Mining Event Patterns in Sensor Networks *

Kay Römer

Institute for Pervasive Computing
ETH Zurich, 8092 Zurich, Switzerland
roemer@inf.ethz.ch

Abstract. Many sensor network applications are concerned with discovering interesting patterns among observed real-world events. Often, only limited apriori knowledge exists about the patterns to be found eventually. Here, raw streams of sensor readings are collected at the sink for later offline analysis – resulting in a large communication overhead. In this paper, we explore the use of in-network data mining techniques to discover frequent event patterns and their spatial and temporal properties. With that approach, compact event patterns rather than raw data streams are sent to the sink. We also discuss issues with the implementation of our proposal and report our experience with preliminary experiments.

1 Introduction

Wireless sensor networks have been successfully applied for detailed observation of a variety of real-world phenomena. Many of these applications are of a highly exploratory nature, where only a very rough idea of the expected findings exists before the experiment. In this case, streams of raw sensor readings from every node are typically delivered to a central sink for later offline analysis in order to find interesting patterns in the data – resulting in a large data volume that has to be delivered through the network (e.g., [5]).

A similar approach is used in the context of monitoring and debugging sensor networks [4]. Experience has shown that subtle real-world influences and large scale are the cause of numerous bugs and indeterministic behavior of sensor networks. Again, since the nature of these problems is often unknown in advance, testbeds and deployment support networks have been proposed to deliver high-volume event logs from every sensor node to a central sink for offline analysis in order to identify patterns that lead to failure.

In the above cases, missing advance knowledge about the patterns to be found eventually in the data limits the applicability of sophisticated in-network data processing and reduction techniques. Rather, raw data streams are delivered to the sink for later analysis. The resulting large data volume is a serious obstacle for deploying long-lived and large-scale sensor networks.

In this position paper, we explore the use of distributed data mining techniques to discover potentially interesting data patterns in a sensor network for the above type

* The work presented in this paper was supported (in part) by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS).

of applications. Rather than transmitting raw data streams from every sensor to the sink, only compact patterns mined at sensor nodes are transmitted to the sink, thus contributing to long-lived and large-scale sensor network deployments.

2 Approach

With our approach, a user can pose a *mining query* to the sensor network. This query is executed by a distributed runtime system in the sensor network. As a result, the user will receive at regular (but long) intervals a set of discovered event patterns from each node. An event pattern is a frequently occurring set of events observed at different nodes in the network.

A mining query specifies the types of events a user is interested in. The notion of event refers to a user-specified state change at a sensor node (e.g., sudden drop of temperature as measured by a sensor). In the context of our work, an event is simply an identifier (e.g., "temperature-drop") plus a timestamp when this event occurred according to some global time scale. We assume that time is partitioned into equal-sized *epochs*.

In addition, a mining query contains a number of constraints on the event patterns the system should look for. These are needed to limit the huge search space of potential event patterns.

The distributed mining algorithm then proceeds as follows. Every node in the network continuously collects event notifications from nodes in a user-defined network *neighborhood* (the size of which is specified by the mining query) using in-network aggregation techniques. The events that have been collected in this way during a user-defined amount of time called *history* (the duration of which is specified by the mining query as an integral number of epochs) are then fed to a mining algorithm. This algorithm executed at node n mines patterns of the following form:

$$A_1 \wedge ... \wedge A_m \Rightarrow E\,[S, C] \tag{1}$$

meaning that an event of type E occurred at node n with support S and confidence C given that antecedents A_i all hold true. Every antecedent A_i is of the form

$$A_i = (E_i, D_i, T_i, N_i) \tag{2}$$

meaning that A_i is true iff a certain type of event E_i occurred N_i times at a distance D_i from node n and T_i time units before E. D_i, T_i, and N_i usually denote intervals such as T_i = *"more than five minutes ago"*, D_i = *"less than 20 meters away"*, or N_i = *"between one and five times"*.

Support S is a number between 0 and 1 indicating how often this pattern could be found over time. Confidence C is a number between 0 and 1 indicating how strong the implication $\Rightarrow$ is. Note that the above patterns are a specific instance of *association rules* [1].

An example pattern discovered by this approach would be:

$$(\mathrm{tempdrop}, [0, 10\mathrm{m}], [0, 5\mathrm{min}], [3, \mathrm{inf}]) \Rightarrow \mathrm{tempdrop}[0.95, 0.8] \tag{3}$$

meaning that node n observed a "temperature drop" event with support 0.95 and confidence 0.8 if at least 3 nodes no more than 20 meters away from n did also observe a "temperature drop" event during the last 5 minutes. The following sections contain details on some selected aspects of the basic approach described above.

