Energy-efficient Self-adapting Online Linear Forecasting for Wireless Sensor Network Applications

Jai-Jin Lim and Kang G. Shin
Real-Time Computing Laboratory, Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI 48109–2122, USA
Email: {jjinlim,kgshin}@eecs.umich.edu

***Abstract*— New energy-efficient linear forecasting methods are proposed for various sensor network applications, including in-network data aggregation and mining. The proposed methods are designed to minimize the number of trend changes for a given application-specified forecast quality metric. They also self-adjust the model parameters, the slope and the intercept, based on the forecast errors observed via measurements. As a result, they incur $O(1)$ space and time overheads, a critical advantage for resource-limited wireless sensors. An extensive simulation study based on real-world and synthetic time-series data shows that the proposed methods reduce the number of trend changes by 20%∼50% over the existing well-known methods for a given forecast quality metric. That is, they are more predictive than the others with the same forecast quality metric.**

I. Introduction

Forecast or prediction is of fundamental importance to all sensor network applications and services, including network management, data aggregation and mining, object tracking, and environmental surveillance. Thus, it should be designed as a middleware component that is predictive, energy-efficient, and self-manageable: (1) being *predictive*, a forecast method should predict as many future measurements as possible with the required forecast quality; (2) being *energy-efficient*, which is a networked version of being predictive when the prediction logic is shared across networked sensors, a forecast method should reduce the number of measurements to be transmitted through the network for given forecast quality requirements; (3) being *self-manageable*, a forecast method should self-adjust the model parameters based on the forecast errors observed during actual measurements.

There are a wide variety of prediction algorithms available in the literature. For example, the Kalman and Extended Kalman Filters [1] (KF/EKF) based predictors have been widely used by many researchers. Selection of a prediction algorithm, however, should be based on the characteristics of application domain and problem at hand. For example, the authors of [2] use double exponential smoothing [3] to build a linear forecast method for predictive tracking of a user position and orientation. Their results showed that when compared against KF/EKF-based predictors, the constructed predictor runs much faster with equivalent prediction performance and simpler implementations.

In this paper, we focus our attention on linear forecast, as it is not only popular (due mainly to its simple implementation and intuitive prediction), but also it has been widely used to characterize the time-series data for many data mining and feature extraction algorithms [4]–[6], which work on a series of piece-wise linear representations (PLRs) extracted from the underlying time-series data.

Main contributions of this paper with respect to the linear forecast are as follows.

- We evaluate the performance of existing well-known linear forecast methods in terms of the number of trend changes for a given forecast quality requirement. Specifically, we analyze the applicability of existing methods in the application context that trades forecast accuracy for the lifetime of sensors.
- We propose new linear forecast methods that outperform the existing methods with respect to the performance metric used above, making themselves more energy-efficient or predictive than the others. Moreover, the proposed methods are designed for online use and self-adjust the model parameters of a linear trend based on the forecast errors observed via actual measurements

The paper is organized as follows. Section II discusses the related work. Section III describes the motivations behind our algorithms development and states the associated problem. Section IV presents our linear forecast methods, as well as existing linear forecast methods, to solve the problem. In Section V, we describe an extensive simulation study of both real-world time series data and synthetic data. Finally, the paper concludes with Section VI.

II. Related Work

Distributed in-network aggregation and approximate monitoring in sensor networks. To reduce the number of transmissions involved in a distributed in-network aggregation, the authors of [7] rely on an aggregation tree rooted at an aggregator. On the other hand, the authors of [8], [9] tackle the same problem in a different way called the *approximate monitoring*, by assuming a new class of error-tolerant applications that can tolerate some degree of errors in the aggregated results.

The authors of [10] extend the latter by defining a way of mapping the user-specified aggregation error bound to the per-

0-7803-9466-6/05/$20.00 ©2005 IEEE

group quality constraint for hierarchically-clustered networks with more than one level. Similar work [11] was also reported to dynamically redistribute the aggregation error bound based on local statistics through which the most beneficiary node from the redistribution is decided.

However, the aggregated result is affected lot by packet loss over a lossy wireless channel; so are all the above-mentioned approaches. To remedy this problem, the authors of [12] proposed a new robust aggregation tree construction algorithm against packet loss over a wireless channel, and designed energy-efficient computation for some classes of aggregation such as `SUM` and `AVG`. Delivery of the linear trend information across networked sensors in our approach is also susceptible to such loss. While the result in [12] can be applied to ensure the robust delivery of the linear trend information in the network, the solution guaranteeing robust trend delivery and recovery from any packet loss would be fully developed in future.

Data mining and piece-wise linear representation. The authors of [6] compare different time-series representations over many real-world data in terms of reconstruction accuracy. They concluded that there is little difference among all well-known approaches in terms of the root mean square error. They also claimed that the choice of an appropriate time-series representation should be made by other features like flexibility, incremental calculation, rather than approximate fidelity.

The authors of [4], [13], [14] studied the use of piece-wise linear representation as the underlying representation of time-series data to allow fast and accurate classification, clustering and relevance feedback. The authors of [5] undertook extensive review and empirical comparison of a variety of algorithms to construct a piece-wise linear representation from time-series data stored in data base systems.

They also proposed the on-line bottom up piece-wise linear segmentation technique, called SWAB (Sliding Window And Bottom-up), that produces high-quality approximations. If SWAB were used in the network, our online linear forecast method can be used as a means to replace the bottom half of the algorithm that produces initial linear segments.

