Optimal Multi-scale Patterns in Time Series Streams

Spiros Papadimitriou
IBM T.J. Watson Research Center
Hawthorne, NY, USA
spapadim@us.ibm.com

Philip S. Yu
IBM T.J. Watson Research Center
Hawthorne, NY, USA
psyu@us.ibm.com

ABSTRACT

We introduce a method to discover optimal local patterns, which concisely describe the main trends in a time series. Our approach examines the time series at multiple time scales (i.e., window sizes) and efficiently discovers the key patterns in each. We also introduce a criterion to select the best window sizes, which most concisely capture the key oscillatory as well as aperiodic trends. Our key insight lies in learning an optimal orthonormal transform *from the data itself*, as opposed to using a predetermined basis or approximating function (such as piecewise constant, short-window Fourier or wavelets), which essentially restricts us to a particular family of trends. Our method lifts that limitation, while lending itself to fast, incremental estimation in a streaming setting. Experimental evaluation shows that our method can capture meaningful patterns in a variety of settings. Our streaming approach requires order of magnitude less time and space, while still producing concise and informative patterns.

1. INTRODUCTION

Data streams have recently received much attention in several communities (e.g., theory, databases, networks, data mining) because of several important applications (e.g., network traffic analysis, moving object tracking, financial data analysis, sensor monitoring, environmental monitoring, scientific data processing).

Many recent efforts concentrate on summarization and pattern discovery in time series data streams [5, 23, 24, 33, 4, 26, 6]. Typical approaches for pattern discovery and summarization of time series rely on fixed transforms, with a predetermined set of bases [24, 33] or approximating functions [5, 23, 4]. For example, the short-window Fourier transform uses translated sine waves of fixed length and has been successful in speech processing [28]. Wavelets use translated and dilated sine-like waves and have been successfully applied to more bursty data, such as images and video streams [32]. Furthermore, selecting a wavelet basis to match the time series' characteristics is a non-trivial problem [32, 27, 30]. Thus, even though these approaches have been useful for pattern discovery in a number of application domains, there is no single method that is best for arbitrary time series.

We propose to address this problem by learning the appropriate approximating functions *directly* from the data, instead of using a fixed set of such functions. Our method estimates the "eigenfunctions" of the time series in finite-length time windows. Similar ideas have been used in other areas (e.g., image denoising or motion capture). However, our approach examines the data at multiple time scales and, to the best of our knowledge, the problem of estimating these eigenfunctions incrementally has not been studied before.

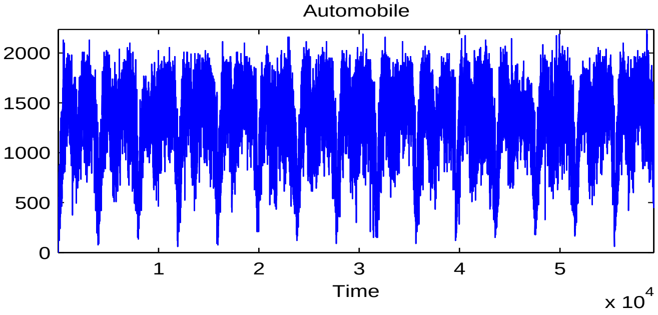


Figure 1: Automobile traffic data (aggregate counts from a west coast interstate).

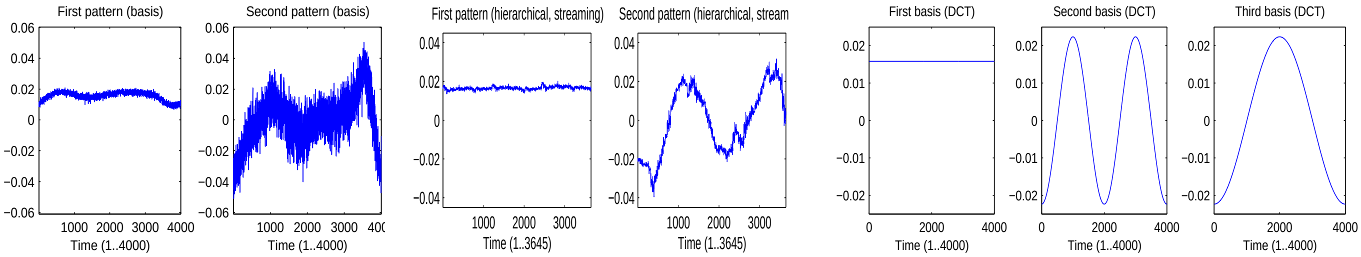
Example

We will illustrate the main intuition and motivation with a real example. Figure 1 shows automobile traffic counts in a large, west coast interstate. The data exhibit a clear daily periodicity. Also, in each day there is another distinct pattern of morning and afternoon rush hours. However, these peaks have distinctly different shapes: the morning one is more spread out, the evening one more concentrated and slightly sharper. What we would ideally like to discover is:

1. The main trend in the data repeats at a window ("period") of approximately 4000 timestamps.
2. A succinct "description" of that main trend that captures most of the recurrent information.

Figure 2a shows the output of our pattern discovery approach, which indeed suggests that the "best" window is

SIGMOD 2006, June 27–29, 2006, Chicago, Illinois, USA.
Copyright 2006 ACM 1-59593-256-9/06/0006 ...$5.00.



(a) Representative patterns (batch, non-hierarchical), 5.4% error.

(b) Representative patterns (streaming, hierarchical).

(c) Fixed bases (DCT) with highest coefficients, 7.7% error (8.4% with two coefficients).

Figure 2: Automobile traffic, best selected window (about 1 day) and corresponding representative patterns.

4000 timestamps. Furthermore, the first pattern captures the average and the second pattern correctly captures the two peaks and also their approximate shape (the first one wide and the second narrower). For comparison, in Figure 2b we show the output of our fast, streaming computation scheme. In order to reduce the storage and computation requirements, our fast scheme tries to filter out some of the "noise" earlier, while retaining as many of the regularities as possible. However, which information should be discarded and which should be retained is once again decided based on the data *itself*. Thus, even though we unavoidably discard some information, Figure 2b still correctly captures the main trends (average level, peaks and their shape).

For comparison, Figure 2c shows the best "local patterns" we would obtain using fixed bases. For illustration, we chose the Discrete Cosine Transform (DCT) on the first window of 4000 points. First, most fixed-basis schemes cannot be easily used to capture information at arbitrary time scales (with the exception of wavelets). More importantly, any fixed-basis scheme (e.g., wavelets, Fourier, etc) would produce similar results which are heavily biased towards the shape of the *a priori* chosen bases or approximating functions.

Contributions

We introduce a method that can learn the key trends in a time series. Our main contributions are:

- We introduce the notion of optimal local patterns in time series, which concisely describe the main trends, both oscillatory as well as aperiodic, within a fixed window.
- We show how to extract trends at multiple time scales (i.e., window sizes).
- We propose a criterion which allows us to choose the best window sizes from the data.
- We introduce an approach to perform all of the above incrementally, in a streaming setting.

We evaluate our approach on real data and show that it discovers meaningful patterns. The streaming approach achieves 1-4 *orders of magnitude* improvement in time and space requirements.

In the rest of this paper we answer the following questions:

1. Given a window, how do we find locally optimal patterns?
2. How can we compare the information captured by patterns at different window sizes?
3. How can we efficiently compute patterns at several different window sizes and quickly zero-in on the "best" window?
4. How can we do all of the above incrementally, in a streaming setting?

After reviewing some of the background in Section 2 and introducing necessary definitions and notation in Section 3, we answer the first two questions in Section 4. We answer the third question in Section 5, which introduces an efficient scheme for pattern discovery at multiple scales. Section 6 answers the fourth question. Section 7 demonstrates the effectiveness of our approach on real data. Finally, Section 8 discusses related work and Section 9 concludes.

2. BACKGROUND

In this section we describe some of the basic background. For more details, the reader can see, e.g., [29]. We use boldface lowercase letters for column vectors, $\mathbf{v} \equiv [v_1 \;\; v_2 \;\cdots\; v_n]^T \in \mathbb{R}^n$, and boldface capital letters for matrices, $\mathbf{A} \in \mathbb{R}^{m \times n}$. Finally, we adopt the notation $\mathbf{a}_j$ for the columns of $\mathbf{A} \equiv [\mathbf{a}_1 \;\; \mathbf{a}_2 \;\cdots\; \mathbf{a}_3]$ and $\mathbf{a}_{(i)}$ for the rows of $\mathbf{A} \equiv [\mathbf{a}_{(1)} \;\; \mathbf{a}_{(2)} \;\cdots\; \mathbf{a}_{(m)}]^T$. Note that $\mathbf{a}_{(i)}$ are also column vectors, not row vectors—we always represent vectors as column vectors. For matrix/vector elements we use either subscripts, a_{ij}, or brackets, $a[i, j]$.

The rows of $\mathbf{A}$ are points in an (at most) n-dimensional space, $\mathbf{a}_{(i)} \in \mathbb{R}^n$, which is the *row space* of $\mathbf{A}$. The actual dimension of the row space is the *rank* r of $\mathbf{A}$. It turns out that we can always find a "special" orthonormal basis for the row space, which defines a new coordinate system (see Figure 3). If $\mathbf{v}_j$ is a unit-length vector defining one of the axes in the row space then, for each row $\mathbf{a}_{(i)}$, it's j-th coordinate in the new axes is the dot product $\mathbf{a}_{(i)}^T \mathbf{v}_j =: p_{ij}$ so that, if $\mathbf{V} := [\mathbf{v}_1 \cdots \mathbf{v}_r]$ and we define $\mathbf{P} := \mathbf{AV}$, then each row of $\mathbf{P}$ is the same point as the corresponding row of $\mathbf{A}$ but with respect to the new coordinate system. Therefore lengths and distances are preserved, i.e. $\|\mathbf{a}_{(i)}\| = \|\mathbf{p}_{(i)}\|$ and $\|\mathbf{a}_{(i)} - \mathbf{a}_{(j)}\| = \|\mathbf{p}_{(i)} - \mathbf{p}_{(j)}\|$, for all $1 \le i, j \le m$.

