Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 20 (2013) 406 – 413
Procedia Computer Science

Complex Adaptive Systems, Publication 3
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2013- Baltimore, MD

AI-WSN: Adaptive and Intelligent Wireless Sensor Network

Gursel Serpen*, Jiakai Li, and Linqian Liu

Electrical Engineering and Computer Science, University of Toledo, Toledo, Ohio, USA

Abstract

This paper proposes employment of artificial neural network techniques to develop in-network "intelligent computation" and "adaptation" capability for wireless sensor networks to improve their functionality, utility and survival aspects. The goal is to introduce computational intelligence capability for the wireless sensor networks to become adaptive to changes within a variety of operational contexts and to exhibit intelligent behaviour. The characteristics of wireless sensor networks bring many challenges, such as the ultra large number of sensor nodes, dense deployment, changing topology structure, and the most importantly, the limited resources including power, computation, storage, and communication capability. All these require the applications and protocols running on wireless sensor network to be not only energy-efficient, scalable and robust, but also "adapt" to changing environment or context, and application scope and focus among others, and demonstrate intelligent behaviour. Feasibility of the proposed approach is demonstrated through a simulation-based case study which entailed a clustering of Iris data using Kohonen's self-organizing map neural network which was embedded across a wireless sensor network.

Keywords: wireless sensor network; artificial neural network; adaptation, artificial intelligence; distributed system; parallel computation

1. Introduction

Wireless sensor networks (WSN) are an emerging technology due to recent advancements in very small-scale manufacturability and high-scale integration of various electronic components in a single packaging. A typical sensor node (or mote) is a standalone package of electronics necessary to hold a number of sensors, an embedded microcontroller, a power unit that has limited capacity, which may or may not be renewable, and a radio trans-receiver at its core. Typical size of a sensor node is anywhere from a matchbox to a coin, but is expected to shrink dramatically in the next decade with the exciting promise of nanotechnology manufacturing and fabrication.

Current and projected application of wireless sensor networks encompasses a wide variety of domains, which have been traditionally challenging to access due to many reasons including potential harm to humans, being at

* Corresponding author. Tel.: 1-419-530-8158; fax: 1-419-530-8146.
E-mail address: gserpen@eng.utoledo.edu.

remote sites or being distributed over very large areas, and being subject to harsh geo or meteorological circumstances among others. Monitoring environment (pollution in a lake or river), forest fires, volcanoes, battlefield troop movements, human body, monitoring structural health of high-rise buildings or bridges, smart home automation, and the last but not the least, smart renewable energy grid monitoring and control are among the countless potential applications.

Given the nature of applications for wireless sensor networks, typical deployment scenario in many cases entails scattered random placement of hundreds or thousands of such nodes in a given geographic area through either dropping aerially from a flying craft or spreading from a moving land vehicle. Resultantly, the set of sensor nodes form an ad hoc wireless computer network. Often such nodes and the network are expected to operate for a period of at least one to two years using the on-board power source depending on the nature of the application without any outside maintenance or repair access since such access may simply be not feasible or practical.

Wireless sensor networks are conceived to be deployed and expected to operate autonomously for a number of years particularly in non-hospitable environments without human involvement. Various factors including geography, climate, and human-induced intentional or non-intentional interference in the electromagnetic spectrum will adversely affect the deployed network, and hence requiring a good level of adaptability to changing circumstances. A wireless sensor network is a dynamic system in the sense that it goes through changes over time, which have important consequences on the operation and requirements of the network. Some of these changes may include revisions to mission or functionality at different scales; changes in static or dynamic node composition; energy consumption profile of nodes and the network over time; destruction or death of certain nodes; and transient effects that may temporarily hinder a node, a cluster of nodes, or a sub-network to function within its normal operating framework.

