On Dynamic Data-Driven Selection of Sensor Streams

Charu C. Aggarwal
IBM T. J. Watson Research Center
Hawthorne, NY, USA
charu@us.ibm.com

Yan Xie
University of Illinois at Chicago
Chicago, IL, USA
yxie8@uic.edu

Philip S. Yu
University of Illinois at Chicago
Chicago, IL, USA
psyu@cs.uic.edu

ABSTRACT

Sensor nodes have limited local storage, computational power, and battery life, as a result of which it is desirable to minimize the storage, processing and communication from these nodes during data collection. The problem is further magnified by the large volumes of data collected. In real applications, sensor streams are often highly correlated with one another or may have other kinds of functional dependencies. For example, a group of sound sensors in a given geographical proximity may pick almost the same set of signals. Clearly, since there are considerable functional dependencies between different sensors, there are huge *redundancies* in the data collected by sensors. These redundancies may also change as the data evolve over time. In this paper, we discuss real time algorithms for reducing the volume of the data collected in sensor networks. The broad idea is to determine the functional dependencies between sensor streams efficiently in real time, and actively collect the data only from a minimal set of sensors. The remaining sensors collect the data passively at low sampling rates in order to detect any changing trends in the underlying data. We present real time algorithms in order to minimize the power consumption in reducing the data collected and show that the resulting data retains almost the same amount of information at a much lower cost.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications

General Terms

Algorithms

1. INTRODUCTION

Recent advances in hardware technology have made it feasible to collect large amounts of data from the physical world using small sensor nodes. Details and surveys on sensor networks may be found in [1, 5, 6, 14]. These sensor nodes are often limited by battery power, and therefore limits on battery life can reduce the life time of the sensor network [1, 14]. When the number of sensor nodes is very large, the aggregate amount of data produced may be very large. This creates the dual problems of too much power consumption and information overload.

Sensor networks are inherently redundant, and have a much larger number of nodes than is needed to collect the data necessary to describe the underlying phenomenon. This is often by design in order to avoid loss of information during special circumstances such as node failure. There are two common kinds of redundancies which are built into sensor networks:

- **Direct Redundancies:** In many cases, sensor nodes may collect very similar kinds of data from related sensor nodes. An example is that of audio sensor nodes in geographically proximate locations. Sensors in geographically proximate locations may pick up almost the same signals from multiple nodes, though there may be some differences because of local differences and noise. In some cases, there may be small *lag correlations* in order to account for small temporal differences between different signals.

- **Implicit Redundancies:** In this case, external events may cause different kinds of characteristic behavior in different sensors. Typically, such characteristic behavior may not show up as the same signal on different sensors, but the signal on one sensor can be predicted from the signal on the other. For example, a seismic signal on one sensor may be correlated with an audio signal on another sensor. Typically such signals may show up as *statistical correlations* in the underlying sensor streams.

While the redundancy in data collection is essential, it comes at a huge cost, because of the aggregate power consumption requirements of the network. This increases the costs to maintain the sensor network effectively, and it also increases the complexity of managing the huge information overload from the large number of sensors. Furthermore, heavy power consumption implies frequent maintenance and replacement of the batteries in order to ensure uninterrupted coverage. Therefore, intelligent and selective choice of transmission of data from sensors is essential in maintaining both the robustness and cost-effectiveness of the signal collection process. A further challenge is that the underlying data streams may evolve over time. Therefore, the dependencies in the data streams may also change over time. Therefore,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'11, August 21–24, 2011, San Diego, California, USA.
Copyright 2011 ACM 978-1-4503-0813-7/11/08 ...$10.00.

it is critical to design *dynamic* methods for reducing the power-consumption and information overload issues by sensor selection.

This paper is organized as follows. The remainder of this section is devoted to related work. In the next section, we will discuss the overall formulation of the problem, and the problem of optimizing the redundancies in the underlying sensor stream. In section 3, we will propose the algorithm for dynamic sensor selection. The experimental results are contains in section 4. Section 5 contains the conclusions and summary.

1.1 Related Work

Since sensors are isolated devices which work on batteries, power is one of the most important constraints in their operation. The work in [6] reduces the data size with data reduction techniques at a particular sensor in order to compress the representation. The technique does not however try to use the correlations across different sensor nodes in order to reduce the collected data. The methods in [6] are in fact orthogonal to our work, and can be combined with our methods, if required. The problem of data reduction from correlated data has been widely studied in the conventional database literature, in the context of principal component analysis [11]. We note that this is simply a *data transformation* process which requires input from all the original dimensions in the data. While this is useful in reducing the *size* of the data, it is not useful from the perspective of power-efficiency or cost of collection. This is because, in our application, the reduction can only be in terms of the original dimensions, since this allows us to decide which sensors to keep in activated mode. Furthermore, the relative costs of different sensors are extremely important in making such decisions.

A number of techniques have been designed in recent years in order to determine correlations between multiple streams in real time [17, 18, 20, 21]. The techniques in [17, 18] use statistical measures in order to find lag-correlations and forecast the behavior of the underlying data stream. The works in [2, 8] propose methods for sensor selection with the use of domain-specific linkage knowledge and utility-feedback, respectively. Methods for observation selection are proposed in [12], when the importance of a sensor-set can be *pre-modeled* as a submodular function. Methods for using adaptive models for time-series prediction were proposed in [19]. The works in [3, 4] propose methods for using adaptive querying in the context of pairwise covariances between sensor streams for reducing the communication costs. The method in [20] uses linear regression in order to determine the correlations in the underlying data and use them for forecasting. The technique in [21] proposes general monitoring techniques to determine statistical correlations in the data in real time. While many of these techniques do detect correlations and redundancies with the use of forecasting and regression techniques, they do not propose a method to systematically reduce the collection complexity in a cost-efficient way with these redundancies. In this paper, we will design cost-sensitive senor selection techniques which are able to minimize loss of information.