III. Motivation and Problem

A. Motivation

Initially, our desire for an energy-efficient self-adapting linear forecast method is generated by the need to design a *unified in-network data aggregation and mining service*, which extends the in-network data aggregation service in such a way that data mining queries are issued and processed within the same network.

Other than the in-network data aggregation service [7], [15] to prolong the lifetime of sensor networks, it may need to develop an energy-efficient in-network data mining service. For example, the following networked version of data mining queries [16] can be issued and processed in the network. *(1) Distributed pattern identification:* Given a query time-series and a similarity measure, find the most similar or dissimilar time-series and any relevant areas in the network. *(2) Summarization:* Given a time-series of measurements, create and report its approximate but essential features. *(3) Anomaly detection:* Given a time-series and some model of normal behavior, find segments showing anomalies and any relevant areas in the network.

Note that both in-network aggregation service and in-network data mining service need to collect continuous measurements, typically at periodic intervals, about various *network states* such as the temperature of some region and position/orientation of moving targets. So, a natural question arises about the representation of the network state, whose data type is, in most cases, numeric in sensor networks.

Obviously, aggregation results like `AVG`, `SUM`, `MIN`, and `MAX`, hide key information on gradual trends or patterns in measurements. However, PLR, used by many data mining applications, not only holds such patterns, but can also reconstruct the original time-series within some error bound, which is then used to produce approximate values of various aggregation functions. Thus, PLR will be appropriate as a basic representation of the network state for a unified in-network aggregation and data mining service.

Other high-level time-series representations can also be used, but comparisons [6] of several time-series representations[1] over many real-world datasets show that there is little difference among these approaches in terms of the root mean square error. Then, as claimed in [6], the choice of an appropriate time-series representation should be made by other features like flexibility, incremental calculation, rather than approximation fidelity.

Using PLR as the network state representation has the following main advantages. First, it is intuitive and easy to calculate. Second, it is one of the most frequently-used representations that support various data mining operations [4]–[6], [17]. Third, it allows distributed incremental computation so that a new linear segment may be built from existing linear segments [4]–[6], [18], without resorting to the original data. Fourth, it can also be used to compute traditional aggregation queries as approximate results, with no delay, through the prediction.

The distributed incremental computation allows initial linear segments to be collected across the network, without collecting all the raw measurements to get initial linear segments. The thus-collected linear segments can later be used to build a new linear segment as needed[2] at resource-abundant sensors, or will be used to produce the approximate aggregation results as needed.

[1] Such representations include Discrete Fourier Transformation (DFT), Singular Value Decomposition (SVD), Discrete Wavelet Transformation (DWT), Piecewise Constant Approximation, Piecewise Aggregate Approximation (PAA), and Piecewise Linear Representation (PLR). They have been proposed in [5], [17] and references thereof.

[2] Much work has been reported in the literature, allowing a new linear segment to be built from existing linear segments [4]–[6], [18], without resorting to the original data. For example, the bottom-up algorithm or SWAB (Sliding Window And Bottom-up) in [5] (or others such as Theorems 3.2 and 3.3 in [18], merging operation in [4], or Theorems 1 and 2 in [6]) enables a new linear segment to be constructed from existing segments with some error metric. To best serve our purpose without any modification, the bottom-up algorithm or SWAB is appropriate.

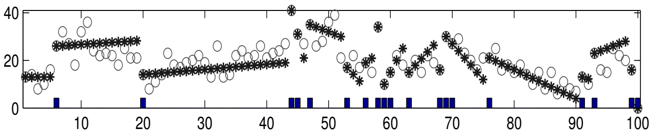


Fig. 1. Illustration of piece-wise linear segmentation with a maximum permissible forecast error $\varepsilon = 10$. Solid circles mean the forecast estimates and empty circles mean actual measurements. Bars at the bottom represent trend-change points.

B. Problem Statement

The problem we want to solve is formally stated as follows.

Problem 1: : Design an energy-efficient self-adaptive online linear forecast method: Given an infinite time-series $x_i, i = 1, 2, \ldots$ and an application-specified forecast quality threshold ε, design an energy-efficient self-adapting online linear forecast method that minimizes the number of trend changes under a given quality measure. It will produce a sequence of piece-wise linear segments, where each piece-wise linear fit to the underlying original time-series data $x_T, x_{T+1}, \ldots, x_{T+l_T-1}$ is represented by $\hat{x}_T(T+\tau)$, such that

$$\hat{x}_T(T+\tau) = \hat{a}_1(T) + \tau\hat{b}_2(T), \tau = 0, \ldots, l_T - 1$$
$$f(< x_T, \ldots, x_{T+l_T-1} >, < \hat{x}_T, \ldots, \hat{x}_{T+l_T-1} >) \leq \varepsilon$$

where $\hat{x}_{T+\tau}$ is an estimate for the actual measurement $x_{T+\tau}$ from the linear fit $\hat{x}_T(T+\tau)$ built at time T, f is the application-specified forecast quality measure governing piece-wise linear segmentation. Note that l_T, the length of the linear segment, is not a decision parameter, but automatically decided as a result of segmentation.

Figure 1 illustrates the application of a piece-wise linear forecast method to real-world data, with the maximum forecast error 10. As can be seen from the figure, the segmentation of actual time-series data is governed by the forecast quality metric ε. Thus, a linear segment can grow unbounded as long as it satisfies the forecast quality measure.

However, a thus-constructed linear segment may degenerate to a single point if it predicts badly or the underlying time-series show random fluctuations, which are unexpected outliers. Figure 1 shows this; some forecasts at the sampling interval between 40 and 65. Depending on the forecast quality measure, such random noises can be ignored with some loss of information about the original time-series data.