There are a set of inter-related optimization processes, i.e. minimum energy, data loss, reliability, robustness, etc., in place during the design and operation of wireless sensor networks. In the typical design and development for wireless sensor networks, a specific set of protocols for medium access, localization and positioning, time synchronization, topology control, security and routing are identified based on the current configuration of the network, the requirements of the application and the topology of their deployment. However, poor performance or unexpected behavior may be experienced for all kinds of reasons following deployment, such as sudden death of sensor nodes, unsatisfactory implementation of application logic, topology changes and mutated network conditions. For instance, it is conceivable that adaptive protocol selection or switching schemes may be developed, which might respond more optimally to changes that affect the wireless sensor network over time. This leads to the necessity of changing the software behavior at both the protocol and application layers after the network has been deployed.

The goal of the project reported in this paper is to introduce ability for "adaptation" and facilities for "computational intelligence" to the wireless sensor networks for significantly-enhanced autonomous behaviour and operation. The expectation from an adaptive and intelligent WSN is that it can readily take into consideration the changes dictated by the dynamic nature of operational and application aspects, and accordingly adapt to changing conditions, circumstances, mission, and operational demands following the deployment. The desirable adaptation capability can be introduced through embedding an artificial neural network, which can be instantiated to any specialized form such as feedforward, self-organizing or recurrent, in fully parallel and distributed mode within the wireless sensor network.

2. Proposed Design

We are proposing embedding artificial neural networks into wireless sensor networks in parallel and distributed computation mode. Wireless sensor networks (WSN) are topologically similar to artificial neural networks. A WSN is constituted from hundreds or thousands of sensor nodes or motes each of which typically has substantial computational power (through the onboard microcontroller). An artificial neural network (ANN) is composed of hundreds or thousands of (computational) nodes or neurons, each of which is assumed to possess or require only very limited computational processing capability. This topological similarity can be the basis to benefit the adaptation and operational aspects of WSNs through leveraging the existing neural network theory in its entirety for all practical purposes. Since a wireless sensor network with thousands of motes or nodes is a distributed system with parallel computing ability, a fusion with another parallel and distributed system, the artificial neural network, is a natural consequence. In fact there is one-to-one correspondence, in that, a sensor mote can act like or implement a neural network neuron or node, while wireless links among the motes are analogous to the weighted connections among neurons. Fusing WSNs with ANNs sets the stage for WSNs suddenly to possess considerably substantial computational intelligence to be able to address a very comprehensive portfolio of problems both at the protocol and application layers. We will refer to the fused system as "WSN-ANN" in the rest of the paper.

The computational power of artificial neural networks is well-documented in the literature and for a comprehensive spectrum of problem domains. Classification, function approximation or regression, optimization, clustering, system identification, and prediction (i.e. time series) are among the leading applications. Artificial neural networks have emerged as a practical technology for function approximation based on a large set of available examplar patterns [1]. Artificial neural networks (ANN) are universal approximators that can be trained on a data set to map multi-dimensional nonlinear functions [2]. In fact, ANNs offer a powerful and general framework for representing non-linear mappings from multi-input to multi-output variables, where a number of adjustable parameters control the form of the mapping [3]. ANNs are particularly attractive since much of univariate approximation methodologies fail to generalize well to higher dimensional spaces: splines and wavelets perform well in regression and signal analysis if the dimensionality of the input space is no more than three [4-6]. There is substantial body of knowledge pertaining to the function approximation capabilities of artificial neural networks. It was theoretically established (along with substantial empirical evidence) that a one-hidden layer feedforward network, whose neuron output functions are sigmoidal (i.e. neuron input is mapped to output through a sigmoid-shaped function), is capable of approximating an arbitrary (continuous) function [1,7-9].

Feedforward neural networks including multilayer perceptron (MLP) and radial basis function (RBF) realizations are well-known for classification and function approximation, while recurrent neural networks like the time-delay neural networks or Elman neural networks are useful for time series forecasting or prediction. For optimization and associative memory applications, Hopfield-style networks including mean-field annealing and Boltzman machine are employed. There are ANN algorithms appropriate for (unsupervised or semi-supervised) clustering applications such as Kohonen's self-organizing map and its many variants such as the family of adaptive resonance theory (ART) neural networks and linear vector quantizer among many others. In conclusion, ANNs are able to address many fundamental problems in computing and as such offer a generic computing tool that is highly versatile, and possesses substantial utility for a comprehensive set of problem domains.