2. SENSOR SELECTION FORMULATION

One solution to increasing the power-efficiency and complexity of data collection is to "switch-off" the redundant sensor nodes, and collect data only at the non-redundant nodes from which all other data can be predicted. Of course, if we completely switch off the redundant sensor nodes, then we have no way of knowing when the underlying data correlations change, and a different set of nodes become redundant. A solution is that the data collection and transmission rates at most sensor networks are highly adjustable. It is possible to operate most nodes in a very low and power-efficient mode which provides sufficient data in order to keep track of the underlying correlations. This reduces both the information overload and the power consumption for data collection. Based on the analysis from this low-rate of sampled data, it is possible to keep track of the non-redundant nodes. Therefore, we assume two modes of sensor data collection:

- **Power-efficient Mode:** All sensor nodes sample in a *normal mode*, which is a power-efficient and uses a low rate of data collection. We assume that the data streams are collected at multiple sensor nodes which are indexed as $\{1 \ldots n\}$. The data at different sensor nodes are received in the form of *ticks*. We assume that the ith tick of the jth sensor is denoted by x_{ij}. Therefore, the values of the ticks in the jth stream are denoted by $x_{1j}, x_{2j} \ldots x_{rj} \ldots$. For simplicity, we assume that the data collection at different sensors happen at the same rate, and the corresponding ticks are synchronized. We note that the synchronization may not happened at the individual sensor nodes, though a variety of simple imputation techniques can be used to convert the data into synchronized ticks at the centralized server at which the data is received. While this sampling mode is sufficient to determine the relative behavior of different streams, it may not be sufficient to make real-time decisions for the sensor network. We note that the data collected in power-efficient mode is essentially considered *training data* which is used to make decisions on the nature of the redundancy in the different nodes at any given moment in time.

- **Active Mode:** A higher mode of sampling may often be required in order to make effective real time decisions in a sensor network. This is expensive both in terms of power consumption and storage costs. In general, we would like to use the active mode of sampling for *only a small subset of nodes* which minimizes the power consumption, without significantly compromising the total amount of information relative to a network in which all nodes operate in the active mode.

The data from the power-efficient mode (or training data) is used to determine the subset of non-redundant nodes which are actively sampled for the data. By minimizing the number of such nodes, we can reduce both the power consumption and the complexity of storing the underlying data. We assume that the (additional) power costs required to activate the different sensor nodes are denoted by $p_1 \ldots p_n$. The different values of p_i are typically not identical, and may vary considerably. This is natural since an audio-sensor would have very different power consumption from a seismic sensor.

A centralized architecture is assumed for the sensor network in which all nodes transmit their signals to a centralized node. We note that we have picked this architecture,

since it is the simplest form of organizing the sensor network. The ideas in this paper can also be generalized to networks with arbitrary clusters with a hierarchical structure, since the redundancy reduction algorithms can be applied within a particular cluster of the hierarchy. The data repository is assumed to have the computational power to process and analyze the sensor streams in real time. It is also assumed that the sampling rate of sensor streams can be controlled in an automated way by using signals from the centralized data repository. Therefore, it is critical to continuously monitor and analyze the streams from the centralized data repository in order to minimize the error in estimation. The aim of this paper is to design an effective method for determining the set of sensors which are actively monitored.

Let us denote the set of active sensor nodes by J. We note that the signals from J can be used to estimate approximations on the active (higher frequency) signals from other (power-efficient) low sampled nodes. A variety of techniques such as lag-correlations or the regression-based MUSCLES technique [18, 20] can be used in order to perform this estimation. Let us denote an estimation model by $\mathcal{M}$. The key is that such a model needs to be efficient and real time in order to handle the large volume of data stream, and react to the evolution of the underlying data. Let i be any sensor node which is not included in J. Then, we define the approximation error of the sensor node i as the error in estimation of the tick values with the use of a model $\mathcal{M}$.

DEFINITION 2.1. *Let $\mathcal{M}$ be a model used in order to estimate the signals on inactive nodes from the actively sampled nodes. We denote $E(J, i, \mathcal{M})$ as the error in estimation of the sensor node i from the set J with the use of model $\mathcal{M}$. Specifically, if $Y(J, i, t, \mathcal{M})$ be the estimated value of the tth tick of sensor i with use of model $\mathcal{M}$ and subset J of nodes, then the estimation error $E(J, i, \mathcal{M})$ is defined as the cumulative error over the last window of length w from current time t. Therefore, we have:*

$$E(J, i, \mathcal{M}) = \sum_{r=0}^{w-1} |Y(J, i, t-r, \mathcal{M}) - x_{(t-r)i}| \quad (1)$$

The corresponding average error over all sensor nodes is defined by $F(J, \mathcal{M}) = \frac{\sum_{i \notin J} E(J,i,\mathcal{M})}{n-|J|}$. Note that the error is defined only over the nodes which are not drawn from J. The key optimization problem is to determine the optimal set J which minimizes the average estimation error over any node which is not in J, subject to a total power consumption constraint on the nodes which are included in J.

