Active Learning Driven Data Acquisition for Sensor Networks

Anish Muttreja[†], Anand Raghunathan[‡], Srivaths Ravi[‡] and Niraj K. Jha[†]

[†]Dept. of Electrical Engineering, Princeton University, NJ 08544
[‡]NEC Labs, Princeton, NJ 08540 *

Abstract

Online monitoring of a physical phenomenon over a geographical area is a popular application of sensor networks. Networks representative of this class of applications are typically operated in one of two modes, viz. *an always-on mode where every sensor reading is streamed to a base station, possibly after in-network aggregation, and a snapshot mode where a user queries the network for an instantaneous summary of the observed field. However, a continuum of data acquisition policies exists between these two extreme modes, depending upon the rate and manner in which each sensor node is queried. In this work, we explore this continuum to improve network energy efficiency.*

We present a data acquisition framework that models the evolution of the observed data field at each sensor location as a function of time and uses an active learning based criterion to intelligently sample each sensor. Sensor nodes in our framework are organized in a clustered hierarchy. Time-dependent models of sensor readings are maintained at cluster-head nodes, which sample nodes in their cluster in a way that minimizes total energy consumption while maintaining confidence bounds on the overall model. We use sparse Gaussian processes to model sensor readings and variance minimization based active learning to intelligently select sensor nodes for querying. Finally, we present simulation results demonstrating up to 70% savings in total network energy, compared to the base case, in which sensors are sampled according to a cyclic schedule.

1 Introduction

Consider a wireless sensor network measuring a data field y. The network maintains an estimate of the field at a central base station (S). The measured field is a *spatio-temporally* correlated process, *i.e.*, the measured value at node x at time t is correlated to the values measured at other nodes and the previous values measured at node x. Typical examples of such a process include measurements of various environmental quantities, *e.g.*, temperature, pressure, rainfall, particle concentrations, *etc.*

An important design task in the above application scenario is the design of a data acquisition policy. Frequently, the user is only interested in an approximation of the measured process. In other words, most sensor network applications can accept a certain degree of distortion. Therefore, it would be redundant to acquire data from all nodes at their maximum sampling capacities. In energy-constrained sensor networks, it would also severely degrade network lifetime. On the other hand, arbitrarily reducing the sampling frequency (in space or time) might lead to unacceptable distortion in the estimate of the measured process. In this paper, we present a data acquisition framework that is able to make an intelligent tradeoff between field distortion and network resource usage. We consider the specific problem of formulating a data acquisition schedule, in order to minimize energy consumption, while maintaining user-specified probabilistic error bounds.

The basic principle underlying our framework is easily explained. We construct a model, $f(x,t)$, at base station S, which approximates the actual readings, $y(x,t)$, at each sensor, x, at time t. The model is used to obtain the most probable estimate $\hat{y}(x,t)$ of $y(x,t)$ along with an estimate of the model's certainty, $c_f(x,t)$, that the estimate does not exceed a given error bound. Put another way, model estimate $\hat{y}(x,t)$ is considered valid at a point $\langle x,t\rangle$ if $c_f(x,t)$ exceeds a minimum level of confidence $c^o(x,t)$. A valid model might be used directly. If the model is invalid at one or more sensor locations, additional readings must be collected to refine it. Intuitively, because of correlation, we do not need to sample all nodes at any time, only enough to improve model confidence sufficiently. Determining the optimal set of nodes to sample is a very hard problem. Instead, we adopt a greedy approach, sequentially sampling more points and refining the model until the confidence requirement at each node is met.

In the machine learning community, the problem of sequentially acquiring data in order to improve model quality with minimum cost is studied under a research focus known as *active learning* [1, 2]. We utilize ideas from active learning to obtain a sampling framework well-suited to sensor networks. We consider one-hop as well as clustered networks and show that the framework maps well to both topologies.

The remainder of this paper is organized as follows. We discuss related work in Section 2 and provide motivation in Section 3. We present our data acquisition framework in detail in Section 4 and results in Section 5. Section 6 concludes.

2 Related Work

Data acquisition and communication mechanisms, which can adapt to varying data characteristics in order to conserve network resources, have received considerable attention in both sensor network and streaming database research. We trace related research in these areas. However, due to the wide range of contexts under which the problem has been examined, this section aims only to be representative rather than being exhaustive.

Resource management research in streaming databases has concerned itself mainly with data filtering [3, 4, 5, 6]. As the name suggests, data filtering techniques are aimed at conserving network bandwidth by filtering out data that may not be relevant in the current context. The filtering may be done at the source [7] or within the network [6]. Lossy compression techniques for sensor networks [8, 9] may also be viewed as a variant of in-network filtering.