Attributes of the proposed WSN-ANN architecture with respect to scalability and complexity are of fundamental interest. Specifically, these attributes include ability of the WSN-ANN to scale with the problem size, the computational complexity in space and time, and the communications or messaging complexity. It is relevant to note that computational complexity aspects of neural networks is a domain that is largely incomplete and fragmented although there have been noteworthy advances during the last decade [10-15]. There are too many and diverse neural network paradigms and countless parameters to consider for a unified and coherent treatment of the subject, which therefore led to only a limited number of computational complexity analyses for specific instances of neural network algorithms and associated learning or training processes.

The time complexity of the proposed computing system is determined by a number of factors depending on the type of the neural network. There are typically two distinct phases that must be considered: training the neural network, which bears a substantial time cost in some cases and the follow-up deployment whose time cost tends to be negligible compared to that of the training. As an example, for feed-forward multi-layer neural networks, the training time is mainly dictated by the convergence properties of the specific problem being addressed, which also affects the topology of the neural network. The convergence properties of multi-layer feedforward networks vary dramatically from one problem domain to another. The empirically specified convergence criterion, i.e. one being cumulative error satisfying a user-defined upper bound, also plays a significant role in the time complexity. There will also be on-board processing time associated with implementing the neuron dynamics signal processing which is negligible compared to other cost elements, and hence may and will be ignored for the rest of the discussion. All of these costs are already inherent in the neural network algorithm regardless of its hardware realization on a specific platform. There is however a new cost component due to implementation of the neural network algorithm on a wireless sensor network. It is expected that there will be delays injected into the learning or training process due to the need to exchange neuron output values among the motes (neurons) through the wireless medium as managed by an appropriate medium access control (MAC) protocol since medium access collisions will necessarily occur and have to be dealt with. What this means is that the MAC protocol and the messaging requirements of a specific neural network as indicated by its inter-neuron connectivity attributes will play a role in the finalization of the time cost assessment. Therefore, it is reasonable to suggest that the time complexity will vary dramatically based on the problem domain, the network type and architecture, parameter settings and others.

Space complexity associated with the WSN-ANN computation is expected to be minimal. In a wireless sensor network, there is no need to create a global or network-wide weight matrix for the neural network! Instead, each neuron stores its own weight vector locally on the mote it resides. Wireless communication channels serve as connections among a transmitting neuron and receiving neurons, and therefore eliminating the need to create and store a single network-wide weight matrix that is typically huge in size. All that is needed is the local (or distributed) storage of weight vectors. Another vector needs to be created locally (say within the read-write memory of microcontroller on-board a mote) to store output values of neurons which are connected to a given neuron. If the worst-case (or maximum) connectivity is N, which cannot be larger than the total number of neurons in the neural network, then the space or memory cost is $O(2\times N)$ real numbers. This translates into $y\times 2\times N$ bytes under the assumption that each real number requires y (a small positive integer) bytes in some digital representation scheme. Typically, for many neural network algorithms, a neuron connects to a number of other neurons which is much smaller than the total number of neurons in a given network. Accordingly, the memory (or space) cost is not expected to be significant.

The communication complexity arises due to the requirement for wireless communications to exchange neuron output values among the neurons, each of which is embedded within a different mote across the WSN toplogy. Typically a given mote will exchange messages with a small number of other motes (compared to the total number of motes in the WSN) which are possibly k-hop neighbors, where k is a small positive integer, since multi-hop communications is preferable to direct mode (single-hop) for a number of reasons including, but not limited to, energy savings. The total number of messages to be exchanged will depend on a number of factors. For instance if the neural network is a multilayer perceptron with a backpropagation-type learning algorithm, the training mode would require a number of iterations for weight updates (until a convergence criterion is met). During training, first outputs of neurons in a given layer are communicated (through the wireless channel) to the neurons in the next layer. Subsequently outputs of neurons from the next layer need to be communicated (again through wireless channel) to neurons in the previous layer for weight updates. This forward-backward signaling continues until a convergence criterion is satisfied, which is problem and neural network instance dependent among others. In conclusion, it is clear that the communication cost (messaging complexity) is the most limiting constraint on the real-time computation, scalability, and power consumption aspects of WSNs.

