on the right by a short code that essentially instructs the sensor to compute a matrix product between a set of inputs and the synaptic weight values stored. For the case of using neural networks, only the synaptic weight values can be modified, as the software code for the sensor does not need modifications.

2.2.2. *C-Mantec constructive neural network algorithm.* C-Mantec (Competitive Majority Network Trained by Error Correction) is a novel neural network constructive algorithm that utilizes competition between neurons and a modified perceptron learning rule to build compact architectures with good prediction capabilities. The novelty of C-Mantec is that the neurons compete for learning the new incoming data, and this process permits the creation of very compact neural architectures. At the single neuronal level, the algorithm uses the thermal perceptron rule, introduced by M. Frean in 1992 [15], that improves the convergence of the standard perceptron for non-linearly separable problems. The activation state (S) of this perceptron depends on the N input signals, ψ_i, and on the actual value of the N synaptic weights (ω_i) and the bias (b) as follows:

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

TABLE 1. A comparison between traditional programming and neural networks code illustrated for the "Pima Indians diabetes" data set. Conditional sentences extracted from a rule generator algorithm are normally used in TP (left). A short code is needed to implement the neural network models, in which only the synaptic weights values should be transmitted every time an updating of the code has to be done (right).

Traditional programming	Neural network programming

```
if (plasma_glucose_concentration <= 0.6181) {
  if (age <= 0.1333) {
    DiagnosticResult(``healthy");
  }
  else {
    if (body_mass_index <= 0.3994) {
      DiagnosticResult(``healthy");
    }
    else {
      if (diastolic_blood_pressure <= 0.7377) {
        if (plasma_glucose_concentration <= 0.5377) {
          DiagnosticResult(``healthy");
        }
        else {
          DiagnosticResult(``diabetes");
        }
      }
      else {
        DiagnosticResult(``healthy");
      }
    }
  }
else {
  if (plasma_glucose_concentration <= 0.8342) {
    if (age <= 0.05) {
      DiagnosticResult(``healthy");
    }
    else {
      DiagnosticResult(``diabetes");
    }
  }
  else {
    DiagnosticResult(``diabetes");
  }
}
```

```
if (NN_Estimulation(inputs)) {
  DiagnosticResult(``diabetes");
}
else {
  DiagnosticResult(``healthy");
}
```

where ϕ is the synaptic potential of the neuron defined as:

$$\phi = \sum_{i=1}^{N} \omega_i \psi_i - b \tag{2}$$

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:

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

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 to the standard perceptron learning rule is that the thermal perceptron incorporates the factor T_{fac}. This factor, whose value

is computed as shown in Equation (4), depends on the value of the synaptic potential and on an artificially introduced temperature (T) that is decreased as the learning process advances.

$$T_{fac} = \frac{T}{T_0} \exp\left(-\frac{|\phi|}{T}\right) \tag{4}$$

C-Mantec, as a CNN algorithm, has in addition the advantage of generating online the topology of the network by adding new neurons during the training phase, resulting in faster training times and more compact architectures [14,17,27,30,38,44].

The topology of a C-Mantec created network consists of a single hidden layer of thermal perceptrons that maps the information to an output neuron that uses a majority function. The choice of the output function as a majority gate is motivated by previous experiments in which very good computational capabilities have been observed for the majority function among the set of linearly separable functions [39]. The results so far obtained with the algorithm [38,40,43] show that it generates very compact neural architectures with state-of-the-art generalization capabilities. It has to be noted that the algorithm incorporates a built-in filtering stage that prevent overfitting of noisy examples.

The C-Mantec algorithm has 3 parameters to be set at the time of starting the learning procedure. Several experiments have shown that the algorithm is very robust against changes of the parameter values and thus C-Mantec operates fairly well in a wide range of values. The three parameters of the algorithm to be set are:

- $I_{\max}$: maximum number of iterations allowed for each neuron present in the hidden layer per learning cycle.
- G_{fact}: growing factor that determines when to stop a learning cycle and include a new neuron in the hidden layer.
- Fi_{temp}: determines in which case an input example is considered as noise and removed from the training dataset according to Equation (5):

$$\forall X \in \{X_1, \ldots, X_N\}, \text{delete}(X) | NTL \geq (\mu + Fi_{temp}\sigma) \tag{5}$$

  where N represents the number of input patterns of the dataset, NTL is the number of times that the pattern X has been learned on the current learning cycle, and the pair $\{\mu, \sigma\}$ corresponds to the mean and variance of the normal distribution that represents the number of times that each pattern of the dataset has been learned during the learning cycle.

