Distributed Pattern Discovery in Multiple Streams

Jimeng Sun[†] **Spiros Papadimitriou**[§] **Christos Faloutsos**[†]

Jan 2006
CMU-CS-06-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

[†] Computer Science Department, Carnegie Mellon University, PA, USA, {jimeng,christos}@cs.cmu.edu
[§] IBM Watson Research Center, NY, USA. This work is done while he was studying at CMU, spapadim@cs.cmu.edu

This material is based upon work supported by the National Science Foundation under Grants No. IIS-0209107 IIS-0205224 INT-0318547 SENSOR-0329549 IIS-0326322 This work is also supported in part by the Pennsylvania Infrastructure Technology Alliance (PITA), a partnership of Carnegie Mellon, Lehigh University and the Commonwealth of Pennsylvania's Department of Community and Economic Development (DCED). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, or other funding parties.

Keywords: Data mining, Stream mining, Distributed mining, Privacy preserving data mining

Abstract

Given m groups of streams which consist of $n_1, \ldots, n_m$ co-evolving streams in each group, we want to: (i) incrementally find local patterns within a single group, (ii) efficiently obtain global patterns across groups, and more importantly, (iii) efficiently do that in real time while limiting shared information across groups. In this paper, we present a distributed, hierarchical algorithm addressing these problems. It first monitors local patterns within each group and further summarizes all local patterns from different groups into global patterns. The global patterns are leveraged to improve and refine the local patterns, in a simple and elegant way. Moreover, our method requires only a single pass over the data, without any buffering, and limits information sharing and communication across groups. Our experimental case studies and evaluation confirm that the proposed method can perform hierarchical correlation detection efficiently and effectively.

1 Introduction

Data streams have received considerable attention in various communities (theory, databases, data mining, networking, systems), due to several important applications, such as network analysis [7], sensor network monitoring [23], financial data analysis [24], and scientific data processing [25]. All these applications have in common that: (i) massive amounts of data arrive at high rates, which makes traditional database systems prohibitively slow, (ii) the data and queries are usually distributed in a large network, which makes a centralized approach infeasible, and (iii) users or higher-level applications, require immediate responses and cannot afford any post-processing (e.g., in network intrusion detection). Data stream systems have been prototyped [1, 18, 4] and deployed in practice [7].

In addition to providing SQL-like support for data stream management systems (DSMS), it is crucial to detect patterns and correlations that may exist in co-evolving data streams. Streams are often inherently correlated (e.g., temperatures in the same building, traffic in the same network, prices in the same market, etc.) and it is possible to reduce hundreds of numerical streams into just a handful of *patterns* that compactly describe the key trends and dramatically reduce the complexity of further data processing. We propose an approach to do this incrementally in a distributed environment.

Limitations of centralized approach: Multiple co-evolving streams often arise in a large distributed system, such as computer networks and sensor networks. Centralized approaches usually will not work in this setting. The reasons are: **(i) Communication constraint**; it is too expensive to transfer all data to a central node for processing and mining. **(ii) Power consumption**; in a wireless sensor network, minimizing information exchange is crucial because many sensors have very limited power. Moreover, wireless power consumption between two nodes usually increases quadratically with the distance, which implies that transmitting all messages to single node is prohibitively expensive. **(iii) Robustness concerns**; centralized approaches always suffer from single point of failure. **(iv) Privacy concerns**; in any network connecting multiple autonomous systems (e.g., multiple companies forming a collaborative network), no system is willing to share all the information, while they all want to know the global patterns. To sum up, a **distributed online algorithm** is highly needed to address all the above concerns.

Motivation and intuition: We describe a motivating scenario, to illustrate the problem we want to solve. Consider a large computer network, in which multiple autonomous sites participate. Each site (e.g., company) wants to monitor local traffic patterns in their system, as well as to know global network traffic patterns so that they can compare their local patterns with global ones to do predication and anomaly detection. The method has to be: **(i) Fast**; early detection is crucially important to reduce the impact of virus spreading. **(ii) Scalable**; a huge number of nodes in the network require a distributed framework that can react in real-time. **(iii) Secure**; an individual autonomous system wants to limit shared information in order to protect privacy.

To address this problem, we propose a hierarchical framework that intuitively works as follows (also see Figure 1): 1) Each autonomous system first finds its local patterns and shares them with other groups (details in section 4). 2) Global patterns are discovered based on the shared

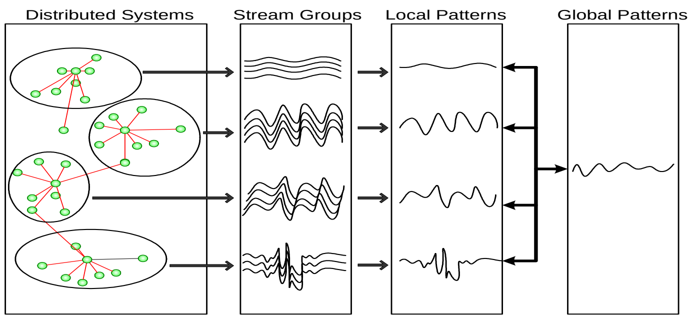


Figure 1: Distributed data mining architecture.

local patterns (details in section 5). 3) From the global patterns, each autonomous system further refines/verifies their local patterns.