The communication requirements among the motes to exchange updated neuron output values for a highly-connected neural algorithm becomes significant at some point as the scale of the problem and the associated neural network increases. This suggests that the real-time computation property is preserved as the scale of the problems increases up to the point when communication bottleneck becomes a determining factor. Therefore, due to parallel and distributed processing ability, the proposed WSN-ANN design offers the potential to deliver real-time computation of solutions for a larger upper bound on the size of the problem than those systems that lack the parallel and distributed processing capability.

Any artificial neural network algorithm can be embedded within a WSN. A non-exhaustive sample of ANN algorithms which can be embedded within a WSN include feed-forward architectures like multi-layer perceptron (MLP) and radial basis function (RBF), self organization algorithms including Kohonen's self-organizing map (SOM), linear vector quantizer (LVQ) and similar, associative memory neural networks including Hopfield associative memory, bi-directional associative memory (BAM), adaptive resonance theory (ART) nets and its many derivatives, and recurrent neural networks including Elman, Jordan, Hopfield, and simultaneous recurrent neural nets (SRN), and time delay neural networks (TDNN). In fact, VLSI implementations of the entire suite of ANN algorithms are already developed in Cichocki et al. [16], which demonstrates feasibility in general and which we will be able to readily leverage to map any type of neural net algorithm to an arbitrary topology of a wireless sensor network.

There are three classes of problems, which the proposed WSN-ANN design addresses. These problems are protocol-level, sensing-based applications including function approximation, estimation, detection, tracking, and localization; and those for generic computations.

At the protocol level, a sample of the problems that can be addressed include clustering or independent set, minimum connected dominating set computation for topology control; maximizing data flow for multiple source-destination pairs in routing [17]; data aggregation as an optimization protocol in data-centric and content-based networking [18-21]; multi-lateration through localized (distributed) solution of the so-called normal equation for the linear least squares problem, which may be addressed by means of QR factorization among many other options; and localized solution of proximity graphs including relative neighborhood graph (RNG), Gabriel graph (GG), Delaunay triangulation (DT), spanning trees (ST), Voronoi diagram, and influence graphs, etc. [22].

At the application level, application support and in-network processing are essential elements of a field-deployed WSN. The list of problems being addressed by the proposed WSN-ANNs cover a large spectrum and a representative sample is discussed here to indicate the potential of employing neural networks for localized or distributed computations of solutions for these problems. In many cases where measured values are correlated in space and time, which is typical for wireless sensor network applications, the optimal tree for collecting all the data transforms into a "traveling salesman" like structure rather than being the shortest-path tree [23]. Hopfield-type neural networks are well known for being able to address such problems which belong to the general class of static (vs. dynamic) combinatorial optimization problems. In the network coding domain, the well-known graph theoretic max-flow/min-cut problem emerges [24], which is another optimization problem for which a number of neural network algorithms have been proposed for computation of optimal solutions. Distributed computation of fast Fourier transform within the context of WSN has been proposed in [25], and this yet is another venue for which neural network algorithms have been proposed and applied. The problem of target detection and tracking involves a number of subtasks including detection, target localization, classification, and tracking. The neural network literature is rich with many neural algorithms devised to address the requirements of all these subtasks associated with target detection and tracking. Boundary or contour (edge) detection problems are also widespread in many WSN applications. Effective neural network algorithms have been proposed to address the distributed or localized computation of solutions for these problems. Another problem that is within the realm of ANN-based function approximation is the field sampling, i.e. estimating a scalar field.

