Boosted Incremental Nelder-Mead Simplex Algorithm: Distributed Regression in Wireless Sensor Networks

Parisa Jalili Marandi[1], Nasrollah Moghadam Charkari[1]

[1]Parallel Processing Lab, Electrical and Computer Engineering Department, Faculty of Engineering, Tarbiat Modares University, Tehran, Iran

{parisa.jalili, charkari}@modares.ac.ir

Abstract Wireless sensor networks (WSNs) have been of great interest among academia and industry due to their diverse applications in recent years. The main goal of a WSN is data collection. As the amount of the collected data increases, it would be essential to develop some techniques to analyze them. In this paper, we propose an in-network optimization algorithm based on Nelder-Mead simplex to incrementally do regression analysis over distributed data. Then, we improve the resulted regressor by the application of boosting concept from machine learning theory. Simulation results show that the proposed algorithm not only increases accuracy but is also more efficient in terms of communication compared to its gradient based counterparts.

1 Introduction

A wireless sensor network (WSN) comprises a group of sensors. Sensors usually have low power supply, limited computational capacity and memory. The main use of WSNs is for data collection. As the amount of collected data increases, some methods to analyze them are required [1, 2]. Machine learning approaches are good solutions, in this regard. Transmitting all the collected data to a fusion center for centrally analyzing the behavior of data and modeling it leads to a high accuracy in the final result. But, since communication capabilities of sensors are limited, this central approach significantly drains the energy of each node and decreases the life time of the network as a whole. In order to deviate with this problem, in-network approach is adapted which eliminates the need for transmitting data to the fusion center [3]. In-network processing increases local computation to prevent energy wasting through a large amount of communications required in central approach. Based on [3] a learning problem can be converted into one of optimization which is much easier to be dealt with. Accordingly, we aim to propose an incremental optimization algorithm to do regression analysis over distributed data which should also adapt to the limitations of WSNs. Distributed optimization for WSNs based on gradient optimization has been previously studied in [1, 4, and 5]. For the algorithm proposed in [1] a Hamiltonian cycle is set on sensors prior to optimization. Then, the estimate for parameter vector is transmitted from one neighboring sensor to the other and each sensor, using incremental sub-

gradient optimization, adjusts the parameters. The algorithm in [4] shows that clustering the network and setting a Hamiltonian cycle within each cluster not only increases the accuracy of final parameters but also makes the algorithm more robust to failures compared to the algorithm proposed in [1]. While [4] sets a Hamiltonian cycle among nodes of each cluster, [5] sets a Hamiltonian cycle among cluster heads and adapts an approach for each sensor to transmit compressed data to the head of the cluster to which it belongs. This algorithm is much more efficient in terms of accuracy, communication cost, and network latency compared to the previously proposed algorithms. In optimization community, when the form of the objective function is known and it is differentiable, the best decision is to use the first order class of optimization algorithms, where incremental sub-gradient is one of them. However, we have some reasons to apply NM simplex to optimization problem in WSNs which has not been studied previously in the field. In this paper, we first develop an incremental version of NM simplex algorithm for WSNs. According to simulation results, although the accuracy achieved with Incremental NM Simplex algorithm is acceptable, yet it is far from the centralized accuracy and a method is required to improve it. The method applied here is the boosting from the machine learning theory. Boosting was originally developed for binary classification [6]. Later, some versions of it were proposed for multi-class classification [7-9]. Other studies such as [10-13] boosted regressors instead of classifiers. Some parallel and distributed versions of boosting have also been proposed in [14-16]. Our experiments show that boosting really improves the accuracy of the regressor, obtained from Incremental NM Simplex. *Thus the main contributions of this paper are: a) to apply NM simplex rather than gradient based optimization and b) to improve the regression accuracy by boosting, in the context of WSNs*. The rest of this paper is organized as follows. Section 2 provides an overview of supervised learning and its application to WSNs and basics about NM simplex optimization. In section 3, assumptions and problem statement are stated. The motivations to use NM simplex are discussed in section 4. In section 5, we present the proposed algorithm. Experimental results are presented in section 6. Finally, in section 7 we conclude the paper and state some of the future works intended.

2 Preliminaries

In this section required basic knowledge are provided.