In general, the specification of the forecast quality measure depends largely on various application requirements. For instance, the segmentation problem can be defined to produce the best representation such that the maximum error for any segment does not exceed a user-specified threshold [5], [19], or the cumulative residual errors are bounded [20], or the dynamic confidence band can be defined [21].

Although these examples are not exhaustive, they are frequently addressed by many data mining and aggregation applications that deal with time-series data. Especially, the first two specifications are so widely used that we may incorporate them as our forecast quality measures. For ease of exposition, X and $\hat{X}$ denote two n vectors of the underlying time-series data and the estimated time-series by the corresponding linear fit. Each element of the two is denoted by $X(i)$ and $\hat{X}(i)$, respectively.

$\mathbf{L_\infty}$ metric. This quality metric, defined as $\max_{i=1}^{n} |X(i) - \hat{X}(i)|$, specifies that the maximum forecast error of any segment should not exceed a user-specified threshold. It is probably the most popular metric in various applications such as data-mining (see Table 5 in [5]) or data-archiving applications to compress time-series data [19]. A similar concept of bounded error-tolerance is also used in approximate monitoring and aggregation applications [8], [9]. Due to the same accuracy constraint, the L_∞ metric can serve both in-network data aggregation and in-network data mining services simultaneously.

Using L_∞ we have the segmentation governed by:

$$\begin{aligned} f(\underline{X}, \underline{\hat{X}}) &\triangleq L_\infty(\underline{X}, \underline{\hat{X}}) \\ &\triangleq \max_{i=1}^{n} |X(i) - \hat{X}(i)| \leq \varepsilon \end{aligned}$$

where ε is a user-specified threshold.

$\mathbf{C_\infty}$ metric. This quality metric, defined as $\sum_{i=1}^{n}(X(i) - \hat{X}(i))$, specifies that the cumulative residual errors of any segment should not exceed a user-specified threshold. This metric is in fact a CUSUM (cumulative sum) metric, one of the most frequently-used metrics for quality control in environmental and manufacturing process management [22]. This metric is even used in a continuous patient monitoring application in an intensive care unit [20]. These areas will become one of major sensor network applications, as explored by the CodeBlue project [23].

Using the C_∞ metric, which is typically controlled from both below and above, we have the segmentation governed by:

$$\begin{aligned} f(\underline{X}, \underline{\hat{X}}) &\triangleq |C_\infty(\underline{X}, \underline{\hat{X}})| \\ &\triangleq |\sum_{i=1}^{n}(X(i) - \hat{X}(i))| \leq \varepsilon \end{aligned}$$

where ε is a user-specified threshold.

Since the method is intended to be used as a middleware component of distributed wireless sensors, the solution should be energy-efficient, lightweight, adaptive, and online:

- *Energy-efficient:* It states that a solution should minimize the number of trend changes, i.e., the number of piece-wise linear segments, given the quality metric.
- *Lightweight:* It states that a solution should incur $O(1)$ space and time overheads in extracting a linear trend information.
- *Adaptive:* It states that a solution should self-adjust the model parameters, the slope and the intercept, of a linear trend according to the forecast errors observed.
- *Online:* It states that there should be no time lag between the availability of a linear trend fit and the underlying time-series data. This means that a new linear trend information is available as soon as the current linear trend violates the forecast quality measure.

IV. Design of Energy-efficient Self-adapting Online Linear Forecast

A. Preliminary

Least-Square-Error based Linear forecast method (LSEL). An intuitive approach to extracting linear trend information is to apply the linear regression method. Forecasting with the linear regression needs to identify the model parameters from past W measurements. Although the thus-constructed linear trend is guaranteed to minimize the sum of the squared errors over the past measurements, it is unclear if the thus-constructed linear trend can still produce a good predictive linear fit to future measurements. The constructed linear trend holds as long as it satisfies the quality constraint specified by $\mathbf{L}_\infty$ or $\mathbf{C}_\infty$ metrics. Otherwise, a new trend should be derived by re-applying the linear regression method.

A *W-period moving linear regression* with $O(W)$ space and time requirements is built as follows. Suppose the current linear trend is no longer valid at time $t \equiv t_e$. Then, past W measurements of $x_i, i \in [t_b, t_e], W \equiv t_e - t_b + 1$, will be used to extract the parameters of a new liner trend.

The intercept $\hat{a}_0$ and the slope $\hat{b}_2$ are calculated as follows [3], [18]:

$$\begin{aligned} \text{slope: } \hat{b}_2(t) &= \sum_{i=t_b}^{t_e} \left(\frac{i - \bar{t}}{\nu_t} \right) (x_i - \bar{x}) \\ \text{intercept: } \hat{a}_0(t) &= \bar{x} - \hat{b}_2 \bar{t} \end{aligned}$$

where $\bar{t} = \sum_{i=t_b}^{t_e} i/W, \bar{x} = \sum_{i=t_b}^{t_e} x_i/W, \nu_t = \sum_{i=t_b}^{t_e} (i - \bar{t})^2$. The intercept at the shifted origin to the time t, $\hat{a}_1(t)$, is given by $\hat{a}_0(t) + t\hat{b}_2(t)$.