A characteristic of sensor networks, which distinguishes them

*Acknowledgments: This work was supported by NSF under grant No. CCF-0428446.

from other types of streaming databases, is the high degree of correlation in sensor data streams, which makes sensor data highly amenable to data aggregation and predictive modeling techniques. Data aggregation [10, 11, 12] refers to in-network transformation and compression of correlated data, in order to conserve network bandwidth and energy. Predictive modeling techniques are more closely related to our work. Models of sensor data have been used to make routing decisions [13], manage data acquisition [14, 15] and regulate sampling rates at individual nodes [16, 17]. The data acquisition framework, BBQ, which is presented in [14, 15], is shown to perform very well on a range of approximate spatial queries. Temporal evolution of sensor data is also supported to an extent, but direct support for continuous data collection queries does not seem to have been provided. BBQ seems to have been designed to support the so-called snapshot mode of operation, where either a higher application layer or the user must decide when data needs to be collected from the network. On the other hand, continuous data collection is the focus of the work in [16, 17], where sampling rates are adjusted according to variations in data characteristics at each node. These approaches, however, do not make use of spatial correlation which can be used to further reduce the number of samples taken across sensors.

The broad paradigm of interleaving model-driven data collection and model refinement has also been explored in backcasting [18], and subsequently in [19, 20] where Gaussian processes and active learning techniques are utilized. All the aforementioned works focus on deciding optimal sensor locations to sample a field before the sensor network is deployed, whereas our work is concerned with evolving a sampling strategy for an already-deployed sensor network. The modeling methodology and active learning heuristics used in our work are also different from previous research in sensor networks.

In summary, it is our view that sensor data acquisition should make transparent use of both spatial and temporal correlation in sensor data. Previous work has focused either exclusively or chiefly on only one of the space and time dimensions. In this paper, we explore a framework that attempts to treat both spatial and temporal dimensions on an equal footing.

3 Motivation

Typically, sensor deployments operate on a cyclic schedule. Each sensor node reports data at a requested sampling rate. It is our contention that the number of samples required can be reduced significantly, without a significant loss in accuracy, if data acquisition is managed using a spatio-temporal model such as the one proposed here. We propose to sample each node only in response to a fall in model confidence. To justify this hypothesis, we applied our data acquisition framework to temperature data collected from weather buoys deployed in the Pacific ocean by the Pacific Marine Environment Laboratory [21]. We experimented with a 400-minute long trace of temperature data collected in May 2000, from seven weather buoys placed at 10° latitude intervals, along the 110°W longitude, from 30°N to 30°S. The trace contained readings taken every ten minutes. Our interest here is to explore whether the sampling frequency can be reduced while still maintaining a good estimate of actual buoy readings. We constructed a model of the temperature sensed at all buoy locations. The model was of the form:

$$T = f(x,t),\ x \in \{1,2\ldots,7\} \tag{1}$$

where T is the predicted temperature, x is a sensor index and t is time. The model was initialized with a trace of ten readings, from each of the seven sensors. As an experiment, for the next

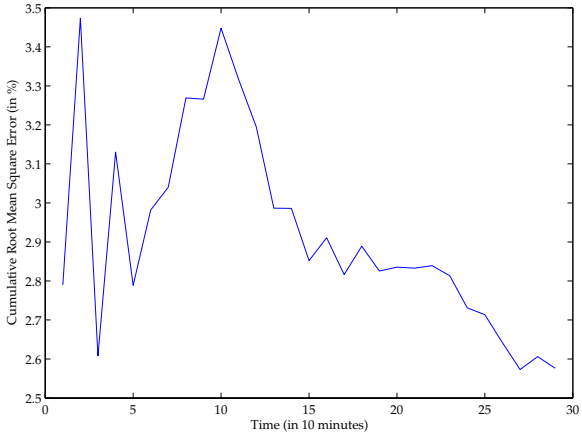


Figure 1. Evolution of temperature sensor model over time

300 minutes, we fixed an *ad hoc* limit on sampling. During every ten-minute interval, at most one sensor was queried from among the seven available and the observed reading was used to update the model. The sensor to be queried was selected according to an active learning criterion described in Section 4.4. Post collection, the model was used to estimate the temperature value at each sensor location during the entire period. We found that even at this drastically reduced level of sampling, a fairly accurate estimate could be obtained.

Fig. 1 plots the cumulative percentage mean square[1] model error. The figure shows the error observed in the model estimate relative to observed data during every ten-minute cycle. Time is shown starting from the 11th cycle, which was the first cycle for which data were not collected from every node. The error can be seen to be at most 3.5%. Thus, for a small loss in accuracy, we were able to reduce the number of sensor readings and associated sensing and communication costs during this period to one-seventh of the baseline. While the example considered in this section came from a deployment that is of a geographical scale and much larger than typical wireless sensor networks, it does serve to make a case for a fine-grained adaptive data acquisition framework.

4 The Data Acquisition Framework

In this section, we discuss our data acquisition framework, beginning with an overview. Key constituent parts of the framework, a modeling methodology and node selection heuristics, are described in later sections. Let X denote the set of all nodes in the network. The centerpiece of our framework is a model $f^S_{\tau,\tau-W}(x,t)$ of data sensed by the network. The subscript indicates that the model evolves with current time τ, and is based only upon values received at node S during the past W time units. We assume that W has been carefully selected to ensure that the model tracks the data stream well[2]. In the remainder of this paper, we will drop the subscript and superscript with the understanding that f^S depends on current time τ and length of sliding window W. The superscript also will only be used if the location where the model is built is not clear from the context. Our objective is to collect data in such a way that model f maintains a user-specified

[1] The absolute root mean square error was divided by the mean of all test readings to express it as a percentage.

