Available online at www.sciencedirect.com
ScienceDirect
Procedia Computer Science 36 (2014) 192 – 197
Procedia Computer Science

Complex Adaptive Systems, Publication 4
Cihan H. Dagli, Editor in Chief
Conference Organized by Missouri University of Science and Technology
2014- Philadelphia, PA

Complexity Analysis of Multilayer Perceptron Neural Network Embedded into a Wireless Sensor Network

Gursel Serpen* and Zhenning Gao

Electrical Engineering and Computer Science, University of Toledo, Toledo, Ohio 43606, USA

Abstract

This paper presents computational and message complexity analysis for a multi-layer perceptron neural network, which is implemented in fully distributed and parallel form across a wireless sensor network. Wireless sensor networks offer a promising platform for parallel and distributed neurocomputing as well as potentially benefiting from artificial neural networks for enhancing their adaptation abilities and computational intelligence. Multilayer perceptron (MLP) neural networks are generic function approximators and classifiers with countless domain-specific applications as reported in the literature. Accordingly, embedding a multilayer perceptron neural network in a wireless sensor network in parallel and distributed mode offers synergy and is very promising. Accordingly, assessing the computational and communication complexity of such hybrid designs, namely an artificial neural network such as a multilayer perceptron network embedded within a wireless sensor network, of interest. This paper presents bounds and results of empirical study on the time, space and message complexity aspects of a wireless sensor network and multilayer perceptron neural network design.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology

Keywords: Distributed and parallel processing; multilayer perceptron neural network; wireless sensor network; computational complexity; communication complexity

1. Introduction

Wireless sensor networks (WSN) and artificial neural networks (ANN) can be hybridized to benefit each other in innovative ways. WSNs offer a promising option as a parallel and distributed neurocomputing platform for ANN implementations. ANNs are poised to provide utility to facilitate the WSNs to become more adaptive through computational intelligence. Prior studies presented several explorations in this regard [1-2] including using the WSN as a parallel and distributed processing (PDP) neurocomputing platform. Examples included Hopfield recurrent neural networks for static optimization, and Kohonen's self-organizing map neural networks for clustering

* Corresponding author. Tel.: 1-419-530-8158; fax: 1-419-530-8140.
E-mail address: gursel.serpen at utoledo dot edu.

Peer-review under responsibility of scientific committee of Missouri University of Science and Technology
doi:10.1016/j.procs.2014.09.078

or dimensionality reduction. Hopfield neural network as a static optimization tool was employed to compute the minimum connected dominating set on a graph model of the WSN topology for re-adaptation dynamically, while the Kohonen's SOM was utilized to discover clusters in an unsupervised context. Another significant ANN algorithm, the multilayer perceptron (MLP) neural network, which has been employed for function approximation and classification across countless problem domains and therefore offers substantial utility and value. This paper presents a study that assesses the computational and communication complexity of implementing MLP neural networks with backpropagation training across a WSN in parallel and distributed processing mode.

2. WSN-MLP Design

Wireless sensor networks (WSN) are topologically similar to artificial neural networks (ANN). A WSN and an ANN possess structural resemblance, which makes it easy to map an ANN algorithm for parallel and distributed processing to a WSN as a computing platform. In fact there is a one-to-one correspondence, in that, a sensor mote can represent and implement computations associated with a neural network neuron or node, while (typically multi-hop) wireless links among the motes are analogous to the connections among neurons. A WSN mote can be considered as a basic computer with a built-in microcontroller and a radio trans-receiver as a wireless communication device as well as a number of sensors. There is sufficient computational power in each mote to implement the computations associated with neuron dynamics for multiple neurons in real time for most applications. In fact, a single mote can satisfy the computation requirements of tens or hundreds of neurons. Consequently, it is possible to prototype a WSN with thousands of motes where each mote can compute dynamics of one or more neurons in real time.

Embedding an ANN for parallel and distributed processing within a WSN entails the following design considerations. Assume each mote houses one or more neurons. Accordingly, a WSN mote will store the weight vector, past several inputs, parameters for learning or training, and the computational model for one or more neurons and perform the needed computations to update the output of those neurons. In such a scheme, neuron outputs on a given mote will need to be communicated to other neurons distributed across many other motes through wireless communication channels. Message packets are subject to delay and drop during wireless transmission due to medium access (such as channel being busy or collision of packets), outgoing or incoming message processing, multi-hop communications, and routing algorithms among many other factors in a wireless communications medium. Meaningful simulation of WSN computations and communications with an ANN embedded into and across requires that such delay and drop be modeled as accurately as reasonably possible.