However, the new coordinate system is "special" in the following sense. Say we keep only the first k columns of P (let's call this matrix $\tilde{\mathbf{P}}$) thus effectively projecting each point into a space with lower dimension k. Also, rows of the matrix

$$\tilde{\mathbf{A}} := \tilde{\mathbf{P}}\tilde{\mathbf{V}}^T \tag{1}$$

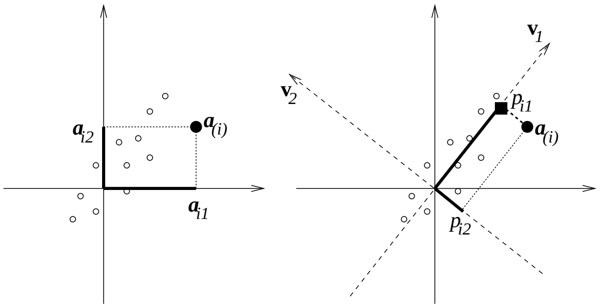


Figure 3: Illustration of SVD (for dimension $n = 2$), with respect to row space. Each point corresponds to a row of the matrix $\mathbf{A}$ and $\mathbf{v}_j$, $j = 1, 2$ are the left singular vectors of $\mathbf{A}$. The square is the one-dimensional approximation of $\mathbf{a}_{(i)}$ by projecting it onto the first singular vector.

are the same points translated back into the original coordinate system of the row space (the square in Figure 3, if $k = 1$), where $\tilde{\mathbf{V}}$ consists of the first k columns of $\mathbf{V}$. Then $\tilde{\mathbf{P}}$ maximizes the sum of squares $\|\tilde{\mathbf{P}}\|_F^2 = \sum_{i,j=1}^{m,k} \tilde{p}_{ij}^2$ or, equivalently, minimizes the sum of squared residual distances (thick dotted line in Figure 3, if $k = 1$), $\|\mathbf{X} - \tilde{\mathbf{X}}\|_F^2 = \sum_{i=1}^{m} \|\mathbf{x}_{(i)} - \tilde{\mathbf{x}}_{(i)}\|^2$.

Therefore, from the point of view of the row space, we can write $\mathbf{A} = \mathbf{P}\mathbf{V}^T$. We can do the same for the column space of $\mathbf{A}$ and get $\mathbf{A} = \mathbf{U}\mathbf{Q}^T$, where $\mathbf{U}$ is also column-orthonormal, like $\mathbf{V}$. It turns out that $\mathbf{U}$ and $\mathbf{V}$ have a special significance, which is formally stated as follows:

Theorem 1 (Singular value decomposition). *Every matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ can be decomposed into*

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where $\mathbf{U} \in \mathbb{R}^{m \times r}$, $\mathbf{V} \in \mathbb{R}^{n \times r}$ and $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, with $r \le \min(m, n)$ the rank of $\mathbf{A}$. The columns $\mathbf{v}_i$ of $\mathbf{V} \equiv [\mathbf{v}_1 \cdots \mathbf{v}_r]$ are the right singular vectors *of $\mathbf{A}$ and they form an orthonormal basis its row space. Similarly, the columns $\mathbf{u}_i$ of $\mathbf{U} \equiv [\mathbf{u}_1 \cdots \mathbf{u}_r]$ are the* left singular vectors *and form a basis of the column space of $\mathbf{A}$. Finally $\mathbf{\Sigma} \equiv \text{diag}[\sigma_1 \cdots \sigma_r]$ is a diagonal matrix with positive values σ_i, called the* singular values *of $\mathbf{A}$.*

From the above, the matrix of projections $\mathbf{P}$ is $\mathbf{P} = \mathbf{U}\mathbf{\Sigma}$. Next, we can formally state the properties of a low-dimensional approximation using the first k singular values (and corresponding singular vectors) of $\mathbf{A}$:

Theorem 2 (Low-rank approximation). *If we keep only the singular vectors corresponding to the k highest singular values ($k < r$), i.e. if $\tilde{\mathbf{U}} := [\mathbf{u}_1\ \mathbf{u}_2 \cdots \mathbf{u}_k]$, $\tilde{\mathbf{V}} := [\mathbf{v}_1\ \mathbf{v}_2 \cdots \mathbf{v}_k]$ and $\tilde{\mathbf{\Sigma}} = \text{diag}[\sigma_1\ \sigma_2 \cdots \sigma_k]$, then $\tilde{\mathbf{A}} = \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^T$ is the best approximation of $\mathbf{A}$, in the sense that it minimizes the error*

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_F^2 := \textstyle\sum_{i,j=1}^{m,n} |a_{ij} - \tilde{a}_{ij}|^2 = \sum_{i=k+1}^{r} \sigma_i^2. \qquad (2)$$

In Equation (2), note the special significance of the singular values for representing the approximation's squared error.

Symbol	Description
$\mathbf{y}$	Vector $\mathbf{A} \in \mathbb{R}^n$ (lowercase bold), always column vectors.
$\mathbf{A}$	Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ (uppercase bold).
$\mathbf{a}_j$	j-th column of matrix $\mathbf{A}$
$\mathbf{a}_{(i)}$	i-th row of matrix $\mathbf{A}$, as a column vector.
$\|\mathbf{y}\|$	Euclidean norm of vector $\mathbf{y}$.
$\|\mathbf{A}\|_F$	Frobenius norm of matrix $\mathbf{A}$, $\|\mathbf{A}\|_F^2 = \sum_{i,j=1}^{m,n} a_{ij}^2$.
$\mathbf{X}$	Time series matrix, with each row corresponding to a timestamp.
$\mathbf{X}^{(w)}$	The delay coordinates matrix corresponding to $\mathbf{X}$, for window w.
$\mathbf{V}^{(w)}$	Right singular vectors of $\mathbf{X}^{(w)}$.
$\mathbf{\Sigma}^{(w)}$	Singular values of $\mathbf{X}^{(w)}$.
$\mathbf{U}^{(w)}$	Left singular vectors of $\mathbf{X}^{(w)}$.
k	Dimension of approximating subspace.
$\tilde{\mathbf{V}}^{(w)}$ $\tilde{\mathbf{\Sigma}}^{(w)}$ $\tilde{\mathbf{U}}^{(w)}$	Same as $\mathbf{V}^{(w)}$, $\mathbf{\Sigma}^{(w)}$, $\mathbf{U}^{(w)}$ but only with with k highest singular values and vectors.
$\tilde{\mathbf{P}}^{(w)}$	Projection of $\mathbf{X}^{(w)}$ onto first k right singular vectors, $\tilde{\mathbf{P}}^{(w)} := \tilde{\mathbf{U}}^{(w)}\tilde{\mathbf{\Sigma}}^{(w)} = \mathbf{X}^{(w)}\tilde{\mathbf{V}}^{(w)}$.
$\tilde{\mathbf{V}}^{(w_0,l)}$ $\tilde{\mathbf{\Sigma}}^{(w_0,l)}$ $\tilde{\mathbf{P}}^{(w_0,l)}$	Hierarchical singular vectors, values and projections, for the k highest singular values.
$\mathbf{V0}^{(w_0,l)}$	Hierarchically estimated patterns (bases).

Table 1: Frequently used notation.

Furthermore, since $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, we have

$$\textstyle\sum_{i=1}^{r} \sigma_i^2 = \|\mathbf{A}\|_F^2 \quad \text{and} \quad \sum_{i=1}^{k} \sigma_i^2 = \|\tilde{\mathbf{A}}\|_F^2. \qquad (3)$$

3. PRELIMINARIES

In this section we give intuitive definitions of the problems we are trying to solve. We also introduce some concepts needed later.

Ideally, a pattern discovery approach for arbitrary time series (where we have limited or no prior knowledge) should satisfy the following requirements:

1. Data-driven: The approximating functions should be derived directly from the data. A fixed, predetermined set of bases or approximating functions may (i) miss some information, and (ii) discovers patterns that are biased towards the "shape" of those functions.
2. Multi-scale: We do not want to restrict examination to a finite, predetermined maximum window size, or we will miss long range trends that occur at time scales longer than the window size.

Many approaches assume a fixed-length, sliding window. In the majority of cases, this restriction cannot be trivially lifted. For example, short-window Fourier cannot say anything about periods larger than the sliding window length. Wavelets are by nature multi-scale, but they still use a fixed set of bases, which is also often hard to choose [27, 32].

The best way to conceptually summarize the main difference of our approach is the following: typical methods first project the data onto "all" bases in a given family (e.g., Fourier, wavelets, etc) and then choose a few coefficients that capture the most information. In contrast, among *all* possible bases we first choose a few bases that are guaranteed to capture the most information and consequently project the data only onto those. However, efficiently determining these few bases and incrementally updating them as new points arrive is a challenging problem.

With respect to the first of the above requirements (data-driven), let us assume that someone gives us a window size. Then, the problem we want to solve (addressed in Section 4) is the following:

PROBLEM 1 (FIXED-WINDOW OPTIMAL PATTERNS). *Given a time series x_t, $t = 1, 2, \dots$ and a window size w, find the patterns that best summarize the series at this window size.*

The patterns are w-dimensional vectors $\mathbf{v}_i \equiv [v_{i,1}, \dots, v_{i,w}]^T \in \mathbb{R}^w$, chosen so that they capture "most" of the information in the series (in a way that we will make precise later).