2.1 Supervised Learning and its Application to WSNs

According to [17], Supervised, Semi-supervised, and Unsupervised are three types of learning. Supervised learning, being the least intelligent, requires a labeled data set indicated as Eq. (1).

$$GD = \{(x_1, y_1), ..., (x_N, y_N)\} \quad (1)$$

Where $X = \{x_k\}_{k=1}^{N}$ and $Y = \{y_k\}_{k=1}^{N}$ are feature and label sets, respectively. Features describe data and labels indicate the class to which data belongs. The

goal of supervised learning is to learn a function (f) to map X to Y such that $y = f(x)$. There are several algorithms for supervised learning one of which is regression, which fits a model to existing data. For further information about regression refer to [18]. Sensors collect lots of data spatially and temporally. In order to gain benefit of the collected data, there must be some analyzing methods. If we consider these data as a kind of labeled data then supervised learning can easily be applied. Throughout this paper, we will consider a network of sensors distributed in an environment which can measure temperature temporally and localize themselves using an existing efficient localization algorithm such as [19]. Here the labeled dataset includes time and location as features and temperature as the label. Thus supervised learning has to discover the function which given a location in the space and a time epoch can predict the temperature with least possible error.

2.2 Nelder-Mead Simplex Algorithm

NM simplex which was first proposed in 1965 [20] is a local optimization algorithm. There are some works done to free NM simplex from local optima such as [21]. NM simplex employs a regular pattern of points in the search space sequentially to obtain the optimizer. Computationally it is relatively uncomplicated, hence easy to implement and quick to debug [22]. One of the major drawbacks of NM simplex is the lack of convergence proof. Further research, study and experimental results are expected to help understand its behavior. Details of NM simplex algorithm implemented in the experiments of this study are the same as [23] where for termination criterion the approach proposed in [24] is employed.

3 Assumptions and Problem Statement

This section introduces the assumptions and outlines the problem.

3.1 Assumptions

The following assumptions are considered throughout the paper:

1. There are n sensors as $S = \{s_1,...,s_n\}$, each of which has collected m data.
2. i , j indices are used to refer to i^{th} sensor and j^{th} data in an arbitrary sensor, respectively ($x \in \{1,..,n\}$, $j \in \{1,...,m\}$). Thus $(x_{i,j}, y_{i,j})$ indicates j^{th} data from i^{th} sensor.
3. Sensors are distributed in a bi-dimensional area. Coordinates of s_i are indicated by dx_i , dy_i .
4. Three features and one label are chosen for describing data such that $x_{i,j} = \lfloor dx_i, dy_i, time_{i,j} \_$ and $y_{i,j} = temperature_{i,j}$, where $time_{i,j}$ and $temperature_{i,j}$ indicate the time of j^{th} measurement and j^{th} temperature in s_i , respectively.

5. Local dataset of s_i is indicated by LD_i, where $|LD_i| = m$.
6. Global dataset, which is the dataset that could be obtained if transmission over long distances was possible, is denoted by GD, where $GD = \bigcup_{i=1}^{n} LD_i$ and $|GD| = N$, where $N = n \times m$.
7. A Hamiltonian path is set among nodes (a distributed algorithm to set a Hamiltonian cycle is described in [25]). This is the routing scheme used in [1]. We selected it because of its simplicity and ability to clarify the main points of the proposed algorithm. Fig. 1 depicts this path. Here we have set a Hamiltonian path rather than a Hamiltonian cycle over the nodes. As every such a cycle can be converted to a Hamiltonian path by removing one of its edges, so the algorithm in [25] is applicable. The reason to set a path rather than a cycle is stated in section 5.2. We assume that s_1 and s_n are the head and the tail nodes of the Hamiltonian path, respectively.
8. As NM simplex is a heuristic method [22], it builds several simplexes to reach the optimizer. The number of local simplexes formed in s_i, which might be different from one sensor to another and depends on the LD_i, is denoted by c_i.
9. Before learning starts, a query dissemination process distributes to all the sensors in the network the user's desired model to fit data. We have followed [26] in fitting a model to data, which suggests some polynomial models among which we chose 'Linear space and quadratic time' which will be called 'quadratic' in the remaining of the paper.

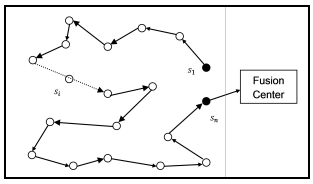


Fig. 1 A Hamiltonian path over the network nodes. Two adjacent nodes on the path can communicate with each other in any direction.

3.2 Statement of the Problem

The goal of the proposed algorithm is to incrementally fit a model to the collected data. Considering the quadratic modeling of the data from section 3.1-Assumption 9, where temperature is to be stated in terms of location and time of measurements, the model is as Eq. (2)

$$F(dx, dy, time) = R[1] \times (dx) + R[2] \times (dy) + R[3] \times (time)^2 + R[4] \times (time) + R[5] \quad (2)$$