Consider a multilayer perceptron (MLP) type artificial neural network with at least three layers of neurons, namely an input layer, one or more hidden layers, and an output layer. The input layer is not considered as a "true" layer since no computation is performed by the neurons in that layer. Neurons in the input layer simply distribute the components of an input pattern vector to neurons in the hidden layer without any other processing. Distribution of training patterns can be accomplished by either a multi-hop routing scheme or by a gateway or clusterhead mote that can reach all the WSN motes directly over the wireless channel.

Outputs of neurons in one layer must be communicated to inputs of neurons in the other layer during training and following the deployment. Since the wireless communication of such packets that carry neuron output values is accomplished through multi-hop routing, it is reasonable to assert that the delay due to medium access, packet processing, and the hopping among others will be mainly affected by the distance (or the equivalently the number of hops) between the sending and receiving neurons or motes. Although it may depend on the actual routing protocol chosen, the distance for the routing path for a packet can be approximated (or underestimated) by the hop count, which is measurable through various estimation schemes [3]. Another aspect plays a significant role in the overall communications and computation process: the likelihood of packet drop carrying a neuron output increases as the number of hops increases between the sender-receiver neuron or the corresponding mote pair. Accordingly, the hop count may be employed as the primary factor affecting the amount of delay and the likelihood of drop for packets carrying neuron outputs. When delay occurs and its value varies and, in the worst case, the packet drop happens, past saved values of the output for the neuron whose most recent output is delayed or dropped are made available to the input of the neuron whose dynamics needs to be updated.

3. Simulation Study

The simulator was custom developed in-house and implemented in C++. It models the "effects" or wireless communication on the messages as delay or drop using statistical models which were derived based on past empirical studies as reported in the literature [4]. It simulates the delay and drop effects on the transmission of neuron outputs, thus provides a highly computationally efficient simulation bypassing the details not relevant for performance assessment associated with application development within a wireless sensor network context.

The simulator implements several phases of data access, initialization, delay and drop instantiation, neural network training, and performance recording. Training of the MLP network entails forward propagation, back propagation, and weights update. After each complete iteration over the entire training dataset, the MLP performance is validated on the testing data. The delay and drop affect transmission of outputs from the hidden layer neurons in the forward propagation step, and the error signals generated at the output layer and communicated back to hidden layer neurons in the backward propagation step.

In a scenario where one or more neurons are embedded into a given mote, the exchange of neuron outputs among motes will be subject to certain delay that is inherent in wireless communications. This delay, which will dictate the duration of a waiting period by a given neuron for its inputs to arrive from other neurons on other motes is not a fixed value but rather a random variable. The delay-induced wait time will be denoted as t_{wait}. Note that t_{wait} is both application dependent and network dependent: its value was found to vary from 10 ms to 3000 ms per the literature survey [4]. For a specific WSN design, the delay between for a pair of motes varies substantially from one pair to another, and even for the same pair of motes the delay variance is significant. Additionally, the maximum delay could be much larger than the mean delay [4]. In simulating a neural network embedded across motes of a wireless sensor network, the t_{wait} may be set according to the mean delay value and the specific network topology to make sure that a good number of inputs successfully arrive for any given neuron to be able to calculate its own output.

In the WSN-MLP case, the parameter t_{wait} is assigned values to vary the delay and drop probabilities. Specifically, the parameter t_{wait} is set as

$$t_{wait} = \vartheta \times \mu \times L_{\max} \tag{1}$$

where ϑ is an empirically determined constant positive real number; μ is the mean of the truncated Gaussian distribution for generating the delay; and L_{max} is the max distance between the node pairs in the WSN topology. In order to exclude the cases where motes are positioned in outlying or extreme areas, the parameter L_{max} is set to a value, which covers approximately 90% of node pairs in the entire WSN topology. The coefficient ϑ is set to different values to cause the neuron output delay to vary. A too small ϑ value results in nearly all the packets to be dropped, while a very large value for this parameter will result in no dropped packets. By experimentation, we determined that (0.3, 2.1) is a reasonable range for ϑ which results in the neuron output delay probability to range from 0.04 to 0.99. Consequently, in this study, we equally divided the range of values for ϑ and therefore set it to 0.3, 0.6, 0.9, 1.2, 1.5, 1.8 and 2.1.

In simulations, we also assumed that there is a gateway mote, which can communicate with each mote in the WSN directly through a single-hop transmission: additionally, potential delays or drop for the communications originating from the gateway mode were not considered. In this study, seven data sets from the UCI Machine Learning repository were employed [5]: these data sets are Iris, Wine, Ionosphere, Dermatology, Numerical, Isolet and Gisette. Simulation on each data set is repeated five times with different initial weight values for the MLP neural network. The only exception is the case of Gisette dataset due to its high computational cost and therefore the simulation was performed only once for this dataset. Additionally, the packets carrying neuron outputs and error signals are subject to delay and drop during the training phase only and not during the testing phase.