In practice, however, we do not know *a priori* the right window size. Therefore, with respect to the second requirement (multi-scale), we want to solve the following problem (addressed in Section 5):

PROBLEM 2 (OPTIMAL LOCAL PATTERNS). *Given a time series x_t and a set of windows $\mathcal{W} := \{w_1, w_2, w_3, \dots\}$, find (i) the optimal patterns for each of these, and (ii) the best window w^* to describe the key patterns in the series.*

So how do we go about finding these patterns? An elementary concept we need to introduce is *time-delay coordinates.* We are given a time series x_t, $t = 1, 2, \dots$ with m points seen so far. Intuitively, when looking for patterns of length w, we divide the series in consecutive, non-overlapping subsequences of length w. Thus, if the original series is a $m \times 1$ matrix (not necessarily materialized), we substitute it with a $\frac{m}{w} \times w$ matrix. Instead of m scalar values we now have a sequence of m/w vectors with dimension w. It is natural to look for patterns among these time-delay vectors.

DEFINITION 1 (DELAY COORDINATES). *Given a sequence* $\mathbf{x} \equiv [x_1, x_2, \dots, x_t, \dots, x_m]^T$ *and a* delay *(or window) w, the* delay coordinates *are a $\lceil m/w \rceil \times w$ matrix with the t'-th row equal to* $\mathbf{X}^{(w)}_{(t')} := [x_{(t'-1)w+1}, x_{(t'-1)w+2}, \dots, x_{t'w}]^T$.

Of course, neither $\mathbf{x}$ nor $\mathbf{X}^{(w)}$ need to be fully materialized at any point in time. In practice, we only need to store the last row of $\mathbf{X}^{(w)}$.

Also, note that we choose non-overlapping windows. We could also use overlapping windows, in which case $\mathbf{X}^{(w)}$ would have $m - w + 1$ rows, with row t consisting of values $x_t, x_{t+1}, \dots, x_{t+w}$. In this case, there are some subtle differences [12], akin to the differences between "standard" wavelets and *maximum-overlap* or *redundant* wavelets [27]. However, in practice non-overlapping windows are equally effective for pattern discovery and also lend themselves better to incremental, streaming estimation using limited resources.

More generally, the original time series does not have to be scalar, but can also be vector-valued itself. We still do the same, only each row of $\mathbf{X}^{(w)}$ is now a concatenation of rows of $\mathbf{X}$ (instead of a concatenation of scalar values). More precisely, we construct the general time-delay coordinate matrix as follows:

```
Procedure 1 DELAY ($\mathbf{X} \in \mathbb{R}^{m \times n}$, $w$)
  $m' \leftarrow \lfloor m/w \rfloor$ and $n' \leftarrow nw$
  Output is $\mathbf{X}^{(w)} \in \mathbb{R}^{m' \times n'}$ {not necessarily materialized}
  for $t = 1$ to $m'$ do
    Row $\mathbf{X}^{(w)}_{(t)} \leftarrow$ concatenation of rows
        $\mathbf{X}_{((t-1)w+1)}, \mathbf{X}_{((t-1)w+2)}, \cdots \mathbf{X}_{(tw)}$
  end for
```

Incremental SVD

Batch SVD algorithms are too costly. For an $m \times n$ matrix $\mathbf{A}$, even finding only the highest singular value and corresponding singular vector needs time $O(n^2 m)$, where $n < m$. Aside from computational cost, we also need to incrementally update the SVD as new rows are added to $\mathbf{A}$.

SVD update algorithms such as [3, 13] can support both row additions as well as deletions. However, besides the right singular vectors $\mathbf{v}_i$, both of these approaches need to store the left singular vectors $\mathbf{u}_i$ (whose size is proportional to the time series length in our case).

```
Algorithm 1 INCREMENTALSVD ($\mathbf{A}$, $k$)
  for $i = 1$ to $k$ do
    Initialize $\mathbf{v}_i$ to unit vectors, $\mathbf{v}_i \leftarrow \mathbf{i}_i$
    Initialize $\sigma_i^2$ to small positive value, $\sigma_i^2 \leftarrow \epsilon$
  end for
  for all new rows $\mathbf{a}_{(t+1)}$ do
    Initialize $\bar{\mathbf{a}} \leftarrow \mathbf{a}_{(t+1)}$
    for $i = 1$ to $k$ do
      $y_i \leftarrow \mathbf{v}_i^T \bar{\mathbf{a}}$ {projection onto $\mathbf{v}_i$}
      $\sigma_i^2 \leftarrow \sigma_i^2 + y_i^2$ {energy $\propto$ singular value}
      $\mathbf{e}_i \leftarrow \bar{\mathbf{a}} - y_i \mathbf{v}_i$ {error, $\mathbf{e}_i \perp \mathbf{v}_i$}
      $\mathbf{v}_i \leftarrow \mathbf{v}_i + \frac{1}{\sigma_i^2} y_i \mathbf{e}_i$ {update singular vector estimate}
      $\bar{\mathbf{a}} \leftarrow \bar{\mathbf{a}} - y_i \mathbf{v}_i$ {repeat with remainder of $\mathbf{a}_{(t+1)}$}
    end for
    $\mathbf{p}_{(t+1)} \leftarrow \mathbf{V}^T \mathbf{a}_{(t+1)}$ {final low-dim. projection of $\mathbf{a}_{(t+1)}$}
  end for
```

We chose an SVD update algorithm (shown above, for given number k of singular values and vectors) that has been successfully applied in streams [25, 34]. Its accuracy is still very good, while it does not need to store the left singular vectors. Since our goal is to find patterns at multiple scales without an upper bound on the window size, this is a more suitable choice. Furthermore, if we need to place more emphasis on recent trends, it is rather straightforward to incorporate an exponential forgetting scheme, which works well in practice [25]. For each new row, the algorithm updates $k \cdot n$ numbers, so total space requirements are $O(nk)$ and the time per update is also $O(nk)$. Finally, the incremental update algorithms need only the observed values and

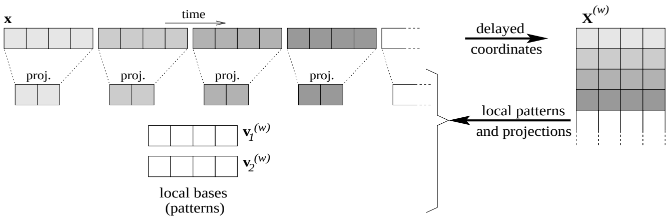


Figure 4: Illustration of local patterns for a fixed window (here, $w = 4$).

can therefore easily handle missing values by imputing them based on current estimates of the singular vectors [25].

4. LOCALLY OPTIMAL PATTERNS

We now have all the pieces in place to answer the first question: for a given window w, how do we find the locally optimal patterns? Figure 4 illustrates the main idea. Starting with the original time series $\mathbf{x}$, we transfer to time-delay coordinates $\mathbf{X}^{(w)}$. The local patterns are the right singular vectors of $\mathbf{X}^{(w)}$, which are optimal in the sense that they minimize the total squared approximation error of the rows $\mathbf{X}^{(w)}_{(i)}$. The detailed algorithm is shown below.

Algorithm 2 LocalPattern ($\mathbf{x} \in \mathbb{R}^m$, w, $k = 3$)

```
    Use delay coord. $\mathbf{X}^{(w)} \leftarrow \textsc{Delay}(\mathbf{x}, w)$
    Compute SVD of $\mathbf{X}^{(w)} = \mathbf{U}^{(w)}\mathbf{\Sigma}^{(w)}\mathbf{V}^{(w)}$
    Local patterns are $\mathbf{v}_1^{(w)}, \ldots, \mathbf{v}_k^{(w)}$
    Power is $\pi^{(w)} \leftarrow \sum_{i=k+1}^{w} \sigma_i^2/w = \big(\sum_{t=1}^{m} x_t^2 - \sum_{i=1}^{k} \sigma_i^2\big)/w$
    $\tilde{\mathbf{P}}^{(w)} \leftarrow \tilde{\mathbf{U}}^{(w)}\tilde{\mathbf{\Sigma}}^{(w)}$ {low-dim. proj. onto local patterns}
    return $\tilde{\mathbf{V}}^{(w)}$, $\tilde{\mathbf{P}}^{(w)}$, $\tilde{\mathbf{\Sigma}}^{(w)}$ and $\pi^{(w)}$
```

For now, the projections $\tilde{\mathbf{P}}^{(w)}$ onto the local patterns $\tilde{\mathbf{v}}_{(i)}$ are not needed, but we will use them later, in Section 5. Also, note that LocalPattern can be applied in general to n-dimensional vector-valued series. The pseudocode is the same, since Delay can also operate on matrices $\mathbf{X} \in \mathbb{R}^{m \times n}$. The reason for this will also become clear in Section 5, but for now it suffices to observe that the first argument of LocalPattern may be a matrix, with one row $\mathbf{x}_{(t)} \in \mathbb{R}^n$ per timestamp $t = 1, 2, \ldots, m$.

When computing the SVD, we really need only the highest k singular values and the corresponding singular vectors, because we only return $\tilde{\mathbf{V}}^{(w)}$ and $\tilde{\mathbf{P}}^{(w)}$. Therefore, we can avoid computing the full SVD and use somewhat more efficient algorithms, just for the quantities we actually need.

Also, note that $\tilde{\mathbf{\Sigma}}^{(w)}$ can be computed from $\tilde{\mathbf{P}}^{(w)}$, since by construction

$$\sigma_i^2 = \|\mathbf{p}_i\|^2 = \textstyle\sum_{j=1}^{m} p_{ji}^2. \qquad (4)$$

However, we return these separately, which avoids duplicate computation. More importantly, when we later present our streaming approach, we won't be materializing $\tilde{\mathbf{P}}^{(w)}$. Furthermore, Equation (4) does not hold exactly for the estimates returned by IncrementalSVD and it is better to use the estimates of the singular values σ_i^2 computed as part of IncrementalSVD.