PROBLEM 2.1. *For any given model $\mathcal{M}$, determine J so as to find the optimum value of $F(J, \mathcal{M})$, while ensuring that $\sum_{i \in J} p_i \leq P$.*

We note that the quality of the reduction also depends upon the model $\mathcal{M}$ which is picked for forecasting the behavior of the passive nodes. This is especially difficult, since the model $\mathcal{M}$ needs to be a *real time model* which can quickly detect the changing inter-dependencies, and correspondingly adjust the set of non-redundant sensor nodes J.

3. LOCAL REGRESSION CLUSTERING

In this section, we will discuss a *regression-clustering* approach for data reduction. In the regression-clustering approach, we create clusters of the different streams. At any given moment in time, each stream belongs to a particular cluster. These clusters may evolve over time as the underlying stream patterns change. The clustering is constructed in conjunction with a *base of active sensor-stream medoids*, around which the data is appropriately clustered. This is essentially a k-medoid based partitioning approach in which the data is clustered around k medoids [13]. These medoids are the representatives around which the different streams are clustered. This base of sensor-streams is also the set of the sensors J which are kept in active mode in order to collect the underlying data. Therefore, the representatives need to be chosen so that the signals for the passive sensors can be accurately predicted from at least one of the active sensor. The continuous clustering provides us with the real-time assignment of the passive sensors to the active sensors. This assignment is utilized in order to perform a one-to-one regression of the underlying data, and fill in the unsampled data points for the passive sensors. In the clustering process, each of sensor streams are assigned to one of the sensor nodes J with the use of a *regression-error based distance function*. This distance function computes the distance as the regression error of assigning a sensor node to each of these alternative sensor-streams, and computes the distance based on the error of the appropriate prediction. Next, we describe the details of the regression distance function used for similarity computation and analysis. Let us consider to sensors i and j such that j is a passively-sampled sensor, in which the last w passively sampled ticks are given by $z_1(j) \ldots z_w(j)$. The corresponding ticks from the sensor i are given by $z_1(i) \ldots z_w(i)$. We would like to compute the predictability of the signals $z_1(j) \ldots z_w(j)$ from the signals $z_1(i) \ldots z_w(i)$. We assume that a lag of r ticks may be required in order to predict the signals $z_1(j) \ldots z_w(j)$ from the signals $z_1(i) \ldots z_w(i)$. Furthermore, since the sensor j is predicted from sensor i, the lag can only be in one direction for the purpose of predictability i.e. j lags i. It is assumed that the maximum lag allowed in terms of the number of ticks is

Algorithm *MaintainActiveS*(Sensors Streams: $[1 \ldots n]$
Power Constraint: P);
{ This algorithm describes one iteration of the process
of maintaining an active set of nodes;}
begin
J = Randomly sampled set of sensor nodes with
aggregate power requirement less than P;
At the next (passively-sampled) tick **do**
repeat
Add a sensor to J, which leads to
maximum decrease in $PError(J)$;
while J violates the power constraint
begin
Drop the sensor from J which leads to
least increase of $PError(J)$
end
until(J did not change in last iteration)
Collect active ticks for current set J;
Predict active ticks for all other sensors using
regression analysis from the current assignment;
end

Figure 1: Dynamically maintaining the active sensor set by local regression clustering

denoted by $maxlag$. For a lag of r ticks, we assume that the regression takes on the following form:

$$z_k(j) = a \cdot z_{k-r}(i) + b \qquad \forall k \in \{r+1 \ldots w\} \tag{2}$$

Here a and b are constants. We would like to pick a and b, so as to minimize the least squares error of estimation. The total square error $e(i,j,r)$ of predicting j from i is given by the following expression:

$$e(i,j,r) = \sum_{k=r+1}^{w} (a \cdot z_{k-r}(i) + b - z_k(j))^2 \tag{3}$$

In order to minimize $e(i,j,r)$ we need to pick a and b such $\delta e(i,j,r)/\delta a = 0$ and $\delta e(i,j,r)/\delta b = 0$. This corresponds to the following two relationships:

$$\sum_{k=r+1}^{w} z_{k-r}(i) \cdot (a \cdot z_{k-r}(i) + b - z_k(j)) = 0 \tag{4}$$

$$\sum_{k=r+1}^{w} (a \cdot z_{k-r}(i) + b - z_k(j)) = 0 \tag{5}$$

We note that these are linear relationships in a and b, which can be easily solved for the optimal values a^* and b^*. Let $e^*(i,j,r)$ be the minimized value of the error, when we set $a = a^*$ and $b = b^*$ respectively. We note that the value of $e^*(i,j,r)$ over different values of r cannot be compared with one another, since we use different window lengths for computing the different errors. Therefore, we compute the *per-tick optimized error* $E^*(i,j,r)$ by averaging the error over the number of ticks over which it is computed. Therefore, we have:

$$E^*(i,j,r) = e^*(i,j,r)/(w-r) \tag{6}$$

The distance function $D(i,j)$ between sensors i and j is defined as the minimum value of $E^*(i,j,r)$ over all possible values of the lag $r \in [0, maxlag]$. Therefore, we have:

$$D(i,j) = \min_{r \in [0,maxlag]} E^*(i,j,r) \tag{7}$$

The optimized value of r denotes the optimal value of the lag at which the sensor signal j can be predicted from the sensor signal i most accurately. This accounts for lag-correlations in the underlying data. Thus, the distance function $D(i,j)$ defines the best predictability of sensor j from sensor i over all possible lag values in the range $[0, maxlag]$. This distance function is used in order to create an effective partition-based clustering algorithm.