There are two main options on where the global patterns are computed. First, all local patterns are broadcast to all systems and global patterns are computed at individual systems. Second, a high-level agent collects all local patterns from different systems, then computes global patterns and sends them back to individual systems. For presentation simplicity, we only focus on latter case[1], though we believe our proposed approaches can apply to the former case with very little modification.

We proved that the quality of our hierarchical method is as good as the centralized approach (where all measurements are sent to the central site). And our method can reduce the communication cost significantly while still preserving the good quality compared with the centralized approach. At the extreme, without any communication to the central agent (e.g., when the central agent is down), local patterns can still be computed independently. Moreover, the hierarchical method hides the original stream measurements (only aggregated patterns are shared) from the central agent as well as other stream groups, so that globally shared information is limited.

Contributions: The problem of pattern discovery in a large number of co-evolving groups of streams has important applications in many different domains. We introduce a hierarchical framework to discover local and global patterns effectively and efficiently across multiple groups of streams. The proposed method satisfies the following requirements:

- It is *streaming*, i.e., it is incremental without any buffering of historical data.
- It scales *linearly* with the number of streams.
- It runs in a distributed fashion requiring small communication cost.
- It avoids a single point of failure, which all centralized approaches have.
- It utilizes the global patterns to improve and refine the local patterns, in a simple and elegant way.
- It reduces the information that has to be shared across different groups, thereby protecting privacy.

[1]Note that the latter case is quite different to centralized approaches, because instead of all measurements to be sent to a central site, here only aggregated information are needed, which is much smaller.

The local and global patterns we discover have multiple uses. They provide a succinct summary to the user. Potentially, users can compare their own local patterns with the global ones to detect outliers. Finally, they facilitate interpolations and handling of missing values [19].

The rest of the paper is organised as follows: Section 2 formally defines the problem. Section 3 presents the overall framework of our algorithm. Section 4 and Section 5 illustrate how to compute local and global patterns, respectively. Section 6 presents experimental studies that demonstrate the effectiveness and efficiency of our approach. Section 7 discusses related work, on stream mining and parallel mining algorithms. Finally, in Section 8 we conclude.

2 Problem formalization

In this section we present the distributed mining problem formally. Given m groups of streams which consist of $\{n_1, \ldots, n_m\}$ co-evolving numeric streams, respectively, we want to solve the following two problems: (i) incrementally find patterns within a single group (*local pattern monitoring*), and (ii) efficiently obtain global patterns from all the local patterns (*global pattern detection*).

More specifically, we view original streams as points in a high-dimensional space, with one point per time tick. Local patterns are then extracted as low-dimensional projections of the original points. Furthermore, we continuously track the basis of the low-dimensional spaces for each group in a way that global patterns can be easily constructed.

More formally, the i-th group S_i consists of a (unbounded) sequence of n_i-dimensional vectors(points) where n_i is the number of streams in S_i, $1 \leq i \leq m$. S_i can also be viewed as a matrix with n_i columns and an unbounded number of rows. The intersection $S_i(t,l)$ at the t-th row and l-th column of S_i, represents the value of the l-th node/stream recorded at time t in the i-th group. The t-th row of S_i, denoted as $S_i(t,:)$, is the vector of all the values recorded at time t in i-th group. Note that we assume measurements from different nodes within a group are synchronized along the time dimension. We believe this constraint can be relaxed, but it would probably lead to a more complicated solution.

With above definition in mind, *local pattern monitoring* can be modelled as a function,

$$F_L : (S_i(t+1,:), G(t,:)) \rightarrow L_i(t+1,:), \tag{1}$$

where the inputs are 1) the new input point $S_i(t+1,:)$ at time $t+1$ and the current global pattern $G(t,:)$ and the output is the local pattern $L_i(t+1,:)$ at time $t+1$. Details on constructing such a function will be explained in section 4. Likewise, *global pattern detection* is modelled as another function,

$$F_G : (L_1(t+1,:), \ldots, L_m(t+1,:)) \rightarrow G(t+1,:), \tag{2}$$

where the inputs are local patterns $L_i(t+1,:)$ from all groups at time $t+1$ and the output is the new global pattern $G(t+1,:)$.

Having formally defined the two functions, Section 3 introduces the distributed mining framework in terms of F_L and F_G. The details of F_L and F_G are then presented in section 4 and section 5, respectively.