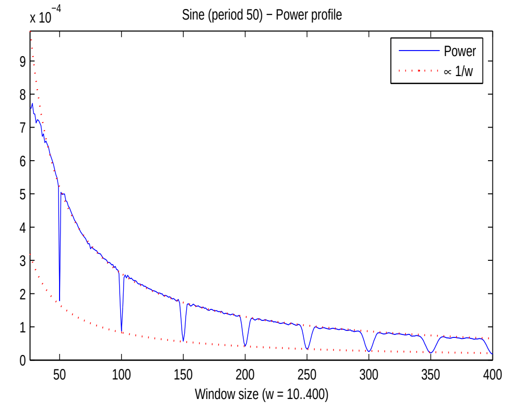


Figure 5: Power profile of sine wave $x_t = \sin(2\pi t/50) + \epsilon_t$, with Gaussian noise $\epsilon_t \sim \mathcal{N}(5, 0.5)$.

Number of patterns

We use a default value of $k = 3$ local patterns, although we could also employ an energy-based criterion to choose k, similar to [25]. However, three (or fewer) patterns are sufficient, since it can be shown that the first pattern captures the average trend (aperiodic, if present) and the next two capture the main low-frequency and high-frequency periodic trends [12]. The periodic trends do not have to be sinusoidal and in real data they rarely are.

Complexity

For a single window w, batch algorithms for computing the k highest singular values of a $m \times n$ matrix ($n < m$) are $O(kmn^2)$. Therefore, for window size w the time complexity is $O\big(k\frac{t}{w}w^2\big) = O(ktw)$. If we wish to compute the local patterns for all windows up to $w_{\max} = O(t)$, then the total complexity is $O(kt^3)$. In Section 5 and 6 we show how we can reduce this dramatically.

4.1 Power profile

Next, let's assume we have optimal local patterns for a number of different window sizes. Which of these windows is the best to describe the main trends? Intuitively, the key idea is that if there is a trend that repeats with a period of T, then different subsequences in the time-delay coordinate space should be highly correlated when $w \approx T$. Although the trends can be arbitrary, we illustrate the intuition with a sine wave, in Figure 5. The plot shows the squared approximation error per window element, using $k = 1$ pattern on a sine wave with period $T = 50$. As expected, for window size $w = T = 50$ the approximation error drops sharply and essentially corresponds to the Gaussian noise floor. Naturally, for windows $w = iT$ that are multiples of T the error also drops. Finally, observe that the error for all windows is proportional to $\frac{1}{w}$, since it is *per window element*. Eventually, for window size equal to the length of the entire time series $w = m$ (not shown in Figure 5, where $m = 2000$), we get $\pi^{(m)} = 0$ since first pattern is the only singular vector, which coincides with the series itself, so the residual error is zero.

Formally, the squared approximation error of the time-delay matrix $\mathbf{X}^{(w)}$ is

$$\epsilon^{(w)} := \textstyle\sum_t \|\tilde{\mathbf{x}}^{(w)}_{(t)} - \mathbf{x}^{(w)}_{(t)}\|^2 = \|\tilde{\mathbf{X}}^{(w)} - \mathbf{X}^{(w)}\|_F^2,$$

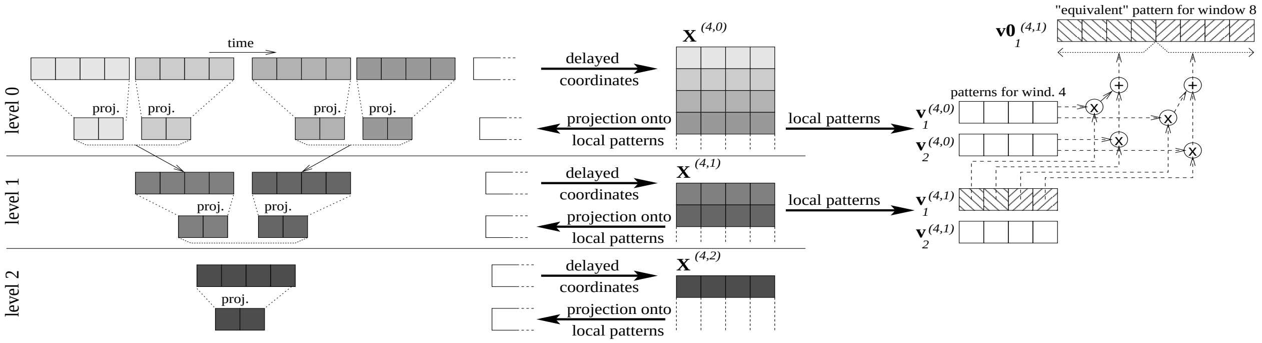


Figure 6: Multi-scale pattern discovery (hierarchical, $w_0 = 4$, $W = 2$, $k = 2$).

where $\tilde{\mathbf{X}}^{(w)} := \tilde{\mathbf{P}}^{(w)}(\tilde{\mathbf{V}}^{(w)})^T$ is the reconstruction (see Equation (1)). From Equation (2) and 3, we have

$$\epsilon^{(w)} = \|\mathbf{X}^{(w)}\|_F^2 - \|\tilde{\mathbf{P}}^{(w)}\|_F^2 \approx \|\mathbf{x}\|^2 - \sum_{i=1}^{k} \big(\sigma_i^{(w)}\big)^2.$$

Based on this, we define the power, which is an estimate of the error per window element.

Definition 2 (Power profile $\pi^{(w)}$). *For a given number of patterns ($k = 2$ or 3) and for any window size w, the power profile is the sequence defined by*

$$\pi^{(w)} := \frac{\epsilon^{(w)}}{w}. \tag{5}$$

More precisely, this is an estimate of the variance per dimension, assuming that the discarded dimensions correspond to isotropic Gaussian noise (i.e., uncorrelated with same variance in each dimension) [31]. As explained, this will be much lower when $w = T$, where T is the period of an arbitrary main trend.

The following lemma follows from the above observations. Note that the conclusion is valid both ways, i.e., perfect copies imply zero power and vice versa. Also, the conclusion holds regardless of alignment (i.e., the periodic part does not have to start at the begining of a windowed subsequence). A change in alignment will only affect the phase of the discovered local patterns, but not their shape or the reconstruction accuracy.

Observation 1 (Zero power). *If $\mathbf{x} \in \mathbb{R}^t$ consists of exact copies of a subsequence of length T then, for every number of patterns $k = 1, 2, \ldots$ and at each multiple of T, we have $\pi^{(iT)} = 0$, $i = 1, 2, \ldots$, and vice versa.*

In general, if the trend does not consist of exact copies, the power will not be zero, but it will still exhibit a sharp drop. We exploit precisely this fact to choose the "right" window.

Choosing the window

Next, we state the steps for interpreting the power profile to choose the appropriate window that best captures the main trends:

1. Compute the power profile $\pi^{(w)}$ versus w.
2. Look for the first window w_0^* that exhibits a sharp drop in $\pi^{(w_0^*)}$ and ignore all other drops occurring at windows $w \approx iw_0^*$, $i = 2, 3, \ldots$ that are approximately multiples of w_0^*.
3. If there are several sharp drops at windows w_i^* that are *not* multiples of each other, then any of these is suitable. We simply choose the smallest one; alternatively, we could choose based on prior knowledge about the domain if available, but that is not necessary.
4. If there are no sharp drops, then no strong periodic/cyclic components are present. However, the local patterns at any window can still be examined to gain a picture of the time series behavior.

In Section 7 we illustrate that this window selection criterion works very well in practice and selects windows that correspond to the true cycles in the data. Beyond this, the discovered patterns themselves reveal the trends in detail.

5. MULTIPLE-SCALE PATTERNS

In this section we tackle the second question: how do we efficiently compute the optimal local patterns for multiple windows (as well as the associated power profiles), so as to quickly zero in to the "best" window size? First, we can choose a geometric progression of window sizes: rather than estimating the patterns for windows of length w_0, $w_0 + 1$, $w_0 + 2$, $w_0 + 3, \ldots$, we estimate them for windows of w_0, $2w_0$, $4w_0, \ldots$ or, more generally, for windows of length $w_l := w_0 \cdot W^l$ for $l = 0, 1, 2, \ldots$. Thus, the size of the window set $\mathcal{W}$ we need to examine is dramatically reduced. Still, this is (i) computationally expensive (for each window we still need $O(ktw)$ time and, even worse, (ii) still requires buffering all the points (needed for large window sizes, close to the time series length). Next, we show how we can reduce complexity even further.

5.1 Hierarchical SVD

The main idea of our approach to solve this problem is shown in Figure 6. Let us assume that we have, say $k = 2$ local patterns for a window size of $w_0 = 100$ and we want to compute the patterns for window $w^{(100,1)} = 100 \cdot 2^1 = 200$. The naive approach is to construct $\mathbf{X}^{(200)}$ from scratch and compute the SVD. However, we can reuse the patterns found from $\mathbf{X}^{(100)}$. Using the $k = 2$ patterns $\mathbf{v}_1^{(100)}$ and $\mathbf{v}_2^{(100)}$ we can reduce the first $w_0 = 100$ points $x_1, x_2, \ldots, x_{100}$ into just two points, namely their projections $p_{1,1}^{(100)}$ and $p_{1,2}^{(100)}$ onto $\mathbf{v}_1^{(100)}$ and $\mathbf{v}_2^{(100)}$, respectively. Similarly, we can reduce the next $w_0 = 100$ points $x_{101}, x_{102}, \ldots, x_{200}$ also into two numbers, $p_{2,1}^{(100)}$ and $p_{2,2}^{(100)}$, and so on. These projections, by construction, approximate the original series well. Therefore, we can represent the first row $\mathbf{x}^{(200)}_{(1)} \equiv [x_1, \dots, x_{200}]^T \in \mathbb{R}^{200}$ of $\mathbf{X}^{(200)}$ with just four numbers, $\mathbf{x}^{(100,1)}_{(1)} \equiv \left[p^{(100)}_{1,1}, p^{(100)}_{1,2}, p^{(100)}_{2,1}, p^{(100)}_{2,2}\right]^T \in \mathbb{R}^4$. Doing the same for the other rows of $\mathbf{X}^{(200)}$, we construct a matrix $\mathbf{X}^{(100,1)}$ with just $n = 4$ columns, which is a very good approximation of $\mathbf{X}^{(200)}$.Consequently, we compute the local patterns using $\mathbf{X}^{(100,1)}$ instead of $\mathbf{X}^{(200)}$. Repeating this process recursively, we can find the local patterns for a window $w^{(100,2)} = 100 \cdot 2^2 = 400$ and so on.