2.1 Mining Queries

A mining query contains the following information:

- **epochlen**: the duration of an epoch in seconds.
- **neighborhood**: the radius of the neighborhood around a node given in network hops or meters.
- **history**: the length of the history given in epochs.
- **minimum support**: a number between 0 and 1 indicating the minimum frequency required for reported patterns.
- **minimum confidence**: a number between 0 and 1 indicating the minimum confidence for the implication $\Rightarrow$ required for reported patterns.
- **distance quantization**: quantization of Euclidean distance between nodes into a small set of partitions. Each partition is assigned a name (e.g., near=(0m,5m], far=(5m,∞]).
- **time quantization**: quantization of time between events into a small set of partitions. Each partition is assigned a name (e.g., now=0, recent=[1, 5epochs], old=(5epochs, ∞]).
- **event frequency quantization**: quantization of number of events into a small set of partitions. Each partition is assigned a name (e.g., none=0, some=[1,∞]).

The purpose of quantization is to cut down the search space to be considered by the mining algorithm by binning event occurrences into a small number of "partitions". Note that quantization is a critical issue as it affects the patterns that will be found eventually.

2.2 Event Collection

As stated earlier, each node in the network collects event notifications from nodes in a neighborhood. This can be achieved by applying a framework that supports neighborhood abstractions such as Abstract Regions [6]. These tools allow a node in the network to define a *neighborhood* that consists of a set of nodes that fulfill certain conditions such as to be within a given distance of the node. Primitives for collecting data from the nodes in a neighborhood are provided. Using the quantization of distance discussed in the previous section, in-network data aggregation is applied to collect the frequency of each event for each distance partition in the neighborhood of the node. Using the example partitions from the previous section, we would count the occurrence of each event at distances "near" and "far" using in-network aggregation.

2.3 Mining Algorithm

The overall approach we apply is to transform the set of events collected from the neighborhood during history into an assignment of {TRUE, FALSE} to a small set of binary variables. The values of these variables characterize the collected events in a compact way and serve as the input for the data mining algorithm.

For the transformation, we make use of the quantization of distances, time, and event frequencies specified by the mining query. Essentially, there is one binary variable $< E > . < D > . < T > . < N >$ for each possible combination of an event $< E >$, a distance partition $< D >$, a time partition $< T >$, and an event frequency partition $< N >$. For example, the variable `tempdrop.near.now.some` would be TRUE, iff event "temperature drop" was observed by at least one node (partition "some"), during this epoch (partition "now"), not more than 5 meters away (partition "near").

In addition, we include a binary variable $< E >$ for each possible event $< E >$ that is TRUE iff this event occurred at the node executing the mining algorithm in the current epoch. For example, the variable `tempdrop` would be TRUE iff the node executing the mining algorithm observed a "temperature drop" event in the current epoch.

By applying this transformation, we obtain the set of binary variables with value TRUE for each epoch. Over time, this results in a stream of sets of binary variables with value TRUE.

To discover event patterns matching the mining query, we need to find sets of the above binary variables that occur with a minimum frequency in the stream. This lower frequency bound is given by the minimum support value in the mining query. Among the resulting frequent candidate sets, the ones satisfying the minimum confidence requirement are selected. The remaining sets can then be easily transformed into patterns of the form given in Equation 1.

Discovering such frequent sets of binary variables from a stream is a standard data mining task. In the literature, this problem is referred to as *mining of frequent itemsets* (each binary variable can be considered as an item that is either present in the set or not). In the recent past, various algorithms have been proposed to solve this problem with constrained resources over streams of itemsets (e.g., [2, 3]). From the resulting frequent itemsets, the ones satisfying the minimum confidence requirement are selected as in [1].

3 Implementation Issues

Clearly the major challenge in implementing our proposal is that of fitting the mining approach described in the previous section into the constrained communication, computational, and memory resources of a sensor node. One of the most challenging aspects is to implement the mining algorithm within the constrained memory resources of a sensor node. The BTnode [7] platform, for example, offers 256 kB of bank-switched RAM.

The memory footprint of algorithms for mining frequent itemsets is largely a function of the number of frequent itemsets discovered from the data stream. To examine the number of such frequent itemsets for a practical application, we performed an experiment using sensor data collected during one month from 54 sensor nodes in the Intel

Research Lab Berkeley [8]. This dataset was collected with an epoch duration of about 30 seconds (resulting in a total of about 65000 epochs) and contains, among others, temperature and light readings.