3.1 Constructing Regression Clusters

In this section, we will discuss the procedure for constructing regression clusters from the underlying data. We will design a clustering algorithm which uses the regression distance function in order to cluster all sensors around a bunch of representative (active) sensors, so as to maximize predictability. Therefore, we will rely on the class of k-median techniques [10] which utilize a bunch of cluster-representatives which are picked from the original data. The traditional k-median algorithm for clustering uses repeated interchange operations on a set of selected cluster representatives. In our algorithm, this set of cluster representatives is the active set J. The algorithm starts off by setting the active set J using random assignment. Subsequently, the algorithm uses an iterative approach in order to improve the quality of the active set J. We note that the quality of the active set J is defined by the accuracy of the prediction of all other sensor signals from the active set J. This is denoted by $PError(J)$, and is defined by assigning each sensor to its closest sensor in J with the use of the distance function $D(\cdot,\cdot)$. The average error of this assignment is denoted by $PError(J)$. This assignment error is calculated from the set J as follows:

for each pair of sensors $j \in [1,n]$ and
  sensor $i \in J$ compute $D(i,j)$;
Assign j to the sensor with the least value of $D(i,j)$;
Let $PError(J)$ be the corresponding average of the
predictability error of the assignment across all sensors;

The overall algorithm for dynamically maintaining the sensor set J is illustrated in Figure 1. This set J serves as the *medoids*, which is also the active sensor set. Typically, k-median algorithms use interchange operations on the medoids in order to compute the representatives from which the cluster partitions are created. However k-median algorithms use a fixed number of representatives, whereas we need to use a power constraint in this case in order to regulate the number of representatives. Therefore, in this case, we separate out the interchange into separate representative addition and deletion operations, and the number of sensors may vary over the course of the algorithm. However, we ensure that the power requirement of the sensors in J is less than the total power constraint P. The algorithm re-computes the active sensor set J after the receipt of each passive tick. This re-computed sensor set is used in order to perform the regression based prediction of the active ticks received between the current passive tick and the next passive tick. Therefore, in each iteration, we first add a sensor to the set J, which minimizes the average assignment error (computed by $D(\cdot,\cdot)$) as much as possible. We note that the addition of this sensor may result in a violation of the power constraint. Therefore, we need to remove some sensors from J, so that the resulting set continues to satisfy the power constraint. To this effect, we remove the sensors, which result in the least increase in prediction error, until the power constraint is satisfied again. We note that it is possible that the same sensor may be removed, which was added during a given iteration. In such a case, the set J does not change. This implies that the current set of active sensors is critical and we can terminate the algorithm. We note that this overall approach is continuously applied in order to maintain the active set J. This set is used in order to make the predictions for the other sensors for which the ticks are not collected actively.

3.2 Speeding up with Ranking Variations

One of the challenges with the use of the technique is that it can sometimes be too slow, when the algorithm requires a large number of iterations. Instead of going through all sensors and finding the one that minimizes the error for every iteration, we only do this process for the first iteration. Subsequently, we rank all the sensor by the errors in terms of regression predictability. In this way, we can get a ranking list of all sensors. From the second iterations, sensors are added sequentially with the use of the ranking list. When there is no change to the active set in a cycle through the ranking list, the process is terminated.

3.3 Leveraging Local-Regression Clusters for Prediction

The prediction of the active ticks from the sensors can be performed in two ways, once the data for the different sensors has been collected:

- For a given sensor j can use a least-squares regression from the sensor $i \in J$ to which the sensor j is assigned in order to predict the corresponding active ticks.
- We can use multi-variate regression on the entire set J in order to predict the active ticks for any sensor i.

We note that the second variation requires more computational power, but can be somewhat more accurate if the remaining sensor set retains information which relates to the prediction effectively. On the other hand, most of the predictive information is encoded in the local sensor node. In such cases, the use of multi-variate regression can add further noise to the predictive process, and in some cases, the quality of multi-variate regression may not be higher. Therefore, it seems like a reasonable tradeoff to use the information in the local sensor node for predictive purposes. In the next section, we will study the experimental effectiveness of the technique over different size of the sensor node sets J. We will see that the local method for prediction turns out to be quite efficient and effective over a wide range of data sets and parameters.

4. EXPERIMENTAL RESULTS

In this section, we will present the experimental evaluation of our algorithm on several real data sets. To be more specific, three real data sets are used for test purposes. The first two are derived from the intel berkeley lab data set [22]. This data set has 54 sensors, and each record includes four values of temperature, humidity, light and voltage. Since the values of light and voltage did not change too much over time, we only used temperature and humidity in our experiments. The third data set is the chlorine data set [16], which is downloadable from [23]. The data set was collected from 166 sensors. In addition, we need to generate a synthetic power requirement for each sensor. In our experiments, the power requirements of the different sensors are uniformly distributed in the range $[0, 1]$.

For each data set, several techniques are adopted to illustrate the effectiveness and efficiency of our proposed approach. The overall approach comprises the process of picking the active sensors and making predictions. The average error in prediction of the sensor values is the key metric, which is used for evaluating the effectiveness of our approach. This section will provide detailed results and is organized as follows. In the first part, we will analyze the effectiveness and accuracy of the technique. In the second part, the efficiency of the algorithm will be analyzed. Finally, we will provide some insights about how the underlying active set of sensors may change over time.