[2] A high value of W guards against noise in the estimate. However, if W is too high, the model will not respond fast enough to changes in the data stream. In practice, W may be selected by cross-validation.

confidence bound, which can vary over space and time. We will use a vector $c^o(x,t)$ to denote the degree of confidence required at node x at time t.

We further assume that nodes are sampled in cycles, each of length t_{cycle}. The assumption is that t_{cycle} is the maximum temporal resolution required by the application. A sensor is queried at most once during a cycle, which is assumed to be long enough to allow all nodes to be queried. Accordingly, we also view the confidence requirement as a set of discrete constraints that must be met at every sensor during every cycle.

In order to enable adaptive data acquisition, we assume that a mechanism to selectively activate sensor nodes is available. This might be accomplished, for example, by using a wake-up radio similar to what is available in the PicoRadio nodes [22].[3]

PicoRadio nodes can be woken up individually without causing neighboring nodes to awaken, and without incurring energy costs for preamble processing. The wake-up beacon is modulated by the node-id, which can be recovered using relatively simple and low-energy front-end hardware, without awakening the complex backend. The availability of a selective wake up mechanism allows us to put nodes to sleep whenever they are not engaged in sensing activity.

The overall sampling algorithm (Section 4.3) can now be described. During each cycle, node S evaluates the model confidence (Section 4.1) at each node. Additional readings need to be taken only if the required confidence bound is not met. An active learning inspired heuristic (Section 4.4) is used to select the most valuable sensor node, which is activated to acquire a reading, and the model is then incrementally retrained (Section 4.2) with the acquired value. This continues until the confidence requirement is met or all nodes are exhausted. The remainder of this section is devoted to describing the model and the selection heuristic in detail.

4.1 Modeling Sensor Data using Gaussian Process Regression

We begin by identifying key characteristics that our framework requires in the model. We then present Gaussian process regression and discuss how it can be used to satisfy these requirements. We identify three basic requirements:

1) The model must be generally applicable, *i.e.,* it should not assume linearity or other forms of special structure in data. It should also provide probabilistic estimates of the modeled data field. Gaussian process regression, as we show in this section, meets both these requirements.

2) It should be possible to train the model *incrementally* with new data, *i.e.,* without having to redo computation for previously seen data from scratch.

3) Efficient active sensor selection requires that it should be possible to efficiently estimate the value or informativeness of a new data point according to some chosen criterion. Moreover, the estimate must be obtained without knowledge of the exact data value being considered.

We now present Gaussian process regression. Assume that we are given N sensor readings of the form $y(x,t), x \in [1,Q]$ and $t \in [1,P]$, where x is the sensor and t is the time at which the reading was taken. The number of available sensors is denoted above by Q and the number of cycles over which data was collected by P. The total number of readings N is therefore upper bounded by the value $Q \times P$ which corresponds to an always-on scenario. In our approach, N is typically much smaller, since only a subset of sensors is queried in each time cycle. Then, in a Gaussian process model, we assume that $\mathbf{y}$ is a collection of random variables, $\mathbf{y} = (y(\mathbf{x}_1), y(\mathbf{x}_2), \dots)$, which have a joint Gaussian distribution

$$P(\mathbf{y}|\mathbf{C}, \mathbf{x_i}) = \frac{1}{Z} exp(-\frac{1}{2}(\mathbf{y}-\mu)^T \mathbf{C}^{-1}(\mathbf{y}-\mu)) \qquad (2)$$

for any set of inputs $\{\mathbf{x}_i\}$, where $\mathbf{x}_i$ is a tuple of coordinates, $\mathbf{x}_i = \langle x, t\rangle$ which identifies the sensor and time at which the reading was taken. $\mathbf{C}$ is the covariance matrix defined by a covariance function $C(\mathbf{x}_n, \mathbf{x}_m; \Theta)$, parametrized by a set of parameters Θ, and μ is the mean function. Given new coordinates $\tilde{\mathbf{x}}$, the model will make a prediction y that is Gaussian-distributed with a predictive mean $\hat{y}$ and posterior variance σ_y.

$$\begin{aligned} \hat{y}(\tilde{\mathbf{x}}) &= \mathbf{k}(\tilde{\mathbf{x}})\mathbf{C}_N^{-1}\mathbf{y} \\ \sigma_y^2(\tilde{\mathbf{x}}) &= C(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - \mathbf{k}(\tilde{\mathbf{x}})\mathbf{C}_N^{-1}\mathbf{k}()\tilde{\mathbf{x}} \end{aligned} \qquad (3)$$

where $\mathbf{k}(\tilde{\mathbf{x}}) = (C(\mathbf{x}_1, \tilde{\mathbf{x}}), \dots, C(\mathbf{x}_N, \tilde{\mathbf{x}}))$ is the covariance between the training data and $\tilde{\mathbf{x}}$, and $\mathbf{C}_N$ is the $N \times N$ covariance matrix of the training data points given the covariance function C. For a Gaussian predictive distribution, bounding the probability of model error exceeding a given value is equivalent to bounding the variance. With this in mind, in the rest of the paper, a minimum confidence requirement is sometimes called a maximum variance requirement.