In our experiment, we consider two types of events: *temperature* and *light* events. Each sensor node with a temperature reading > 23 degrees Celsius in an epoch emits a temperature event in this epoch. Every sensor node with a light reading > 300 Lux emits a light event. Otherwise, we use the quantization from the example given in Sect. 2.1 with a neighborhood size of 10 meters and a history duration of 10 epochs.

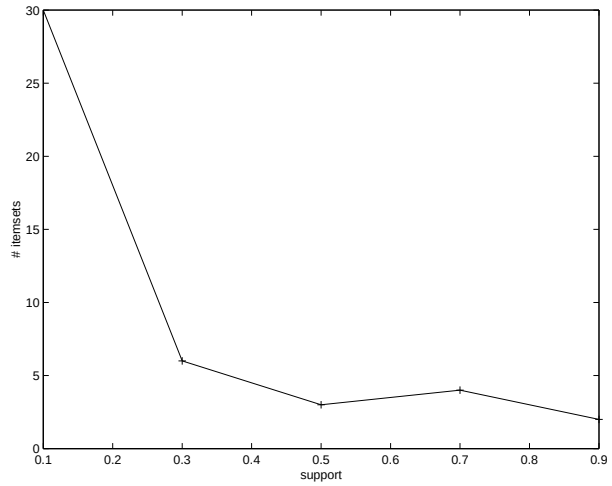

Fig. 1. Number of discovered maximal itemsets for different support values.

For our experiment, we consider the mining algorithm that would be executed on mote with ID 1 in the dataset. With the above settings, mote 1 generates a temperature event in about 23% of all epochs and a light event in about 14% of all epochs. We would expect a strong correlation of the occurrence of these events on mote 1 with the light and temperature events in the neighborhood of the mote.

Applying the method given in Sect. 2.3, we create a stream of itemsets which is then fed to an algorithm to discover maximal itemsets (a variant of [1]) for different values of minimum support. Here, a maximal itemset is a frequent itemset that has no proper supersets that are also frequent.

The results shown in Figure 3 are encouraging as the number of maximal itemsets is very small over the whole range of considered support values. Note that a single itemset in this experiment can be represented with 26 bits, since there are 26 different binary variables.

For a minimum support of 0.9 we obtain two maximal itemsets that result in the following patterns for node 1 in the format of Eq. 1:

$$(\mathrm{t}, \mathrm{now}, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{t}, \mathrm{recent}, *, \mathrm{some}) \wedge (\mathrm{t}, \mathrm{old}, *, \mathrm{some}) \Rightarrow t\ [0.96, 0.38]$$
$$(\mathrm{t}, *, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{l}, \mathrm{now}, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{l}, \{\mathrm{old}, \mathrm{recent}\}, *, \mathrm{some}) \Rightarrow l\ [0.92, 0.32]$$

Here, “t” and “l” refer to temperature and light events as defined above. The notations “{...,...}” and “*” mean that the enclosing antecedent is valid for the set of given partition identifiers or for all possible partition identifiers, respectively. The rules indicate that – as expected – the occurrence of temperature/light events at mote 1 is correlated with the occurrence of these events in the neighborhood of the node.

4 Conclusions and Future Work

We have examined the use of data mining techniques to discover frequent event patterns and their spatio-temporal relationships within a sensor network, such that compact patterns rather than raw data streams would have to be transmitted from nodes to the sink. In particular, such a system would be helpful to support exploratory settings, where only a rough idea of the actual findings exists before the experiment.

We have also discussed challenges in implementing this proposal on sensor nodes. In particular, we have identified the memory consumption of itemset discovery algorithms. We have performed an experiment with real-world data to motivate that an implementation is feasible. The work reported in this paper is a first step towards a distributed event-pattern-mining system for sensor networks.

References

1. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *VLDB 1994*, Santiago de Chile, Chile, September 1994.
2. H.-F. Li, S.-Y. Lee, and M.-K. Shan. An Efficient Algorithm for Mining Frequent Itemsets over the entire History of Data Streams. In *First Intl. Workshop on Knowledge Discovery in Data Streams 2004*, Pisa, Italy, September 2004.
3. G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *VLDB 2002*, Kong Kong, China, August 2002.
4. N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the Sensor Network Debugger. In *Sensys 2005*, San Diego, USA, November 2005.
5. R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a Sensor Network Expedition. In *EWSN 2004*, Berlin, Germany, January 2004.
6. M. Welsh and G. Mainland. Programming Sensor Networks Using Abstract Regions. In *NSDI 2004*, Boston, USA, March 2004.
7. BTnodes. www.btnode.ethz.ch.
8. Intel Lab Sensor Data. http://berkeley.intel-research.net/labdata/.