Online Latent Variable Detection in Sensor Networks

Jimeng Sun Spiros Papadimitriou Christos Faloutsos
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA, USA
{jimeng,spapadim,christos}@cs.cmu.edu

Abstract

Sensor networks attract increasing interest, for a broad range of applications. Given a sensor network, one key issue becomes how to utilize it efficiently and effectively. In particular, how can we detect the underlying correlations (latent variables) among many co-evolving sensor measurements? Can we do it incrementally? We present a system that can (1) collect the measurements from the real wireless sensors; (2) process them in real-time; and (3) determine the correlations (latent variables) among the sensor streams on the fly.

1 Introduction

As sensor network infrastructures become more and more mature, an important issue that needs to be explored even further is: how to effectively make good use of the data from senor networks. Can we find some deeper knowledge from the data? Can we perform the traditional data mining tasks in the context of sensor networks? For example, how can we quickly discover the major trends in the whole system? More specifically, in sensor networks, what are the underlying trends/correlations among all the sensors? How can we monitor those correlations in real time? How can we interpret the correlations? To answer those questions, we present a system prototype that has the following features: 1) it communicates with multiple wireless sensors (or, optionally, a simulator plug-in) to monitor the measurements online; 2) it analyzes the data in real time (with a single pass) to track the latent variables; 3) it can help spot outliers; 4) it can easily handle missing sensor measurements.

In the following, we will first describe the system's architecture in section 2; then we elaborate on some of the demonstration examples in section 3.

2 Architecture

First, we introduce the different data sources supported by the system. Second, we describe the *system frontend*. Third, we illustrate how the pattern discovery and prediction tasks are performed in the *system backend*.

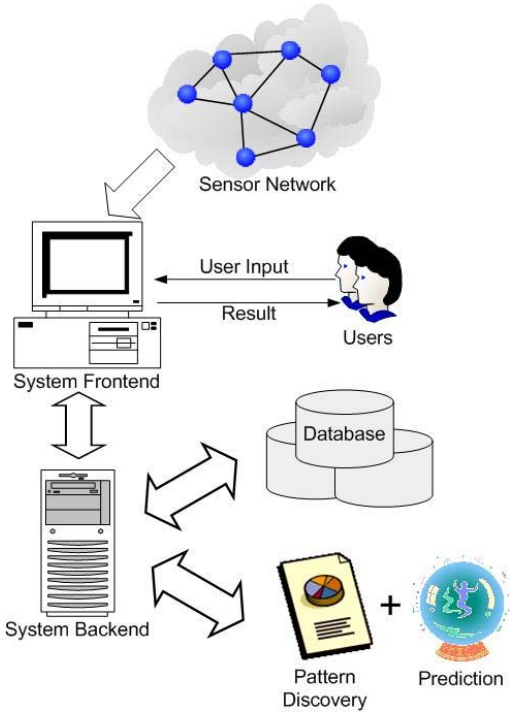

Figure 1. `Architecture`

2.1 Data Sources

We developed our system in Java. Figure 1 shows the overall architecture. Data are produced by *sources*; our system supports three types: real sensors, synthetic sources (e.g., generated by equations), and playback. We describe each next.

Generally speaking, the data source is a real sensor network. Specifically, in our system, it is directly collected from wireless sensors (Berkeley motes). We also study the wireless sensor motes. We use MICA2 wireless motes with MTS310 sensor boards, which have the multiple sensing modalities (e.g. light intensity, temperature, acoustic, etc.). Our system communicates with the motes through MIB510 interface board with 433MHz radio frequency.