Symbol	Description
n_i	number of streams in group i
m	total number of stream groups
n	total number of streams in all groups ($\sum_1^m n_i$)
S_i	i-th stream group ($1 \leq i \leq m$) consisting of n_i streams
$S_i(t,l)$	the value from l-th stream at time t in group i
$S_i(t,:)$	a n_i-dimensional vector of all the values recorded at time t in group i
$\hat{S}_i(t,:)$	the reconstruction of $S_i(t,:)$, $\hat{S}_i(t,:) = L_i(t,:) \times W_{i,t}$
$W_{i,t}$	a $k_i \times n_i$ participation weight matrix at time t for group i
$k(k_i)$	number of global(local) patterns
$L_i(t,:)$	a k_i-dimensional projection of $S_i(t,:)$ (local patterns at t for group i)
$G(t,:)$	a k-dimensional vector (global patterns at time t)
$F_L(F_G)$	the function computing local(global) patterns.
$E_{t,i}$	Total energy captured by group i at time t.
$\hat{E}_{t,i}$	Total energy captured by the local patterns of group i at time t.
$f_{i,E}$, $F_{i,E}$	Lower and upper bounds on the fraction of energy for group i.

Table 1: Description of notation.

3 Distributed mining framework

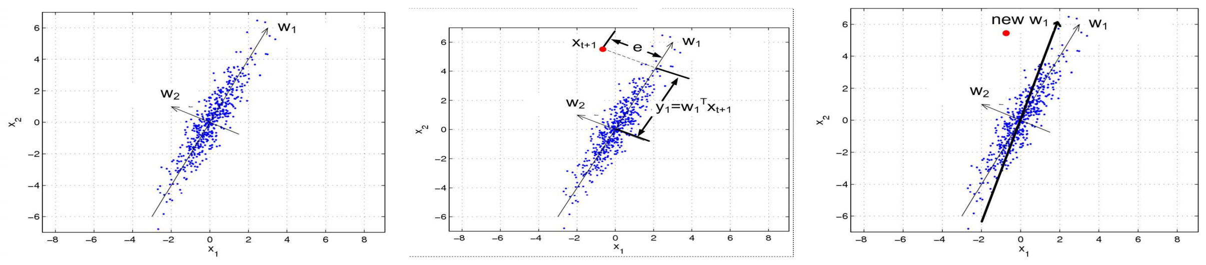
In this section, we introduce the general framework for distributed mining. More specifically, we present the meta-algorithm to show the overall flow, using F_L (*local patterns monitoring*) and F_G (*global patterns detection*) as black boxes.

Intuitively, it is natural that global patterns are computed based on all local patterns from m groups. On the other hand, it might be a surprise that the local patterns of group i take as input both the stream measurements of group i and the global patterns. Stream measurements are a natural set of inputs, since local patterns are their summary. However, we also need global patterns as another input so that local patterns can be represented consistently across all groups. This is important at the next stage, when constructing global patterns out of the local patterns; we elaborate on this later. The meta-algorithm is the following:

```
Algorithm DISTRIBUTEDMINING
0. (Initialization) At $t = 0$, set $G(t,:) \leftarrow$ null
1. For all $t > 1$
    (Update local patterns) For $i \leftarrow 1$ to $m$, set $L_i(t,:) := F_L(S_i(t,:), G(t-1,:))$
    (update global patterns) Set $G(t,:) := F_G(L_1, \ldots, L_m)$
```

4 Local pattern monitoring

In this section we present the method for discovering patterns within a stream group. More specifically, we explain the details of function F_L (Equation 1). We first describe the intuition behind the algorithm and then present the algorithm formally. Finally we discuss how to determine the number of local patterns k_i.



(a) Original $W_{i,t}(1,:)$ (b) Update process (c) Resulting $W_{i,t+1}(1,:)$

Figure 2: Illustration of updating $W_{i,t}(1,:)$ when a new point $S_i(t,:)$ arrives.

The goal of F_L is to find the low dimensional projection $L_i(t,:)$ and the participation weights $W_{i,t}$ so as to guarantee that the reconstruction error $\|S_i(t,:) - \hat{S}_i(t,:)\|^2$ over time is predictably small. Note that the reconstruction of $S_i(t,:)$ is defined as $\hat{S}_i(t,:) = L_i(t,:) \times W_{i,t}$. The $W_{i,t}$ can be considered as the basis on which the original high-dimensional points are projected.

This formulation is closely related to PCA [15], which tries to find both a low rank approximation Y and the participation weights W from original data X, such that $(Y, W) = argmin\|X - Y \times W\|^2$. But a fundamental difference is that PCA requires the entire data matrix X available up front, while in our setting S_i grows continually and rapidly, without bound. This requires a fast incremental algorithm that finds the projection quickly and is also capable of updating the participation weights as the data distribution changes.

If we assume that the $S_i(t,:)$ are drawn according to some distribution that does not change over time (i.e., under *stationarity* assumptions), then the weight vectors $W_{i,t}$ converge to the true principal directions. However, even if there are non-stationarities in the data (i.e., gradual drift), we can still deal with these very effectively, as we explain shortly.

Tracking local patterns: The first step is, for a given k_i, to incrementally update the $k \times n_i$ participation weight matrix $W_{i,t}$, which serves as a basis of the low-dimensional projection for $S_i(t,:)$. Later in this section, we describe the method for choosing k_i. For the moment, assume that the number of patterns k_i is given.