Thus, the τ-step ahead linear forecast $\hat{x}_t(\tau)$ into future measurements after t is given by $\hat{a}_1(t) + \tau\hat{b}_2(t) = (\hat{a}_0(t) + t\hat{b}_2(t)) + \tau\hat{b}_2(t), \tau = 0, 1, \ldots$

Non-seasonal Holt-Winters Linear (NHWL) forecast. A commonly-used linear trend forecast algorithm with $O(1)$ space and time overheads is NHWL [24]. It uses exponential smoothing to extract the model parameters of a linear trend. It has been applied widely to detect aberrant network behavior [21], [25] for its simplicity. NHWL uses separate smoothing parameters, α, β $(0 < \alpha, \beta < 1)$, for the intercept and the slope updates at time t as follows:

$$\text{intercept: } \hat{a}_1(t) = \alpha x_t + (1 - \alpha) \left(\hat{a}_1(t-1) + \hat{b}_2(t-1) \right)$$

$$\text{slope: } \hat{b}_2(t) = \beta \left(\hat{a}_1(t) - \hat{a}_1(t-1) \right) + (1 - \beta)\hat{b}_2(t-1)$$

where the determination of smoothing parameters is left to the user or application.

Thus, the τ-step ahead linear forecast $\hat{x}_t(\tau)$ into future measurements after t is then given by $\hat{a}_1(t) + \tau\hat{b}_2(t), \tau = 0, 1, \ldots$

Double-Exponential-Smoothing based Linear (DESL) forecast. Another $O(1)$ space and time linear forecast algorithm that uses exponential smoothing is DESL [3]. It has two major differences from the above two algorithms. First, unlike NHWL, it has only one single smoothing parameter. Second, it is shown in [3] that DESL minimizes the weighted sum of the squared errors, unlike LSEL minimizing the sum of the squared errors.

Given a smoothing constant α, DESL requires that the following exponential smoothing statistics S_t and $S_t^{[2]}$ are updated at every sampling period [3]:

$$\begin{aligned} S_t &= \alpha x_t + (1 - \alpha) S_{t-1} \\ S_t^{[2]} &= \alpha S_t + (1 - \alpha) S_{t-1}^{[2]} \end{aligned}$$

where the notation $S_t^{[2]}$ implies double exponential smoothing, not the square of a single smoothed statistic. Then, the following relations are known [3]:

$$\begin{aligned} \text{intercept: } \hat{a}_1(t) &= 2S_t - S_t^{[2]} \\ \text{slope: } \hat{b}_2(t) &= \frac{\alpha}{1 - \alpha} (S_t - S_t^{[2]}) \end{aligned}$$

Equivalently [3]:

$$\begin{aligned} \text{slope: } \hat{b}_2(t) &= \alpha \left(S_t - S_{t-1} \right) + (1 - \alpha)\hat{b}_2(t-1) \\ \text{intercept: } \hat{a}_1(t) &= S_t + \frac{1 - \alpha}{\alpha} \hat{b}_2(t). \end{aligned}$$

Note that the slope is first computed in an equivalent form, which is then used to calculate the intercept.

Thus, the τ-step ahead linear forecast $\hat{x}_t(\tau)$ into future measurements after t is then given by $\hat{a}_1(t) + \tau\hat{b}_2(t), \tau = 0, 1, \ldots$

B. The Proposed Linear Forecast Methods

Directly Smoothed Slope based Linear (DSSL) forecast. DSSL is logical and intuitive, but is not based on sound criteria like the least square errors or the weighted least square errors.

Recall that in NHWL the estimate of the slope is simply the smoothed difference between two successive estimates of the intercept. Similarly, DESL estimates the slope by calculating the smoothed difference between two successive smoothed inputs. In some sense, the difference between two successive estimates in NHWL or DESL can be considered as an estimate for the instantaneous slope between two successive measurements in the sampling period.

DSSL extends this interval of instantaneous slope calculation from between two successive sampling measurements to between two successive linear segments. In other words, DSSL estimates the slope by anchoring a base to the intercept of the current linear trend. Specifically, suppose that the current linear trend was built at time t_o with the intercept $\hat{a}_1(t_o)$. Then, at any time t after t_o in the sampling period, a slope estimate s_t is defined by

$$s_t \triangleq \frac{1}{t - t_o} (x_t - \hat{a}_1(t_o)).$$

Then, DSSL applies the same update procedures for the slope and the intercept as in NHWL, except for using s_t rather than the successive difference of two intercept estimates:

$$\text{intercept: } \hat{a}_1(t) = \alpha x_t + (1-\alpha)\Big(\hat{a}_1(t-1) + \hat{b}_2(t-1)\Big)$$
$$\text{slope: } \hat{b}_2(t) = \beta s_t + (1-\beta)\hat{b}_2(t-1).$$

Thus, the τ-step ahead linear forecast $\hat{x}_t(\tau)$ into future measurements after t is then given by $\hat{a}_1(t) + \tau\hat{b}_2(t), \tau = 0, 1, \ldots$

Directly Averaged Slope based Linear (DASL) forecast. A common problem with any exponential smoothing system is that it is difficult to determine an appropriate smoothing parameter under which a system is expected to yield better performance. To circumvent such difficulty, DASL uses another incremental relationship using s_t

$$\begin{aligned}\text{intercept: } \hat{a}_1(t) &= x_t\\ \text{slope: } \hat{b}_2(t) &\triangleq \frac{1}{t-t_o}\sum_{i=t_o+1}^{t} s_i\\ &= \hat{b}_2(t-1) + \frac{1}{t-t_o}(s_t - \hat{b}_2(t-1)).\end{aligned}$$

Thus, the τ-step ahead linear forecast $\hat{x}_t(\tau)$ into future measurements after t is then given by $\hat{a}_1(t) + \tau\hat{b}_2(t), \tau = 0, 1, \ldots$

C. Intercept correction under the quality metric

All the algorithms described above are susceptible to violate the L_∞ metric constraint, especially when the new intercept estimate is either above or below the actual measurement by more than the specified L_∞ constraint ε. To avoid this problem, we set the current measurement x_t to the new intercept estimate, implying that a new linear trend always starts with the actual measurement that causes the violation of the specified forecast quality metric. Moreover, we adhere to the same policy even in the case of using the C_∞ metric in order to make algorithms simple.