4.1 Accuracy Analysis

In this section, we will provide an accuracy analysis of the sensor data reduction technique. The aim of the approach is to use cluster-based regression techniques to provide a high quality reduction of the sensors over which the data needs to be collected. The best way to measure the effectiveness is to analyze the accuracy of the prediction. To test the effectiveness, we compared the results of several variations of our proposed technique for regression and prediction. In addition, we used a couple of baselines in order to measure the qualitative accuracy. The different algorithms are discussed below, of which the last two are baselines:

- **Univariate Power Efficient Selection(*PES Univariate*)**: Note that at the end of the process, each sensor gets assigned to at least one other sensor. The assigned sensor is used for prediction purposes with the use of univariate regression.
- **Multivariate Power Efficient Selection(*PES Multivariate*):** We use multi-variate regression from a subset of the sensors in J in order to make the prediction. For regression purpose, the number of such sensors picked is the minimum of the number of active sensors and $w-maxlag-2$. In each iteration, the error-minimization criterion is used in order to add sensors to the active set. A multivariate prediction approach is used in conjunction with the selected sensors in order to perform the prediction.
- **Multivariate Power Efficient Selection with Ranking List (*PES Multivariate(RL)*):** In this case, we use the same technique as PES Multivariate for the first iteration. We also obtain a ranking list of sensors as well based on their predictive power. For the remaining iterations within a particular update interval, we do not use the error minimization criterion, but we use the next senor in the ranking list. Multivariate regression is used to perform the prediction.
- **Univariate sampling: Baseline (*Sampled Univariate*):** In this case, we use sampling in order to determine the set of active sensors. The number of sampled sensors is the same as the *PES Univariate* technique. As in the case of the *PES Univariate* technique, we use univariate regression in order to perform the prediction.
- **Multivariate sampling: Baseline (*Sampled Multivariate*):** As in the previous case, we use sampling in order to determine the set of active sensors. The number of sampled sensors is the same as the PES multivariate technique. As in the case of the *PES Multivariate* technique, we use multivariate regression in order to perform the prediction.

We will test the effectiveness of the technique over a wide range of parameters. The most relevant parameter is the power constraint, since it defines the resources which are relevant to the data reduction process. Clearly, a larger power allowance provides greater accuracy for the prediction process, since it allows us to retain a larger number of active sensors. The variations in effectiveness with increasing power constraint for the different data sets are illustrated in Figures 2(a), 2(b) and 2(c) respectively. In the experiment, we use a window size w of 10, a maximum lag $maxlag$ of 5 and an update interval of 5. The power constraint is illustrated on the X-axis, and the error is illustrated on the Y-axis. It is clear that in each case, a larger power constraint provides greater accuracy. The power efficient variations provided much greater accuracy than the sampled variations. An additional observation was that the multivariate versions were

significantly more accurate than the univariate prediction techniques. The reason for the greater accuracy of the multivariate prediction techniques was the fact that a larger number of sensors could be used to make more accurate predictions. In each case, the *PES Multivariate technique* provided the greatest accuracy. The ranked technique approximated the *PES Multivariate technique* quite closely. As we will see, this is an excellent practical alternative for the optimal multivariate scheme.

The window size is a parameter related to the number of data points used for regression. The power constraint P was set to 3, the maximum lag $maxlag$ was set to 5, and the update interval was set to 5. With a fixed max lag, a larger window size leads to the use of greater number of data points. Extremely small window sizes leads to lack of robustness in prediction. At the same time, very large window sizes can lead to smoothing of the trends in the data which have predictive power. The results of the three data sets are illustrated in Figures 3(a), 3(b) and 3(c). The window size is illustrated on the X-axis, and the error is illustrated on the Y-axis. It is evident that different window sizes work better with different data sets. In practice, this corresponds to the core periodicity of the different data sets. However, it is also evident that the *effectiveness and efficiency ordering among the different techniques* does not vary across the different ranges of parameters tested.

The choice of update interval is a key parameter which decides the refresh rate for our algorithm. The results are illustrated in Figures 4(a), 4(b) and 4(c). The update interval is illustrated on the X-axis and the error is illustrated on the Y-axis. The value of the power constraint P was 3, the value of the window size w was 3, and the value of the maximum lag $maxlag$ was 5. It is evident that higher update intervals lead to better quality results. This is because higher update intervals lead to the use of less stale data and will therefore naturally lead to more accurate results. On the other hand, a higher update interval also requires more computational processing. This is a natural tradeoff between computational processing and accuracy.

The regression analysis uses a *maximum lag value* to perform the prediction. The larger the value of this maximum lag is, the greater the flexibility of the algorithm is in being able to perform the regression analysis. The results are illustrated in Figures 5(a), 5(b) and 5(c) respectively. The power constraint P was set to 3, the window size w was set to 10, and the update interval was set to 5. The maximum lag is illustrated in the X-axis, and the error is illustrated on the Y-axis. It is clear that the error reduces with the increase in the maximum lag value. The tradeoff is that it results in some delay in the predictive data received from the sensors.