An ANN that is embedded within a WSN which can be seen as a general-purpose computer that also happens to be massively parallel and fully distributed offers vast potential to perform computations of generic nature and high utility. In fact, the WSN-ANN design can solve large-scale problems in real time due to its fully parallel and massively distributed architecture. A non-exhaustive list covers problems from the domains of linear, quadratic, and linear complementarity problems; systems of linear equations; least squares problem; minimax ($L_{infinity}$-norm) solution of over-determined system of (linear) equations, and least absolute deviation (L_1-norm) solution of systems of equations; discrete Fourier transform; matrix algebra problems including inversion, LU decomposition, QR factorization, spectral factorization, SVD, Lyapunov's equation for generalized matrices and PCA adaptive estimation; graph theoretic problems; and static optimization problems. In fact, VLSI realizations (within parallel and distributed computation framework) of many neural network-based algorithms configured for solutions of a very large spectrum of problems are detailed in [16]. It is possible to leverage readily this existing work to implement the mapping of these problems to the WSN-ANN architecture. The set of problems addressed in [16] is very diverse, extensive and includes all those mentioned in this paragraph and more.

3. Simulation Study

The simulation study is intended to show feasibility of the proposed WSN-ANN design. The aim is to simulate a wireless sensor network embedded with a specific neural network configured for a domain problem of interest in a fully-distributed and parallel-computation scenario. The simulation study will demonstrate the application of Kohonen's self-organizing map (SOM) neural network algorithm to a specific problem domain through its parallel and distributed realization on a wireless sensor network. Performance profile of the specific simulation scenario including quality of solutions, computational complexity, messaging cost will be monitored and established. We used the probabilistic wireless network simulator PROWLER for simulations. PROWLER is an event-driven simulator for wireless distributed systems and runs under MATLAB whose current target platform is the Berkeley MICA motes executing the TinyOS embedded operating system [26]. It is available at http://www.isis.vanderbilt.edu/projects/nest/prowler.

The mapping of a two-layer SOM ANN to a wireless sensor network entails embedding one output-layer neuron into a single mote. This is desirable for fully parallel and maximally distributed computation. Input layer neurons in a two-layer SOM network can also be embedded on a one-neuron-per-a-single-mote basis or a single mote can be designated to take over the role of the entire input layer in the SOM. If a single mote is designated to store and supply the entire set of training patterns, then it can also serve as a synchronizer for the phases of overall SOM computation, which includes deployment and initialization, training, and unsupervised clustering. Such a mote will be designated as the "Supervisory Mote" or SM for short. Each (generic) mote (GM) that is responsible for hosting a neuron will need to store the weight vector of the associated SOM neuron, compute the output for the same neuron, and communicate the neuron output value to the supervisory mote, and update the neuron weight vector if it is the so-called best matching unit (BMU) or in the neighborhood of another BMU as determined by the neighborhood function.

The problem domain of interest is the classification of Iris data set [27] which has four attributes as sepal length, sepal width, petal length and petal width. This dataset contains 3 classes of 50 instances each where each class represents a type of Iris plant. One class is linearly separable from the other two, while the other two are not linearly separable from each other. A sample pattern from each class is shown in Table 1.

Table 1. Sample pattern for each of three classes in Iris dataset

Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Setosa
7.0	3.2	4.7	1.4	Versicolor
6.3	3.3	6.0	2.5	Virginica

The Iris dataset has been processed through the MATLAB SOM toolbox using the *som_read_data* and *som_normalize* functions [28]. The SOM topology of 10×10 neurons is used for the MATLAB SOM toolbox solution, and the results after 5 rough training and 5 fine tuning are shown in Figure 1. The quantization and topographic errors, and Huang's accuracy are 0.337, 0.007 and 0.887, respectively.