V. Evaluation

We now evaluate the proposed linear forecast methods over various real-world and synthetic time-series data. Some of data used in our evaluation are obtained from the following sites.

- The SensorScope [26] which is an indoor wireless sensor network testbed at the EPFL (École Polytechnique Fédérale de Lausanne) campus. The network currently consists of around 20 TinyOS mica2 and mica2dot motes, equipped with a variety of sensor boards such as light, temperature, and acoustic sensors.
- The tsunami, i.e., tidal waves due to seismic activity, research program at PMEL (Pacific Marine Environmental Laboratory) of NOAA (National Oceanic and Atmospheric Administration) [27]. The PMEL tsunami research program seeks to mitigate tsunami hazards to Hawaii, California, Oregon, Washington and Alaska. The program focuses on improving tsunami warning and mitigation.

TABLE I
Statistical properties of samples

	mean	std	range	msd	stdsd
Temperature	553.44	19.37	159.00	0.9431	1.28
Light	359.08	418.56	1023.00	6.3778	45.64
Water-level	1.73	1.32	10.60	0.0077	0.06
Random-walk	254.53	254.37	1160.00	2.5075	1.44

In addition to the above, synthetic random walk time-series data are generated by the relation: $x_1 = 20, x_{t+1} = x_t + U(-5, 5)$.

A total of 100,000 temperature and light readings are collected from the SensorScope project during the period of Nov. 25 through Dec. 31, 2004. As for water-level samples, the archived measurement data [28] from several sites participating in the tsunami research program are used. These 100,000 samples per each category are then grouped into non-overlapping 200 sub-groups of size 500 samples each.

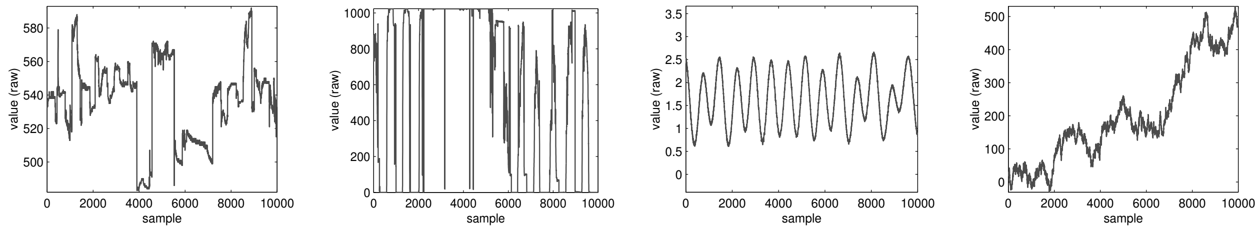
Figure 2 shows snapshots of 10,000 readings for both real-world data and synthetic data. Table I summarizes statistical properties of a total of 100,000 samples used in our evaluation. The statistics like `mean`, `std`, and `range` are defined as usual, representing the mean, the standard deviation, and the range in samples. The `msd` statistic, short for the mean successive difference, is defined as $\sum_{i=2}^{n} |x_i - x_{i-1}|/(n-1)$. Finally the `stdsd` statistic represents the standard deviation of the successive difference between samples.

Using these 500 samples per each group we evaluate our algorithm and get results on the performance metrics below. The results are then averaged from 200 independent runs.

- **Communication overhead**: the number of trend changes under the user-specified $\mathbf{L}_\infty$ or $\mathbf{C}_\infty$ metrics. This measure is calculated as a fraction of sampling points that trigger the linear trend change. By definition, this performance measure is directly related to the energy consumption in the communication, a major source of energy consumption in wireless sensor networks.
- **The mean absolute deviation**: the accuracy of the predicted value as compared to the actual value. Notice that any linear trend fit will be valid and equivalent to each other as long as it satisfies the user-specified $\mathbf{L}_\infty$ or $\mathbf{C}_\infty$ metrics. Nonetheless, a linear trend construction algorithm minimizing the mean absolute deviation is preferred in that it can produce a better linear fit to underlying time series data.

In plotting the performance results, we use the *relative performance* since we are interested in deciding which method is a better fit to the underlying time series. As for the communication overhead, all the performance results are normalized with that of NHWL, whereas the tolerance bound ε in unit of the mean absolute deviation is used for evaluating the relative performance of the mean absolute deviation per each method.

In addition, in specifying the smoothing parameters for forecast methods other than LSEL and DASL, we use the relationship between an exponential smoothing system with



(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 2. Snapshots of time series (raw) data used in our evaluation

a smoothing parameter α and a W-period moving window system. For instance, making the average age of data in both system equal gets the following result [3]: $\alpha = 2/(W+1)$.

A. Communication overhead

Figure 3 shows the relative communication overhead of all forecast methods, normalized with NHWL, under the L_∞ metric. The results are obtained with the parameters $W = 2$ (LSEL), $\alpha = 2/(W+1) = 0.67$ (DSEL, NHWL, DSSL) and $\beta = 0.67$ (NHWL, DSSL). The tolerance bound ε in the L_∞ metric is specified in unit of the integer times the mean absolute deviation (`msd`). We leave out any evaluation result for methods whose relative performance with respect to NHWL exceeds 150%. For instance, LSEL is not plotted in Figure 3(a) and Figure 3(c) since its relative performance is almost always beyond 150% in the two cases evaluated.