Definition 3 (Level-(w_0, l) window). *The level-(w_0, l) window corresponds to an original window size (or* scale*) $w_l := w_0 \cdot W^l$. Patterns at each level l are found recursively, using patterns from the previous level $l - 1$.*

In the above example, we have $w_0 = 100$ and $l = 0, 1$. Since w_0 and W are fixed for a particular sequence of scales w_l, we will simply refer to level-l windows and patterns. The recursive construction is based on the level-l delay matrix and corresponding patterns.

Definition 4 (Level-l delay matrix $\mathbf{X}^{(w_0,l)}$). *Given a starting window w_0 and a scale factor W, the level-l delay matrix is simply $\mathbf{X}^{(w_0,0)} := \mathbf{X}^{(w_0)}$ for $l = 0$ and for $l = 1, 2, \dots$ it is recursively defined by*

$$\mathbf{X}^{(w_0,l)} := \textsc{Delay}\big(\tilde{\mathbf{P}}^{(w_0,l-1)}, W\big),$$

where $\tilde{\mathbf{P}}^{(w_0,l)} := \mathbf{X}^{(w_0,l)}\tilde{\mathbf{V}}^{(w_0,l)}$ is the projection onto the level-l patterns $\tilde{\mathbf{V}}^{(w_0,l)}$ which are found based on $\mathbf{X}^{(w_0,l)}$. The level-l delay matrix is an approximation of the delay matrix $\mathbf{X}^{(w_l)}$ for window size $w_l = w_0 W^l$.

In our example, the patterns extracted from $\mathbf{X}^{(100,1)}$ are four-dimensional vectors, $\mathbf{v}^{(100,1)}_i \in \mathbb{R}^4$, whereas the patterns for $\mathbf{X}^{(200)}$ would be 200-dimensional vectors $\mathbf{v}^{(200)}_i \in \mathbb{R}^{200}$. However, we can appropriately combine $\mathbf{v}^{(100,1)}_i$ and $\mathbf{v}^{(100,0)}_i \equiv \mathbf{v}^{(100)}_i$ to estimate $\mathbf{v}^{(200)}_i$.

Definition 5 (Level-l local pattern $\mathbf{v0}^{(w_0,l)}_i$). *The level-l pattern $\mathbf{v0}^{(w_0,l)}_i$, for all $i = 1, 2, \dots, k$, corresponding to a window of $w_l = w_0 W^l$ is simply $\mathbf{v0}^{(w_0,0)}_i := \mathbf{v}^{(w_0)}_i$ for $l = 0$ and for $l = 1, 2, \dots$ it is defined recursively by*

$$\mathbf{v0}^{(w_0,l)}_i[(j-1)w_{l-1} + 1 : jw_{l-1}] := \mathbf{V0}^{(w_0,l-1)}\big(\mathbf{v}^{(w_0,l)}_i[(j-1)k + 1 : jk]\big), \quad (6)$$

for $j = 1, 2, \dots, W$. It is an approximation of the local patterns $\mathbf{v}^{(w_l)}_i$ of the original delay matrix $\mathbf{X}^{(w_l)}$, for window size $w_l = w_0 W^l$.

Consider $\mathbf{v0}^{(100,1)}_1$ in our example. The first $k = 2$ out of $kW = 4$ numbers in $\mathbf{v}^{(100,1)}_1$ approximate the patterns among the 2-dimensional vectors $\mathbf{p}^{(100,0)}_{(j)}$, which in turn capture patterns among the 100-dimensional vectors $\mathbf{x}^{(100,0)}_{(i)}$ of the original time-delay matrix. Thus, but forming the appropriate linear combination of the 100-dimensional patterns $\mathbf{v}^{(100,0)}_i \equiv \mathbf{v0}^{(100,0)}_i$ (i.e., the columns of $\tilde{\mathbf{V}}^{(100,0)} \equiv \mathbf{V0}^{(100,0)}$), weighted according to $\mathbf{v}^{(100,1)}_1[1:2]$, we can construct the first half of the 200-dimensional pattern $\mathbf{v0}^{(100,1)}_1[1 : 100]$ (left-slanted entries in Figure 6). Similarly, a linear combination of the columns of $\tilde{\mathbf{V}}^{(100,0)} \equiv \mathbf{V0}^{(100,0)}$ weighted according to $\mathbf{v}^{(100,1)}_1[3 : 4]$ gives us the second half of the 200-dimensional pattern $\mathbf{v0}^{(100,1)}_1[101 : 200]$ (right-slanted entries in Figure 6). For level $l = 2$ we similarly combine the columns of $\mathbf{V0}^{(100,1)}$ according to $\mathbf{v}^{(100,2)}_1[1:2]$ (for the first half, $\mathbf{v0}^{(100,2)}_1[1 : 200]$) and to $\mathbf{v}^{(100,2)}_[3 : 4]$ (for the second half, $\mathbf{v0}^{(100,2)}_1[201 : 400]$) and so on, for the higher levels.

Lemma 1 (Orthonormality of $\mathbf{v0}^{(w_0,l)}_i$). *We have $\|\mathbf{v0}^{(w_0,l)}_i\| = 1$ and, for $i \neq j$, $\big(\mathbf{v0}^{(w_0,l)}_i\big)^T\big(\mathbf{v0}^{(w_0,l)}_j\big) = 0$, where $i, j = 1, 2, \dots, k$.*

Proof. For level $l = 0$ they are orthonormal since they coincide with the original patterns $\mathbf{v}^{(w_0)}_i$ which are by construction orthonormal. We proceed by induction on the level $l \geq 1$. Without loss of generality, assume that $k = 2$ and, for brevity, let $\mathbf{B} \equiv \mathbf{V0}^{(w_0,l-1)}$ and $\mathbf{b}_{i,1} \equiv \mathbf{v}^{(w_0,l)}_i[1 : k]$, $\mathbf{b}_{i,2} \equiv \mathbf{v}^{(w_0,l)}_i[k+1 : k]$, so that $\mathbf{v}^{(w_0,l)}_i = [\mathbf{b}_{i,1}, \mathbf{b}_{i,2}]$. Then

$$\begin{aligned} \|\mathbf{v0}^{(w_0,l)}_i\|^2 &= [\mathbf{B}\mathbf{b}_{i,1}\ \ \mathbf{B}\mathbf{b}_{i,2}]^2 = \|\mathbf{B}\mathbf{b}_{i,1}\|^2 + \|\mathbf{B}\mathbf{b}_{i,2}\|^2 \\ &= \|\mathbf{b}_{i,1}\|^2 + \|\mathbf{b}_{i,2}\|^2 = \|\mathbf{v}^{(w_0,l)}_i\|^2 = 1, \end{aligned}$$

and

$$\begin{aligned} \big(\mathbf{v0}^{(w_0,l)}_i\big)^T\big(\mathbf{v0}^{(w_0,l)}_j\big) &= [\mathbf{B}\mathbf{b}_{i,1}\ \ \mathbf{B}\mathbf{b}_{i,2}]^T[\mathbf{B}\mathbf{b}_{j,1}\ \ \mathbf{B}\mathbf{b}_{j,2}] \\ &= \mathbf{b}^T_{i,1}\mathbf{B}^T\mathbf{B}\mathbf{b}_{j,1} + \mathbf{b}^T_{i,2}\mathbf{B}^T\mathbf{B}\mathbf{b}_{j,2} \\ &= \mathbf{b}^T_{i,1}\mathbf{b}_{j,1} + \mathbf{b}^T_{i,2}\mathbf{b}_{j,2} \\ &= \big(\mathbf{v}^{(w_0,l)}_i\big)^T\big(\mathbf{v}^{(w_0,l)}_j\big) = 0, \end{aligned}$$

since $\mathbf{B}$ preserves dot products as an orthonormal matrix (by inductive hypothesis) and $\mathbf{v}^{(w_0,l)}_i$ are orthonormal by construction. □

The detailed hierarchical SVD algorithm is shown below. In practice, the maximum level L is determined based on the length m of the time series so far, $L \approx \log_W(m/w_0)$.

Algorithm 3 Hierarchical ($\mathbf{x} \in \mathbb{R}^m$, w_0, W, L, $k = 6$)

```
{Start with level $l = 0$, corresponding to window $w_0$}
$\tilde{\mathbf{V}}^{(w_0,0)}$, $\tilde{\mathbf{P}}^{(w_0,0)}$, $\tilde{\mathbf{\Sigma}}^{(w_0,0)}$, $\pi^{(w_0,0)} \leftarrow$
                    LocalPattern($\mathbf{x}, w_0, k$)
{Levels $l$, corresponding to window $w_l = w_0 \cdot W^l$}
for level $l = 1$ to $L$ do
    $\tilde{\mathbf{V}}^{(w_0,l)}$, $\tilde{\mathbf{P}}^{(w_0,l)}$, $\tilde{\mathbf{\Sigma}}^{(w_0,l)}$, $\pi^{(w_0,l)} \leftarrow$
                    LocalPattern($\tilde{\mathbf{P}}^{(w_0,l-1)}, W, k$)
    Compute patterns $\mathbf{v0}^{(w_0,l)}_i$ for window size $w_l$ are based on Equation (6)
end for
```

Choosing the initial window

The initial window w_0 has some impact on the quality of the approximations. This also depends on the relationship of k to w_0 (the larger k is, the better the approximatio: and if $k = w_0$ then $\tilde{\mathbf{P}}^{(w_0,1)} = \mathbf{X}^{(w_0)}$, i.e., no informatio: is discarded at the first level). However, we want k to b relatively small since, as we will see, it determines the buffe ing requirements of the streaming approach. Hence, we fi $k = 6$. We found that this simple choice works well fo real-world sequences, but we could also use energy-base thresholding [18], which can be done incrementally.