If the covariance function is suitably selected to reflect the actual covariance structure of the data, Gaussian processes can be used to obtain probabilistic estimates of a wide variety of spatial data. Clearly, therefore, Gaussian process regression models meet the first of our aforementioned conditions.

4.2 Sparse Approximation for Gaussian Process Regression

Unfortunately, as defined above, Gaussian process regression does not meet the second requirement mentioned in Section 4.1, since training a Gaussian process is not an incremental process. Each time the data set changes, the entire model must be relearned. To mitigate this problem, we use a *sparse online* approximation to Gaussian processes [25].

The approximation gives a model which can be trained incrementally and is sparse, *i.e.*, the final representation of the model is much smaller than the training dataset. The origin of sparsity can be understood by recognizing that a positive-definite covariance function $C(\mathbf{x}_1, \mathbf{x}_2)$ might be viewed as the inner product of so-called feature vectors $\Phi(\mathbf{x}_1)$ and $\Phi(\mathbf{x}_2)$. Feature vector $\Phi(\mathbf{x})$ is a function that maps input $\mathbf{x}$ to a higher-dimensional feature space. The covariance matrix C_N can thus be thought of as expressed using dot-products of feature vectors.

Sparse approximations to Gaussian processes rely on identifying a *basis* set that spans the feature vector space. The input vectors $\{\mathbf{x}_1, \mathbf{x}_2 \dots, \mathbf{x}_M\}$ corresponding to the basis set are known as *basis vectors*. Given the basis set, the entire model, including the predictive Equations (3), can be expressed in terms of the basis vectors, as follows

$$\begin{aligned} \hat{y}(\tilde{\mathbf{x}}) &= \mathbf{W}^T\mathbf{k}(\tilde{\mathbf{x}}) \\ \sigma_y^2(\tilde{\mathbf{x}}) &= C(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - \mathbf{k}(\tilde{\mathbf{x}})\mathbf{C}_M^{-1}\mathbf{k}(\tilde{\mathbf{x}}) \end{aligned} \qquad (4)$$

where $\mathbf{k}(\tilde{\mathbf{x}}) = (C(\mathbf{x}_1, \tilde{\mathbf{x}}), \dots, C(\mathbf{x}_M, \tilde{\mathbf{x}}))$ is the covariance between the basis vectors and $\tilde{\mathbf{x}}$, and $\mathbf{C}_M$ is the $M \times M$ covariance matrix of the basis vectors given the covariance function C. $\mathbf{W}$ is a vector of weights that must be estimated during learning. Note that the difference between Equations (3) and (4) is very small. Ef-

[3] Other notable wake up techniques were given in STEM [23] and recently in [24].

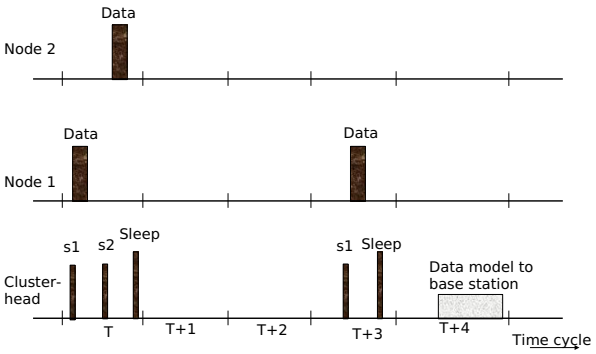


Figure 2. Example communication schedule in a cluster with three sensor nodes

fectively, we have used the reduced covariance matrix $\mathbf{C}_M$ instead of the full matrix $\mathbf{C}_N$ and have introduced the vector of weights $\mathbf{W}$. We refer the reader to [25] for further analysis of sparse online approximation and limit the discussion here to its consequences for our framework, which are two-fold:

1) Sparse approximation allows incremental training and *detraining* of the model. Since we only want the model to reflect data seen during a sliding window, old data must be discarded as new data arrive, a process we call detraining. With a sparse approximation, detraining is equivalent to deleting some basis vectors from the model. Adding a new point to the model or removing an old basis vector are both $O(M^2)$ operations, where M is the size of the basis set. Moreover, the *real* time requirement can be bounded by specifying the maximum number of basis vectors that the model is allowed to have. In comparison, training a full Gaussian model requires $O(N^3)$ operations.

2) Model sparsity also allows the model to be used as a *compression* device. This leads to a natural extension of the data acquisition framework to clustered network topologies such as that proposed for LEACH [8]. Instead of maintaining a centralized model of the entire network at S, cluster-level models are maintained at each cluster-head. Children nodes are sampled by the cluster-head using the sensor selection policy described in Section 4.4. Every W time units, the entire model (basis vectors and the weights) are reported to the base station.