, in which R is a vector of unknown constants. Given a set of basis functions as $(1, time, time^2, dx, dy)$ the algorithm aims to estimate their coefficients such that the final model fits data with less possible error (Similar to the approach used in [26]). Based on [1, 3], the learning problem of F can be converted to an

optimization problem to compute R , such that applying least-square error Eq. (3) is minimized:

$$G(R)=\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\left(\begin{array}{l}R[1]\times(dx_i)+R[2]\times(dy_i)+R[3]\times(time_{i,j})^2+R[4]\times(time_{i,j})\\+R[5]-temperature_{i,j}\end{array}\right)^2 \quad (3)$$

That is to say, we would like to determine R in a way that the final model fits all the data with the least possible error. Here, optimization is the same as minimization. So, for quadratic modeling of temperature the problem of learning is converted into a minimization problem with five parameters. As mentioned previously in section 1, it is impossible or at least difficult to centrally compute G , as Eq. (3) is highly dependent on individual data and their transmission to fusion center is energy-consuming. So, it is not feasible to have this formula centrally, but distributed. In fact, there are n sub formulas in the form of Eq. (4):

$$g_i(R)=\frac{1}{m}\sum_{j=1}^{m}\left(\begin{array}{l}R[1]\times(dx_i)+R[2]\times(dy_i)+R[3]\times(time_{i,j})^2+R[4]\times(time_{i,j})\\+R[5]-temperature_{i,j}\end{array}\right)^2 \quad (4)$$

, which when added up give the central formula of Eq. (3). Following this consideration, Eq. (3) is rewritten as Eq. (5):

$$G(R)=\frac{1}{n}\sum_{i=1}^{n}g_i(R) \quad (5)$$

Where $g_i:\Re^L\to\Re$, $G:\Re^L\to\Re$ and L , the length of R , is the number of parameters to be estimated. So, the goal of the proposed algorithm is to do the regression analysis by fitting a pre-specified model to the existing data in a distributed manner and to compute the final parameters as a vector R_G $(|R_G|=L)$.

4 Motivations

The reason to use gradient methods in the previous works was the fact that when the objective function is in hand, having the formula of its first derivative is inevitable. Thus, there is a compelling reason to apply gradient-based optimization. But, examining the previous works revealed some deficiencies that made us to apply another optimization algorithm. Here we have listed the shortcomings encountered:

1. For the incremental sub-gradient method to work, there must be an estimate of Θ , a non-empty, closed, and convex subset of $\Re^L$ in which optimizer is expected to exist [1, 4]. Determining such a subset prior to algorithm execution seems to be a difficult job and the distributed nature of the data makes it even worse.
2. If in any stage of the algorithm execution, the estimate falls out of Θ , a projection must be done to keep the value in the boundary. The experiments showed that the final results highly depend on the projection procedure.
3. In incremental gradient method, at the end of the cycle, parameters suffer from an error. Experiments show that the obtained accuracy for our objective func-

tion is far from the central results. It must be stated that the behavior of optimization methods depends on the objective function and hence, inaccurate results of one method over a special function does not label it as a non efficient method.

4. When the objective function is quadratic, [1] estimates that often one cycle suffices to find the optimizer with a low error. However, their experiments showed that, in one special function, 45 cycles led to the answer, which means large energy consumption.

Based on these, it is desired to propose an algorithm which reduces the final error and frees the user or programmer from specifying Θ as well as the projection procedure. Reduction of communication in the expense of computation increase is another goal followed. NM simplex is selected to fulfill these desires. One of the reasons for applying NM simplex rather than any other optimization method was its popularity among practitioners, despite the absence of any general proof for its convergence. So, further experiments will be helpful to discover the nature of NM simplex. The other main reason to choose NM simplex was its computationally light procedure, which is consistent with the sensors limited computational capacity.

5 Proposed Algorithm

Based on fundamentals of NM simplex described in section 2.2 and the motivations of section 4, in this section we describe the proposed algorithm.

5.1 IS: Incremental NM Simplex

Incremental NM Simplex algorithm, IS henceforth, is illustrated in Fig. 2. Starting from the first sensor on the path, each sensor runs a NM simplex algorithm on the local data and sends the computed parameters to the neighboring sensor. The neighboring sensor uses received parameters as the start point for its local simplex execution. In step II of Fig. 2 the notion of R_{i,c_i} indicates the coefficients of the final local regressor in s_i. As mentioned in section 3.1-Assumption 8, there are c_i locally built simplexes in s_i numbered through 1 to c_i, where c_i th simplex leads to the final local optimizer. At the end of the algorithm, s_n includes $R_G = R_{n,c_n}$ and $N = N_{\{1,\ldots,n\}}$. Let's rename R_G as $R_{G,(IS)}$ to distinguish it from global regressor obtained from BIS in the next section. Also the need to calculate N is explained in the next section.

5.2 BIS: Boosted Incremental NM Simplex Algorithm

Although IS solves deficiencies of incremental gradient, yet the accuracy achieved needs to be improved. The idea here is to apply boosting in order to reduce regression error.