From the figure, we can draw the following conclusions. First, the forecast method based on the linear regression shows the worst performance in all the cases evaluated, challenging the adequacy of it as a good forecast method in our context. Second, NHWL and DESL, when the same single smoothing parameter is used, show almost the same performance due to their similarity in the slope estimation. Third, DSSL shows the best performance in almost all the cases evaluated, making itself promising for an energy-efficient linear forecast method in wireless sensor networks. It turns out that the proposed DSSL and DASL save the communication overhead by up to 50% as compared to NHWL or DESL. Fourth, DASL, without any hassle of determining smoothing parameter, performs as good as and even better than DSSL in a particular data set.

Similar results are also observed with the C_∞ metric, as shown in Figure 4. DSSL performs fairly well as compared to the best method in each data set, contributing up to about 20% energy savings as compared to NHWL. For some method, the overhead shown in both Figure 3 and Figure 4 appears to be increasing as the tolerance bound ε gets larger. However, it does not mean that actual absolute overhead increases proportionally. Instead, it just shows the relative performance among the forecast methods.

The evaluation of the communication overhead shows that the proposed DSSL outperforms the other methods in almost all data set evaluated, regardless of the quality metric, resulting in saving more energy up to 20% $\sim$ 50%. Notice that a very short-term memory or a way of weighing more recent measurements is used for the forecasting evaluation in both figures. However, a long-term memory does not overturn the conclusion drawn above. This will be shown later in the sensitivity analysis, where we perform the sensitivity of DSSL to different smoothing parameters – but the same conclusion will be drawn as can be seen later.

B. The mean absolute deviation

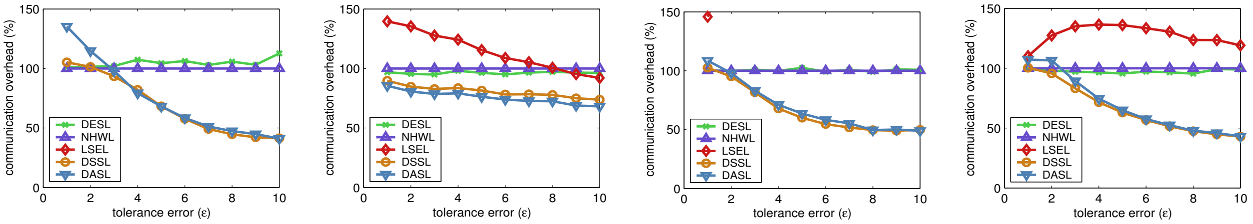
Any forecast method is equivalent as long as it satisfies the user-specified L_∞ or C_∞ metrics. Thus any method described in the paper will be used as a valid forecast method. However, it is still worth investigating how good a fit each method is to underlying time series. Figure 5 and Figure 6 show evaluation results, measuring the `msd` of each method with respect to the input tolerance bound ε. As can be seen in both figures, all the forecast methods show similar features. Under the L_∞ metric, the `msd`'s stay between 30%$\sim$40%, whereas the deviation gets small below 20% under the C_∞ metric. Such difference between under two forecast quality measures relates to the communication overhead. The more often a trend change occurs, the closer the forecast is the underlying time series and the smaller deviation.

C. Sensitivity to the smoothing parameters

Finally, we perform the sensitivity analysis of DSSL to the smoothing parameters α and β. Note that in the DSSL method α is used as the intercept smoothing parameter whereas β as the slope smoothing parameter. Figure 7 and Figure 8 show evaluation results under the L_∞ metric. Another evaluation was conducted for the C_∞ metric, but due to both the space limit and the similar features, its result is omitted in the paper. In general, DSSL turns out to be better than NHWL no matter what combination of the α and β smoothing parameters is chosen.

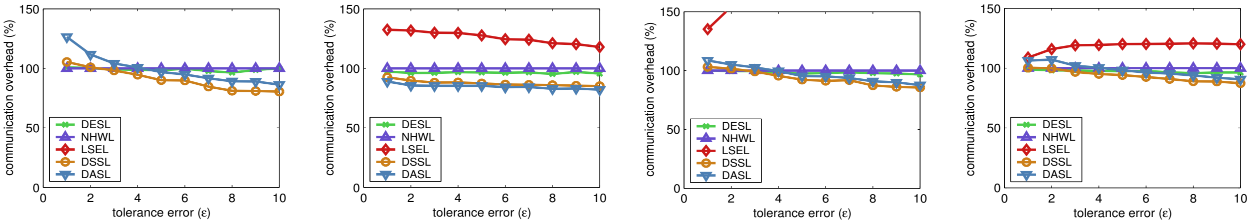
Specifically, when experimented with the same α and β, the result in Figure 7 shows that a short-term memory is sufficient to get a good forecast method in most cases, except for the Light readings. When experimented with the constant α and the varying β, which evaluates the sensitivity of the slope smoothing only, a short-term memory is better in all the cases. Recall that a short-term memory in DSSL or NHWL means that the forecast model is more responsive to recent forecast errors.