A hypothetical example communication schedule is depicted in Fig. 2. The example scenario is based on a cluster of three sensor nodes, one of which is acting as the cluster-head. The cluster-head maintains a sparse Gaussian process model, with a history of length five. Fig. 2 shows five sampling cycles $T, T+1, \ldots, T+4$. In the first cycle, the cluster-head incrementally samples Nodes 1 and 2. The sample requests are shown as s1 and s2 in the figure. The samples lead to a sufficient increase in confidence for cycle T as well as the next two cycles. We assume that two cycles without sampling is a long interval, so that it is energy-efficient to switch off the node front-end.[4] Children nodes wake up, *i.e.*, switch their front-end receivers on, in cycle $T + 3$, and Node 1 is sampled before the network is put to sleep again. In cycle $T+4$, the cluster-head transmits its data model to the base station.

[4]Switching off all nodes in the cluster completely requires that the sleep signal be processed by each node. Therefore, it may not always be more energy-efficient than just leaving the low energy front-end hardware awake. The decision will depend on front-end and backend power consumption, as well as on the total sleep time.

4.3 The Data Acquisition Algorithm

We are now in a position to present the overall adaptive data acquisition algorithm (Algorithm 1) in detail. The algorithm might be run on the base station S, or on a cluster-head node. Next, we describe the actions of node S during a given time cycle τ. Since model f is maintained only for data received during a sliding window W, at the beginning of each cycle, we remove readings that were not received during W from f. This is accomplished by a call to subroutine *RemoveOldBasisVectors* in line 1 of the algorithm. Line 2 is a check to see if the pruned model's confidence c_f satisfies user requirements c^o. If the requirement is met, the algorithm returns and node S waits for the next cycle. Otherwise, in line 6, we incrementally select the best sensor $\tilde{x}$ and incorporate it into the model until there is no location $x \in X$ at which model confidence does not satisfy the minimum requirement. Sensor $\tilde{x}$ is selected using the $ActiveSensorSelection$ subroutine in line 7, following which it is removed from the candidate set of sensors $\tilde{X}$. The removal ensures that a node is sampled at most once during a cycle. Sensor $\tilde{x}$ is then activated and model f is incrementally trained with the newly acquired reading $y(\tilde{x}, \tau)$. Next, we present our procedure to select the best sensor $\tilde{x}$.

Algorithm 1: Overall Adaptive Data Acquisition Algorithm

input: Current cycle τ, model f, confidence bound $c^o(x,t)$, node set X, history length W

1 $f = \text{RemoveOldBasisVectors}(f, W)$;
2 **if** $c_f(x,\tau) \geq c^o(x,\tau)\ \forall x \in X$ **then**
3 return;
4 **else**
5 $\tilde{X} = X$;
6 **while** $\exists x \in X, c_f(x,\tau) < c^o(x,\tau)$ **do**
7 $\tilde{x} = \text{ActiveSensorSelection}(f, \tilde{X}, X)$;
8 $\tilde{X} = \tilde{X} \setminus \tilde{x}$;
9 $f = \text{TrainWithNewPoint}(y(\tilde{x}, \tau))$;
10 **end**
11 **end**

4.4 Active Sensor Selection

During a sensing cycle t, if model confidence c_f does not meet our confidence requirement c^o, the current model f must be incrementally updated, until the confidence requirement is met. In this section, we concern ourselves with designing a criterion to select the best sensor $\tilde{x}$ that should be incorporated in the model at each step.

Maximizing confidence for Gaussian-distributed predictive models is equivalent to minimizing variance. This leads to a straightforward and popular heuristic, which is to always select the sensor x which shows the greatest current variance $\sigma_f(x,t)$ [26], thereby hoping to bound the variance. This simple criterion, however, may lead to suboptimal choices in some situations since it does not directly optimize the desired objective, which is to meet specified variance bounds on all sensors, rather than merely minimizing the maximum variance. As an example scenario, consider two sensors a and b such that the current variance at both sensors is equal, *i.e.*, $\sigma_y(a,t) = \sigma_y(b,t)$. However, b is in a region of uniformly high variance, whereas sensor a is displaying much higher variance than its neighbors. This would typically indicate that the neighborhood around b has not been sampled for some time. On the other hand, the isolated peak of variance at a probably indi-

cates noise or sensor failure. In such a case, sampling b might be a better choice because it would reduce variance over the entire neighborhood of b. Reading the value from a on the other hand may not provide very useful information. Unfortunately, the maximum variance criterion does not distinguish between a and b.