A summary of the C-Mantec pseudo-code algorithm is described in Table 2. This learning procedure is essentially based on the idea that patterns are learned by those neurons, the thermal perceptrons in the hidden layer of the neural architecture, whose output differs from the target value (wrongly classified the input) and for which its internal temperature is higher than the set value of G_{fact}. In the case in which more than one thermal perceptron in the hidden layer satisfies these conditions at a given iteration, the perceptron that has the highest temperature is the selected candidate to learn the incoming pattern. A new single neuron is added to the network when there is no a thermal perceptron that complies with these conditions and a new learning cycle starts. The learning process ends when there are no more patterns to be learned, as all of them are classified correctly or are outside of the initial dataset because are considered noisy by an internal built-in filter.

2.2.3. *COOJA simulator.* As mentioned before, COOJA sensor network simulator has been used to carry out all the experiments. COOJA is a power profiling tool that enables accurate network-scale energy measurements in a simulated environment. In order to carry out the simulations, we used Contiki [10] which is an open source, highly portable,

TABLE 2. Brief pseudo-code summary of the C-Mantec learning algorithm

C-Mantec learning algorithm

```
Initialize the parameters of the algorithm;

while (exists patterns to be learned) {
  input a random pattern;
  if (pattern target value == network ouput) {
    remove temporarily the pattern from the dataset;
  }
  else {
    the pattern has to be learned by the network;
    select the wrong neuron with highest temperature;
    if (Tfac >= Gfact) {
      the neuron will learn the pattern;
      update its synaptic weights according to the thermal perceptron rule;
    }
    else {
      a new neuron is added to the network;
      this new neuron learns the pattern;
      iteration counters are reset;
      noisy patterns are deleted from the training dataset;
      reset the set of patterns;
    }
  }
}
```

multi-tasking operating system for memory-efficient networked embedded systems and wireless sensor networks. The sensor network reprogramming was simulated thanks to a routing protocol called mesh which is provided by Contiki. This protocol allows the communication of two sensor nodes in a multi-hop way.

A square grid topology with 25 nodes (5×5) has been used to carry out the reprogramming experiments with the three different analyzed models (TP, CM and FF). This is a quite standard configuration and it could also represent very well the environment in which the chosen datasets presented in Section 2.1 could be implemented. Namely, if a sensor network is deployed in a Day Care Center, the nodes attached to the people would be reprogrammed when they are sleeping in their rooms, thus, the nodes would be distributed according to the physical position of the rooms in the floor of a building, which can be usually approximated by a grid.

The simulations for calculating the resources needed in reprogramming the nodes were carried out for three different scenarios: Best, Medium and Worst case results, according to the position of the node where the update process starts and finishes. The study of these scenarios is motivated by the missing packets that are introduced by the physical distance between nodes with the corresponding increase on the energy consumption.

1. Best Case results: Control center starts the reprogramming task in order to modify the code of three nodes located one hop away from it.
2. Medium Case results: Control center starts the reprogramming task so as to update the code of two nodes located three and four hops away from it.
3. Worst Case results: Control center starts the reprogramming task to change the code of the node located the farthest from it (8 hops).

2.2.4. *Parameter setting and measurements methodology.* The initialization of the parameter values used in the simulations, for the different machine learning algorithms, were selected as follows: i) for the feedforward neural network, the number of neurons in the hidden layer were set to 9, 4 and 14 respective for Parkinson, Pima Indians diabetes and Red wine quality datasets. In each case of study, "tansig" was the transfer function used. And ii), for C-Mantec, on each of the three datasets the parameter $I_{\max}$ was set to 100.000. The pair of parameters $\{G_{fact}, FI_{temp}\}$ were set to $\{0.25, 3\}$, $\{0.1, 0.75\}$ and $\{0.15, 1.5\}$ respective for Parkinson, Pima Indians diabetes and Red wine quality datasets.

For each of the analyzed datasets (Parkinson, Pima Indians diabetes and Red wine quality), the three different classifiers (Rule-based, FF and C-Mantec) were trained with all available examples. For the three methodologies used, models of similar performance were chosen in order to obtain similar values for the correct classification of positive and negative instances. Positive and negative classifications were analyzed separately because some of the datasets contain very unbalanced classes. As NN models have a known random component, 50 experiments have been carried out (randomizing 10 times the distribution of patterns inside the dataset and then iterating 5 times for each randomized distribution) per benchmark dataset in order to average and obtain the final performance of the NN models.

A comparison between the different methodologies used for programming the sensors, has been carried out as follows:

1. TP: Rule extraction was done using an inductive algorithm. It allows us to obtain the code (set of rules *if-then*) needed to evaluate each one of the selected problems (Parkinson, Pima Indians Diabetes and Red Wine) for simulating the dynamic reprogramming of a WSN using traditional programming (TP). Table 1 (left) describes an example of the code needed for the case of the Pima Indians diabetes dataset. For each of the problems presented in Section 2.1, the size of the code needed is computed.
2. NN (FF and C-Mantec): For these two methodologies only the values of the synaptic weights need to be sent, everytime the sensor code has to be modified. The number of bytes needed to be transmitted is defined by Equation (6), where the number 8 indicates the fact that double data types are used to represent the synaptic weight values (real numbers) and N represents the number of input signals for the neural network model.