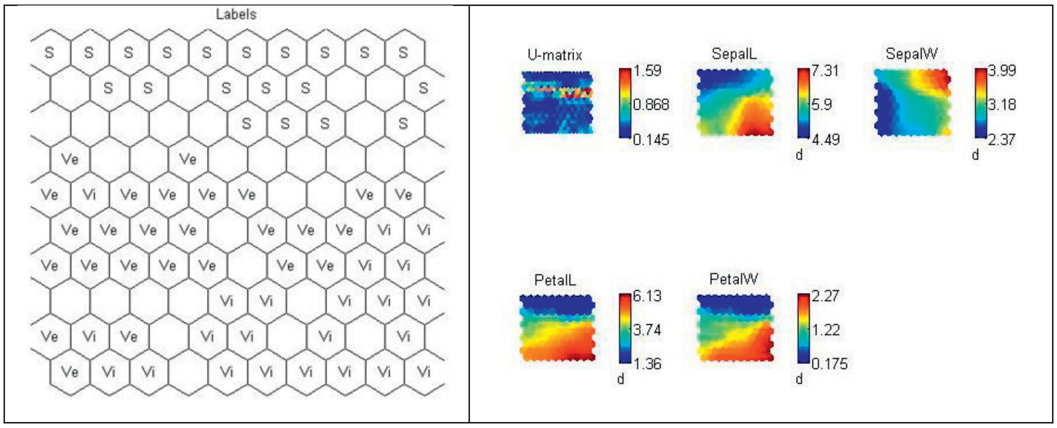

Fig. 1. Component Maps and U-Matrix Using MATLAB SOM Toolbox for Iris Dataset

From the U-Matrix map in Figure 1, it is easy to see the top three rows form a well-defined cluster, which appears to represent the Setosa subspecies. The other two subspecies Versicolor and Virginica are mixed in the other cluster; the U-Matrix map does not show clear separation of them. The Labeled component map in Figure 1 indicates that they correspond to two different parts of the cluster. The component map of petal length and petal width as shown by the last two maps in Figure 1 are closely related to each other. Considering the four component maps with the labeled map in Figure 1 collectively, Setosa has small petals (small petal length seen from component map of petal width and small petal width seen from component map of petal width), and short and wide sepals (seen from component map of sepal length and sepal width). From the maps, no noteworthy difference can be observed as the distinguishing factor when comparing Versicolor with Virginica. In fact, the only distinguishing factor for these two subspecies is that Virginica has bigger leaves.

The MATLAB SOM toolbox uses the complete dataset as the training data and also as the test data. To be able to compare the results with those of the MATLAB SOM Toolbox implementation, no separation of training and test data is implemented for the PROWLER simulation. The WSN deployment topology in PROWLER has 10×10 motes (neurons) plus one supervisory mote, which are randomly positioned using a uniform distribution. Parameter values, errors, and message counts are presented in Table 2. Figure 2 presents the distribution of SOM neurons based on their responses to patterns presented from three classes. Figure (b) is an alternate representation of Figure (a) by removing the rectangles representing the motes, and uses square, five-pointed star and circle for Setosa, Versicolor and Virginica, respectively, and dot for the motes not chosen as the BMU for any data pattern. Two lines are drawn in Figure 2 (b) to separate the three species. The Setosa is linearly separated from the other two species, while for the two species, there is nonlinear separation boundary with most Virginica (circle) being above Versicolor (five-pointed star).

Table 2: Simulation Study Results for Iris Dataset with 5% Message Loss Tolerance

Simulation study parameters	Values
Iterations	10
Number of motes	100 (+1 supervisory mote)
Quantization error	0.690
Topographical error	0.288
Huang's accuracy	0.973
Incoming message count (average per mote)	90,871
Outgoing message count (average per mote)	12,593
PROWLER simulation time	72.03 hours

Comparing the results of WSN-SOM with those through the MATLAB SOM toolbox, for both of them, Setosa is clearly separated from the other two, and there is a defined separation line between Versicolor and Virginica. Smaller quantization error of MATLAB SOM toolbox results indicates that the final weight are more closer to the original data, and the MATLAB SOM toolbox has smaller topographic error which means the topology preservation is better than WSN-SOM results. The WSN-SOM results have larger Huang's accuracy meaning it has better clustering performance than MATLAB SOM toolbox. In conclusion, the evalution of results indicate that the WSN-SOM computation is comparable to that of the MATLAB SOM.

