WindMine: Fast and Effective Mining of Web-click Sequences

Yasushi Sakurai
NTT Communication Science Labs
yasushi.sakurai@acm.org

Lei Li
Carnegie Mellon University
leili@cs.cmu.edu

Yasuko Matsubara
Kyoto University
y.matsubara@db.soc.i.kyoto-u.ac.jp

Christos Faloutsos
Carnegie Mellon University
christos@cs.cmu.edu

Abstract

Given a large stream of users clicking on web sites, how can we find trends, patterns and anomalies? We have developed a novel method, *WindMine*, and its fine-tuning sibling, *WindMine-part*, to find patterns and anomalies in such datasets. Our approach has the following advantages: (a) it is effective in discovering meaningful "building blocks" and patterns such as the lunch-break trend and anomalies, (b) it automatically determines suitable window sizes, and (c) it is fast, with its wall clock time linear on the duration of sequences. Moreover, it can be made sub-quadratic on the number of sequences (*WindMine-part*), with little loss of accuracy.

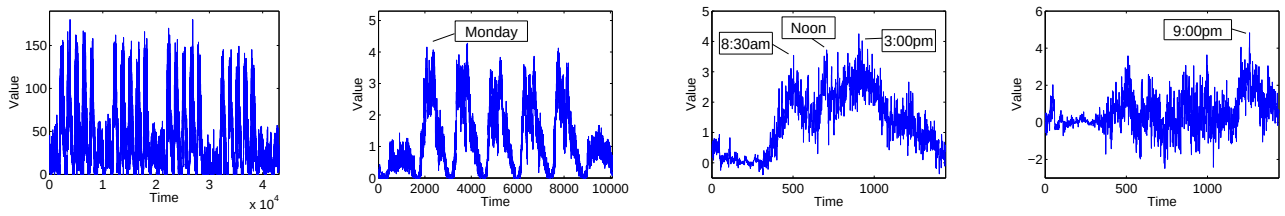
We examine the effectiveness and scalability by performing experiments on 67 GB of real data (one billion clicks for 30 days). Our proposed *WindMine* does produce concise, informative and interesting patterns. We also show that *WindMine-part* can be easily implemented in a parallel or distributed setting, and that, even in a single-machine setting, it can be an order of magnitude faster (up to 70 times) than the plain version.

1 Introduction

Many real applications generate log data at different time stamps, such as web click logs and network packet logs. At every time stamp, we might observe a set of logs, each consisting of a set of events, or time stamped tuples. In many applications the logging rate has increased greatly with the advancement of hardware and storage technology. One big challenge when analyzing these logs is to handle such large volumes of data at a very high logging rate. For example, a search web could generate millions of logging entries every minute, with information of users and URLs. As an illustration, we will use the web-click data as a running target scenario, however, our proposed method will work for general datasets as we demonstrate experimentally.

There has been much recent work on summarization and pattern discovery for web-click data. We formulate the web-click data as a collection of time stamped entries, i.e. ⟨user-id, url, timestamp⟩. The goal is to find anomalies, patterns, and periodicity for such datasets in a systematic and scalable way. Analyzing click sequences can help practitioners in many fields: (a) ISPs would like to undertake provisioning, capacity planning and abuse detection by analyzing historical traffic data; (b) web masters and web-site owners would like to detect intrusions or target designed advertisements by investigating the user-click patterns.

A common approach to analyzing the web-click tuples is to view them as multiple event sequences, one for each common URL. For example, one event sequence could be $\{\langle Alice, 1\rangle, \langle Bob, 2\rangle, \dots\}$, i.e., Alice hits url_1 at time 1 sec., and Bob at 2 sec. Instead of studying this at the individual click level, our approach is designed to find patterns at the aggregation level to allow us to detect common behavior or repeated trends. Formally, these event sequences are aggregated into multiple time series for m websites (or URLs). Each of them counts the number of hits (clicks) per $\Delta t = 1$ minute and has an aligned duration of n. Given such a dataset with multiple time series, we would like to develop a method to find interesting patterns and anomalies. For example, the leftmost plot in Figure 1 shows the web-click records from a business news site. The desired patterns for this particular data include (a) the adaptive cycles (e.g., at a daily or a weekly level); (b) the spikes in the morning and at lunch time that some sequences exhibit; and (c) the fact that these spikes are only found on weekdays. For a few number of sequences (e.g. $m = 5$), a human could eye-ball them, and derive the above patterns. The mining task is even more challenging if the number of sequences increases tremendously. How can we accomplish this task automatically for thousands or even million of sequences? We will show later that our proposed system can



Web-click sequence | Weekly component | Weekday component | Weekend component

Figure 1: Illustration of trend discovery. (a) Original web-click sequence (access count from a business news site). (b) Weekly trend, which shows high activity on weekdays for business purposes. (c) Weekday trend, which increases from morning to night and reaches peaks at 8:30 am, noon, and 3:00 pm. (d) Weekend trend, which is different from the weekday trend pattern.

automatically identify these patterns all at once, and how it solves scalably.