$$Size(b) = 8 * ((N * HiddenNeuronsNumber) + HiddenNeuronsNumber) \tag{6}$$

3. Sensor nodes communicate between themselves only by sending packets of 20 bytes length (payload). Thus, the number of bytes needed to be transmitted in order to update the program codes will determine the exact number of packets to be sent. Table 3 shows the total number of packets that a node has to send to the WSN according to the three different techniques used to program the nodes.
4. The energy consumption has been obtained by using the model approach from [11]. Due to the fact that in our experiments the parameter that most affects the energy consumption is the current draw of the communication in the receiver mode, a shorter version of the model has been used:

$$E = i\ t_i\ V \tag{7}$$

where i is the current draw of the communication, t_i reflects the time of the communication of the node i and V indicates the supplied voltage. As our goal was to measure the energy consumption of the entire network, the above expression was modified in the following way:

$$E = i \frac{\sum_{k=1}^{20} t_k}{20} V \tag{8}$$

where $i = 19.7$ mA (data obtained from the datasheet of the TelosB) and $V = 3$ V.

2.3. **Benchmark results.** We first show in Table 4 the values for the classification performance of the different algorithms measured by the sensitivity or true positive rate (TPR) and the specificity or true negative rate (TNR). Instead of taking an average and displaying a single classification error, TPR and TNR are prefered estimators for the case of unbalanced datasets. The results have been selected from among several parameter tests, trying to obtain similar operating conditions for the three algorithms used. The number of decision rules obtained to implement TP and the number of neurons of the final architectures used (FF & CM) are also shown in the table. Further, the time needed (in minutes) to obtain FF results was $\{21.10, 42.35, 57.01\}$ respective for Parkinson, Pima Indians diabetes and Red wine quality datasets. On the other hand, CM results were obtained in $\{10.04, 25.73, 40.48\}$ respective for each of the benchmark datasets. It is remarkable the difference between FF and CM models although in our case this process is less important because it could be done off-line on a simple PC.

Table 5 shows a comparison of the results obtained when reprogramming a WSN using the three techniques. Time, energy and number of packets have been measured using the COOJA simulator for all the cases considered. The three included tables show the different results for the three analyzed datasets: Top table shows the results for Parkinson disease data, Middle table for the Pima Indians diabetes diagnosis set and the bottom table shows those results corresponding to Red wine quality data. For each one of the problems analyzed, also three different results are reported, corresponding to Best, Medium and Worst communication scenarios according to the position of the sensors to be updated,

TABLE 3. Size (in bytes) and corresponding number of packets of the message needed to be sent to update the code of the sensors everytime an update is done, computed for the three different methods used in this work: TP, FF and CM

	Parkinson disease		Pima Indians Diabetes		Red wine quality	
	Size (bytes)	Packets	Size (bytes)	Packets	Size (bytes)	Packets
TP	7884	395	7270	364	8192	410
FF	1656	83	288	15	1344	68
CM	368	19	144	8	192	10

TABLE 4. Classification results for the 3 problems using TP, a feedforward multilayer perceptron (FF) and C-Mantec (CM)

	Parkinson disease			Pima Indians Diabetes			Red wine		
	% TPR	% TNR	Rules / Neurons	% TPR	% TNR	Rules / Neurons	% TPR	% TNR	Rules / Neurons
TP	99.3	91.7	9	77.2	78	7	48.8	97.3	20
FF	100 ±0	95.58 ±19.73	9	67.2 ±5.84	86.96 ±12.64	4	67.45 ±4.22	98.25 ±0.33	14
CM	95.98 ±0.9	99.75 ±0.66	2 ±0	75.72 ±2.89	77.79 ±2.31	1.46 ±0.49	85.83 ±1.07	79.81 ±0.72	1.07 ±0.26

TABLE 5. Time (ms), energy (mW) and number of packets needed to reprogram the simulated sensor nodes for three different problems analyzed: Parkinson disease (top table), Pima Indians diabetes (medium table) and red wine quality (bottom table) and for the three different technologies used: traditional programming (TP), feedforward neural network (FF) and constructive neural network C-Mantec algorithm. For each dataset and technology best, medium and worst cases are analyzed depending on the position of the nodes (see the text for specific details).