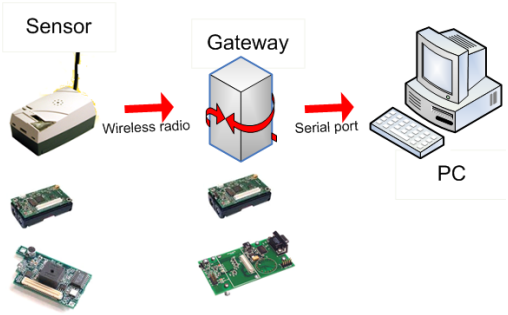

Figure 2. Sensor dataflow

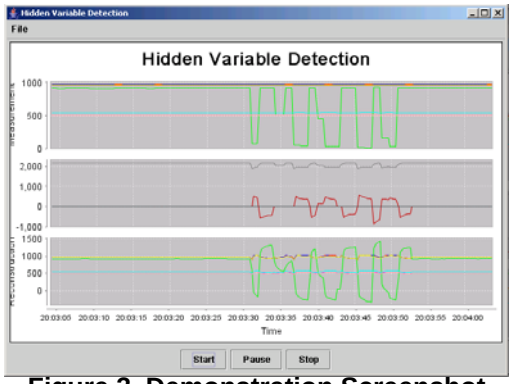

Figure 3. Demonstration Screenshot

2.2 System Frontend

After the raw data are obtained from the *source*, they are fed into the *system frontend*. The *system frontend* serves the following purposes:

1. It communicates with *data sources* to obtain raw data.
2. It cleans and calibrates the data before feeding them into the *system backend*.
3. It communicates with the user via the *user interface* to obtain certain parameters. These are also fed into the backend.

2.3 System Backend

The major task of the *system backend* is to perform online analysis of the incoming data in the stream processing module. This determines the number of latent variables and monitor their values. The number of latent variables essentially is the number of correlated groups among the sensors. Each sensor has a participation weight for each of these latent variables, which can be viewed as the participation strength of the sensor belongs to that group. Another important task of this key module is to forecast future values, with the help of the latent variables.

The algorithm we apply here does incremental/adaptive SVD on the multiple data streams. The details and performance evaluation are presented elsewhere [1]. The important features are: *(1)* efficient tracking of the latent variables (comparable to static SVD); *(2)* limited parameter tuning (all parameters are well-understood and defaults suffice for the vast majority of applications); *(3)* very good scalability; *(4)* dynamic change of the number of latent variables based on the sensor measurements.

3 Demonstration

The data source consists of 8 sensor streams, 4 temperature streams and 4 light intensity streams. Each sensor stream comes from a MTS310 sensor board that connects to a MICA2 wireless mote in real time. Figure 2 shows the data flow. The top window of Figure 3 shows the measurements of light intensity and temperatures [1]. All streams have constant trend originally. Hence, only one constant latent variable are needed in Figure 3 at the beginning. The reconstruction can be trivially obtained by multiplying different weights for each stream.

When the environment changes, for example when someone covers up one of the sensors (and thus the light intensity from it decreases), the changes directly affect the observable measurements in the top window of Figure 3. More importantly, it affects the latent variable in the middle window of Figure 3 too. Specifically, the original constant latent variable varies according the change of the original sensor measurements. The reconstruction is still fairly good in Figure 3. After that, when the measurements fall back to the normal/constant trend, the system drops the additional latent variable and becomes normal again (see Figure 3).

Our system is intuitive and informative for the users. The users can interact with the system directly at the demonstration without much explanation. By doing so, they can easily understand/appreciate the algorithm behind. In practice, this system can help humans monitor a large number of sensors in real time. It detects the trends, patterns and correlations among them, and summarizes the input sources into a few latent variables. When such a correlation is broken, the system spots it easily, introducing a new latent variable; the fact that there is a change in the number of latent variables, can be flagged as an anomaly.

4 Conclusion

We have presented a system that can handle multiple co-evolving streams. Specifically, it can track the correlations incrementally, in real-time; it can discover corresponding latent variables quickly, and it can spot broken correlations, flagging them as anomalies.

References

[1] S. Papadimitriou, J. Sun, and C. Faloutsos. Viper:incremental pattern discovery on multiple streams. In *Submitted for publication*, 2004.

[1]Note that the actual unit of the measurements are not important. In order to show two different kinds of measurements on the same plots, we scale the data so that they are comparable.