Contributions Our main contribution is the proposal of *WindMine*, which is a novel method for finding patterns in a large collection of click-sequence data. *WindMine* automatically detects daily periodicity (unsurprisingly), huge lunch-time spikes for news-sites as shown in Figure 1 (reasonable, in retrospect), as well as additional, surprising patterns. Additional contributions are as follows:

1. *Effective*: We apply *WindMine* to several real datasets, spanning 67 GB. Our method finds both expected and surprising patterns, completely on its own.
2. *Adaptive*: We propose a criterion that allows us to choose the best window size of trend patterns from the sequence dataset. The choice of window sizes is data-driven and fully automatic.
3. *Scalable*: The careful design of *WindMine* makes it linear on the number of time-ticks in terms of wall clock time. In fact, it is readily parallelizable, which means that it can also scale well with a large number of sites m.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 introduces preliminary concepts that are relevant to an understanding of this paper. Sections 4 presents our proposed method, and Section 5 introduces some useful applications of our method, and evaluates our algorithms based on extensive experiments. Section 6 concludes the paper.

2 Related work

There are several pieces of work related to our approach, including (a) dimensionality reduction; (b) time series indexing; (c) pattern/trend discovery and outlier detection.

Dimensionality reduction for time-series data: Singular value decomposition (SVD) and principal component analysis (PCA) [7, 28] are commonly used tools to discover hidden variables and low rank patterns from high dimensional data. In contrast to the traditional SVD for batch data, Yi et al. [30] proposed an online autoregressive model to handle multiple sequences. Gilbert et al. [4] used wavelets to compress the data into a fixed amount of memory by keeping track of the largest Haar wavelet coefficients. Papadimitriou et al. [17] proposed the SPIRIT method to discover linearly correlated patterns for data streams, where the main idea was to calculate the SVD incrementally. Sun et al. [23] took a step forward by extending SPIRIT to handle data from distributed sources, so that each node or sensor device could calculate local patterns/hidden variables, which are then summarized later at a center node. Our proposed WindMine is related to a scheme for partitioning the data, calculating them individually and finally integrating them in the end. Moreover, our method could detect patterns and anomalies even more effectively.

Indexing and representation: Our work is also related to the theories and methods for time-series representation [15, 14, 22], and indexing [8, 21, 9, 2]. Various methods have been proposed for representing time-series data using shapes, including velocity and shape information for segmenting trajectory [15]; symbolic aggregate approximation (SAX) [14] and its generalized version for indexing massive amounts of data (iSAX) [22]. Keogh [8] proposed a search method for dynamic time warping (DTW). [21] proposed the FTW method with successive approximations, refinements and additional optimizations, to accelerate "whole sequence" matching under the DTW distance. Keogh et al. used uniform scaling to create an index for large human motion databases [9]. [2] presented SPIRAL, a fast search method for HMM datasets. To reduce the search cost, the

method efficiently prunes a significant number of search candidates by applying upper bounding approximations when estimating likelihood. Tensor analysis is yet another tool for modeling multiple streams. Related work includes scalable tensor decomposition [10] and incremental tensor analysis [25, 24, 26].

Pattern/trend discovery: Papadimitriou et al. [18] proposed an algorithm for discovering optimal local patterns, which concisely describe the multi-scale main trends. [20] proposed BRAID, which efficiently finds lag correlations between multiple sequences. SPRING [19] efficiently and accurately detects similar subsequences without determining window size. Kalman filters are also used in tracking patterns for trajectory and time series data [27, 13]. Other remotely related work includes the classification and clustering of time-series data and outlier detection. Gao et al. [3] proposed an ensemble model to classify time-series data with skewed class distributions, by undersampling the dominating class and oversampling or repeating the rare class. Lee et al. [12] proposed the TRAOD algorithm for identifying outliers in a trajectory database. In their approach, they first partition the trajectories into small segments and then use both distance and density to detect abnormal sub-trajectories. This paper mainly focuses on web-click mining as an application of our method, thus, our work is also related to topic discovery for web mining. There has been a large body of work on statistical topic models [5, 1, 16, 29], which uses a multinomial word distribution to represent a topic. These techniques are also useful for web-click event analysis while our focus is to find local components/trends in multiple numerical sequences.

3 Background

In this section we briefly introduce some necessary background material. Table 1 is a list of symbols and their definitions.

3.1 Principal component analysis Given a collection of n-dimensional vectors $\mathbf{x_i} \in \mathbb{R}^n$, $i = 1, 2, \ldots, m$, the first principal direction $\mathbf{b_1} \in \mathbb{R}^n$ is the vector that minimizes the sum of squared residuals, i.e.,

$$\mathbf{b_1} := \arg\min_{||\mathbf{b}||=1} \sum_{i=1}^{m} ||\mathbf{x_i} - (\mathbf{b}\mathbf{b}^T)\mathbf{x_i}||^2. \quad (3.1)$$

The projection of $\mathbf{x_i}$ on $\mathbf{b_1}$ is the first principal component (PC) $y_{i,1} := \mathbf{b_1}^T\mathbf{x_i}$, $i = 1, \ldots, m$. Note that, since $||\mathbf{b_1}|| = 1$, we have $(\mathbf{b_1}\mathbf{b_1}^T)\mathbf{x_i} = (\mathbf{b_1}^T\mathbf{x_i})\mathbf{b_1} = y_{i,1}\mathbf{b_1} := \hat{\mathbf{x}}_\mathbf{i}$, where $\hat{\mathbf{x}}_\mathbf{i}$ is the projection of $y_{i,1}$ back into the original n-D space. That is, $\hat{\mathbf{x}}_\mathbf{i}$ is a reconstruction of the original measurements from the first PC $y_{i,1}$. More generally, PCA will produce k vectors $\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_k}$, such that, if we represent each n-D data point $\mathbf{x_i} := [x_{i,1} \ldots x_{i,n}]$ with k-D projection $\mathbf{y_i} = [\mathbf{b_1^T}\mathbf{x_i} \ldots \mathbf{b_k^T}\mathbf{x_i}]^T$, then this representation minimizes the squared error $\sum_i ||\mathbf{x_i} - \hat{\mathbf{x}}_\mathbf{i}||^2$. Furthermore, the principal directions are orthogonal, so the principal components $y_{i,1}$,, $1 \leq j \leq k$ are, by construction, uncorrelated, i.e., if $\mathbf{y}^{(j)} := [y_{1,j}, \ldots, y_{i,j}]^T$ is the sequence of the j-th principal component, then $(\mathbf{y}^{(j)})^T\mathbf{y}^{(k)} = 0$ if $j \neq k$.