4.2 Computational Efficiency

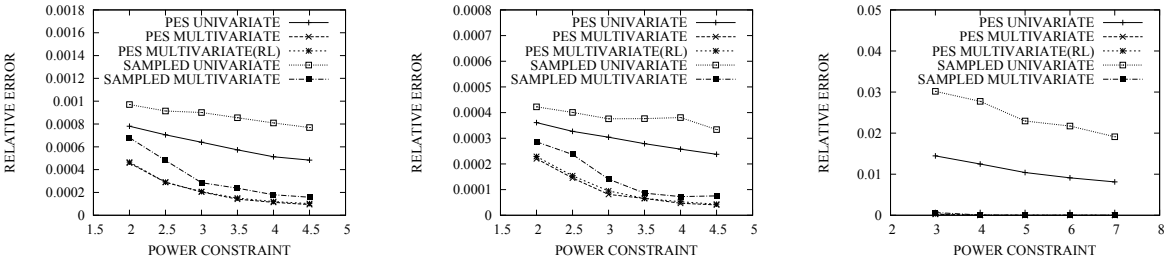
All experiments were performed on a Microsoft Windows XP machine with Intel Core2 Duo 2.5GHz CPU and 1.5GB main memory. The algorithm was implemented in C++. We will examine the computational complexity of performing each update. Since the sampling techniques do not require such a computational process, they will not be included in this analysis. The running times for updates are mostly determined by the underlying parameter setting. These settings are set to the same values as those used for the effectiveness results. Since the efficiency results for the three data sets were extremely similar, we present the results only on the *intel-humidity* data set.

Before examining each parameter separately, we will provide a brief overview of the relative trends between the different figures. The *PES Multivariate* technique required the greatest amount of time, because it requires an iterative approach in order to determine the optimal set. Furthermore, it requires us to determine the prediction with the use of optimal multivariate analysis. Between *PES Univariate* and *PES Multivariate(RL)*, the two techniques were very close to one another in terms of running time. This is essentially because most of the time was spent in the update process, and the difference between univariate and multivariate analysis was relatively small.

The results with increasing power constraint are illustrated in Figures 6(a). For test purpose, we fixed window size w to be 10, maximum lag to be 5, and update interval to be 5. As the power constraint increases, it is easy to understand that the running time will become longer. This is because the use of a higher power constraint leads to the use of a larger number of sensors. The use of a larger number of sensors naturally increases the running time, because the regression analysis and optimization needs to be performed over a larger number of sensors. However, it is important to note that the running time increases only linearly with the power constraint. This means that the scheme can be implemented fairly efficiently over a large number of sensors.

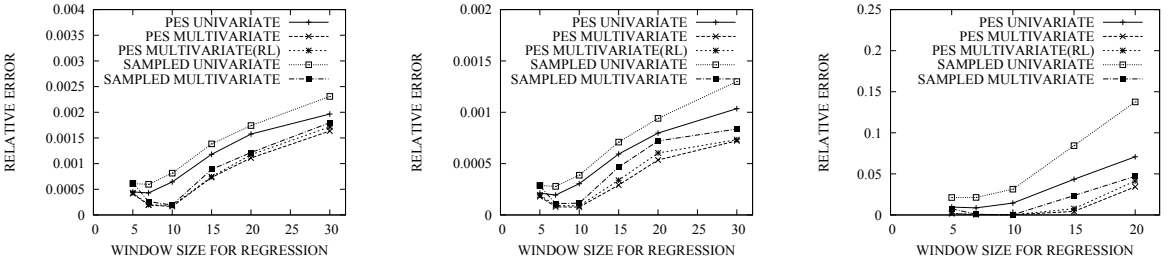
It is generally expected that a larger window size should increase the running time. However, this is not always the case. The reason is that the running time is also dependent upon the number of iterations required by the algorithm. When the active sensor set changes considerably from one iteration to the other, a larger number of iterations are required. This change over different runs dictates the running time to a greater extent than the window size. The results with increasing window size for the humidity data set are illustrated in Figure 6(b). Here, the power constraint P was 3, the maximum lag $maxlag$ was 5, and the update interval was 5. It is clear that the time is not necessarily monotonically increasing. This will be explained elaborately in similarity analysis. This also means that the scheme can be used over larger window sizes when required without substantially increasing the running time.

The sensitivity of the running time to the maximum lag for regression analysis is illustrated in Figures 6(c). The maximum lag is illustrated on the X-axis and running time is illustrated on the Y-axis in each of the figures. The power constraint P was set to 3, the window size w was set to 10, and the update interval was set to 5. We note that the maximum lag affects the number of possible computations performed by the algorithm. Clearly, a larger lag increases the number of computations, since a larger number of possibilities need to be checked for optimality. While it has been shown that this leads to higher quality results, the corresponding cost is the higher running time. Therefore, the running times also increase with increasing value of the maximum lag. It is evident that the running times increase only modestly with the increase in the maximum lag. Therefore, all the three trend figures on running time show modest scalability with different parameters such as the power constraint, maximum lag, and window size. This suggests good scalability characteristics with increasing data size.