We use an algorithm based on adaptive filtering techniques [22, 13], which have been tried and tested in practice, performing well in a variety of settings and applications (e.g., image compression and signal tracking for antenna arrays).

The main idea behind the algorithm is to read the new values $S_i(t+1,:) \equiv [S_i(t+1,1), \ldots, S_i(t+1, n_i)]$ from the n_i streams of group i at time $t+1$, and perform three steps: **(1) Compute** the low dimensional projection $y_j, 1 \leq j \leq k_i$, based on the *current* weights $W_{i,t}$, by projecting $S_i(t+1,:)$ onto these.**(2) Estimate** the reconstruction error ($\vec{e}_j$ below) and the energy.**(3) Compute** $W_{i,t+1}$ and output the *actual* local pattern $L_i(t+1,:)$. To illustrate this, Figure 2(b) shows the $\vec{e}_1$ and y_1 when the new data $S_i(t+1,:)$ enters the system. Intuitively, the goal is to adaptively update $W_{i,t}$ so that it quickly converges to the "truth."

In particular, we want to update $W_{i,t}(j,:)$ more when $\vec{e}_j$ is large. The magnitude of the update also takes into account the past data currently "captured" in the system, which is why the update step size is inversely proportional to d_i. Finally, the update takes into account the global pattern

$G(t, j)$—only when a global pattern is not available, we use the local projection $\vec{y}_j$. This is crucial to ensure that the local patterns are represented consistently among groups. It is a simple and elegant way by which the global patterns direct the projections into the local subspaces and improve the quality of the local patterns, while sharing very limited information. Moreover, it simplifies the global pattern detection algorithm (see Section 5)

```
Algorithm $F_L$
Input: new vector $S_i(t+1,:)$, old global patterns $G(t,:)$
Output: local patterns ($k_i$-dimensional projection) $L_i(t+1,:)$
1. Initialize $\vec{x}_1 := S_i(t+1,:)$.
2. For $1 \le j \le k$, we perform the following in order:
        $y_j := \vec{x}_j W_{i,t}(j,:)^T$                          ($y_j$ = projection onto $W_{i,t}(j,:)$)
        If $G(t,:) = \text{null}$, then $G(t,j) := y_j$            (handling boundary case)
        $d_j \leftarrow \lambda d_j + y_j^2$                       (local energy, determining update magnitude)
        $\vec{e} := \vec{x}_j - G(t,j) W_{i,t}(j,:)$               (error, $\vec{e} \perp W_{i,t}(j,:)$)
        $W_{i,t+1}(j,:) \leftarrow W_{i,t}(j,:) + \frac{1}{d_j} G(t,j)\vec{e}$   (update participation weight)
        $\vec{x}_{j+1} := \vec{x}_j - G(t,j) W_{i,t+1}(j,:)$       (repeat with remainder of $\vec{x}$).
3. Compute the new projection $L_i(t+1,:) := S_i(t+1,:) W_{i,t+1}^T$
```

For each j, $\vec{x}_j$ is the component of $S_i(t+1,:)$ in the orthogonal complement of the space spanned by the updated weight $W_{i,t+1}(j',:), 1 \le j' < j$. The vectors $W_{i,t+1}$ are in order of importance (more precisely, in order of decreasing eigenvalue or energy). It can be shown that, under stationarity assumptions, these updated $W_{i,t+1}$ converge to the true principal directions.

The term λ is an exponential forgetting factor between 0 and 1, which helps adapt to more recent behavior. For instance, $\lambda = 1$ means putting equal weights on all historical data, while smaller λ means putting higher weight on more recent data. This allows us to follow trend drifts over time. Typical choices are $0.96 \le \lambda \le 0.98$ [13]. We chose 0.96 throughout all experiments. As long as the values of S_i do not vary wildly, the exact value of λ is not crucial.

So far, we understand how to track the local patterns for each individual group. Next we illustrate how to decide the number of local patterns k_i.

Detecting the number of local patterns: In practice, we do not know the number k_i of local patterns. We propose to estimate k_i on the fly, so that we maintain a high percentage $f_{i,E}$ of the *energy* $E_{i,t}$. Energy thresholding is a common method to determine how many principal components are needed [15] and corresponds to a bound on the total squared reconstruction error. Formally, the energy $E_{i,t}$ (at time t) of the sequence of $\vec{x}_t$ is defined as

$$E_{i,t} := \tfrac{1}{t}\sum_{\tau=1}^{t} \|S_i(\tau,:)\|^2 = \tfrac{1}{t}\sum_{\tau=1}^{t}\sum_{j=1}^{n_i} S_i(\tau,j)^2.$$

Similarly, the energy $\hat{E}_{i,t}$ of the reconstruction $S_i(\hat{t},:)$ is defined as

$$\hat{E}_{i,t} := \tfrac{1}{t}\sum_{\tau=1}^{t} \|\hat{S}_i(\tau,:)\|^2 = \tfrac{1}{t}\sum_{\tau=1}^{t} \|L_i(t,:) \times W_{i,t}\|^2 = \tfrac{1}{t}\sum_{\tau=1}^{t} \|L_i(t,:)\|^2.$$

For each group, we have a low-energy and a high-energy threshold, $f_{i,E}$ and $F_{i,E}$, respectively. We keep enough local patterns k_i, so the retained energy is within the range $[f_{i,E} \cdot E_{i,t}, F_{i,E} \cdot E_{i,t}]$.