If w_0 is too small, then we discard too much of the variance too early. If w_0 is unnecessarily big, this increases buffering requirements and the benefits of the hierarchical approach diminish. In practice, a good compromise is a value in the range $10 \leq w_0 \leq 20$.

Finally, out of the six patterns we keep per level, the first two or three are of interest and reported to the user, as explained in Section 4. The remaining are kept to ensure that $\mathbf{X}^{(w_0,l)}$ is a good approximation of $\mathbf{X}^{(w_l)}$.

Choosing the scales

As discussed in Section 4.1, if there is a sharp drop of $\pi^{(T)}$ at window $w = T$, then we will also observe drops at multiples $w = iT$, $i = 2, 3, \ldots$. Therefore, we choose a few different starting windows w_0 and scale factors W that are relatively prime to each other. In practice, the following set of three choices is sufficient to quickly zero in on the best windows and the associated optimal local patterns:

$$k = 6 \quad \text{and} \quad (w_0, W) \in \{(9,2), (10,2), (15,3)\}$$

Complexity

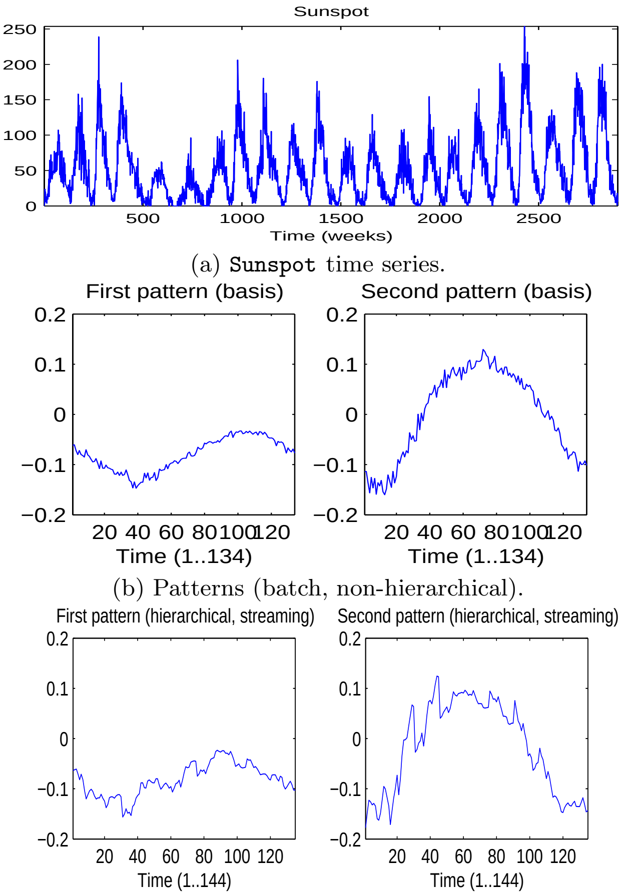
For a total of $L \approx \log_W(t/w_0) = O(\log t)$ levels we have to compute the first k singular values and vectors of $\mathbf{X}^{(w_0,l)} \in \mathbb{R}^{t/(w_0W^l)\times Wk}$, for $l = 1, 2, \ldots$. A batch SVD algorithm requires time $O\big(k\cdot(Wk)^2\cdot\frac{t}{w_0W^l}\big)$, which is $O\big(\frac{W^2k^2t}{W^l}\big)$ since $k < w_0$. Summing over $l = 1, \ldots, L$, we get $O(W^2k^2t)$. Finally, for $l = 0$, we need $O\big(k\cdot w_0^2\frac{t}{w_0}\big) = O(kw_0t)$. Thus, the total complexity is $O(W^2k^2t + kw_0t)$. Since W and w_0 are fixed, we finally have the following

Lemma 2 (Batch hierarchical complexity). *The total time for the hierarchical approach is $O(k^2t)$, i.e., linear with respect to the time series length.*

This is a big improvement over the $O(t^3k)$ time of the non-hierarchical approach. However, we still need to buffer all the points. We address this problem in the next section.

6. STREAMING COMPUTATION

In this section we explain how to perform the necessary computations in an incremental, streaming fashion. We designed our models precisely to allow this step. The main idea is that we recursively invoke only one iteration of each loop in IncrementalSVD (for LocalPattern) and in Hierarchical, as soon as the necessary number of points has arrived. Subsequently, we can discard these points and proceed with the next non-overlapping window.



(a) `Sunspot` time series.

(b) Patterns (batch, non-hierarchical).

(c) Patterns (streaming, hierarchical).

Figure 7: `Sunspot`, best selected window (about 11 years) and corresponding representative trends.

Modifying LocalPattern

We buffer consecutive points of $\mathbf{x}$ (or, in general, rows of $\mathbf{X}$) until we accumulate w of them, forming one row of $\mathbf{X}^{(w)}$. At that point, we can perform one iteration of the outer loop in IncrementalSVD to update all k local patterns. Then, we can discard the w points (or rows) and proceed with the next w. Also, since on higher levels the number of points for SVD may be small and close to k, we may choose to initially buffer just the first k rows of $\mathbf{X}^{(w)}$ and use them to bootstrap the SVD estimates, which we subsequently update as described.

Modifying Hierarchical

For level $l = 0$ we use the modified LocalPattern on the original series, as above. However, we also store the k projections onto the level-0 patterns. We buffer W consecutive sets of these projections and as soon as kW values accumulate, we update the k local patterns for level $l = 1$. Then we can discard the kW projections from level-0, but we keep the k level-1 projections. We proceed in the same way for all other levels $l \geq 2$.

Complexity

Compared to the batch computation, we need $O\big(k\cdot Wk\cdot\frac{t}{w_0W^l}\big) = O\big(\frac{kt}{W^{l-1}}\big)$ time to compute the first k singular values and vectors of $\mathbf{X}^{(w_0,l)}$ for $l = 1, 2, \ldots$. For $l = 0$ we need $O\big(k\cdot w_0\cdot\frac{t}{w_0}\big) = O(kt)$ time. Summing over $l =$

$0, 1, \ldots, L$ we get $O(kt)$. With respect to space, we need to buffer w_0 points for $l = 0$ and Wk points for each of the remaining $L = O(\log t)$ levels, for a total of $O(k \log t)$. Therefore, we have the following

Lemma 3 (Streaming, hierarchical complexity). *Amortized cost is $O(k)$ per incoming point and total space is $O(k \log t)$.*

Since $k = 6$, the update time is constant per incoming point and the space requirements grow logarithmically with respect to the size t of the series. Table 2 summarizes the time and space complexity for each approach.

	Time		Space	
	Non-hier.	Hier.	Non-hier.	Hier
Batch	$O(t^3k)$	$O(tk^2)$	all	all
Incremental	$O(t^2k)$	$O(tk)$	$O(t)$	$O(k \log t)$

Table 2: Summary of time and space complexity.

7. EXPERIMENTAL EVALUATION

In this section we demonstrate the effectiveness of our approach on real data. The questions we want to answer are:

1. Do our models for mining locally optimal patterns correctly capture the best window size?
2. Are the patterns themselves accurate and intuitive?
3. How does the quality of the much more efficient, hierarchical streaming approach compare to the exact computation?

We performed all experiments in Matlab, using the standard functions for batch SVD. Table 3 briefly describes the datasets we use.

Dataset	Size	Description
`Automobile`	59370	Automobile counts on interstate
`Sunspot`	2904	Sunspot intensities (years 1755–1996)
`Mote`	7712	Light intensity data

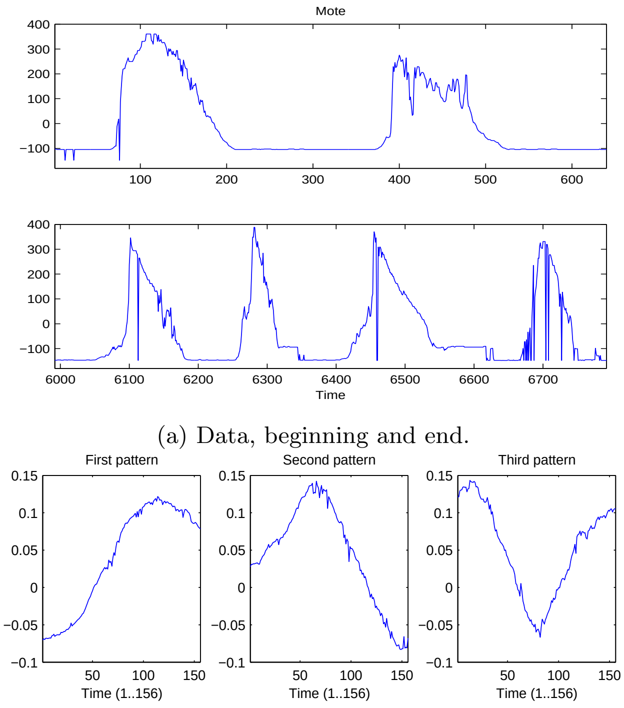
Table 3: Description of datasets.