Table 1: Symbols and definitions.

Symbol	Definition
n	Duration: number of time-ticks
m	Number of sequences
w	Window size
M	Number of subsequences
k	Number of components
X	Data sequence of length n
x_t	Value of X at time $t = 1, \ldots, n$
$\hat{X}$	Window matrix of X
$A = [a_{i,j}]$	Mixing matrix
B	Independent component
C_w	CEM score of w

3.2 Independent component analysis To find the directions of minimal entropy the well known fastICA algorithm [6] requires us to transform the dataset into white space, i.e., the dataset must be centered and normalized so that it has unit variance in all directions. This may be achieve from the eigenvalue decomposition of the covariance matrix (i.e., $V \cdot \Lambda \cdot V^T := \Sigma$ where V is an orthonormal matrix consisting of the eigen vectors, and Λ is a diagonal matrix($\Lambda = diag(\lambda_1, \ldots, \lambda_d)$). The matrix $\Lambda^{-1/2}$ is a diagonal matrix with the elements $\Lambda^{-1/2} = diag(\sqrt{1/\lambda_i}, \ldots, \sqrt{1/\lambda_d})$. The fastICA algorithm then determines a matrix B that contains the independent components. This matrix is orthonormal in white space but not in the original space. FastICA is an iterative method that finds $B = (b_1, \ldots, b_d)$ by optimizing the vectors b_i using the following updating rule:

$$b_i := E\{y \cdot g(b_i^T \cdot y)\} - E\{g'(b_i^T \cdot y)\} \cdot b_i \quad (3.2)$$

where $g(s)$ is a non-linear contrast function (such as $tanh(s)$) and $g'(s) = \frac{d}{ds}g(s)$ is its derivative. We denote the expected value with $E\{\ldots\}$. After each application of the update rule to $b_i, \ldots, b_d$, the matrix B is orthonormalized. This is repeated until convergence. The de-mixing matrix A^{-1}, which describes the overall transformation from the original data space to the independent components, can

be determined as

$$A^{-1} = B^T \Lambda^{-1/2} \cdot V^T, A = V \cdot \Lambda^{+1/2} \cdot B \quad (3.3)$$

and, since V and B are orthonormal matrices, the determinant of A^{-1} is simply the determinant of $\Lambda^{-1/2}$, i.e,

$$det(A^{-1}) = \prod_{1 \le i \le d} \sqrt{1/\lambda_i}. \quad (3.4)$$

4 Proposed Method

4.1 Problem definition Web-log data consist of tuples of the form *(user-id, url, timestamp)* . We turn them into sequences X_1, ..., X_m, one for each URL of interest. We compute the number of hits per Δt= 1 minute (or second), and thus we have m sequences of duration n. One of the sequences, X, is a discrete sequence of numbers $\{ x_1, \ldots, x_t, \ldots, x_n \}$, where x_n is the most recent value.

Our goal is to extract the main components of click sequences, to discover common trends, hidden patterns, and anomalies. As well as the components of the entire sequences, we focus on components of length w to capture local trends. We now define the problems we are trying to solve and some fundamental concepts.

PROBLEM 1. (LOCAL COMPONENT ANALYSIS) *Given m sequences of duration n and window size w, find the subsequence patterns of length w that represent the main components of the sequences.*

The window size w is given in Problem 1. However, with real data, w for the component analysis is not typically known in advance. Thus the solution has to handle subsequences of multiple window sizes. This gives rise to an important question: whenever the main components of the 'best' window size are extracted from the sequences, we expect there to be many other components of multi-scale windows, which could potentially flood the user with useless information. How do we find the best window size automatically? The full problem that we want to solve is as follows:

PROBLEM 2. (CHOICE OF BEST WINDOW SIZE) *Given m sequences of duration n, find the best window size w and the subsequence patterns of length w that represent the main components of the sequences.*

An additional question relates to what we can do in the highly likely case that the users need an efficient solution while in practice they require high accuracy. Thus, our final challenge is to present a scalable algorithm for the component analysis.