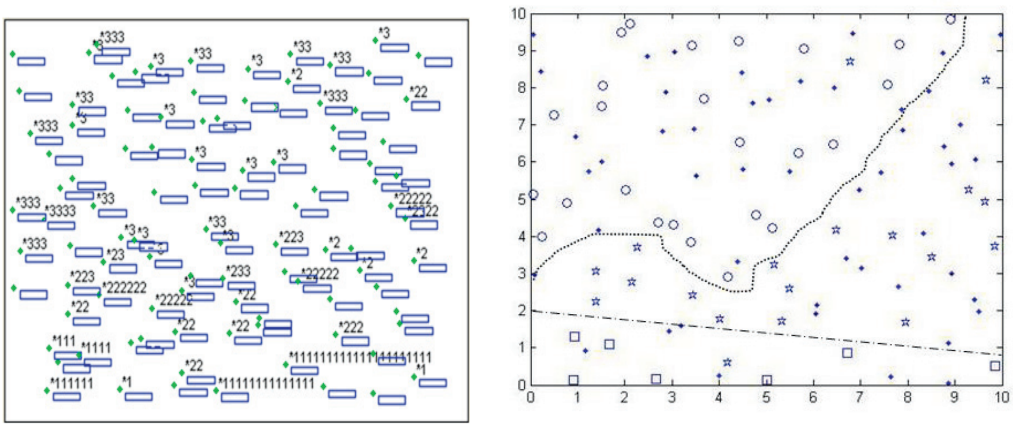

Fig. 2. (a) Simulation results shown in Prowler GUI (1: Setosa, 2: Versicolor, 3: Virginica, label information starts with *, '*233' means the node has been chosen as BMU for 3 times; one time for Versicolor, and two times for Virginica; (b) WSN-SOM results component map (square: Setosa, five-pointed star: Versicolor, circle:Virginica, dot: not chosen as the BMU for any training pattern)

4. Conclusions

This paper proposed embedding an artificial neural network within a wireless sensor network to infuse capability for adaptation and intelligence. A wireless sensor network embedded with a neural network can solve a large class problems to develop capability to adapt to changes in a dynamic environment or possess computational intelligence to address challenges during its operational phase following deployment. A case study using the Kohonen's self-organizing neural network over a wireless sensor network was simulated using the PROWLER platform to identify clusters in the Iris data set. Solutions computed by the proposed design compared favourably with those reported in the literature and those computed through the MATLAB neural network toolbox. Simulation results indicated the feasibility of the proposed design.