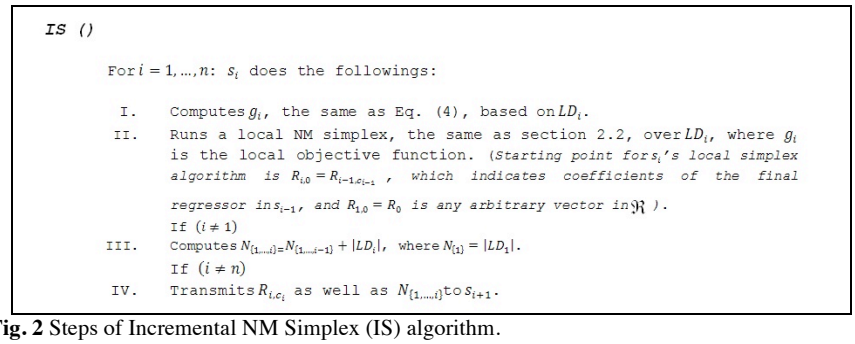


Fig. 2 Steps of Incremental NM Simplex (IS) algorithm.

5.2.1 Motivation to apply Boosting

The main idea of boosting as an ensemble learning method is to train several weak learners serially and to combine them in some way in order to compute a final strong learner [8]. Features mentioned in this sentence, were the motivation for the application of boosting to regression in WSNs. In these networks, sensors collect data independently of each other and don't have access to each others' datasets. Therefore, in case each sensor is to train a regressor individually, that regressor will behave weakly over global dataset which could be obtained centrally if transmission of data over long distances was possible. Serial behavior of boosting on the one hand, and setting a Hamiltonian path among nodes on the other, was the second motivation for proposing this method. Based on [15], there are three types of errors present in a learning algorithm which threaten the accuracy of the learner and can be reduced by the boosting. One is the systematic error of prediction technique which is called bias and the other of these three is variance that is engaged with the sample set. The third error is not of interest here (for more information refer to [15]). Boosting is applied to simplex method as it suffers from both bias and variance. The former is due to the dependence of simplex on the starting point and as the objective function is made up of local samples, the latter type is also present. This is the third compelling reason to apply boosting in this paper.

5.2.2 Boosting Procedure

In boosting, the first regressor is trained over the entire dataset where all data are equally important [12]. Then this regressor is evaluated over the entire dataset and weights are assigned to data so that correctly learned data get smaller weights while higher weights are assigned to wrongly learned data (If the weight assigned to a correctly learned data is zero, it means that these data are simply eliminated from the dataset). Then a second regressor is trained over the new weighted dataset which mostly concentrates over data with higher weights. This procedure continues until a desired level of accuracy is achieved. Furthermore, for each weak learner a weight is assigned which expresses its ability in global regression. Finally, all the regressors are combined to compute the global regressor which is much more accurate than any of the individual models [8]. Now, if boosting is to be applied to regression in WSNs, this procedure must be simulated in some way.

5.2.3 Simulation of Boosting Procedure in WSNs

Before simulating the boosting procedure in a WSN, an important point must be highlighted: the difference between the first weak learner and the others. For the first learner, all data are equally important, whereas, the others pay more attention to some partitions of the entire dataset. Since as mentioned in section 5.2.2, for a new weak learner, the dataset is reweighed according to the previous learners' behavior such that the weights of correctly learned data are decreased while the weights of the remaining are increased. Assuming a weight of zero is assigned to correctly learned data, this can be viewed as for the new learner the correctly learned data with the previous regressors is omitted. Therefore, instead of the entire dataset, the learners other than the first consider only one partition of dataset.

5.2.3.1 Simulation of the First Regressor

For the first step of boosting procedure to be simulated in a WSN, all the data must be present in a single node; nevertheless it is not possible because of the restricted power supply of small sensors. The simulation of this step is the IS from section 5.1, which aims at incrementally obtaining a global regressor. Although the result of this simulation is not as accurate as that of the central approach in the first step of boosting, yet it is an appropriate estimation. Now in order to arrive at the weighted dataset needed in the next step, $R_{G,(IS)}$ must be evaluated over the entire dataset, a weight must be assigned to it and the correctly learned data must be eliminated from the sensors (assuming a weight of zero is assigned to correctly learned data). To accomplish these goals a second pass over the nodes is started in which each sensor's dataset is shrunk to exclude correctly learned data with $R_{G,(IS)}$. Each sensor also calculates a partial weight for $R_{G,(IS)}$ by evaluating it over the local dataset and then by giving the partial weight to its neighbor on the path contributes at computing a global weight for $R_{G,(IS)}$, the sum of all the partial weights.