An alternative selection criterion, which directly optimizes the variance (and for non-biased models, the mean squared error), was suggested in [1].[5] One estimates the change in variance that reading sensor $\tilde{x}$ would cause over the entire region of interest, which is typically the entire set of nodes. We observed, however, that it is possible to further focus sampling efforts in order to improve accuracy, by designing a distribution of interest over the network. This is, in our view, a major advantage of using such a heuristic. We will come back to our design of the interest distribution later in this section. Thus, if $\{\varepsilon\}$ is the set of sensors we are interested in, and $\tilde{x}$ is a sensor whose utility we wish to evaluate, we estimate $\Delta\sigma_{y(\varepsilon)}(\tilde{x})$ and take an average (possibly weighted) of $\Delta\sigma_{y(\varepsilon)}(\tilde{x})$ over $\{\varepsilon\}$. $\tilde{x}$ is selected to minimize this average. Then, a closed-form approximation to $\Delta\sigma_{y(\varepsilon)}(\tilde{x})$ can be obtained.

$$\Delta\sigma_{y(\varepsilon)}(\tilde{x}) = \frac{\left(\mathbf{k}_M \mathbf{C}_M^{-1}\mathbf{m} - C(\tilde{\mathbf{x}}, \langle\varepsilon, t\rangle)\right)^2}{C(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}) - \mathbf{m}^T \mathbf{C}_M^{-1}\mathbf{m}} \tag{5}$$

where $\tilde{\mathbf{x}} = \langle\tilde{x}, t\rangle$ and $\mathbf{k}_M$ and $\mathbf{m}$ are vectors of covariances and $\mathbf{C}_M$ is the covariance matrix in just the basis vectors. $\mathbf{k}_M$ and $\mathbf{m}$ are defined as $\mathbf{k}_M = [C(\mathbf{x}_1, \langle\varepsilon, t\rangle), \ldots, C(\mathbf{x}_M, \langle\varepsilon, t\rangle)] \in \mathbb{R}^M$ and $\mathbf{m} = [C(\mathbf{x}_1, \tilde{\mathbf{x}}), \ldots, C(\mathbf{x}_M, \tilde{\mathbf{x}})] \in \mathbb{R}^M$. The approximation was obtained by neglecting the sparse approximation we introduced in Section 4.2 and instead modeling the scenario as a full Gaussian process. Estimating $\Delta\sigma_{y(\varepsilon)}(\tilde{x})$ for the sparse approximation would require estimating the value $y(\tilde{x}, t)$ and retraining the model with it. Since our interest in this section is only in an approximate, but efficient, selection heuristic, this is a useful compromise. $\tilde{x}$ can be derived from:

$$\tilde{x} = \arg\min_{\tilde{x}} E_{q(\varepsilon)}\left[\Delta\sigma_{y(\varepsilon)}(\tilde{x})\right],\ \tilde{x} \in X \tag{6}$$

where $q(\varepsilon)$ is a probability distribution, using which we can express a different degree of interest in each sensor location. This ability to design $q(\varepsilon)$ can be used in various ways, *e.g.*, to balance energy consumption across the network or increase sensing accuracy. In this paper, we exploit $q(\varepsilon)$ to focus interest on sensors that have seen unpredictable changes. While we do not have a direct measure of unpredictability, an approximate measure can be obtained by comparing actual readings at a sensor with the value predicted by the model *before* the reading is incorporated into the model. To this end, we store the last few relative error values observed at each sensor ε. Relative error $\delta(\varepsilon, t)$ is obtained by dividing the squared error by the square of the observed sensor value and then taking the square root:

$$\delta(\varepsilon, t) = \sqrt{\frac{\left(\hat{y}(\varepsilon, t) - y(\varepsilon, t)\right)^2}{y^2(\varepsilon, t)}} \tag{7}$$

If the last W_h relative error values are recorded for each sensor, the weight $\hat{q}(\varepsilon)$ for each sensor is calculated as follows.[6] The probability distribution $q(\varepsilon)$ is obtained by normalizing $\hat{q}(\varepsilon)$.

$$\hat{q}(\varepsilon) = \sum_{j}^{W_h} \frac{\delta(\varepsilon, t-j)}{j} \tag{8}$$

[5] Other heuristics have been suggested in [19, 20].

[6] A similar metric is used in [16].

We have thus obtained a selection heuristic that is driven both by model variance and observed error.

5 Experimental Results

Since we did not have access to an actual sensor network deployment, we tested our framework on simulated data. A data field Z was generated using a Gaussian random field simulation, with the following covariance function, adapted from [27].

$$E[Z(x_1, y_1, t_1), Z(x_2, y_2, t_2)] = e^{\left(\alpha\left((x_1-x_2)^2+(y_1-y_2)^2\right)+\beta(t_1-t_2)^2\right)^\kappa} \tag{9}$$

where α and β are correlation coefficients along the space and time axes, respectively, and x, y and t are space and time coordinates. Results are reported for the values $\alpha = 0.5$, $\beta = 0.05$ and $\kappa = 1$, which represent a field with reasonable isotropic spatial correlation but little temporal correlation. The simulation was conducted over a 10m $\times$ 10m grid for 100 time units. We used the RandomFields [28] package for the simulation. The field was measured by placing a sensor at the center of every grid cell.