From both figures, it is concluded that DSSL with a short memory outperforms most of time any other methods investigated in the paper, in terms of the number of trend chanes or the number of linear segments produced. This gets us back to the results in Figure 3 or Figure 4. Since DASL is also



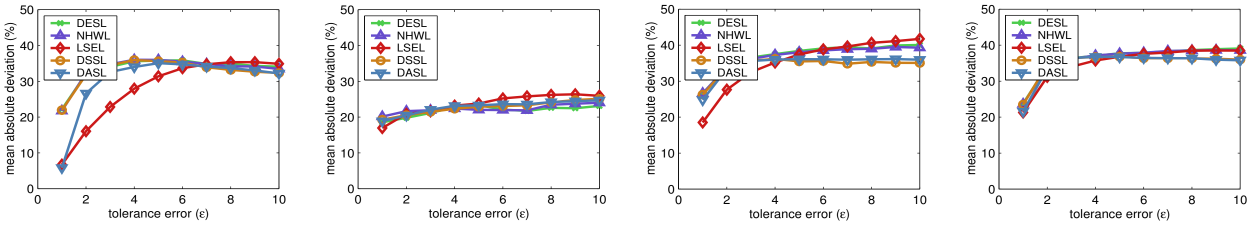
(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 3. Communication overhead (%, with respect to the NHWL) vs. the specified tolerance error ε (in unit of the integer times msd) under the L_∞ metric, with parameters of $W = 2$ (LSEL), $\alpha = 2/(W + 1) = 0.67$ (DSEL, NHWL, DSSL), and $\beta = 0.67$ (NHWL, DSSL).



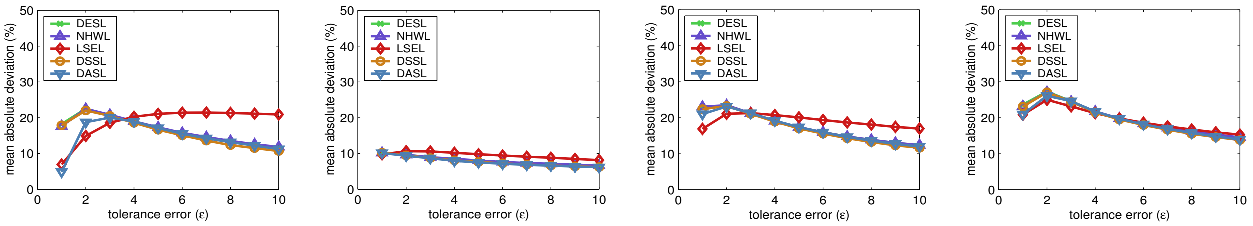
(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 4. Communication overhead (%, with respect to the NHWL) vs. the specified tolerance error ε (in unit of the integer times msd) under the C_∞ metric, with parameters of $W = 2$ (LSEL), $\alpha = 2/(W + 1) = 0.67$ (DSEL, NHWL, DSSL), and $\beta = 0.67$ (NHWL, DSSL).



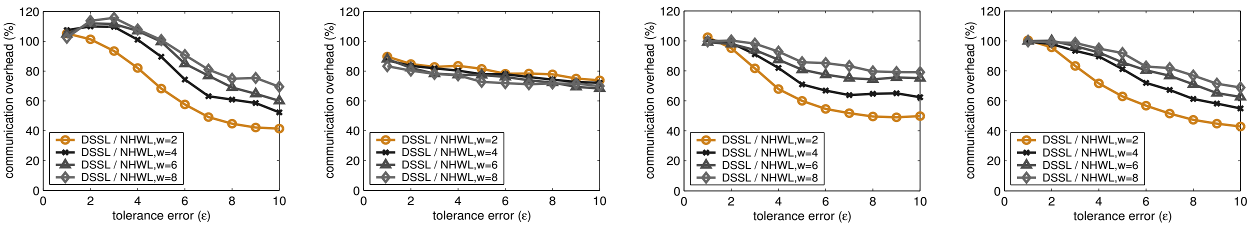
(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 5. The mean absolute deviation (%, with respect to the tolerance error) vs. the specified tolerance error ε (in unit of the integer times msd) under L_∞ metric with parameters of $W = 2$ (LSEL), $\alpha = 2/(W + 1) = 0.67$ (DSEL, NHWL, DSSL), and $\beta = 0.67$ (NHWL, DSSL).



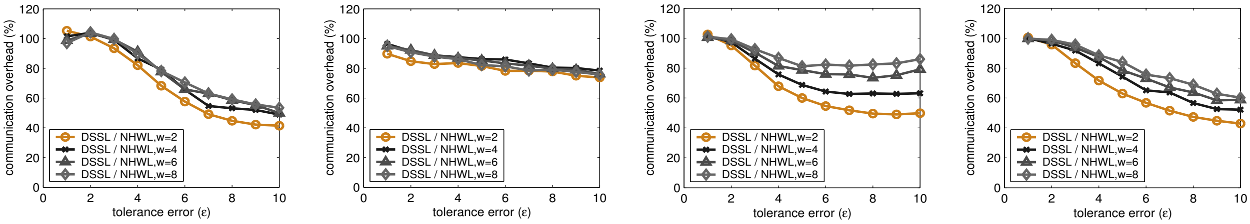
(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 6. The mean absolute deviation (%, with respect to the tolerance error) vs. the specified tolerance error ε (in unit of the integer times msd) under C_∞ metric with parameters of $W = 2$ (LSEL), $\alpha = 2/(W + 1) = 0.67$ (DESL, NHWL, DSSL), and $\beta = 0.67$ (NHWL, DSSL).



(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 7. The DSSL communication overhead relative to NHWL vs. the specified tolerance error ε (in unit of the integer times msd) under L_∞ metric with parameters of $W = 2, 4, 6, 8$ and $\alpha = \beta = 2/(W + 1) = 0.67, 0.40, 0.28, 0.22$.