References

1. Pinkus, A., Approximation theory of the MLP model in neural networks, Acta Numerica (1999), pp. 143-195.
2. Ferrari, S. and Stengel, R. F., Smooth function approximation using neural networks, IEEE Transactions on Neural Networks, vol. 16 no. 1 (2005), pp. 24 – 38.
3. Bishop, C. M. Neural Networks for Pattern Recognition, Oxford University Press (1995)
4. Lyche, T. Morken, K. and Quak, E. "Theory and Algorithms for Nonuniform Spline Wavelets," Multivariate Approximation and Applications, N. Dyn, D. leviathan, D. Levin, A. Pinkus, Eds., Cambridge University Press, Cambridge, UK, 2001.
5. Cox, M. G. "Practical Spline Approximation," Lecture Notes in Mathematics 965: Topics in Numerical Analysis, P. R. Turner, Ed., Springer-Verlag, New York, NY 1982.
6. Antoniadis, A. and Pham, D. T. "Wavelets and Statistics," Lecture Notes in Statistics 103, Springer Verlag, New York, NY 1995.
7. Chen, T., Chen, H. and Liu, R. W.: Approximation capability in by multilayer feedforward networks and related problems, IEEE Transactions on Neural Networks (1) (1995), 25-30.
8. Maiorov, V. and Pinkus, A. Lower bounds for approximation by MLP neural networks. 81-91 1999 25 Neurocomputing 1-3.
9. Kainen, P. C., Kurkova, V. and Vogt, A. Best approximation by linear combinations of characteristic functions of half-spaces, Journal of Approximation Theory 122: 151-159, 2003.
10. Serpen, G. "Managing spatio-temporal complexity in Hopfield neural network simulations for large-scale static optimization," *Mathematics and Computer in Simulation.* Vol. 64, no.2, pp. 279-293, 2004
11. Orponen, P., Computational complexity of neural networks: A survey. Nordic Journal of Computing 1 (1994), 94--110.
12. Sima, J. and Orponen, P. "General-purpose computation with neural networks: A survey of complexity theoretic results," *Neural Computation*, Vol. 15(12), pp2727-2778, 2003.
13. Sima, J. and Orponen, P., Computational Taxonomy and Survey of Neural Network Models. Neural Computation, 2001.
14. Horne, B. G., Hush, D. R.: Bounds on the complexity of recurrent neural network implementations of finite state machines. Neural Networks, 9, 243--252, 1996.
15. Maas, W., Bounds for the computational power and learning complexity of analog neural nets. In Proc. 25th Annu. ACM Sympos. Theory Comput., pages 335--344. ACM Press, New York, NY, 1993.
16. Cichocki, A. and Unbehauen, R. *Neural Networks for Optimization and Signal Processing*, John Wiley & Sons Ltd. & B. G. Teubner, Stuttgart, 1994.
17. Srinivasan, V. Chiasserini, C. F. Nuggehalli, P. and Rao, R. R. Optimal rate allocation and traffic splits for energy efficient routing in ad hoc networks, in proc. of IEEE Infocom, New York, 2002.
18. Kalpakis, K., Dasgupta, K. and Namjoshi, P. Maximum lifetime data gathering and aggregation in wireless sensor networks, In Proc of the IEEE International Conference on Networking (ICN), pages 685-696, Atlanta, GA, 2002.
19. Kalpakis, K., Dasgupta, K. and Namjoshi, P. Efficient algorithms for maximum lifeteime data gathering and aggregation in wireless sensor networks, Computer Networks, 42: 697, 2003/
20. Dasgupta, K. Kalpakis, K. and Namjoshi, P. An efficient clustering-based heuristic for data gathering and aggregation in sensor networks, in proc. of the IEEE Wireless Communications and Networking Conference (WCNC), New Orleans, LA, March 2003.
21. Dasgupta, K. Kukreja, M. and Kalpakis, K. Topology-aware placement and role assignment for energy-efficient information gathering in sensor network. In proc. of 8th IEEE Symposium on Computers and Communications (ISCC), pp. 341-348, Kemer, Turkey, 2003.
22. Adamatzky, A. When algorithms became local: solving computational geometry problems with neural networks, RNNS/IEEE Symp. on Neuroinformatics and Neurocomputers, vol. 2, pp. 1044-1055, 1992
23. Cristescu, R. and Vetterli, M. Power efficient gathering of correlated data: optimization, NP-completeness and heuristics. In proc. o fthe 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc), Annapolis, MD. 2003.
24. Ahlswede, R., Cai, N. Li, S.-Y. R. and Yeung, R. W. Networks and information flow. IEEE Trans. On Information Theory, 46(4): 1204-1216, 2000.
25. Chiasserini, C.-F. and Rao, R. On the concept sof distributed digital signal processing in wireless sensor networks. In proc. of the IEEE Military Communications Conference (MILCOM), Anaheim, CA, 2002.
26. MEMSIC (formerly Crossbow), "MICA Wireless Measurement System Datasheet". Retrieved from http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA.pdf.
27. "UCI Machine Learning Repository". Internet: http://archive.ics.uci.edu/ml/. April. 19, 2012..
28. E. Alhoniemi, J. Himberg, J. Parhankangas and J. Vesanto, SOM Toolbox, Helsinki University of Technology, 2005.