Table 4 shows the best windows identified from the power profiles. Figure 9 shows the exact power profiles computed using the batch, non-hierarchical approach and Figure 10 shows the power profiles estimated using the streaming, hierarchical approach.

	Best window		
	`Sunspot`	`Mote`	`Auto`
Batch	134, 120, 142	156, 166, 314	4000
Streaming	144	160, 320	3645

Table 4: Best windows—see also Figures 9 and 10.

`Automobile`
The results on this dataset were already presented in Section 1. Our method correctly captures the right window for the main trend and also an accurate picture of the typical daily pattern. The patterns discovered both by the batch and the streaming approach are very close to each other and equally useful.



(a) Data, beginning and end.

(b) Patterns

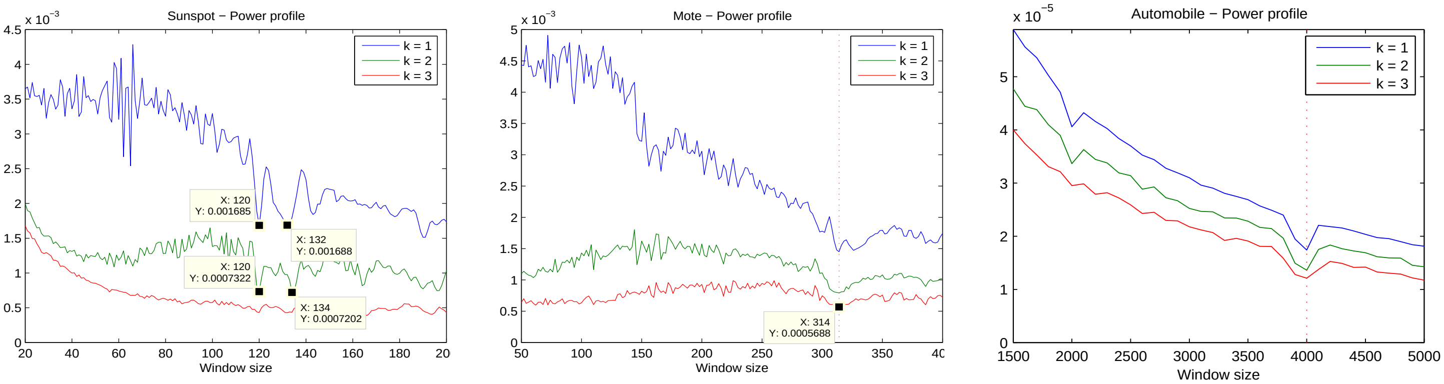
Figure 8: `Mote` data and patterns.

`Sunspot`
This dataset consists of sunspot intensities for a period of over 200 years, measured at a monthly interval (see Figure 7a). This phenomenon exhibits a cyclic pattern, with cycles lasting about 11 years (varying slightly between 10 and 12). Indeed, the best window identified by our model is 134 months (i.e., 11 years and 2 months), with 120 (i.e., 10 years) and 142 (i.e., 11 years and 10 months) being close runner-ups. The fast, streaming approach identifies 144 (or 12 years) as the best window, which is the scale closest to the average cycle duration. Both the exact patterns (Figure 7b) as well as those estimated by the fast streaming approach (Figure 7c) provide an accurate picture of what typically happens within a cycle.

`Mote`
This dataset consists of light intensity measurements collected in a lab by a wireless sensor device. Like `Sunspot`, it exhibits a varying cycle but with much bigger changes. Originally a cycle lasts about 310–320 time ticks, which progressively shortens to a cycle of about half that length (140–160), as shown in Figure 8a. The power profile drops sharply in the vicinity of both cycle lengths. If we examine the patterns, they show the same shape as the data. The first two capture the low-frequency component with peaks located about half a window size apart, while the third pattern captures the higher-frequency component with two peaks (see Figure 8b). Patterns by the fast, streaming approach convey the same information.

7.1 Quality of streaming approach

In this section we compare the fast, streaming and hierarchical approach against the exact, batch and non-hierarchical. Even though the fast streaming approach potentially introduces errors via (i) the hierarchical approximation, and (ii)



(a) Sunspot (b) Mote (c) Automobile

Figure 9: Power profiles (non-hierarchical, batch).

the incremental update of the SVD, we show that it still achieves very good results. The quality should be compared against the significant speedups in Table 5.

	Auto	Sunspot	Mote
Exact	15252	2.0	11.9
Streaming	6.5	0.6	1.3
Ratio	×2346	×3.3	×9.1
Length	59370	2904	7712
$w_{\max}$	5000	400	600

Table 5: Wall-clock times (in seconds). The maximum window $w_{\max}$ for the exact approach and the length of each series are also shown.

First, the streaming approach correctly zeros in to the best windows, which generally correspond to the scale that is closest to the "true" best windows identified by the exact, batch approach (see Table 4).

Our next goal is to characterize the quality of the patterns themselves, at the best windows that each approach identifies. To that end, we compare the reconstructions of the original time series $\mathbf{x}$ for the two methods. If $\mathbf{v}_i^{(w^*)}$ are the optimal patterns for the best window w^* (or $\mathbf{v0}_i^{(w_0^*,l^*)}$ for the streaming approach, with corresponding window $w_l^* = w_0^* \cdot W^{l^*}$), we first zero-pad $\mathbf{x}$ into $\mathbf{x}_0$ so its length is a multiple of w^*, then compute the local pattern projections $\tilde{\mathbf{P}}^{(w^*)} = \textsc{Delay}(\mathbf{x}_0, w^*)\tilde{\mathbf{V}}^{(w^*)}$, project back to the original delay coordinates $\tilde{\mathbf{X}}^{(w^*)} = \tilde{\mathbf{P}}^{(w^*)}\left(\tilde{\mathbf{V}}^{(w^*)}\right)^T$. Finally, we concatenate the rows of $\tilde{\mathbf{X}}^{(w^*)}$ into the reconstruction $\tilde{\mathbf{x}}$ of $\mathbf{x}$, removing the elements corresponding to the zero-padding. The quality measures we use are

$$Q := \frac{\|\mathbf{x}\|^2 - \|\tilde{\mathbf{x}}_{\text{stream}} - \mathbf{x}\|^2}{\|\mathbf{x}\|^2 - \|\tilde{\mathbf{x}}_{\text{exact}} - \mathbf{x}\|^2} \quad \text{and} \quad C := \frac{|\tilde{\mathbf{x}}_{\text{stream}}^T \tilde{\mathbf{x}}_{\text{exact}}|}{\|\tilde{\mathbf{x}}_{\text{stream}}\| \, \|\tilde{\mathbf{x}}_{\text{exact}}\|}.$$

The first one is a measure of the information retained, with respect to the total squared error: we compare the fraction of information retained by the streaming approach, $\left(\|\mathbf{x}\|^2 - \|\tilde{\mathbf{x}}_{\text{stream}} - \mathbf{x}\|^2\right)/\|\mathbf{x}\|^2$, to that retained by the exact approach, $\left(\|\mathbf{x}\|^2 - \|\tilde{\mathbf{x}}_{\text{exact}} - \mathbf{x}\|^2\right)/\|\mathbf{x}\|^2$. The second is simply the cosine similarity of the two reconstructions, which penalizes small "time shifts" less and more accurately reflects how close are the overall shapes of the reconstructions.

We use the reconstruction $\tilde{\mathbf{x}}_{\text{exact}}$ as our baseline. For pattern discovery, using the original series $\mathbf{x}$ itself as the baseline is undesirable, since that contains noise and other irregularities that should not be part of the discovered patterns.

Finally, we show the cosine similarity of the patterns themselves, for the best window w_l^* identified by the streaming approach. This characterizes how close the direction of the hierarchically and incrementally estimated singular vectors is to the direction of the "true" singular vectors. Table 6 shows the results. Figures 2 and 7 allow visual comparison of the patterns (at slightly different best windows). In summary, the streaming approach's patterns capture all the essential information, while requiring 1–4 orders of magnitude less time and 1–2 orders of magnitude less space.

Dataset	Auto	Sunspot	Mote
Quality ratio	0.97	0.89	0.84
Cosine	0.98	0.94	0.89
Pattern similarity (cosine)	1.00 0.85 0.80	0.99 0.93 0.94	0.89 0.75 0.93

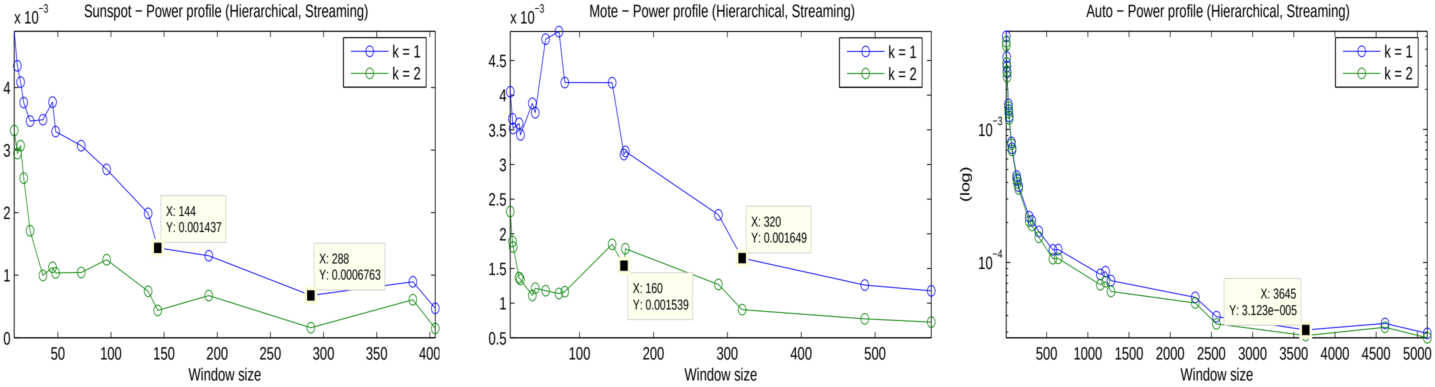
Table 6: Batch non-hierarchical vs. streaming hierarchical ($k = 3$, best window from Table 4).