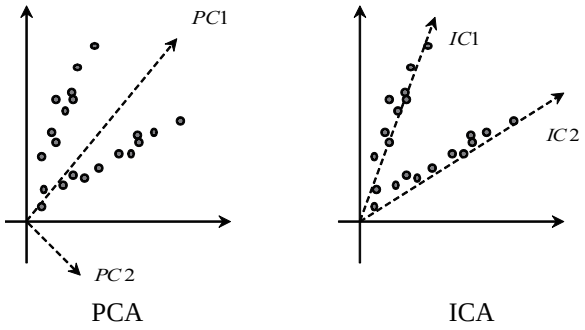


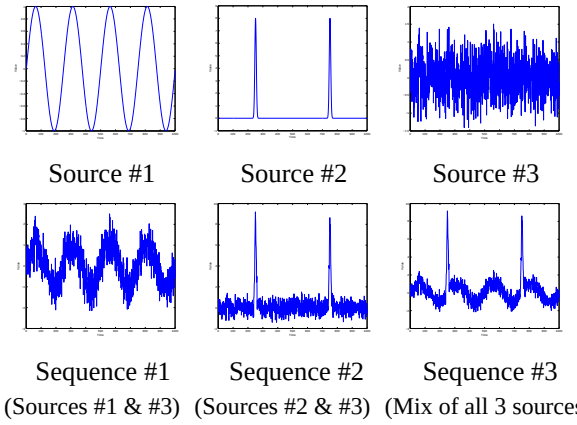
Figure 2: PCA vs ICA. Note that PCA vectors go through empty space; ICA/WindMine components snap on the natural lines (leading to sparse encoding).

4.2 Multi-scale local component analysis For a few time sequences, a human could eye-ball them, and derive the above patterns. But, how can we accomplish this automatically for thousands of sequences? The first idea would be to perform principal component analysis (PCA) [7], as employed in [11]. However, PCA and singular value decomposition (SVD) have pitfalls. Given a cloud of n-D points (sequences with n time-ticks), PCA will find the best line that goes through that cloud; and then the second best line (orthogonal to the first), and so on. Figure 2 highlights this pitfall: If the cloud of points looks like a pair of scissors, then the first principal component will go in the *empty* area marked "PC1", and the second principal component will be on the equally empty area that is perpendicular to the first PC.

APPROACH 1. *We introduce independent component analysis (ICA) for data mining of numerical sequences.*

Instead of using PCA, we propose employing ICA [6], also known as *blind source separation*. ICA will find the directions marked IC1 and IC2 in Figure 2 exactly, because it does not require orthogonality, but needs a stronger condition, namely, independence. Equivalently, this condition results in sparse encoding: the points of our initial cloud will have a sparse representation in the new set of (non-orthogonal) axes, that is, they will have several zeros.

EXAMPLE 1. *Figure 3 shows an example of component analysis. The sample dataset includes three sequences: (1) sinusoidal waves with white noise, (2) large spikes with noise, and (3) a combined sequence. We compute three components each for PCA and ICA, from the three original sequences. Unlike PCA, which is confused by these components, ICA recognizes them successfully and separately.*



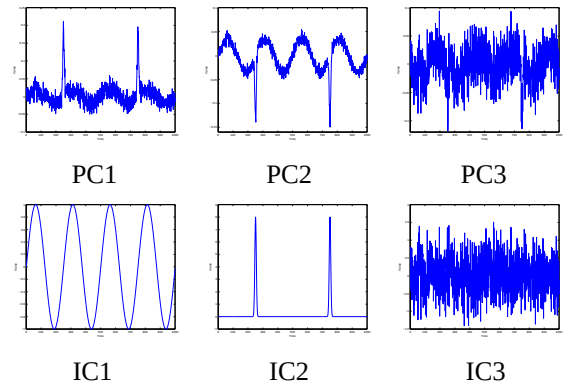


Figure 3: Example of PCA and ICA components. Top row: sources; second row: sequences that are linear combinations of the three sources; 3rd row: the sources recovered by PCA; 4th row: the sources recovered by ICA. Notice how much more clear is the separation of sources that ICA achieves. PCA suffers from the 'PCA confusion' phenomenon.

In the preceding discussion we introduced ICA and showed how to analyze entire full length sequences to obtain their 'global' components. We then describe how to find the local components using ICA.

APPROACH 2. *We propose applying a short-window approach to ICA, which is a more powerful and flexible approach for component analysis.*

DEFINITION 1. (WINDOW MATRIX) *Given a sequence* $X = \{ x_1, \ldots, x_n \}$ *and a window size* w*, the window matrix of* X*,* $\hat{X}$*, is a* $\lceil n/w \rceil \times w$ *matrix, in which the* i*-th row is* $\{ x_{(i-1)w+1}, \ldots, x_{iw} \}$.

When we have m sequences, we can locally analyze their common independent components using the short-window approach. We propose *WindMine* for local component analysis.

DEFINITION 2. (*WindMine*) *Given* m *sequences of duration* n*, and a window size* w*, the local independent components are computed from the* $M \times w$ *window matrix of the* m *sequences, where* $M = m \cdot \lceil n/w \rceil$.

The size of the local components typically depends on the given datasets. Our method, *WindMine*, handles multi-scale windows to analyze the properties of the sequences.

APPROACH 3. *We introduce a framework based on multi-scale windows to discover local components.*

Starting with the original sequences $\{ X_1, \ldots, X_m \}$, we divide each one into subsequences of length w, construct their window matrix $\hat{X}_w$, and then compute the local components from $\hat{X}_w$. We vary the window size w, and repeatedly extract the local components B_w with the mixing matrix A_w for various window sizes w.