PARKINSON DISEASE MONITORING PROBLEM

	Best Case			Medium Case			Worst Case		
	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets
TP	141038	17606	56537	686156	18188	66935	1769194	21517	155856
FF	38819	3750	12649	129646	3770	14399	323220	4477	30941
CM	8462	891	2713	50233	964	4861	75965	1097	8309

PIMA INDIANS DIABETES PROBLEM

	Best Case			Medium Case			Worst Case		
	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets
TP	137713	16222	53886	597482	16400	60668	1724786	20136	149988
FF	6132	713	2411	28669	738	2705	60708	890	6635
CM	3745	406	1234	13791	410	1535	41258	525	4553

RED WINE PROBLEM

	Best Case			Medium Case			Worst Case		
	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets
TP	179840	18262	50756	686100	18487	67835	1891400	22515	165149
FF	31913	3083	10374	128235	3144	12450	254026	3662	26041
CM	5370	493	1584	14520	499	1874	43347	621	4692

as it was indicated before in Section 2.2.3. In particular, it is important to highlight the big difference that exists between the Best and Worst scenario in some measures such as the Time to transmit the packets to the sensor nodes. Moreover, there are also important positive results obtained using our proposed algorithm C-Mantec in comparison to the ones obtained with Traditional Programming or Feedforward that are considerably worse on each of the datasets and measures considered in this work.

Figure 3 reflects, in general, the benefits of using neural networks (particularly C-Mantec) in comparison with the traditional programming when the sensor networks needs to carry out dynamic reprogramming. The colored bars on the graphic are overlapped and these final results are calculated by averaging the best, medium and worst cases shown in Table 5. The graphic shows that FF neural networks outperform traditional programming between 80% – 95% in all the three analyzed features (energy consumption, the time needed to carry out the reprogramming and number of packets sent). Moreover, C-Mantec improves further the efficiency of the transmission as it outperformed the TP results on approximately 95% at least in all the cases studied.

3. **Case of Study: Fall Detection Problem.**

3.1. **Fall detection problem overview.** In this section we apply our proposal to the problem of fall detection, an important problem in social health, affecting a large percentage of the elderly population [5,36,50]. The usual framework for this problem are

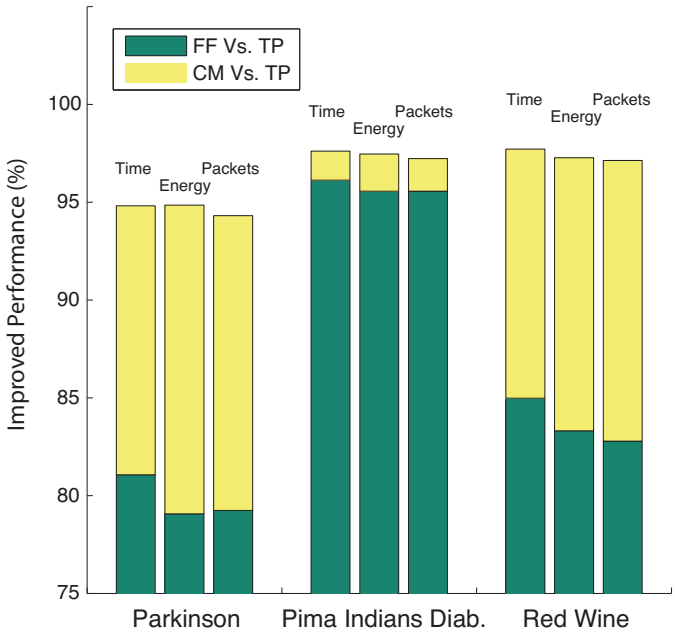

FIGURE 3. Improvement on performance reflecting the benefits of using neural networks. The graphics show the average (between best, medium and worst cases) improvement between traditional programming (TP) and feed-forward neural networks (FF) (top graphs), and between TP and C-Mantec (bottom graphs) for the three analyzed datasets.

elderly people residences where it is needed to detect efficiently when a person living in the residence falls, either in their room or in a communal area. A practical way to detect when a person falls is to use wireless sensor networks. The first step consists of attaching a small sensor node with a 3-axis accelerometer to each person that allows the system to monitor the tilt angle and motion of the carrier person. This configuration detects any anomalous movement in the individual wearing it, such as a sharp downward movement or any sudden or violent movement. These body sensors together with other ones fixed to the walls of the residence will form a wireless sensor network which will permit the data to be relayed from the location where the event occurs to the base station where the information is going to processed.

Detecting a fall from data from the sensor is quite a difficult task, in particular for standard programming algorithms, as a typical "falling pattern" should be extracted from the accelerometer readings (3-axis data) gathered by the sensor, in order to implement the program. However, artificial neural networks offer a simple and versatile way to solve this problem. Specifically, the Multi-LayerPerceptron (FF) and C-Mantec (CM) are trained with real data composed of the accelerometer readings at two instants of times and a variable that indicates whether the relation between the readings correspond to a fall or not. The neural network training process manages to divide the input space into two classes, separating the patterns associated with a fall from those which are not. This kind of learning classification problems is better suited for nonlinear models like MLP, because they can solve a greater number of problems (of different complexity) than linear models. Specifically in this neural model, the number of hidden layers, together with the number of neurons, varies depending on the complexity of the problem posed. The

usefulness of the hidden layers is to project the input patterns into another space in which the classification can be solved linearly.