4. Complexity Analysis for WSN-MLP Design

This section presents computational and message complexity analysis for the WSN-MLP design where any mote implements a single neuron.

4.1. Space Complexity

Space or memory complexity of an MLP ANN embedded within a WSN is negligible for the most part since neurons are distributed over motes. In one extreme, a single mote can house one neuron and therefore needs to store the data structures including inputs, which are outputs from all the other neurons, parameters and computation model of the neuron, which all collectively require minimal memory space allocation. The most costly data structure is the one-dimensional neuron input array, which will grow linearly in the number of neurons, which send their outputs as inputs to a given neuron. Therefore, the space complexity for one-neuron-per-mote case is linear in the number of neurons connected to the single neuron under consideration. If there are multiple neurons housed by a single mote, then the space complexity is still linear since the space complexity due to a single neuron is multiplied by the number of neurons housed on that single mote to determine the resultant space complexity.

4.2. Time Complexity

Assume that there are $|P_T|$ patterns in the training set and $|P_V|$ patterns in the validation or testing set, respectively. Patterns in the training set P_T are provided to the MLP network (with two layers: one hidden and one output) one at a time. Processing of each pattern is realized in parallel at the level of individual neuron through distributed (and asynchronous) computation. Processing time by individual neurons can be incorporated into the cumulative delay, t_{wait}, which is mainly affected by the delays originating due to medium access control (MAC) and routing protocol related requirements. This delay parameter is a random variable and its value depends on numerous factors. Number of iterations, n_{iter}, needed for convergence to a solution is a random variable also and its value depends on the training algorithm, the initial weight and parameter values, the stopping criterion, the data set characteristics and its presentation order among other factors. Consequently, the time complexity, TC, for a WSN-MLP architecture can be estimated by the following equation:

$$TC = n_{iter} \times (|P_T| + |P_V|) \times \mathrm{E}\{t_{wait}\}, \tag{2}$$

where $\mathrm{E}\{\}$ is the expected value operator. t_{wait} value is set in the simulation according to the Equation 1; and the mean value of truncated Gaussian distribution μ is relocated to be 1 in the simulation. To get the actual value for time, μ needs to be multiplied by the value of per hop delay. From the literature survey [4], the range of per hop delay is 2 ms to 226 ms and the expected value is 65 ms. TC calculation utilized the parameter value set obtained through the simulation study. For instance, the mean value for the iteration count was calculated as the average of all values for a dataset. Similarly, mean value for the simulation duration was calculated by averaging values across all trials. Values for other parameters appearing in Table 1, which present associated data for all datasets, are constant as they were employed in the simulation study.

From Table 1, the iteration count values don't vary significantly. The number of patterns varies, and based on that variation, the hidden layer neuron count also varies across these datasets. The number of patterns varies by an order of magnitude from 150 to 7797: In fact, it is the main factor that affects the value of time complexity TC. The variation in the number of hidden neurons and output neurons will cause a change in the value of the parameter L_{max} which in turn results the value of t_{wait} to increase. In overall, simulation duration (or the time cost/complexity) increases linearly with the increase in pattern count (or neuron count in both hidden and output layers). Empirical findings are in agreement with the theoretical bound in Equation 1.

Table 1. Parameters affecting time complexity for WSN-MLP design

Data set	Iterations	Patterns	Hidden neurons	Output neurons	t_{wait} (ms)	Simulation duration (hrs)
Iris	179	150	4	3	187.2	1.47
Wine	194	175	9	3	234.0	1.73
Ionosphere	126	351	11	2	218.4	2.28
Dermatology	139	358	16	6	312.0	4.07
Numerical	113	2000	53	10	468.0	28.95
Isolet	170	7797	131	26	702.0	261.90
Gisette	106	7000	141	2	577.2	119.88