Dataset	n	k	**Description**
`Chlorine`	166	2	Chlorine concentrations from EPANET.
`Motes-Light`	48	2–4	Light sensor measurements.
`Motes-Humid`	48	2	Humidity sensor measurements.
`Motes-Temp`	48	2	Temperature sensor measurements.
`Motes-Volt`	48	2	Battery voltage measurements.

Table 2: Description of datasets

Whenever we get outside these bounds, we increase or decrease k_i. In more detail, the steps are: **(1) Estimate** the full energy $E_{i,t+1}$ from the sum of squares of $S_i(\tau, :)$. **(2) Estimate** the energy $\hat{E}_{i,t+1}$ of the k_i local patterns. **(3) Adjust** k_i if needed. We introduce a new local pattern (update $k_i \leftarrow k_i + 1$) if the current local patterns maintain too little energy, i.e., $\hat{E}_{i,t+1} < f_{i,E} E_{i,t}$. We drop a local pattern (update $k_i \leftarrow k_i - 1$), if the maintained energy is too high, i.e., $\hat{E}_{i,t+1} > F_{i,E} E_{i,t+1}$. The energy thresholds $f_{i,E}$ and $F_{i,E}$ are chosen according to recommendations in the literature [15]. We set $f_{i,E} = 0.95$ and threshold $F_{i,E} = 0.98$.

5 Global pattern detection

In this section we present the method for obtaining global patterns over all groups. More specifically, we explain the details of function F_G (Equation 2).

First of all, what is a global pattern? Similar to local pattern, global pattern is low dimensional projections of the streams from all groups. Loosely speaking, assume only one global group exists which consists of all streams, the global patterns are the local patterns obtained by applying F_L on the global group—this is essentially the centralized approach. In other words, we want to obtain the result of the centralized approach without centralized computation.

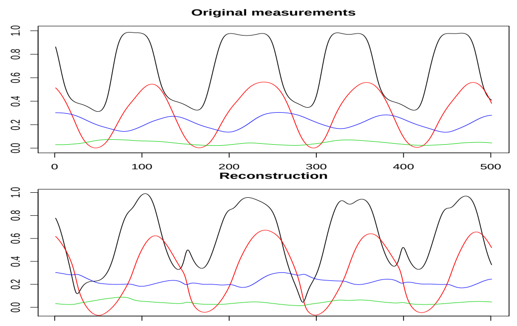
As explained before, F_L has been designed in a way that it is easy to combine all different local patterns into global patterns. More specifically, we have:

Lemma 1. *Assuming the same k_i for all groups, the global patterns from the centralized approach equal the sum of all local patterns. i.e., $G(t,:) = F_L([S_1(t,:), \ldots, S_m(t,:)])$ equals $\sum_{i=1}^{m} L_i(t,:)$*

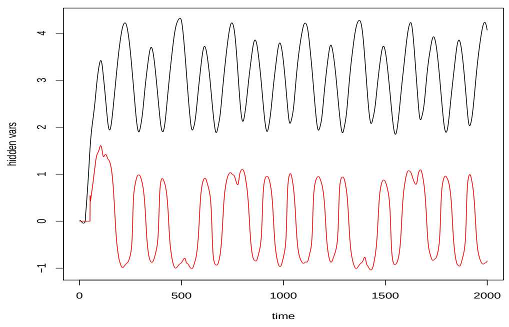
Proof. Let $G(t+1,:) = \sum_{i=1}^{m} L_i(t,:)$ be the global patterns from distributed approach and $L_i = S_i(t,:) \times W_{i,t}^T$ from algorithm F_L. Therefore, $G(t+1,:) = [S_1(t,:), \ldots, S_m(t,:)] \times [W_{1,t}, \ldots, W_{m,t}]^T$. That is exactly the result of global approach (i.e., considering all streams as one group and applying F_L on that). Other update steps can be proved using similar arguments, therefore omitted. □

The algorithm exactly follows the lemma above. The j-th global pattern is the sum of all the j-th local patterns from m groups.

```
Algorithm $F_G$
Input: all local patterns $L_1(t,:), \ldots, L_m(t,:)$
Output: global patterns $G(t,:)$
0. Set $k := max(k_i)$ for $1 \le i \le m$
1. For $1 \le j \le k$, set $G(t,j) := \sum_{i=1}^{m} L_i(t,j)$ (if $j > k_i$ then $L_i(t,j) \equiv 0$)
```



(a) Measurements and reconstruction



(b) Local patterns

Figure 3: `Chlorine` dataset: (a) Actual measurements and reconstruction at four junctions. We plot only 500 consecutive timestamps (the patterns repeat after that). (b) Two local (and also global) patterns.