3.2. **Model evaluation.** In order to generate the neural network that detects falls it is necessary to get a dataset to train the network. To obtain this training dataset in this case study we attached a sensor node to the waist of a voluntary person. The sensor was placed on the waist because for this part typical movements (such as walking and sitting down) do not significantly affect the readings of the accelerometer sensor. On the other hand, when the sensors are attached to the waist falls have relatively simple recognizable patterns.

After attaching the sensor common movements such as sitting down, walking and stopping were reproduced. The sensor periodically gathers accelerometer data each 100 ms. This set of data was identified with the pattern $[x_{ini}\ y_{ini}\ z_{ini}\ x_{end}\ y_{end}\ z_{end}\ 0]$ where x, y and z represent the accelerometer readings for every axis and 0 indicates that the pattern does not correspond to a fall. After that, we simulated falls to obtain patterns related with them. This set of data was represented by patterns with the form $[x_{ini}\ y_{ini}\ z_{ini}\ x_{end}\ y_{end}\ z_{end}\ 1]$.

Once the neural network was trained, some tests were carried out in order to see how well the combination of sensor and neural networks performed to detect falls. The results obtained were satisfactory (see Tables 6 and 7) since almost all simulated falls were detected correctly by the sensor node. If after the deployment of the network the application needs modifications to take into consideration new data or new behavioral patterns, the use of TP would imply reprogramming the whole or a large part of the program in all the nodes attached to each person. Following our approach, however, this task is considerably simplified as only the value of synaptic weights need to be transmitted.

The time needed to carry out the training process using FF was 172.2 minutes while CM results were obtained in 121.5 minutes. Once again, it is remarkable the difference between FF and CM models although in our case this process is less important because it could be done off-line on a simple PC. Table 8 shows a comparison of the results obtained when reprogramming a WSN using the three techniques. Time, energy and number of packets have been measured using the COOJA simulator and three different results are reported, corresponding to Best, Medium and Worst communication scenarios according to the position of the sensors to be updated, as it was indicated before in Section 2.2.3. As it was expected, there is a big difference between the Best and Worst scenario in some measure. Furthermore, there are also important positive results obtained using our proposed algorithm C-Mantec in comparison to the ones obtained with Traditional Programming or Feedforward that are considerably worse on each of measures considered in this work.

TABLE 6. Size (in bytes) and corresponding number of packets of the message needed to be sent to update the code of the sensors everytime an update is done, computed for the practical case of study in this work: TP, FF and CM

	Fall Detection	
	Size (bytes)	Packets
TP	4137	207
FF	840	42
CM	104	6

TABLE 7. Classification results for the practical case study using TP, a feedforward multilayer perceptron (FF) and C-Mantec (CM)

	Fall detection		
	% TPR	% TNR	Rules / Neurons
TP	99.98	99.99	7
FF	98 ±14.14	100 ±0	10
CM	99.85 ±0.06	99.91 ±0.1	1.05 ±0.23

TABLE 8. Time (ms), energy (mW) and number of packets needed to reprogram the simulated sensor nodes for the practical case study analyzed (fall detection) and for the three different technologies used: traditional programming (TP), feedforward neural network (FF) and constructive neural network C-Mantec algorithm. Best, medium and worst cases are analyzed depending on the position of the nodes (see the text for specific details).

FALL DETECTION PROBLEM

	Best Case			Medium Case			Worst Case		
	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets	Time (ms)	Energy (mW)	Packets
TP	84285	8986	29113	351023	10337	35326	954742	12423	84236
FF	17226	2102	7256	74892	2423	8141	164229	2478	17526
CM	2513	268	752	10126	281	1137	23719	365	2678

Figure 4 reflects, in general, the benefits of using neural networks (particularly C-Mantec) in comparison with the traditional programming when the sensor networks needs to carry out dynamic reprogramming. The colored bars on the graphic are overlapped and these final results are calculated by averaging the best, medium and worst cases shown in Table 8. The graphic shows that FF neural networks outperform traditional programming between 77% – 82% in all the three analyzed features (energy consumption, the time needed to carry out the reprogramming and number of packets sent). Moreover, C-Mantec improves further the efficiency of the transmission as it outperformed the TP results on approximately 97% at least in all the features.

4. **Conclusions.** In this paper, we propose the use of neural network based technologies to carry out the dynamic reprogramming of WSN. In order to show the benefits of our proposal we first present a comparative analysis of the resources needed to reprogram the WSN for three test problems related to the use of sensor networks obtained from public benchmark datasets. Energy consumption, time needed and number of packets sent were measured for a simulated sensor network considering best, medium and worst reprogramming cases. Traditional programming based on rules from decision trees, standard FF neural networks and a novel constructive neural network algorithm were used as tools for reprogramming the code of the sensors and for these three technologies the comparative study was carried out. Further, to test the practical application we apply our method to a real problem where sensors are directly involved and needed for the task of fall detection, an important problem in social health. The obtained results for this case of study (Section 3.2) confirmed the benchmark tests.