We constructed a sparse Gaussian Process Regression model f^S of the entire field at S with history length $W = 5$ time units. The model was trained for W time units and used to guide sensor selection until the data were exhausted. We report total network energy consumption and the estimation error for different confidence requirements. Energy consumption was measured using the LEACH extension for NS2 [29, 30].

The data acquisition framework was implemented in MATLAB. We used an exponential covariance function (kernel) of the form

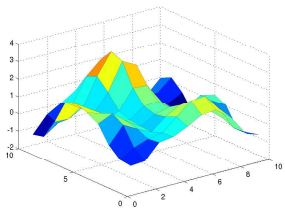
$$C(x_1, t_1, x_2, t_2) = e^{\left(w_1(x_1-x_2)^2+w_2(t_1-t_2)^2\right)} \tag{10}$$

where, as before, x and t are the sensor index and time, respectively. The model, thus, does not make use of location information. It is possible that accuracy can be improved further by using a location-aware kernel as the covariance function. w_1 and w_2 are hyperparameters that were learnt during the initialization phase by evidence maximization, as suggested in [31]. Once learnt, the hyperparameters were not changed during the experiment. The model was configured to have at most 50 basis vectors. The maximum limit was not met in our experiments.

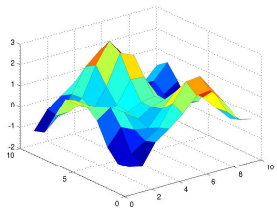
We performed two sets of experiments, on a one-hop network, where each node communicated directly with the base station, and on a clustered topology, respectively. In each case, a base station S was placed at the center of the field. Experiments were performed using two uniform maximum variance thresholds, $\sigma = 0.2$ and $\sigma = 0.1$. Data were acquired during each cycle, until the variance at every node was less than the stipulated maximum. We report the total network energy consumption and the observed average test mean square error.

For the second set of experiments, with a clustered topology, five clusters were obtained using the static clustering algorithm (LEACH-C) [29]. A model was maintained at each cluster-head. The maximum number of basis vectors allowed was set to one-tenth the maximum size of the training set, *i.e.*, $Q * W/10$, where Q denotes the number of nodes in the cluster and W the history length.

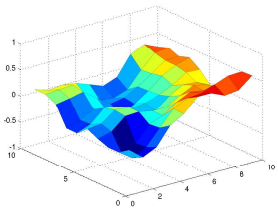
In Fig. 3, a snapshot of the evolving estimate is shown. The snapshot was taken at the 37th cycle during the simulation of the one-hop network with $\sigma = 0.1$ and shows the original data filed, the estimate at the base station and the error (estimated value $-$ original value).



(a) Original data field



(b) Estimated data field



(c) Error

Figure 3. Original and estimated data fields

In Table 1, we report the estimation error and network energy consumption for all cases. Root mean squared error, averaged over all cycles, is reported. The baselines correspond to every reading being reported to the base station and thus show zero error. Energy estimates were obtained using [29]. In addition to the energy values listed in [29], we assumed that in the idle mode, the node consumes 100μW of power. In the clustered networks, the cluster-heads communicated compressed data to the base station once every $W = 5$ cycles. Since in [29], no implementation for the compression algorithm used at cluster-head nodes is provided, we assumed that it is at least as efficient as the model used in our work. Thus, we assumed, that the energy spent in transmitting data from cluster-head nodes to the base station is equal in both cases. It can be seen from Table 1 that network energy can be reduced significantly by using our framework while keeping distortion within reasonable limits. The clustered topology shows somewhat higher error than the one-hop network. This is, in all probability, due to the models being trained over much smaller regions of the field. Reduced training area also adversely affects the energy consumption as more, possibly correlated, nodes are sampled across clusters. Another possible source of error is that the choice of the covariance function, Equation (10), was made without considering the actual covariance model. Performance can probably be improved by using a better matched covariance function.

Table 1. Estimation error and network energy

Topology	Base	Active data acquisition			
	Energy	σ	Error	Energy(mJ)	Savings
One-hop	39.1	0.1	6%	16.7	57%
		0.2	7%	11.8	69%
Clustered	20.0	0.1	8%	12.1	40%
		0.2	10%	9.3	55%

6 Conclusions

In this work, we presented a model-driven data acquisition framework that can be used to reduce energy consumption for networks measuring scalar fields. We presented a modeling methodology that can transparently exploit spatial and temporal correlations. A heuristic to actively select sensor readings was also presented. An important component of the selection heuristic is the interest distribution, which was presented in Section 4.4. We believe that a suitably-designed interest distribution can be used as a high-level policy to obtain different network goals, for example, to focus sampling effort on regions with unpredictable data, or balance energy consumption equally over the network, *etc*. We will explore this as part of future research.

References

[1] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *J. Artificial Intelligence Research*, vol. 4, pp. 129–145, Mar. 1995.

[2] D. Angluin, "Queries and concept learning," *Machine Learning*, vol. 2, no. 4, pp. 319–342, 1988.

[3] C. Olston, B. T. Loo, and J. Widom, "Adaptive precision setting for cached approximate values," *SIGMOD Rec.*, vol. 30, no. 2, pp. 355–366, June 2001.