EXAMPLE 2. *Figure 4 illustrates multi-scale local component analysis using* WindMine. *The total duration of a sequence* X *is* $m = 8$*. We have four disjoint windows each of length* $w = 2$*, thus* $\hat{X}_2$ *is a* 4×2 *matrix. We extract two local components for* $w = 2$ *in this figure.*

4.3 CEM criterion: best window size selection Thus far, we have assumed that the window size was given. The question we address here is how to estimate a good window size automatically when we have multiple sequences. We would like to obtain a criterion that will operate on the collection of subsequences, and dictate a good number of subsequence length w for the local component analysis. This criterion should exhibit a spike (or dip) at the "correct" value of w. Intuitively, our observation is that if there is a trend of length w that frequently appears in the given sequences, the computed local component is widely used to represent their window matrix $\hat{X}_w$. We want to find the "sweet spot" for w.

We therefore propose using the mixing matrix A_w to compute the criterion for selecting the window size. Notice that straightforward approaches are unsuitable, because they are greatly affected by specific, unpopular components. For example, if we summed up the weight values of each column of the mixing matrix and then chose the component with the highest value, the component would be used to represent a limited number of subsequences. Our goal boils down to the following question: *What function of* w *reaches an extreme value when we hit the 'optimal' number of window sizes* w_{opt}*?* It turns out that 'popular' (i.e., widely used)

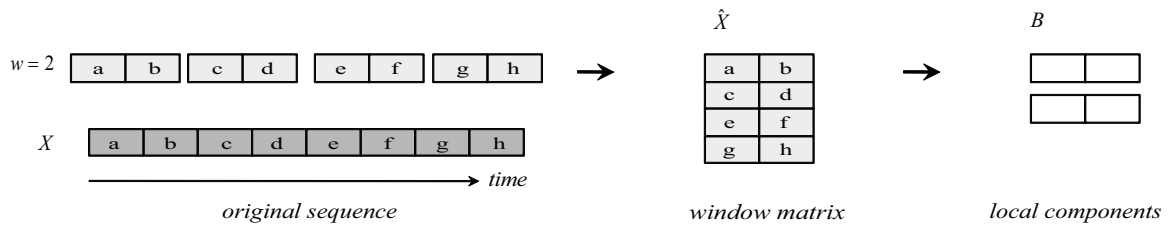


Figure 4: Illustration of *WindMine* for window size $w = 2$. It creates a window matrix of 4 disjoint windows, and then finds their two major trends/components.

components are suitable for selection as local components that capture the local trends of the sequence set.

APPROACH 4. *We introduce a criterion for window size selection, which we compute from the entropy of the weight values of each component in the mixing matrix.*

We propose a criterion for estimating the optimal number of w for a given sequence set. The idea is to compute the probability histogram of the weight parameters of each component in the mixing matrix, and then compute the entropy of each component.

The details are as follows: For a window size w, we provide the mixing matrix $A_w = [a_{i,j}]$ (i = 1,. . .,M; j = 1,. . .,k) of given sequences, where k is the number of components and M is the number of subsequences. Optionally, we normalize the weight values for each subsequence.

$$a'_{i,j} = a_{i,j} / \sum_j a^2_{i,j}. \tag{4.5}$$

We then compute the probability histogram P_j = $\{p_{1,j}, \ldots, p_{M,j}\}$ for the j-th component.

$$p_{i,j} = \|a'_{i,j}\| / \sum_i \|a'_{i,j}\|. \tag{4.6}$$

Intuitively, P_j shows the size of the j-th component's contribution to each subsequence. Since we need the most popular component among k components, we propose using the entropy of the probability histogram for each component.

Therefore, our proposed criterion, which we call component entropy maximization (CEM), or the CEM score, is given by

$$C_{w,j} = -\frac{1}{\sqrt{w}} \sum_i p_{i,j} \log p_{i,j}, \tag{4.7}$$

where $C_{w,j}$ is the CEM score of the j-th component for the window size w. We want the best local component of length w that maximizes the CEM score, that is, $C_w = \max_j C_{w,j}$.

Once we obtain C_w for every window size, the final step is to choose w_{opt}. Thus, we propose

$$w_{opt} = arg \max_w C_w. \tag{4.8}$$

4.4 Scalable algorithm: *WindMine-part* In this subsection we tackle an important and challenging question, namely how do we efficiently extract the best local component from large sequence sets? In Section 4.2 we present our first approach for multi-scale local component analysis. We call this approach *WindMine-plain*[1].

Algorithm 1 *WindMine-part*(w, $\{X_1, \ldots, X_m\}$)

```
for each sequence X_i do
  Divide X_i by ⌈m/w⌉ subsequences
  Append the subsequences to the window matrix X̂
end for
for level h = 1 to H do
  Initialize X̂_new
  Divide the subsequence set of X̂ into ⌈M/g⌉ groups
  for group number j = 1 to ⌈M/g⌉ do
    Create the j-th submatrix S_j of X̂
    Compute the local components of S_j with their mixing matrix A
    Compute the CEM score of each component from A
    Append the best local component(s) to X̂_new
  end for
  X̂ = X̂_new
  M = ⌈M/g⌉
end for
Report the best local component(s) in X̂
```

Although important, this approach is insufficient to provide scalable processing. What can we do in the highly likely case that the users need an efficient solution for large datasets while in practice they require high accuracy? To reduce the time needed for local component analysis

[1]We use '*WindMine*' as a general term for our method and its variants.

and overcome the scalability problem, we present a new algorithm, *WindMine-part*.

APPROACH 5. *We introduce a partitioning approach for analyzing a large number of subsequences hierarchically, which yields a dramatic reduction in the computation cost.*

Specifically, instead of computing local components directly from the entire set of subsequences, we propose partitioning the original window matrix into submatrices, and then extracting local components each from the submatrices.