On overall, the results show that the use of FF neural networks can save much of the resources in reprogramming the nodes, with a further improvement for the case of the

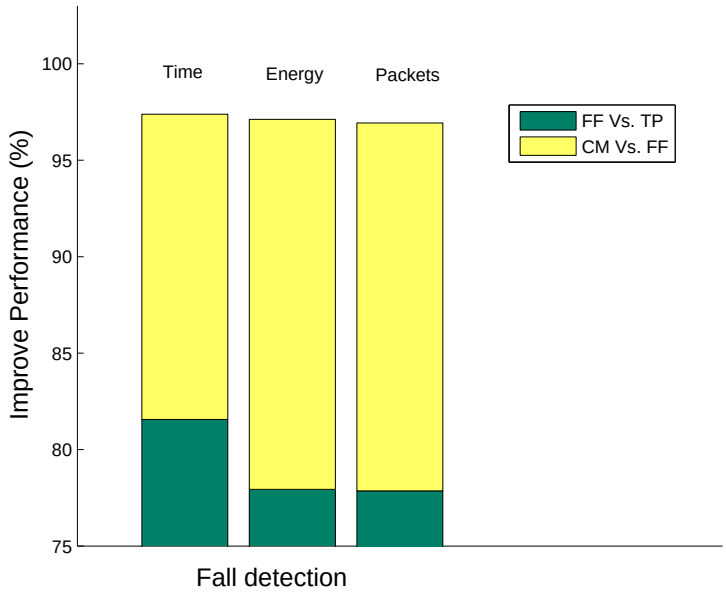

FIGURE 4. Improvement on performance reflecting the benefits of using neural networks. The graphics show the average (between best, medium and worst cases) improvement between traditional programming (TP) and feed-forward neural networks (FF) (top graphs), and between TP and C-Mantec (bottom graphs) for the case study.

C-Mantec algorithm above 95% in almost all considered cases. The large difference in resources observed between TP and neural based technologies (FF & CM) arises mainly from the way traditional programming operates, sending the whole set of conditional sentences. This process is very inefficient most of the time as it is independent to whether the change of the code was large or small. On the other hand, if a neural network based technology can be used the updating process is much more efficient as only the synaptic weight values are sent between nodes, saving much of the resources. Regarding the use of the C-Mantec constructive algorithm, the saving on resources is even larger as this method generates very small neural networks and thus less numbers of synaptic weights are needed. Based on these analyse and considerations, it seems that neural network based technologies, and in particular constructive neural networks like C-Mantec, are very promising tools to be applied in the area of WSNs, given their powerful computational capabilities and flexibility of use. On the other hand, one of the disadvantages of the present approach might be that the use of neural networks involves obtaining training times usually longer than other standard classification method, but it is worth mentioning that this process is done off-line, and thus is not related to the time needed to reprogram the sensors. Eventually, the introduction in the near future of smarter sensors might permit in the case of small modification of the programming code to alter only specific parts of the code allowing TP to be more efficient at the time of reprogramming sensors. Nevertheless, these potential benefits will apply only to minor modifications of the code based on TP.

Acknowledgments. The authors acknowledge support from CICYT (Spain) through grants TIN2008-04985, TIN2008-03107 and TIN2010-16556, and from Junta de Andalucía (Spain) through grants P08-TIC-04026 (including FEDER funds).

REFERENCES

[1] G. Anastasi, M. Conti, M. Di Francesco and A. Passarella, Energy conservation in wireless sensor networks: A survey, *Ad Hoc Networks*, vol.7, pp.537-568, 2009.

[2] G. Anastasi, O. Farruggia, G. L. Re and M. Ortolani, Monitoring high-quality wine production using wireless sensor networks, *Hawaii International Conference on System Sciences*, pp.1-7, 2009.

[3] N. Aslam, W. Philips, W. Robertson and S. S. Kumar, A multi-criterion optimization technique for energy efficient cluster formation in wireless sensor networks, *Information Fusion, Elsevier*, vol.12, pp.202-212, 2011.

[4] J. Barbancho, C. León, F. J. Molina and A. Barbancho, Using artificial intelligence in routing schemes for wireless networks, *Comput. Commun.*, vol.30, pp.2802-2811, 2007.

[5] J. Chen, K. Kwong, D. Chang, J. Luk and R. Bajcsy, Wearable sensors for reliable fall detection, *Annual International Conference of the IEEE Engineering in Medicine and Biology*, vol.7, pp.3551-3554, 2005.

[6] M. Cordina and C. Debono, Increasing wireless sensor network lifetime through the application of som neural networks, *The 3rd International Symposium on Communications, Control, and Signal Processing*, pp.467-471, 2008.