5.2.3.2 Simulation of Next Regressors

If in a WSN partitions mentioned at the beginning of this section could be obtained, the rest of the boosting procedure would also be simulated. If we assume that each sensor monitors its own range of region, then the local data of one will be distinct from that of the others. Based on this, we can further assume that each local dataset (which is now shrunk due to evaluation of $R_{G,(IS)}$) is the same as a partition in the boosting procedure and thus a local regressor obtained over a shrunk dataset matches a weak learner. Let's refer to this as isolated learning in each sensor. But in reality there might be some common parts among different sensors observation areas in such a way that a larger part of a region is monitored by s_i and a smaller part by s_{i+a} $\left(a \mid a+i \in \{1,...,n\}\right)$, thus in order to train a more perfect local weak learner in s_i, it is desirable to transfer the data collected from smaller part of the region by s_{i+a} to s_i. Similarity in observations is usually among adjacent sensors, leading to data exchange over short distances. However, as the transmission of data is costly even over the short distances, it was decided

to transfer the regressors forward instead of transmitting data backward, which was somehow similar to the approach used in [14]. This means that a regressor is computed in s_i and then transmitted to s_{i+a}. Obviously if s_{i+a} includes any similar data to that of s_i, the regressor will fit them correctly. If such data exists, s_{i+a} simply excludes these data from its local dataset, avoiding unnecessary training of another regressor over repeated data. Such a procedure was tested against isolated learning in each sensor. The increased accuracy of the former was negligible in contrast to drained energy, so it was decided not to do this transfer and simply to run a local regressor in individual nodes. Thus to simulate next steps of the boosting procedure:

1. A second pass over the network is started from s_n back to s_1. (This is the same started in the simulation of first regressor in subsection 5.2.3.1).
2. $R_{G,(IS)}$ is evaluated over s_i's local dataset (LD_i) and each local dataset is shrunk to exclude correctly learned data leading to $Shrunk(LD_i)$.
3. A local NM simplex is run over the shrunk dataset and a weak regressor is obtained (R_{i,c_i}).
4. Each weak regressor is evaluated and is assigned a weight in relation to global dataset, whose size is calculated during the first pass over the network in IS.
5. Each sensor also takes part at computing $R_{\{n,\ldots,i\}}$, the partial combination of weighted local regressors.

BIS ()

1. $R_{G,(IS)} = IS()$ *(from section 5.1,figure 2)*
2. For $i = n,\ldots,1$: s_i does the followings:
   - I. Computes $W(R_{G,(IS)})_i = RWP(R_{G,(IS)}, LD_i, N)$. *($s_n$ knows N ,global dataset size, as IS() is first executed over the network.)*
   - II. Shrinks LD_i to include those data wrongly learned by $R_{G,(IS)}$ and refers to new local dataset as $Shrunk(LD_i)$.
   - III. Runs a local NM simplex over $Shrunk(LD_i)$ and computes R_{i,c_i}. *(Starting point for s_i's local simplex algorithm might be any arbitrary point independent of other sensors, and R_{i,c_i} is the regressor obtained after c_i execution of local NM simplex in s_i over $Shrunk(LD_i)$).*
   - IV. *Computes* $W(R_{i,c_i}) = RWP(R_{i,c_i}, Shrunk(LD_i), N)$.
   - *If* $(i \neq n)$
   - V. Computes $W(R_{G,(IS)})_{\{n,\ldots,i\}} = W(R_{G,(IS)})_{\{n,\ldots,i+1\}} + W(R_{G,(IS)})_i, W(R_{G,(IS)})_{\{n\}} = W(R_{G,(IS)})_n$.
   - VI. Computes $R_{\{n,\ldots,i\}} = R_{\{n,\ldots,i+1\}} + R_{i,c_i} \times W(R_{i,c_i})$, $R_{\{n\}} = R_{n,c_n} \times W(R_{n,c_n})$.
   - If $(i \neq 1)$
   - VII. Transmits $R_{G,(IS)}$, $W(R_{G,(IS)})_{\{n,\ldots,i\}}$, $R_{\{n,\ldots,i\}}$, N to s_{i-1} as the Hamiltonian path is being traversed in the reverse order.
   - If $(i = 1)$
   - VIII. Computes $R_{G,(BIS)} = R_{G,(IS)} \times W(R_{G,(IS)})_{\{n,\ldots,1\}} + R_{\{n,\ldots,1\}}$.

Fig. 3 Steps of Boosted Incremental NM Simplex (BIS) algorithm.