DEFINITION 3. (MATRIX PARTITIONING) *Given a window matrix $\hat{X}$, and an integer g for partitioning, the j-th submatrix of $\hat{X}$ is formed by taking rows from $(j-1)g+1$ to jg.*

Our partitioning approach is hierarchical, which means that we reuse the local components of the lower level for local component analysis on the current level.

DEFINITION 4. (*WindMine-part*) *Given a window matrix on the h-th level, we extract k local components from each submatrix that has g local components of the $(h-1)$-th level. Thus, the window matrix on the h-th level includes $M \cdot (k/g)^{h-1}$ local components (i.e., $M \cdot (k/g)^{h-1}$ rows).*

After extracting the local components from the original window matrix on the first level $h = 1$, we create a new window matrix from the components of $h = 1$ on the second level ($h = 2$), and then compute the local components of $h = 2$. We repeatedly iterate this procedure for the upper levels. Algorithm 1 provides a high-level description of *WindMine-part*.

5 Experimental Results

To evaluate the effectiveness of *WindMine*, we carried out experiments on real datasets. We conducted our experiments on an Intel Core 2 Duo 1.86GHz with 4GB of memory, and running Linux. Note that all components/patterns presented in this section are generated by the scalable version, *WindMine-part*, while both versions provide useful results for the applications.

The experiments were designed to answer the following questions:

1. How successful is *WindMine* in local component analysis?
2. Does *WindMine* correctly find the best window size for mining locally patterns?
3. How does *WindMine* scale with the number of subsequences m in terms of computational time?

5.1 Effectiveness

5.1.1 Mining Web-click sequences In this subsection we describe some of the applications for which *WindMine* proves useful. We present case studies of real web-click datasets to demonstrate the effectiveness of our approach in discovering the common trends for each sequence.

Ondemand TV This dataset is from the *Ondemand TV* service of 13,231 programs that users viewed in a 6-month period (from May 14th to November 15th, 2007). The data record the use of *Ondemand TV* by 109,474 anonymous users. It contains a list of attributes (e.g., content ID, the date the user watched the content, and the ID of the user who watched the content).

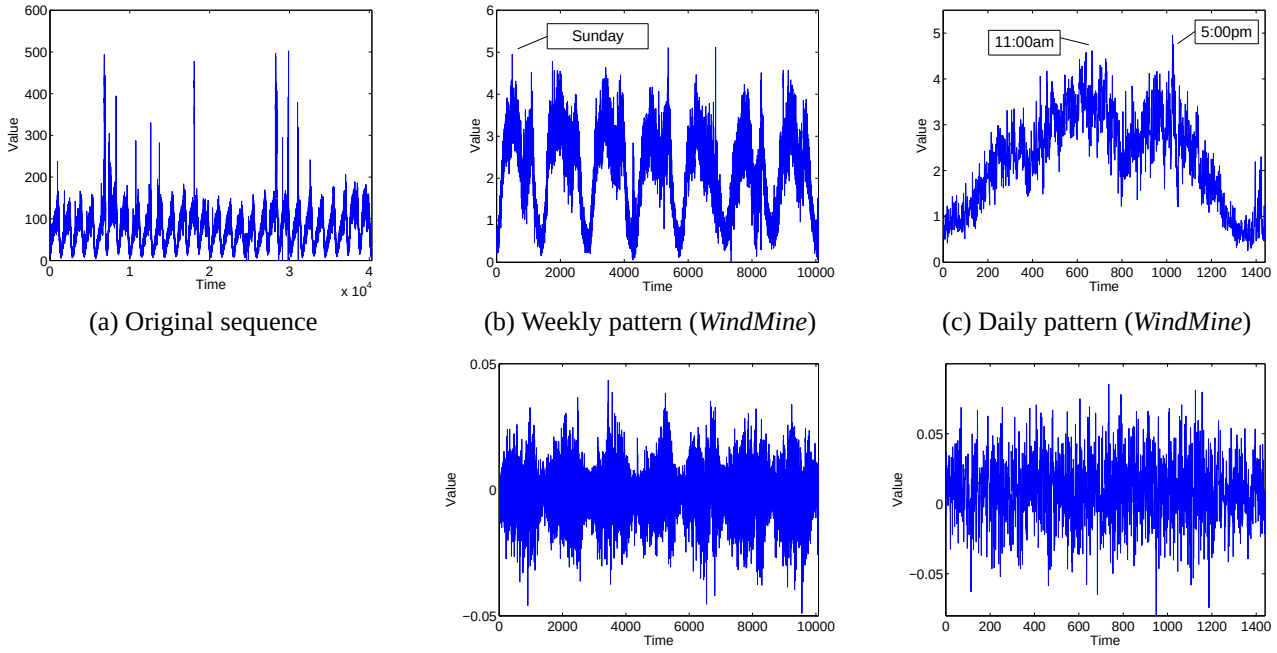
Figure 5 (a) shows the original sequence on the *Ondemand TV* dataset. It exhibits a cyclic daily pattern. There are anomaly spikes each day at about lunch time. Figure 5 (b)-(c) show that *WindMine* successfully captures the weekly and daily patterns from the dataset. It can be easily used to capture information for arbitrary time scales. For comparison, Figure 5 (d)-(e) show the best local patterns using the PCA technique. As shown in these figures, it is not robust against noise and anomaly spikes, and it cannot produce good results.