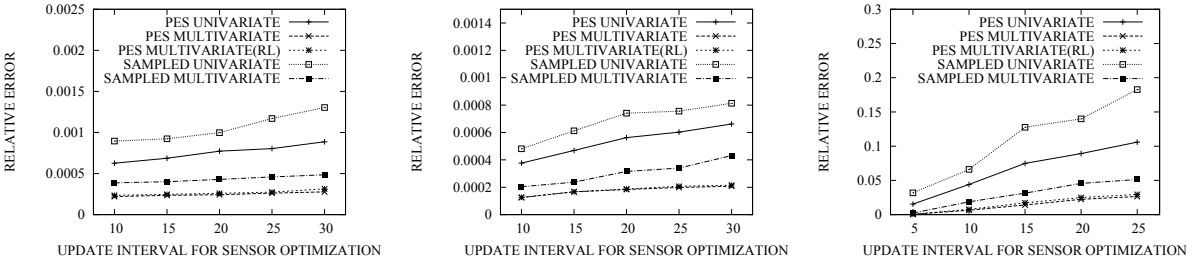
(a) Data Set *intel-humidity* (b) Data Set *intel-temperature* (c) Data Set *Chlorine*

Figure 2: Error Plot on Power Constraint



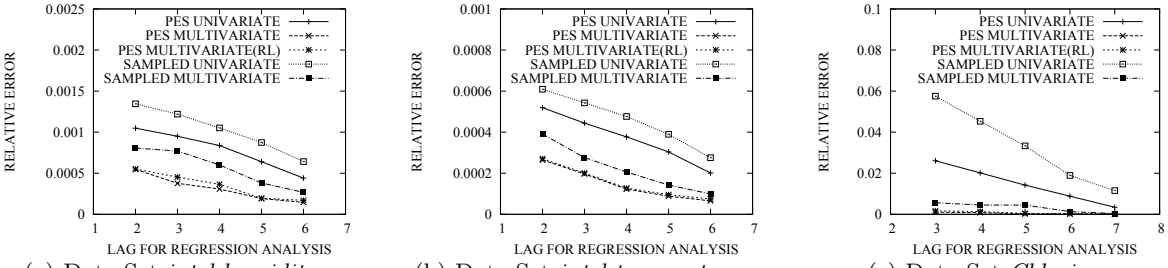
(a) Data Set *intel-humidity* (b) Data Set *intel-temperature* (c) Data Set *Chlorine*

Figure 3: Error Plot on Window Size



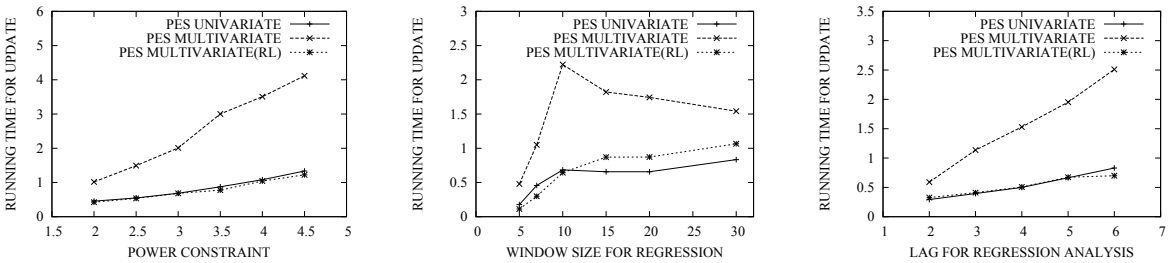
(a) Data Set *intel-humidity* (b) Data Set *intel-temperature* (c) Data Set *Chlorine*

Figure 4: Error Plot on Update Interval



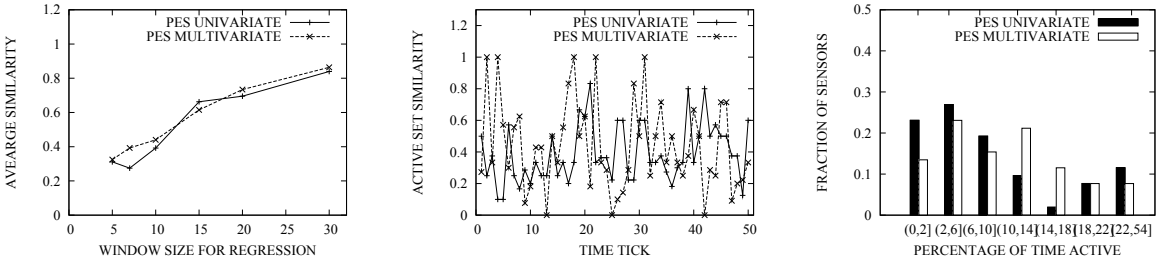
(a) Data Set *intel-humidity* (b) Data Set *intel-temperature* (c) Data Set *Chlorine*

Figure 5: Error Plot on Maximum Lag



(a) Power Constraint (b) Window Size (c) Lag for Regression Analysis

Figure 6: Running Time (Data Set *intel-humidity*)



(a) Average Similarity Plot (b) Similarity Plot (c) Duration Plot

Figure 7: Sensitivity Analysis (Data Set *intel-humidity*)

4.3 Similarity Analysis

As stated in previous sections, the dependencies between different sensors may change as the data evolves over time. When the active set changes considerably from one iteration to the next, this affects both the prediction errors and the running time. Therefore, it is useful to examine how the sensor set changed during the progression of the algorithm. One way of doing so is to examine the change in the active sensor set from one iteration to the next. To quantify the change of two sets, we adopt the definition of jaccard similarity coefficient between the sensor sets between two iterations. Given two sets A and B, the Jaccard similarity is defined as: $J(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$.

According to the previous analysis, the prediction error for different window sizes is influenced by the ability of the system to adapt to changes in the underlying data distribution. Since the prediction error was rather sensitive to window size, we test the average similarity against window size in Figure 7(a). The results clearly show that the average similarity increases with window size. This suggests that the active set changes more slowly with increasing window size. This is because regressions which are based on larger window sizes, tend to change the active set more slowly. As a result the prediction error is better for smaller window sizes, in which the system is able to adjust more rapidly to the changes in the underlying data distribution. This is also the reason for our earlier counterintuitive observation that the execution time is sometimes longer even though we have a smaller window size. If the similarity measure is low, it indicates that it requires more update time to determine the new set.