Finally s_1 computes $R_{G,(BIS)}$, the final strong learner in the boosting procedure, as the sum of weighted local regressors and weighted $R_{G,(IS)}$ and transmits it to the fusion center. Boosted Incremental NM Simplex algorithm, BIS henceforth, has two steps and is illustrated more formally in Fig. 3. As IS terminates in s_n ,

second step of BIS starts from it, thus avoiding an extra direct communication from s_n to s_1 for transmitting two values of $N, R_{G,(IS)}$. Even though this seems to be a minor saving in energy consumption, it is valuable in WSNs context. This is why a Hamiltonian path is set over the nodes rather than a Hamiltonian cycle. Accuracy of each regressor is reflected in its weight, which is the fraction of correctly learned data in relation to the global dataset. We have used a pre-specified threshold to decide if the data is learned correctly. This weighing procedure is suggested in [10] and is illustrated in Fig. 4. So the reason to calculate N in IS from section 5.1 is for computing weights of regressors in the second step of BIS. Also calls to RWP () in Fig. 3, refer to the weighting procedure of Fig. 4.

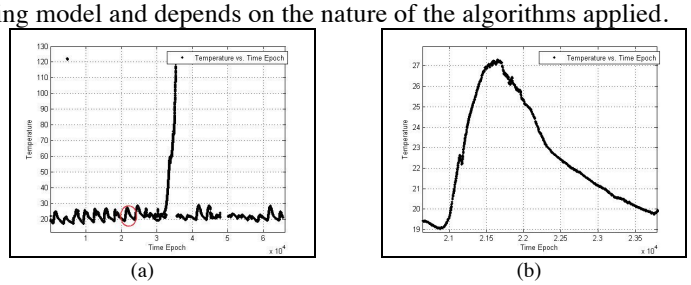
RWP (R, D, DatasetSize)

- If the number of correctly learned data with R over D is C, then: $W(R) = \frac{C}{DatasetSize}$

Fig. 4 Regressor Weighting Procedure

6 Experimental Results

We used the publicly available Intel Lab dataset which contains data collected from 54 sensors deployed in the Intel Berkeley Research Lab. Mica2Dot sensors with weather boards has collected time stamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds [27]. Fig. 5 depicts relation between temperature and time epochs for an arbitrary sensor. All the sensors in the network show the same behavior. It is evident from the figure that except some noisy measurements, a polynomial model, repeated over time intervals, relates temperature to time epochs. Here we evaluate algorithms over such a randomly selected interval. $n = 48$ sensors which contained uniformly distributed measurements over the interval were selected. For each sensor $m = 20$ data were selected. Obviously a single sensor's measured temperatures are constantly related to its location. But for multiple sensors distributed over an area, temperature varies with changes in location. We have decided to consider a linear model for location. Thus the intended model is comprised of some basis functions as $(1, time, time^2, dx, dy)$ which is also shown in Eq. 2 in section 3. 2. Additional basis functions such as $time^3, time^4 \ldots$ might improve the regression accuracy. But the important is the relative accuracy of different algorithms, which is independent of the fitting model and depends on the nature of the algorithms applied.

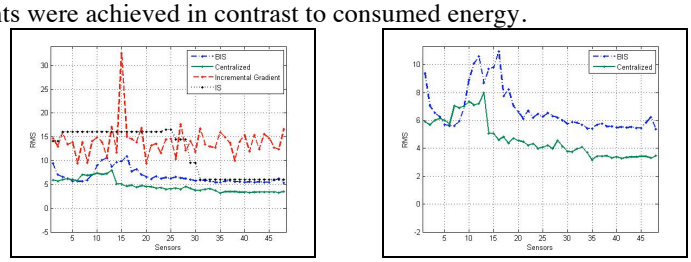


(a) (b)

Fig. 5 (a) Temperature variation over time for a randomly selected sensor. (b) Temperature over a randomly selected time interval which is also shown by an oval in (a).

6.1 Regression Accuracy

Fig.6 (a) depicts Root Mean Square error (RMS) of regressors obtained from IS, BIS, Incremental Gradient (IG), and Centralized approach. Results shown for IG are for one pass over the network. As it was repeated for more passes, minor improvements were achieved in contrast to consumed energy.



(a) (b)

Fig. 6 (a) RMS of the final regressor for IG, Centralized approach, IS and BIS. As it is evident from the curves BIS has the least RMS compared to its distributed counterparts. (b) RMS of the BIS and the Centralized approach. For the Centralized curve, RMS in s_i is for the regressor trained over $\bigcup_{k=1}^{i} LD_k$.

A better accuracy was achieved for 36 more passes over the network, and improved very little after that, which was yet far from that of others. As it is evident from the figure, BIS is superior to its distributed counterparts. Fig. 6 (b) depicts the accuracy of BIS and Centralized algorithm. As it is expected, in both methods, except in some sensors, the overall RMS is decreasing as parameters reach the last sensor, which means that dataset is growing and more data is included. Although RMS of the BIS is not as good as that of the Centralized approach, yet it is better than any other distributed algorithm. The curve of IS in Fig. 6 (a) is also more stable in contrast to that of IG.