[7] L. Dehni, F. Krief and Y. Bennani, Power control and clustering in wireless sensor networks, *IFIP International Federation for Information Processing*, vol.197, pp.31-40, 2006.

[8] L. D. Toffola, S. Patel, B.-R. Chen, Y. M. Ozsecen, A. Puiatti and P. Bonato, Development of a platform to combine sensor networks and home robots to improve fall detection in the home environment, *Proc. of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp.5331-5334, 2011.

[9] N. D. Tian, A node scheduling scheme for energy conservation in large wireless sensor networks, *Wireless Communications and Mobile Computing*, vol.3, pp.271-290, 2003.

[10] A. Dunkels, B. Grövall and T. Voigt, Contiki – A lightweight and flexible operating system for tiny networked sensors, *Proc. of Conference on Local Computer Networks*, pp.455-462, 2004.

[11] A. Dunkels, F. Osterlind, N. Tsiftes and Z. He, Software-based on-line energy estimation for sensor nodes, *Proc. of the 4th Workshop on Embedded Networked Sensors, EmNets*, pp.28-32, 2007.

[12] N. Enami, R. A. Moghadam, K. Dadashtabar and M. Hoseini, Neural network based energy efficiency in wireless sensor networks: A survey, *International Journal of Computer Science and Engineering Survey*, vol.1, pp.39-55, 2010.

[13] J. Eriksson, F. Osterlind, N. Finne, A. Dunkels, N. Tsiftes and T. Voigt, Accurate network-scale power profiling for sensor network simulators. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol.5432, pp.312-326, 2009.

[14] M. Frean, The upstart algorithm: A method for constructing and training feedforward neural networks, *Neural Comput.*, vol.2, pp.198-209, 1990.

[15] M. Frean, A "thermal" perceptron learning rule, *Neural Comput.*, vol.4, pp.946-957, 1992.

[16] H. Frey, S. Rúhrup and I. Stojmenović, Routing in wireless sensor networks, *Guide to Wireless Sensor Networks*, Chapter 4, pp.81-111, 2009.

[17] N. García-Pedrajas and D. Ortiz-Boyer, A cooperative constructive method for neural networks for pattern recognition, *Pattern Recogn.*, vol.40, pp.80-98, 2007.

[18] J. Gehrke and L. Liu, Guest editors' introduction: Sensor-network applications, *IEEE Internet Computing*, vol.10, pp.16-17, 2006.

[19] A. Hosseingholizadeh and A. Abhari, A neural network approach for wireless sensor network power management, *Proc. of the 2nd International Workshop on Dependable Network Computing and Mobile Systems*, 2009.

[20] J. Ibriq and I. Mahgoub, Cluster-based routing in wireless sensor networks: Issues and challenge, *Proc. of the 2004 Symposium on Performance Evaluation of Computer Telecommunication Systems*, pp.759-766, 2004.

[21] Y. Jin, J. Jo and Y. Kim, Energy-efficient multi-hop communications scheme in clustered sensor networks, *International Journal of Innovative Computing, Information and Control*, vol.4, no.7, pp.1741-1749, 2008.

[22] S. Liang, Y. Tang and Q. Zhu, Passive wake-up scheme for wireless sensor networks, *ICIC Express Letters*, vol.2, vol.2, pp.149-154, 2008.

[23] Y. T. Liao, C.-L. Huang and S.-C. Hsu, Slip and fall event detection using Bayesian belief network, *Pattern Recognition*, vol.45, no.1, pp.24-32, 2011.

[24] X. Liu, K. Hou and H. Shi, Efficient and portable reprogramming method for high resource-constraint wireless sensor nodes, *International Conference on Wireless and Mobile Computing, Networking and Communications*, pp.455-460, 2011.

[25] M. Liu, H. Li, Y. Shen, J. Fan and S. Huang, Elastic neural network method for multi-target tracking task allocation in wireless sensor network, *Computers and Mathematics with Applications*, vol.57, pp.1822-1828, 2009.

[26] K. Mohamed, O. Mirza and J. Kawtharani, Barc: A battery aware reliable clustering algorithm for sensor networks, *Journal of Network and Compute Applications*, vol.32, pp.1183-1193, 2009.

[27] M. C. Nicoletti and J. R. Bertini, An empirical evaluation of constructive neural network algorithms in classification tasks, *Int. J. Innov. Comput. Appl.*, vol.1, pp.2-13, 2007.

[28] Y. Obashi, H. Chen, H. Mineno and T. Mizuno, An energy-aware routing scheme with node relay willingness in wireless sensor networks, *International Journal of Innovative Computing, Information and Control*, vol.3, no.3, pp.565-574, 2007.

[29] F. Oldewurtel and P. Mahonen, Neural wireless sensor networks, *The 2nd International Conference on Systems and Networks Communications*, pp.28-28, 2006.

[30] R. Parekh, J. Yang and V. Honavar, Constructive neural-network learning algorithms for pattern classification, *IEEE Trans. Neural Netw.*, vol.11, pp.436-451, 2000.