WebClick This dataset consists of the web-click records from *www.goo.ne.jp*, obtained over one month (from April 1st to 30th, 2007). It contains one billion records with 67 GB of storage. Each record has 3 attributes: user ID (2,582,252 anonymous users), URL group ID (1,797 groups), and the time stamp of the click. There are various types of URLs, such as "blog", "news", "health", and "kids".

Figures 6, 7 and 8 show the effectiveness of our method. Specifically, Figures 6 and 7 show the local components of the Q & A site and job-seeking site. The left, middle and right columns in Figure 6 show the weekly, daily, and weekend patterns, respectively. Our method identifies the daily period, which increases from morning to night and reaches a peak. This trend appears strongly, especially on weekends. In contrast, Figure 7 describes "business" trends. Starting from Monday, the daily access decreases as the weekend approaches. At 9:00 am, workers arrive at their office, and they look at the job-seeking website during a short break. Additionally, the right figure shows that there is a large spike during the lunch break.

Figure 8 shows the local patterns of other websites. We can observe interesting daily trends according to various lifestyles.

(a) Dictionary: Figure 8 (a) shows the daily trend of the



(a) Original sequence (b) Weekly pattern (*WindMine*) (c) Daily pattern (*WindMine*)

(d) Weekly pattern (PCA) (e) Daily pattern (PCA)

Figure 5: Original sequence and weekly and daily components for *Ondemand TV*. (b) Note the daily periodicity, with NO distinction between weekdays and weekends. (c) The main daily pattern agrees with our intuition: peaks in the morning and larger peaks in the evening, with low activity during the night. In contrast, PCA discovers trends that are not as clear, which suffer from the 'PCA confusion' phenomenon.

dictionary site. The access count increases from 8:00 am and decreases from 11:00 pm. We consider this site to be used for business purposes since this trend is strong on weekdays.

(b) Kids: Our method discovered a clear trend from an educational site for children. From this figure, we can recognize that they visit this site after school at 3:00 pm.

(c) Baby: This figure shows the daily pattern of the website as regards pregnancy and baby nursery resources. The access pattern shows the presence of several peaks until late evening, which is very different from the kids site. This is probably because the kids site is visited by elementary school children whereas the main users of the baby site will be their parents, rather than babies!

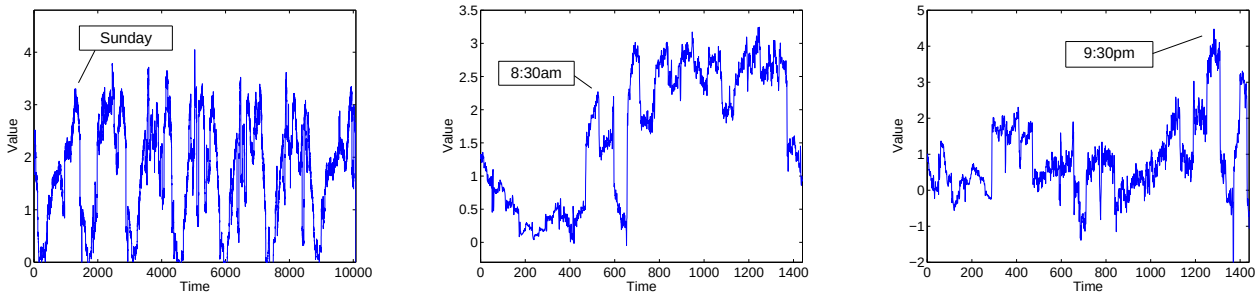
(d) Weather news: This website provides official weather observations, weather forecasts and climate information. We observed that the users typically check this site three times a day. We can recognize a pattern of behavior. They visit this site in the early morning and at noon before going outside. In the early evening, they check their local weather for the following day.

(e) Health: This is the main result of the healthcare site. The result shows that the users rarely visit website late in the evening, which is indeed good for their health.

(f) Diet: This is the main daily pattern of an on-line magazine site that provides information about diet, nutrition and fitness. The access count increases rapidly after meal times. We also observed that the count is still high in the middle of the night. We think that perhaps a healthy diet should include an earlier bed time.

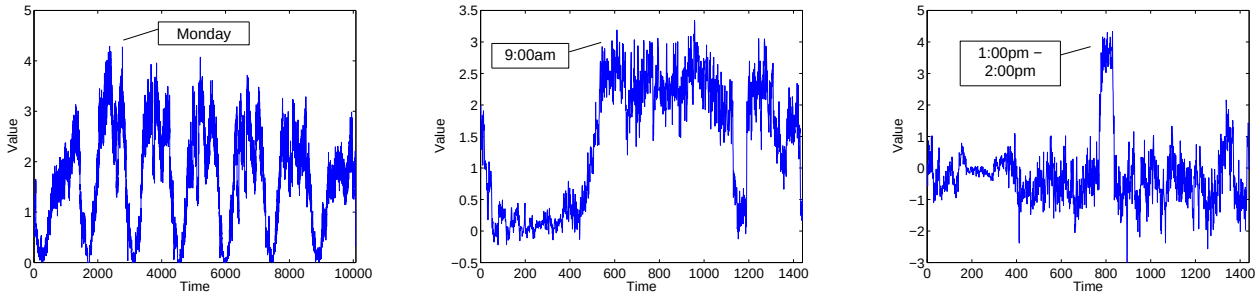
5.1.2 Generalization of WindMine We demonstrate the effectiveness of our approach in discovering the trends for other types of sequences.