6 Experimental evaluation

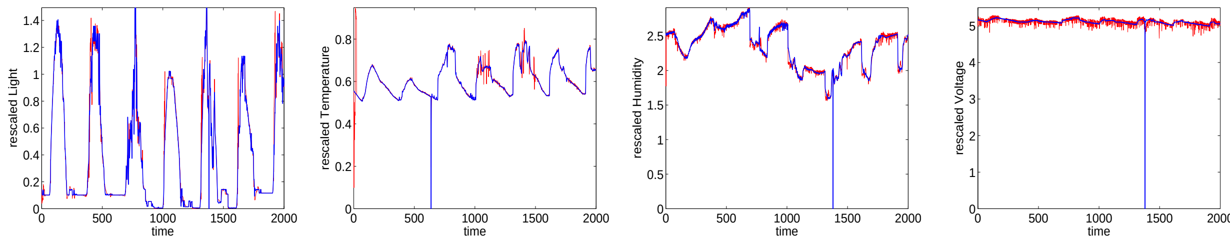
In this section we first present case studies on real and realistic datasets to demonstrate the effectiveness of our approach in discovering patterns among distributed streams. Then we discuss performance issues. In particular, we show that (i) identified patterns have natural interpretations, (ii) very few patterns can achieve good reconstruction, (iii) processing time per stream is constant, and (iv) communication cost is small.

6.1 Case Study I — Chlorine concentrations

Description The `Chlorine` dataset was generated by EPANET 2.0 [8] which accurately simulates the hydraulic and chemical phenomena within drinking water distribution systems. We monitor the chlorine concentration level at 166 junctions(streams) for 4310 timestamps during 15 days (one time tick every five minutes). The data was generated using the input network with the demand patterns, pressures and flows specified at each node. We partition the junctions into 4 groups of roughly equal size based on their geographical proximity.

Data characteristics This dataset is an example of homogeneous streams. The two key features are: (1) A clear global periodic pattern (daily cycle, dominating residential demand pattern). Chlorine concentrations reflect this, with few exceptions. (2) A slight time shift across different junctions, which is due to the time it takes for fresh water to flow down the pipes from the reservoirs. Thus, most streams exhibit the same sinusoidal-like pattern, but with gradual "phase shift" as we go further away from the reservoir.

Results The proposed method can successfully summarize the data using two local patterns per group (i.e., eight local patterns in total) plus two global patterns, as opposed to the original 166 streams) . Figure 3(a) shows the reconstruction for four sensors from one group over 500 timestamps. Just two local patterns give very good reconstruction. Since the streams all have similar periodic behavior, the pair of global patterns is similar to the pairs of local patterns. Overall reconstruction error is below 4%, using the L_2 norm (see also 6.3).



(a) Light measurements (b) Temperature (c) Humidity (c) Voltage

Figure 4: `Mote` dataset: original measurements (blue) and reconstruction (red) are very close. And the method converges very quickly with 20-50 time ticks.

Interpretation The two local/global patterns (Figure 3(b)) reflect the two key dataset characteristics: (1) The first hidden variable captures the global, periodic pattern. (2) The second one also follows a very similar periodic pattern, but with a slight "phase shift." It turns out that the two hidden variables together are sufficient to express any other time series with an arbitrary "phase shift."

6.2 Case study II — Mote sensors

Description The `Motes` dataset consists of 4 groups of sensor measurements (i.e., light intensity, humidity, temperature, battery voltages) collected using 48 Berkeley Mote sensors at different locations in a lab, over a period of a month. This is an example of heterogeneous streams. All the streams are scaled to have unit variance, to be comparable across different measures. In particular, streams from different groups behave very differently. This can be considered as a bad scenario for our method. The goal is to show that the method can still work well even when the groups are not related. If we do know the groups are unrelated up front, we can treat them separately without bothering to find global patterns. However, in practice, such prior knowledge is not available. Our method is still a sound approach in this case.

Data characteristics The main characteristics (see the blue curves in Figure 4) are: (1) Light measurements exhibit a clear global periodic pattern (daily cycle) with occasional big spikes from some sensors (outliers), (2) Temperature shows a weak daily cycle and a lot of bursts. (3) Humidity does not have any regular pattern. (4) Voltage is almost flat with a small downward trend.

Results The reconstruction is very good (see the red curves in Figure 4(a)), with relative error below 6%. Furthermore, the local patterns from different groups are correlated well with the original measurements (see Figure 6). The global patterns (in Figure 5) are combinations of different patterns from all groups and reveal the overall behavior of all the groups.

6.3 Performance evaluation

In this section we discuss performance issues. First, we show that the proposed method requires very limited space and time. Next, we elaborate on tradeoff between accuracy and communication cost.