[31] I. Park and M. Takeshi, Energy reduction in wireless sensor networks through measurement estimation with second order recurrent neural networks, *The 3rd International Conference on Networking and Services*, pp.103-103, 2007.

[32] R. Rajagopalan and P. Varshney, Data aggregation techniques in sensor networks: A survey, *IEEE Communications Surveys & Tutorials*, vol.8, pp.48-63, 2006.

[33] H. Shahbazi, M. Araghizadeh and M. Dalvi, Minimum power intelligent routing in wireless sensors networks using self organizing neural networks, *International Symposium on Telecommunications*, pp.354-358, 2008.

[34] R. Shaikh, V. Thakare and R. Dharaskar, Efficient code dissemination reprogramming protocol for WSN, *International Journal of Computer and Network Security*, vol.2, pp.116-122, 2010.

[35] Y. Shen and B. Guo, Wavelet neural network approach for dynamic power management in wireless sensor networks, *Proc. of the International Conference on Embedded Software and Systems*, pp.376-381, 2008.

[36] S. Y. Sim, H. S. Jeon, G. S. Chung, S. K. Kim, S. J. Kwon, W. K. Lee and K. S. Park, Fall detection algorithm for the elderly using acceleration sensors on the shoes, *Proc. of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp.4935-4938, 2011.

[37] D. Smith and S. Singh, Approaches to multisensor data fusion in target tracking: A survey, *IEEE Transactions on Knowledge and Data Engineering*, vol.18, pp.1696-1710, 2006.

[38] J. L. Subirats, J. M. Jerez and L. Franco, A new decomposition algorithm for threshold synthesis and generalization of boolean functions, *IEEE Transactions on Circuits and Systems*, vol.1, pp.3188-3196, 2008.

[39] J. L. Subirats, L. Franco, I. Gómez and J. M. Jerez, Computational capabilities of feedforward neural networks the role of the output function, *Proc. of the XII CAEPIA07*, Salamanca, Spain, pp.231-238, 2008.

[40] J. L. Subirats, L. Franco and J. M. Jerez, C-Mantec: A novel constructive neural network algorithm incorporating competition between neurons, *Neural Networks*, vol.26, pp.130-140, 2012.

[41] M. K. Suh, C. A. Chen, J. Woodbridge, M. K. Tu, J. I. Kim, A. Nahapetian, L. Evangelista and M. Sarrafzadeh, A remote patient monitoring system for congestive heart failure, *Journal of Medical Systems*, vol.35, pp.1165-1179, 2011.

[42] A. Tsanas, M. Little, P. E. McSharry and L. Ramig, Accurate telemonitoring of parkinson's disease progression by non-invasive speech tests, *IEEE Transactions on Biomedical Engineering*, vol.57, pp.884-893, 2009.

[43] D. Urda, J. L. Subirats, L. Franco and J. M. Jerez, Constructive neural networks to predict breast cancer outcome by using gene expression profiles, *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol.6096, pp.317-326, 2010.

[44] P. E. Utgoff and D. J. Stracuzzi, Many-layered learning, *Neural Comput.*, vol.14, pp.2497-2529, 2002.

[45] Q. Wang, Y. Zhu and L. Cheng, Reprogramming wireless sensor networks: Challenges and approaches, *IEEE Network*, vol.20, pp.48-55, 2006.

[46] M. Winter and G. Favier, A neural network for data association, *IEEE Int. Conf. Acoustics, Speech and Signal Processing*, vol.2, pp.1041-1044, 1999.

[47] J. Yick, B. Mukherjee and D. Ghosal, Wireless sensor network survey, *Computer Networks*, vol.52, pp.2292-2330, 2008.

[48] S. Yun, Y. Youk and S. Kim, Study on applicability of self-organizing maps to sensor network, *International Symposium on Advanced Intelligent Systems*, Sokcho, Korea, 2007.

[49] Y. Zhang, J. Yang and W. Li, Towards energy-efficient code dissemination in wireless sensor networks, *Proc. of the 22nd IEEE International Parallel and Distributed Processing Symposium*, pp.1-5, 2008.

[50] Z. Zhang, U. Kapoor, M. Narayanan, N. H. Lovell and S. J. Redmond, Design of an unobtrusive wireless sensor network for nighttime falls detection, *Proc. of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp.5275-5278, 2011.

Appendix A. **Glossary of acronyms.** The different acronyms used among the article are

WSN.	Wireless sensor network
TP.	Rule-based programming
NN.	Neural networks
CNN.	Constructive neural networks
FF.	Feedforward neural networks
CM.	C-Mantec algorithm
TDR.	Traditional dynamic reprogramming
NNDR.	Neural network dynamic reprogramming
TPR.	True positive rate
TNR.	True negative rate
OAP.	Old age pensioner