6.2 Computation Requirements for IS and BIS

At the first glance, it seems that local computation for the proposed algorithm is much more than that of the IG, but experimental results show that in average, the number of local simplexes formed is low. For IS, the average number of local simplexes formed was 3. For BIS, some additional computations were required in the second step, which had an average of 15 local simplexes produced. Other computations include simple addition and multiplication which are compatible with sensors limited computational capacity. So, altogether computation burden of the proposed algorithms is affordable for sensors.

6.3 Communication Requirements

There are two parameters transmitted among the nodes in IS:

1. Partial global dataset size, which is an integer denoted by $N_{\{1,\ldots,i\}}$.
2. Coefficients of a locally obtained regressor which is a vector of size L .

And four parameters in the second step of BIS:

1. Coefficients of the regressor obtained from IS, which is a vector of size L
2. Partial weight of $R_{G,(IS)}$, which is a double denoted by $W\left(R_{G,(IS)}\right)_i$.
3. Global dataset size which is an integer denoted by N.
4. Partial weighted combination of local regressors which is a double denoted by $R_{\{n,\ldots,i\}}$.

Hence $L+1$ and $L+3$ parameters in IS and in the second step of BIS are transmitted between two adjacent nodes respectively. In IG there is one parameter transmitted: Local regressor which is a vector of size L. Let's denote the number of passes over the network for IG algorithm by P. In the Central approach there are N vectors transmitted from sensors to the fusion center each of which has a size of 3 for assumed labeled dataset as stated in section 2.1. Following [28] and considering the case where n nodes are uniformly distributed in a unit square, the average distance between two successive nodes over a Hamiltonian path is: $O\left(\sqrt{\log^2 n/n}\right)$.Whereas in the Centralized approach the average distance between a sensor and the fusion center is 1 over the unit square. Based on these considerations, Table. 1 shows communication order of four algorithms. Upon termination of BIS, IS, and IG there is a transmission of the final regressor from the last node on the path to the fusion center. As this is common in all three algorithms, it is eliminated from Table. 1. If $P>1$ and $L>1$ then $P\times L \geq (L+2) > (L+1)$, thus BIS and IS are more efficient in terms of communication than IG and as usually $L << m$, BIS, IS, and IG are much more efficient than Centralized approach. Compression of transmitted data and other similar strategies can decrease energy consumption even more. Thus in BIS, with two passes over the network a good estimation of regerssor parameters are obtained, which is more accurate than parameters obtained from several passes of IG over the network. Thus a good balance point for the tradeoff between energy consumption and regression accuracy is achieved.

Table 1. Communication order of Centralized approach, (IG), IS, BIS.

Algorithm	Communication Cost
Centralized	$O(3mn) \cong O(mn)$
IG	$O\left(P\times L\times\sqrt{\log^2 n/n}\right)$
IS	$O\left((L+1)\times\sqrt{\log^2 n/n}\right)$
BIS	$O\left(2(L+2)\times\sqrt{\log^2 n/n}\right) \cong O\left((L+2)\times\sqrt{\log^2 n/n}\right)$

7 Conclusions

In this paper we proposed an in-network optimization technique for distributed regression in WSNs. To overcome deficiencies of incremental gradient optimization, NM simplex was applied and an incremental version of it (IS) was developed. Although, the accuracy of IS was higher than that of the incremental gradient, yet improvements were needed. Hence boosting was applied, and the global accuracy did really improve. Experiments also illustrated the actual effect of boosting in improving the accuracy. Efficiency of the proposed algorithm was also analyzed from the point of computation and communication. The conclusion is that, the proposed BIS algorithm is more efficient in terms of accuracy, communication cost, and local computations compared to its gradient based predecessors. Although the accuracy of BIS is closer to that of the central approach, further improvements are required. We have used the least-square error for converting regression to optimization; other error functions which are more robust to noise might be applied. Other optimization algorithms rather than NM simplex should be considered, as well. Examining the evolutionary algorithms and comparing their performance with that of this paper is put for a later time.

Acknowledgements. The authors wish to thank Iran Telecommunication Research Center (ITRC) for partial funding for this research and the PhD candidate Muharram Mansoorizadeh for his constant help during the research. And also thanks to Intel Berkeley Lab for collected data.