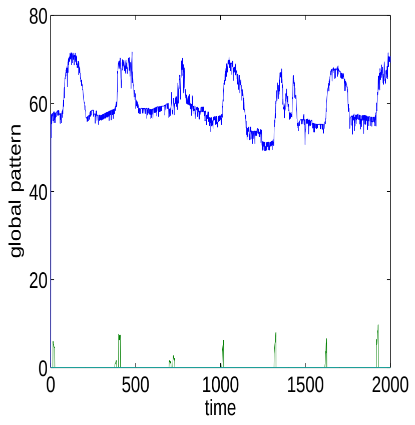
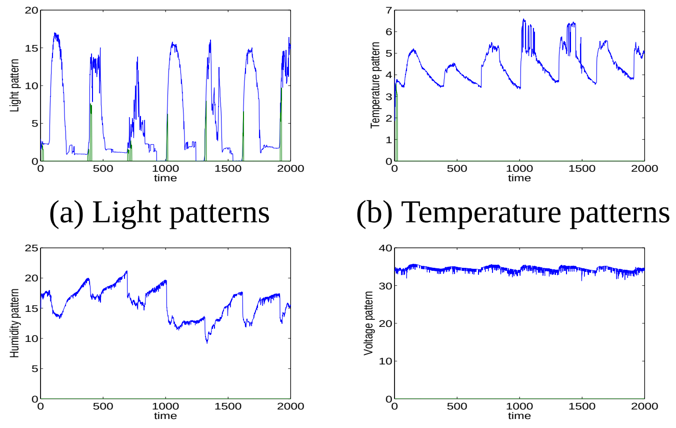


Figure 5: Global patterns



(a) Light patterns (b) Temperature patterns

(c) Humidity patterns (d) Voltage patterns

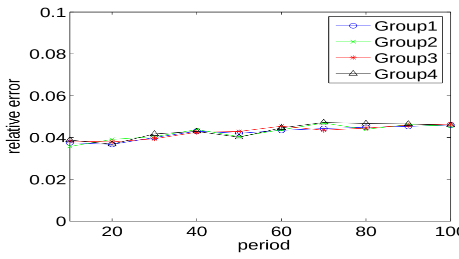
Figure 6: Local patterns

Time and space requirements: For local pattern monitoring, time scales linearly with respect to (i) stream size t, (ii) number of streams n_i, and (iii) number of local patterns k_i. This is because, for each time tick t, we have to update the weights $W_{i,t}$, which consist of n_i numbers for each of the k_i local patterns. The space requirements also scale linearly with n_i and k_i and are *independent* of stream size. For global pattern detection, the time scales linearly with the number of groups m. Space depends only on the number of global patterns $k = \max_i k_i$.

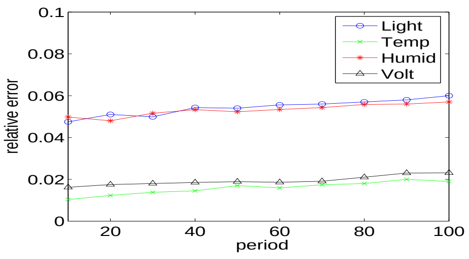
Accuracy and communication cost: In terms of accuracy, everything boils down to the quality of the summary provided by the local/global patterns. To this end, we use the relative reconstruction error ($\|S - \hat{S}\|^2/\|S\|^2$ where S are the original streams and $\hat{S}$ are the reconstructions) as the evaluation metric. The best performance is obtained when accurate global patterns are known to all groups. But this requires exchanging up-to-date local/global patterns at every timestamp among all groups, which is prohibitively expensive. One efficient way to deal with this problem is to increase the communication period, which is the number of timestamps between successive local/global pattern transmissions. For example, we can achieve a 10-fold reduction on communication cost by changing the period from 10 to 1000 timestamps. Figure 7 shows reconstruction error vs. communication period for both real datasets. Overall, the relative error rate increases very slowly as the communication period increases. This implies that we can dramatically reduce communication with minimal sacrifice of accuracy.

7 Related work

Stream mining Many stream processing and mining techniques have been studied (see tutorial [10]). Much of the work has focused on finding interesting patterns in a single stream. Ganti et al. [9] propose a generic framework for stream mining. Guha et al. [11] propose a one-pass k-median clustering algorithm. Recently, [14, 20] address the problem of finding patterns over concept drifting streams.



(a) Relative error rate on `Chlorine`



(b) Relative error rate on `Motes`

Figure 7: Accuracy drops slowly as the update period increases.

Guha et al. [12] study on discovering correlations among multiple streams, by first doing dimensionality reduction with random projections, and then periodically computing the SVD. However, the method incurs high overhead because of the SVD re-computation. Yuan et al. [21] developed an incremental subspace learning algorithm, with application in text data. Papadimitriou et al. [19] improve on correlation discovering, which requires constant processing time per timestamp without any buffering. However, both methods still use a centralized approach that has all the undesirable properties as we list in Section 1.