(a) Temperature (b) Light (c) Water-level (d) Random-walk

Fig. 8. The DSSL communication overhead relative to NHWL vs. the specified tolerance error ε (in unit of the integer times msd) under L_∞ metric with parameters of $\alpha = 0.67, \beta = 2/(W+1) = 0.67, 0.40, 0.28, 0.22$.

performing as good as DSSL with a short memory, DASL can be chosen as an alternative to DSSL without worrying about the determination of smoothing parameters.

VI. Conclusion

In this paper, we proposed energy-efficient, self-adapting, online linear forecast methods for various wireless sensor network applications. The proposed methods self-adjust the model parameters using the forecast errors observed via experimental measurements, and minimize the number of trend changes for a given forecast quality metric. Moreover, the computation involved in the self-adjustment incurs $O(1)$ space and time overheads, which is critically important for resource-limited wireless sensors.

An extensive simulation study based on both real-world and synthetic random walk time-series data shows that the proposed methods reduce the number of trend changes by 20$\sim$50% compared to the other existing methods. It is also empirically shown that the performance of forecasting based on linear regression is not suitable both qualitatively and quantitatively, but forecasting based on smoothing techniques can be general solutions for various sensor network applications.

We plan to augment the TinyDB, a de-facto standard data aggregation application in the TinyOS-based sensor networks, with our forecast capability. The augmented TinyDB is then expected to support both in-network aggregation and data mining with a more unified framework. We will also design a robust linear forecast method to combat measurement noises. Finally, we need to develop a new definition of robust forecast quality metric for various sensor network applications, especially for data aggregation and mining.

References

[1] P. Zarchan and H. Musoff, "Fundamentals of kalman filtering: A practical approach," AIAA (American Institute of Aeronautics and Astronautics, 2000.

[2] J. LaViola, "Double exponential smoothing: an alternative to kalman filter-based predictive tracking," in *International Immersive projection technologies workshop*, 2003.

[3] D. C. Montgomery, L. A. Johnson, and J. S. Gardiner, "Forecasting and time series analysis," 2nd Edition, McGraw-Hill Book Company, 1990.

[4] E. Keogh and M. Pazzani, "An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback," in *Fourth International Conference on Knowledge Discovery and Data Mining (KDD'98).*

[5] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani, "An online algorithm for segmenting time series," in *Proceedings of the IEEE International Conference on Data Mining (ICDM'01).*

[6] T. P. Themis, "Online amnesic approximation of streaming time series," in *International Conference on Data Engineering*, 2004.

[7] S. Maden, M. Franklin, J. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," in *Symposium on Operating System Design and Implementation*, Dec. 2002.

[8] C. Olston, B. T. Loo, and J. Widom, "Adaptive precision setting for cached approximate values," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data (SIGMOD '01).*

[9] C. Olston, J. Jiang, and J. Widom, "Adaptive filters for continuous queries over distributed data streams," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data (SIGMOD '03).*

[10] X. Yu, S. Mehrotra, N. Venkatasubramanian, and W. Yang, "Approximate monitoring in wireless sensor networks," http://www.ics.uci.edu/ xyu/pub/aggXYU.pdf.

[11] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, "Hierarchical in-network data aggregation with quality guarantees," in *Proceedings of the 9th International Conference on Extending DataBase Technology (EDBT)*, March 2004.

[12] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," in *Proceedings of the IEEE SNPA'03*, 2003.

[13] E. Keogh and P. Smyth, "A probabilistic approach to fast pattern matching in time series databases," in *Third International Conference on Knowledge Discovery and Data Mining*, 1997.

[14] E. J. Keogh, "Fast similarity search in the presence of longitudinal scaling in time series databases," in *ICTAI*, 1997, pp. 578–584.

[15] Y. Yao and J. Gehrke, "Query processing in sensor networks," in *1st Biennial Conference Inovative Data System Research (CIDR).* ACM Press, 2003.

[16] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery (DMKD '03).*

[17] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Locally adaptive dimensionality reduction for indexing large time series databases," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data.*

[18] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, "Multi-dimensional regresssion analsis of time-series data streams," in *Proc. of the 28th VLDB Conference*, Oct. 2002.

[19] I. Lazaridis and S. Mehrotra, "Capturing sensor-generated time series with quality guarantees," in *IEEE International Conference on Data Engineering*, 2003.

[20] S. Charbonnier, G. Becq, and L. Biot, "On-line segmentation algorithm for continuously monitored data in intensive care units," in *IEEE Transactions on Biomedical Engineering*, March 2004, pp. 484–492.

[21] J. D. Brutlag, "Aberrant behavior detection in time series data for network monitoring," in *Proceedings of the 14th USENIX Systems Administration Conference (LISA)*, 2000.

[22] D. M. Hawkins and D. H. Olwell, "Cumulative sum charts and charting for quality improvement," Springer, 1997.

[23] CodeBlue, "Wireless sensor networks for medical care," http://www.eecs.harvard.edu/ mdw/proj/codeblue/.

[24] P. Broackwell and R. Davis, "Introduction to time series and forecasting," Springer, 1996.

[25] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen, "Sketch-based change detection: methods, evaluation, and applications," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement (IMC '03).*

[26] EPFL, "The sensorscope project," http://sensorscope.epfl.ch/.

[27] NOAA, "The tsunami research program," http://www.pmel.noaa.gov/tsunami.

[28] ——, "Tsunami events and data," http:// www.pmel.noaa.gov /ftp /tsunami /database /t20030925hokkaido /nos.