References

1. Rabbat, M., Nowak, R.: Distributed optimization in sensor networks. In International Symposium on information processing in sensor networks, ACM Press, Berkley California USA, (2004)
2. Wang, B., He, Z.: Distributed Optimization Over Wireless Sensor Networks using Swarm Intelligence. In: IEEE International Symposium on Circuits and Systems, pp. 2502 - 2505, (2005)
3. Predd, J. B., Kulkarni, S. R., Poor, H. V.: Distributed Learning in Wireless Sensor Networks. J. Signal Processing, vol. 23, pp. 56-69. (2006)
4. Son, S. H., Chiang, M., Kulkarni, S. R., Schwartz, S. C.: The value of clustering in distributed estimation for sensor networks. In: Proceedings of International Conference on Wireless Networks, Communications and Mobile Computing, pp. 969–974, vol. 2, IEEE, Maui, Hawaii, (2005)
5. Charkari, N. M., Marandi, P. J.: Distributed Regression based on gradient optimization in Wireless sensor networks. In: proceedings of first Iranian Data Mining Conference, Tehran, Iran, (2007)
6. Schapire, R.: The strength of weak learnability. J. Machine Learning, vol. 5, pp. 197-227. (1990)
7. Freund, Y., Schapire, R.: A decision Theoretic generalization of on-line learning and an application to boosting. J. Computer and System Sciences, vol. 55, pp.119-139. (1995)
8. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proceedings of 13th International Conference on Machine Learning, Morgan Kaufmann Press, pp. 148-156. (1996)
9. Li, L.: Multiclass boosting with repartitioning. In: Proceedings of the 23rd international conference on Machine learning, ACM Press, vol. 148, pp. 569 - 576. (2006)
10. Solomatine, D. P., Shrestha, D. L.: AdaBoost.RT: a Boosting Algorithm for Regression Problems. In: IEEE International Joint Conference on Neural Networks, IEEE Press, vol. 2, pp. 1163 - 1168. (2004)

11. Wang, L., Zhu, X.: A Modified Boosting Based Neural Network Ensemble Method for Regression and Forecasting. In: 2nd IEEE Conference on Industrial Electronics and Applications: IEEE Press, pp. 1280-1285 (2007)
12. Avnimelech, R., Intrator, N.: Boosting regression estimators. J. Neural Computation, vol. 11, pp. 491--513. (1999)
13. Drucker, H.: Improving Regressors using Boosting Techniques. In: Proceedings of the 14th International Conference on Machine Learning, pp. 107--115. (1997)
14. Lazarevic, A., Obradovic, Z.: Boosting Algorithms for Parallel and Distributed Learning. In: Distributed and Parallel Databases, Kluwer Academic Press, vol. 11, pp. 203–229. (2002)
15. Yu, C., Skillicorn, D. B.: Parallelizing Boosting and Bagging. Technical Report, Queen's University, Kingston, Ontario, Canada K7L 3N6 February (2001)
16. Lozano, F., Rangel, P.: Algorithms for parallel boosting. In: Proceedings. Fourth International Conference on Machine Learning and Applications, (2005)
17. Chapelle, O., Scholkopf, B., Zien, A.: Semi Supervised Learning. MIT Press, (2006)
18. Draper, N. R., Smith, H.: Applied Regression Analysis. Wiley Press, (1998)
19. Langendoen, K., Reijers, N.: Distributed localization in wireless sensor networks: a quantitative comparison. In: J. Computer and Telecommunications Networking, vol. 43, pp. 499-518. (2003)
20. Nelder, J. A., Mead, R.: A simplex method for function minimization. J. *Computer* vol. 7, pp. 308-313. (1965)
21. Pedroso, J. P.: Simple Metaheuristics Using the Simplex Algorithm for Non-linear Programming, vol.4638, pp. 217-221. Springer Berlin, (2007)
22. Reklaitis, G. V., Ravindran, A., Ragsdell, K. M.: Engineering Optimization: Methods and Applications. John Willey Press, (1983)
23. Lagarias, J. C., Reeds, J. A., Wright, M. H., Wright, P. E.: Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM J. Optima,* vol. 9, pp. 112-147. (1998)
24. Padmanabhan, V., Rhinehart, R. R.: A Novel Termination Criterion for Optimization. In: Proceedings of the Americal Control Conference, vol. 4, pp. 2281 - 2286. Portland, OR, USA, (2005)
25. Petit, J.: Hamiltonian cycles in faulty random geometric networks. In: proceedings of International Workshop on Approximation and Randomization Algorithms in Communication Networks, BRICS Aarhus, Denmark., (2001)
26. Guestrin, C., Bodi, P., Thibau, R., Paskin, M., Madde, S.: Distributed regression: An efficient framework for modeling sensor network data. In: proceedings of third international symposium on Information processing in sensor networks, ACM Press, pp. 1–10. Berkeley, California, USA, (2004)
27. "http://berkeley.intel-research.net/labdata/."
28. Rabbat, M., Nowak, R.: Quantized Incremental Algorithms for Distributed Optimization. IEEE J.Sel Areas Commun, Vol.23, no. 4, pp. 798-808. (2005)