Figure 7(b) shows the details of the similarity change over time. Here we use a window size of 10 as an example. The results show that the similarity values suddenly drop at specific times. These correspond to the times at which the patterns in the regression stream change a lot, and correspondingly the active set changes. Thus, this figure illustrates the adaptivity of our approach in dynamically changing the sensor set in responses to changes in the underlying sensor stream patterns and correlations.

The other measurement of the change of the active set used in our experiment is the distribution of number of appearances, which is illustrated in Figures 7(c). The X-axis of the histograms is the percentage of time that a particular sensor stays active, while the Y-axis shows the proportion of the sensors whose activity lies within that range. It is clear that most sensors do not stay active for a very long time span. In most cases, they appear to be active only for a few ticks. This suggests that in practical scenarios, it may be possible to use the redundancy inherent in a sensor network and collect most of the relevant information, while keeping only a small fraction of the sensors active at a given time.

5. CONCLUSIONS AND SUMMARY

In this paper, we presented a new power efficient data reduction scheme for sensor networks. The key idea of the technique is to use regression analysis in order to determine a small active set with minimal power requirements. This small active set is used to predict the values on many of the other redundant sensors. We show the effectiveness of using such a technique and show its applicability for a number of different data sets. We present several variations of the scheme and show its efficiency and effectiveness over a number of data sets. The results suggest that it is possible to use a small number of sensors to continuously predict all the streams for the purpose of mining. This allows significant savings in power and storage requirements, while retaining predictive accuracy.

Acknowledgements

The first author's work was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.
The work of the second and third authors is supported in part by NSF through grants IIS-0905215, DBI-0960443, OISE-0968341 and OIA-0963278.

6. REFERENCES

[1] C. C. Aggarwal (ed). Data Streams: Models and Algorithms. *Springer*, 2007.

[2] C. C. Aggarwal, A. Bar-Noy, and S. Shamoun. On Sensor Selection in Linked Information Networks. *IEEE DCOSS Conference*, 2011.

[3] C. Anagnostopoulos, N. M. Adams, and D. J. Hand. Streaming Covariance Selection with Applications to Adaptive Querying in Sensor Networks. *The Computer Journal*, 53(9): pp. 1401–1414, 2010.

[4] T. Arici, T. Akgun, and Y. Altunbasak. A prediction error-based hypothesis testing method for sensor data acquisition. *ACM Transactions on Sensor Networks (TOSN)*, Vol. 2, pp. 529–556, 2006.

[5] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing Historical Information in Sensor Networks. *ACM SIGMOD Conference* pp. 527–538, 2004.

[6] A. Deligiannakis, and Y. Kotidis. Data Reduction Techniques in Sensor Networks. *IEEE Data Engineering Bulletin* 28(1): pp. 19–25, 2005.

[7] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model Driven Data Acquisition for Sensor Networks. *VLDB Conference*, pp. 588–599, 2004.

[8] D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. *IPSN Conference*, pp. 220–231, 2010.

[9] M. Greenwald, and S. Khanna. Power Conserving Computation of Order Statistics over Sensor Networks. *ACM PODS Conference*, pp. 275–285, 2004.

[10] A. Jain, and R. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1998.

[11] I. T. Jolliffee. Principal Component Analysis. *Springer*, 2002.

[12] A. Krause, and C. Guestrin. Near-optimal observation selection using submodular functions. *AAAI Conference*, pp. 1650–1654, 2007.

[13] R. T. Ng, and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. *VLDB Conference*, pp. 144–155, 1994.

[14] G. Kollios, J. Byers, J. Considine, M. Hadjieleftheriou, and F. Li. Robust Aggregation in Sensor Networks. *IEEE Data Engineering Bulletin*, 28(1): pp. 26–32. 2005.

[15] C. Olston, J. Jiang, and J. Widom. Adaptive Filters for Continuous Queries over Distributed Data Streams. *ACM SIGMOD Conference*, pp. 563–574, 2003.

[16] A. Ostfeld(et.al.) The battle of the water sensor networks (BWSN): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, vol. 134, no. 6, pp. 556–568, 2008. [Online]. Available: `http://link.aip.org/link/?QWR/134/556/1`

[17] S. Papadimitriou, J. Sun, and C. Faloutsos. Dimensionality Reduction and Forecasting of Time-Series Data Streams. *Data Streams: Models and Algorithms, Ed. Charu Aggarwal, Springer*, Chapter 12, pp. 261–288, 2007.

[18] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. BRAID: Stream Mining through Group Lag Correlations. *ACM SIGMOD Conference*, pp. 599–610, 2005.

[19] L. Yann-Ael, S. Santini, and G. Bontempi. Adaptive model selection for time series prediction in wireless sensor networks. *Signal Processing*, Vol. 87, pp. 3010–3020, 2007.

[20] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. V. Jagadish, C. Faloutsos, and A. Biliris. Online Data Mining for Co-Evolving Time Sequences. *ICDE Conference*, pp. 13–22, 2000.

[21] Y. Zhu, and D. Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. *VLDB Conference*, pp. 358–369, 2002.

[22] Available online at: `http://berkeley.intel-research.net/labdata/`

[23] Available online at: `http://www.cs.cmu.edu/afs/cs/project/spirit-1/www/`