4.3. Message Complexity

Message complexity is measured by estimating the number of messages sent which carry neuron output values since this aspect causes the major component of the communication cost. More specifically, each original or retransmission of a given message that carries a neuron output value is counted as a basic unit of measurement. During forward propagation cycle of the MLP training phase through backpropagation with momentum, n_{hid} neurons in the hidden layer send their outputs on the average to n_{out} neurons in the output layer. Each wireless communication handover or hop requires retransmission of a given message. The hop distance between hidden neuron i and output neuron j will be denoted by h_{ij}. Therefore, the total number of message transmissions (including retransmissions) for a single training pattern during forward propagation cycle as represented by c_{fp} is given by

$$c_{fp} = \sum_{i=1}^{n_{hid}} \sum_{j=1}^{n_{out}} h_{ij} \ .$$

This cost is incurred for each training pattern in the training and validation sets, namely P_T and P_V. The message complexity, MC, for the entire training episode will depend on the size of the data set as well as problem attributes, which, in conjunction with other factors such as the initial values of weights and learning parameters to name a few, will dictate the number of iterations to convergence. Accordingly, the message complexity for the entire forward propagation phase is estimated by the following equation:

$$MC_{FP} = n_{iter} \times (|P_T| + |P_V|) \times c_{fp}.$$

During the backward propagation phase, n_{out} output-layer neurons transmit their outputs to n_{hid} hidden-layer neurons, which results in the same number of messages as c_{fp}. Assuming that online or incremental learning is implemented, the above cost is incurred for each pattern in the training set for the duration of training, which will continue for a number of iterations to convergence. Hence, the message complexity for the backward propagation phase is estimated by

$$MC_{BP} = n_{iter} \times |P_T| \times c_{fp}.$$

The overall message complexity for the training and validation combined is given by

$$MC = MC_{FP} + MC_{BP} = n_{iter} \times (2|P_T| + |P_V|) \times \sum_{i=1}^{n_{hid}} \sum_{j=1}^{n_{out}} h_{ij} \ .$$

The message complexity measurements for all the simulation experiments are presented in Table 2, which presents the mean value of message complexity variable for each data set. The message complexity depends on

number of iterations, number of training and testing patterns and the sum of hop distances between every node pairs. In Table 2, as the number of neurons (which is calculated as a function of pattern count) in the hidden and output layers increases, the sum of distances (as measured by the number of hops) between any two neurons (motes) also increases as does the number of messages (packets). For a hundred-fold increase in the neuron count, the total hop count increases thousand-fold, and the number of messages increases on the order of thousand-fold as well. The message complexity is worse than linear, and therefore, the message complexity sets the dominant bound for the scalability of the WSN-MLP algorithm compared to the time and space complexities.

Table 2. Parameters affecting message complexity for WSN-MLP design

Data set	Iterations	Training patterns	Testing patterns	Hidden neurons	Output neurons	c_{fp}	Message packets
Iris	179	100	50	4	3	22	985,671
Wine	194	117	58	9	3	52	2,858,248
Ionosphere	126	234	117	11	2	45	3,260,205
Dermatology	139	239	119	16	6	269	23,755,353
Numerical	113	2000	666	53	10	2128	804,279,201
Isolet	170	5198	2599	131	26	20724	2,549,444,060
Gisette	106	4667	2333	141	2	1446	1,381,217,200

5. Conclusions

The analyses of space, time and message complexities for a multilayer perceptron (MLP) neural network distributed over a wireless sensor network (WSN) on a single neuron per mote basis is performed. Space complexity of the proposed WSN-MLP design is minimal due to distributed storage of the required algorithm or the memory space. The analysis of time complexity shows it is mainly affected by the number of patterns in a particular dataset as well as the number of neurons in the hidden and output layers. The message complexity, on the other hand, is primarily affected by the number of hidden neurons and output neurons, and the total hop distances in the WSN. The message complexity increases much faster than the time complexity. Overall, the scalability for the WSN-MLP is promising but bounded above by the communication cost or the message complexity. There are a number of options to manage the communication or message cost, which includes highly efficient routing protocols, increasing the number of neurons embedded in a single mote at the expense of reduction in parallelism, and exploring variants of batch learning (vs. incremental learning).

References

1. Gürsel Serpen, Jiakai Li, Linqian Liu: AI-WSN: Adaptive and Intelligent Wireless Sensor Network. Complex Adaptive Systems 2013: 406-413.
2. Gürsel Serpen, Jiakai Li, Linqian Liu, Zhenning Gao: WSN-ANN: Parallel and distributed neurocomputing with wireless sensor networks.IJCNN 2013: 1-8.
3. Xiao Chen and Benliang Zhang, “Improved DV-Hop Node Localization Algorithm in Wireless Sensor Networks,” International Journal of Distributed Sensor Networks, vol. 2012, Article ID 213980, 7 pages, 2012. doi:10.1155/2012/213980
4. Gao, Zhenning. "Parallel and Distributed Implementation of A Multilayer Perceptron Neural Network on A Wireless Sensor Network." Electronic Thesis or Dissertation. University of Toledo, OhioLINK Electronic Theses and Dissertations Center. 09 Jun 2014.
5. “UCI Machine Learning Repository”. Internet: http://archive.ics.uci.edu/ml/. March. 22, 2014.