Distributed data mining Most of works on distributed data mining focus on extending classic (centralized) data mining algorithms into distributed environment, such as association rules mining [6], frequent item sets [17]. Web is a popular distributed environment. Several techniques are proposed specifically for that, for example, distributed top-k query [3] and Bayes-net mining on web [5]. But our focus are on finding numeric patterns, which is different.

Privacy preserving data mining Recent year many researchers start to study the privacy issues associated with data mining. The most related discussion is on how much privacy can be protected using subspace projection method [2, 16]. Liu et al. [16] discuss the subspace projection method and propose a possible method to breach the protection using Independent component analysis(ICA). All the method provides a good insight on the issues on privacy protection. Our method focuses more on incremental online computation of subspace projection.

8 Conclusion

We focus on finding patterns in a large number of distributed streams. More specifically, we first find local patterns within each group, where the number of local patterns is automatically determined based on reconstruction error. Next, global patterns are identified, based on the local patterns from all groups. Our proposed method has the following desirable characteristics:

- It discovers underlying correlations among multiple stream groups incrementally, via a few patterns.
- It automatically estimates the number k_i of local patterns to track, and it can automatically adapt, if k_i changes.

- It is distributed, avoiding a single point of failure and reducing communication cost and power consumption.
- It utilizes the global patterns to improve and refine the local patterns, in a simple and elegant way.
- It requires limited shared information from different groups, while being able to successfully monitor global patterns.
- It scales up extremely well, due to its incremental and hierarchical nature.
- Its computation demands are low. Its space demands are also limited: no buffering of any historical data.

We evaluated our method on several datasets, where it indeed discovered the patterns. We gain significant communication savings, with small accuracy loss.

References

[1] Daniel J. Abadi, Don Carney, Ugur Cetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Michael Stonebraker, Nesime Tatbul, and Stan Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] C. Agrawal and P. Yu. A condensation approach to privacy preserving data mining. In *EDBT*, 2004.

[3] B. Babcock and C. Olston. Distributed Top-K Monitoring. In *SIGMOD*, 2003.

[4] Sirish Chandrasekaran, Owen Cooper, Amol Deshpande, Michael J. Franklin, Joseph M. Hellerstein, Wei Hong, Sailesh Krishnamurthy, Samuel Madden, Vijayshankar Raman, Fred Reiss, and Mehul A. Shah. Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[5] R. Chen, S. Krishnamoorthy, and H. Kargupta. Distributed Web Mining using Bayesian Networks from Multiple Data Streams. In *ICDM*, pages 281–288, 2001.

[6] D. W. Cheung, V. T. Ng, A. W. Fu, and Y. Fu. Efficient Mining of Association Rules in Distributed Databases. *TKDE*, 8:911–922, 1996.

[7] Chuck Cranor, Theodore Johnson, Oliver Spataschek, and Vladislav Shkapenyuk. Gigascope: a stream database for network applications. In *SIGMOD*, 2003.

[8] EPANET. `http://www.epa.gov/ORD/NRMRL/wswrd/epanet.html`.

[9] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Mining data streams under block evolution. *SIGKDD Explorations*, (2):1–10, 2002.

[10] Minos N. Garofalakis, Johannes Gehrke, and Rajeev Rastogi. Querying and mining data streams: you only get one look a tutorial. In *SIGMOD*, 2002.

[11] S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams: Theory and practice. *IEEE TKDE*, 15(3):515–528, 2003.

[12] Sudipto Guha, Dimitrios Gunopulos, and Nick Koudas. Correlating synchronous and asynchronous data streams. In *KDD*, 2003.

[13] S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 1992.

[14] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *KDD*, 2001.

[15] I.T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[16] K. Liu, H. Kargupta, and J. Ryan. Multiplicative noise, random projection, and privacy preserving data mining from distributed multi-party data. In *TKDE*, 2005.

[17] K. K. Loo, I. Tong, B. Kao, and D. Cheung. Online Algorithms for Mining Inter-Stream Associations From Large Sensor Networks. In *PAKDD*, 2005.

[18] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma. Query processing, resource management, and approximation in a data stream management system. In *CIDR*, 2003.

[19] Spiros Papadimitriou, Jimeng Sun, and Christos Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[20] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proc.ACM SIGKDD*, 2003.

[21] J. Yan, Q. Cheng, Q. Yang, and B. Zhang. An incremental subspace learning algorithm to categorize large scale text data. In *APWeb*, 2005.

[22] Bin Yang. Projection approximation subspace tracking. *IEEE Trans. Sig. Proc.*, 43(1):95–107, 1995.

[23] Y. Yao and J. Gehrke. Query processing in sensor networks. In *CIDR*, 2003.

[24] Y. Zhu and D. Shasha. Statstream: Statistical monitoring of thousands of data streams in real time. In *VLDB*, 2002.

[25] Y. Zhu and D. Shasha. Efficient elastic burst detection in data streams. In *KDD*, 2003.