8. RELATED WORK

Initial work on time series representation [1, 11] uses the Fourier transform. Even more recent work uses fixed, predetermined bases or approximating functions. APCA [4] and other similar approaches approximate the time series with piecewise constant or linear functions. DAWA [14] combines the DCT and DWT. However, all these approaches focus on compressing the time series for indexing purposes, and not on pattern discovery.

AWSOM [24] first applies the wavelet transform. As the authors observe, just a few wavelet coefficients do not capture all patterns in practice, so AWSOM subsequently captures trends by fitting a linear auto-regressive model at each time scale. In contrast, our approach learns the *best* set of orthogonal bases, while matching the space and time complexity of AWSOM. Finally, it is difficult to apply wavelets on exponential-size windows that are not powers of two.

The work in [19] proposes a multi-resolution clustering scheme for time series data. It uses the average coefficients (low frequencies) of the wavelet transform to perform k-means clustering and progressively refines the clusters by incor-



(a) `Sunspot` (b) `Mote` (c) `Automobile`

Figure 10: Power profiles (hierarchical, streaming).

porating higher-level, detail coefficients. This approach requires much less time for convergence, compared to operating directly on the very high dimension of the original series. However, the focus here is on clustering multiple time series, rather than discovering local patterns.

The problem of *principal components analysis (PCA)* and SVD on streams has been addressed in [25] and [13]. Again, both of these approaches focus on discovering linear correlations among multiple streams and on applying these correlations for further data processing and anomaly detection [25], rather than discovering optimal local patterns at multiple scales. Also related to these is the work of [8] which uses a different formulation of linear correlations and focuses on compressing historical data, mainly for power conservation in sensor networks. Finally, the work in [2] proposes an approach to combine segmentation of multidimensional series with dimensionality reduction. The reduction is on the segment representatives and it is performed across dimensions (similar to [25]), not along time, and the approach is not applicable to streams.

The seminal work of [7] for rule discovery in time series is based on sequential patterns extracted after a discretization step. Other work has also focused on finding *representative trends* [17]. A representative trend is a subsequence of the time series that has the smallest sum of distances from all other subsequences of the same length. The proposed method employs random projections and FFT to quickly compute the sum of distances. This does not apply directly to streams and it is not easy to extend, since each section has to be compared to all others. Our approach is complementary and could conceivably be used in place of the FFT in this setting. Related to representative trends are *motifs* [26, 6]. Intuitively, these are frequently repeated subsequences, i.e., subsequences of a given length which match (in terms of some distance and a given distance threshold) a large number of other subsequences of the same time series. More recently, vector quantization has been used for time series compression [21, 20]. The first focuses on finding good-quality and intuitive distance measures for indexing and similarity search and is not applicable to streams, while the second focuses on reducing power consumption for wireless sensors and not on pattern discovery. Finally, other work on stream mining includes approaches for periodicity [10] and periodic pattern [9] discovery.

Approaches for regression on time series and streams include [5] and *amnesic functions* [23]. Both of these estimate the best fit of a given function (e.g., linear or low-degree polynomial), they work by merging the estimated fit on consecutive windows and can incorporate exponential-size time windows placing less emphasis on the past. However, both of these approaches employ a fixed, given set of approximating functions. Our approach might better be described as agnostic, rather than amnesic.

A very recent and interesting application of the same principles is on correlation analysis of complex time series through change-point scores [16]. Finally, related ideas have been used in other fields, such as in image processing for image denoising [22, 15] and physics/climatology for nonlinear prediction in phase space [35]. However, none of these approaches address incremental computation in streams. More generally, the potential of this general approach has not received attention in time series and stream processing literature. We demonstrate that its power can be harnessed at very small cost, no more than that of the widely used wavelet transform.

9. CONCLUSION

We introduce a method to that can learn the key trends in a time series. Our main contributions are:

- We introduce the notion of optimal local patterns in time series.
- We show how to extract trends at multiple time scales.
- We propose a criterion which allows us to choose the best window sizes from the data.
- We introduce an approach to perform all of the above incrementally, in a streaming setting.

Furthermore, our approach can be easily extended to an ensemble of multiple time series, unlike fixed-basis methods (e.g., FFT, SFT or DWT), and also to deal with missing values. Besides providing insight about the behavior of the time series, the discovered patterns can also be used to facilitate further data processing. In fact, our approach can be used anywhere a fixed-basis orthonormal transform would be used.

Especially in a streaming setting, it makes more sense to find the best bases and project only onto these, rather than use an *a priori* determined set of bases and then try to find the "best" coefficients. However, computing these bases incrementally and efficiently is a challenging problem. Our streaming approach achieves 1-4 orders of magnitude improvement in time and space over the exact, batch approach. Its time and space requirements are comparable to previous multi-scale pattern mining approaches on streams [24, 23, 5], while it produces significantly more concise and informative patterns, without any prior knowledge about the data.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. N. Swami. Efficient similarity search in sequence databases. In *FODO*, 1993.

[2] E. Bingham, A. Gionis, N. Haiminen, H. Hiisilä, H. Mannila, and E. Terzi. Segmentation and dimensionality reduction. In *SDM*, 2006.

[3] M. Brand. Fast online SVD revisions for lightweight recommender systems. In *SDM*, 2003.

[4] K. Chakrabarti, E. Keogh, S. Mehotra, and M. Pazzani. Localy adaptive dimensionality reduction for indexing large time series databases. *TODS*, 27(2), 2002.

[5] Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang. Multi-dimensional regression analysis of time-series data streams. In *VLDB*, 2002.

[6] B. Chiu, E. Keogh, and S. Lonardi. Probabilistic discovery of time series motifs. In *KDD*, 2003.

[7] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth. Rule discovery from time series. In *KDD*, 1998.

[8] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos. Compressing historical information in sensor networks. In *SIGMOD*, 2004.

[9] M. G. Elefky, W. G. Aref, and A. K. Elmagarmid. Using convolution to mine obscure periodic patterns in one pass. In *EDBT*, 2004.

[10] M. G. Elefky, W. G. Aref, and A. K. Elmagarmid. WARP: Time warping for periodicity detection. In *ICDM*, 2005.

[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *SIGMOD*, 1994.

[12] M. Ghil, M. Allen, M. Dettinger, K. Ide, D. Kondrashov, M. Mann, A. Robertson, A. Saunders, Y. Tian, F. Varadi, and P. Yiou. Advanced spectral methods for climatic time series. *Rev. Geophys.*, 40(1), 2002.

[13] S. Guha, D. Gunopulos, and N. Koudas. Correlating synchronous and asynchronous data streams. In *KDD*, 2003.

[14] M.-J. Hsieh, M.-S. Chen, and P. S. Yu. Integrating DCT and DWT for approximating cube streams. In *CIKM*, 2005.

[15] A. Hyvärinen, P. Hoyer, and E. Oja. Sparse code shrinkage for image denoising. In *WCCI*, 1998.

[16] T. Idé and K. Inoue. Knowledge discovery from heterogeneous dynamic systems using change-point correlations. In *SDM*, 2005.

[17] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In *VLDB*, 2000.

[18] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2nd edition, 2002.

[19] J. Lin, M. Vlachos, E. J. Keogh, and D. Gunopulos. Iterative incremental clustering of time series. In *EDBT*, 2004.

[20] S. Lin, D. Gunopulos, V. Kalogeraki, and S. Lonardi. A data compression technique for sensor networks with dynamic bandwidth allocation. In *TIME*, 2005.

[21] V. Megalooikonomou, Q. Wang, G. Li, and C. Faloutsos. A multiresolution symbolic representation of time series. In *ICDE*, 2005.

[22] D. D. Muresan and T. W. Parks. Adaptive principal components and image denoising. In *ICIP*, 2003.

[23] T. Palpanas, M. Vlachos, E. J. Keogh, D. Gunopulos, and W. Truppel. Online amnesic approximation of streaming time series. In *ICDE*, 2004.

[24] S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, unsupervised stream mining. *VLDB J.*, 13(3), 2004.

[25] S. Papadimitriou, J. Sun, and C. Faloutos. Streaming pattern discovery in multiple time series. In *VLDB*, 2005.

[26] P. Patel, E. Keogh, J. Lin, and S. Lonardi. Mining motifs in massive time series databases. In *ICDM*, 2002.

[27] D. B. Percival and A. T. Walden. *Wavelet Methods for Time Series Analysis*. Cambridge Univ. Press, 2000.

[28] M. R. Portnoff. Short-time Fourier analysis of sampled speech. *IEEE Trans. ASSP*, 29(3), 1981.

[29] G. Strang. *Linear Algebra and Its Applications*. Brooks Cole, 3rd edition, 1998.

[30] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics*, pages 15–87. SIGGRAPH Course notes, 1996.

[31] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *J. Royal Stat. Soc. B*, 61(3), 1999.

[32] J. Villasenor, B. Belzer, and J. Liao. Wavelet filter evaluation for image compression. *IEEE Trans. Im. Proc.*, 4(8), 1995.

[33] M. Vlachos, C. Meek, Z. Vagena, and D. Gunopulos. Identifying similarities, periodicities and bursts for online search queries. In *SIGMOD*, 2004.

[34] B. Yang. Projection approximation subspace tracking. *IEEE Trans. Sig. Proc.*, 43(1), 1995.

[35] P. Yiou, D. Sornette, and M. Ghil. Data-adaptive wavelets and multi-scale singular-spectrum analysis. *Physica D*, 142, 2000.