[4] A. Jain, E. Y. Chang, and Y.-F. Wang, "Adaptive stream resource management using Kalman filters," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, June 2004, pp. 11–22.

[5] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, June 2003, pp. 563–574.

[6] I. Lazaridis, Q. Han, X. Yu, S. Mehrotra, N. Venkatasubramanian, D. V. Kalashnikov, and W. Yang, "QUASAR: Quality aware sensing architecture," *SIGMOD Rec.*, vol. 33, no. 1, pp. 26–31, Mar. 2004.

[7] C. Olston and J. Widom, "Best-effort cache synchronization with source cooperation," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2002, pp. 73–84.

[8] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocols for wireless microsensor networks," in *Proc. Hawaii Int. Conf. System Sciences*, vol. 8, Jan. 2000, pp. 1–10.

[9] D. Ganesan, D. Estrin, and J. Heidemann, "DIMENSIONS: Why do we need a new data handling architecture for sensor networks?" in *Proc. ACM Wkshp. Hot Topics in Networks*, Oct. 2002, pp. 143–148.

[10] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The impact of data aggregation in wireless sensor networks," in *Proc. Int. Conf. Distributed Computing Systems*, July 2002, pp. 575–578.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: A Tiny AGgregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, 2002.

[12] A. Boulis, S. Ganeriwal, and M. Srivastava, "Aggregation in sensor networks: An energy-accuracy trade-off," *Elsevier Ad Hoc Networks Journal (Special Issue, Sensor Network Protocols and Applications)*, vol. 1, no. 2-3, pp. 317–331, 2003.

[13] A. Scaglione and S. Servetto, "On the interdependence of routing and data compression in multi-hop sensor networks," *Wireless Networks*, vol. 11, no. 1-2, pp. 149–160, Jan. 2005.

[14] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong, "Model-driven data acquisition in sensor networks," in *Proc. Int. Conf. Very Large Scale Databases*, Sept. 2004, pp. 588–599.

[15] A. Deshpande, C. Guestrin, and S. R. Madden, "Using probabilistic models for data management in acquisitional environments," in *Proc. Conf. Innovative Data Systems Research*, Jan. 2005, pp. 26 – 38.

[16] A. Jain and E. Y. Chang, "Adaptive sampling for sensor networks," in *Proc. Wkshp. Data Manangement for Sensor Networks*, Aug. 2004, pp. 10–16.

[17] A. D. Marbini and L. E. Sacks, "Adaptive sampling mechanisms for sensor networks," in *Proc. London Communications Symp*, Sept. 2003.

[18] R. Willett, A. Martin, and R. Nowak, "Backcasting: Adaptive sampling for sensor networks," in *Proc. Int. Symp. Information Processing in Sensor Networks*, Apr. 2004, pp. 124–133.

[19] N. Ramakrishnan, C. Baiely-Kellog, S. Tadepalli, and V. N. Pandey, "Gaussian processes for active data mining of spatial aggregates," in *Proc. SIAM Data Management*, Apr. 2005.

[20] C. Guestrin, A. Krause, and A. Singh, "Near optimal sensor placement in Gaussian processes," School of Computer Science, Carnegie Mellon University, Tech. Rep., 2005.

[21] M. J. McPhaden, "Tropical atmospheric ocean project," 2001. [Online]. Available: http://www.pmel.noaa.gov/tao/

[22] L. C. Zhong, R. Shah, C. Guo, and J. M. Rabaey, "An ultra-low power and distributed access protocol for broadband wireless sensor networks," in *Proc. IEEE Broadband Wireless Summit*, May 2001.

[23] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava, "Optimizing sensor networks in the energy-latency-density design space," *IEEE Trans. Mobile Computing*, vol. 1, no. 1, pp. 70–80, Jan.-Mar. 2002.

[24] L. Gu and J. A. Stankovic, "Radio-triggered wake-up for wireless sensor networks," *Real-Time Syst.*, vol. 29, no. 2-3, pp. 157–182, Mar. 2005.

[25] L. Csatò, "Gaussian Processes - Iterative Sparse Approximation," Ph.D. dissertation, Aston University, 2002.

[26] D. J. C. Mackay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 589–603, July 1992.

[27] R. Cristescu and M. Vetterli, "On the optimal density for real-time data gathering of spatio-temporal processes in sensor networks," in *Proc. Int. Symp. Information Processing in Sensor Networks*, Apr. 2005, pp. 159–164.

[28] "Simulation and analysis of random fields." [Online]. Available: http://pbil.univ-lyon1.fr/library/RandomFields/html/00Index.html

[29] "MIT μAMPS NS code extensions." [Online]. Available: http://www.ece.rochester.edu/research/wcng/code/index.htm

[30] "NS2 LEACH implementation." [Online]. Available: http://www.internetworkflow.com/resources/ns2leach.pdf

[31] D. J. C. Mackay, "Gaussian processes, Tutorial," *NIPS*, vol. 10, Dec. 1998.