Automobile This dataset consists of automobile traffic count for a large, west coast interstate. The top row of Figure 9 (a) exhibits a clear daily periodicity. The main trend repeats at a window of approximately 4000 timestamps. Also, during each day there is another distinct pattern of morning and afternoon rush hours. However, these peaks have distinctly different shapes: the evening peak is more spread out, the morning peak is more concentrated and slightly sharper.



(a) Weekly pattern (b) Daily pattern (c) Weekend pattern

Figure 6: Frequently used components for the Q & A site of *WebClick*. (a) Major weekly trend/component, showing similar activity during all 7 days of the week. (b) Major daily trend - note the low activity during sleeping time, as well as the dip at dinner time. (c) Major weekday pattern - note the spike during lunch time.



(a) Weekly pattern (b) Daily pattern (c) Weekday additional pattern

Figure 7: Frequently used components for the job-seeking site of *WebClick*. (a) Major weekly trend, showing high activity on weekdays. (b) Major daily pattern. (c) Daily pattern, which is mainly applicable to weekdays.

The bottom row of Figure 9 (a) shows the output of *WindMine* for the *Automobile* dataset. The common trend seen in the figure successfully captures the two peaks and also their approximate shape.

Temperature We used temperature measurements (degrees Celsius) in the Critter data set, which comes from small sensors within several buildings. In this dataset there are some missing values, and it exhibits a cyclic pattern, with cycles lasting less than 2000 time ticks. This is the same dataset that was used in our previous study [20].

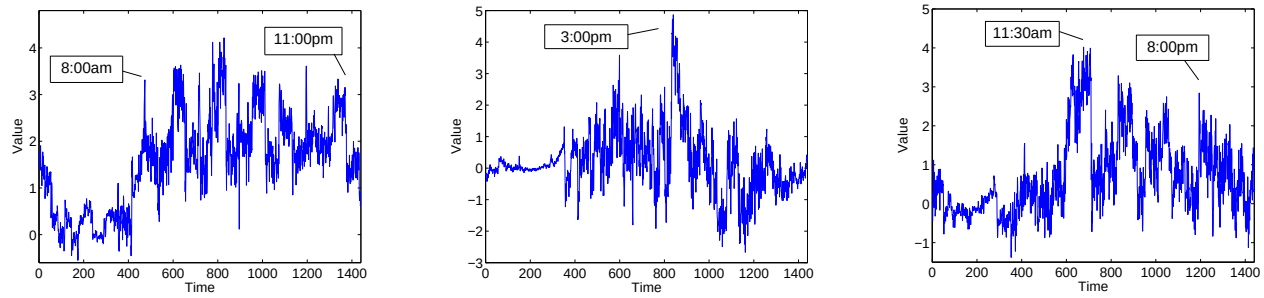
Our method correctly captures the right window for the main trend and also an accurate picture of the typical daily pattern. As shown in Figure 9 (b), there are similar patterns that fluctuate significantly with the weather conditions (which range from 17 to 27 degrees). Actually, *WindMine* finds the daily trend when the temperature fluctuates between cool and hot.

Sunspots We know that sunspots appear in cycles. The top row of Figure 9 (c) indicates the number of sunspots per day. For example, during one 30-year period within the so-called "Maunder Minimum", only about 50 sunspots were observed, as opposed to the normal 40,000-50,000 spots. The average number of visible sunspots varies over time, increasing and decreasing in a regular cycle of between 9.5 and 11 years, averaging about 10.8 years [2].

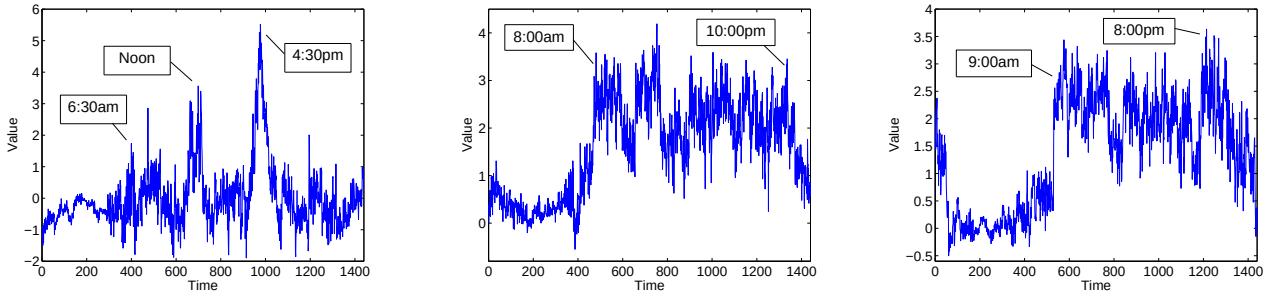
WindMine can capture bursty sunspot periods and identify the common trends in the *Sunspot* dataset. The bottom row in Figure 9 (c) shows that our method provides an accurate picture of what typically happens within a cycle.

5.2 Choice of best window size We evaluate the accuracy of the CEM criterion for window size selection. Figure 10 (a) presents the CEM score for *Ondemand TV*, for various window sizes. This figure shows that *WindMine* can deter-

[2] http://csep10.phys.utk.edu/astr162/lect/sun/sscycle.html

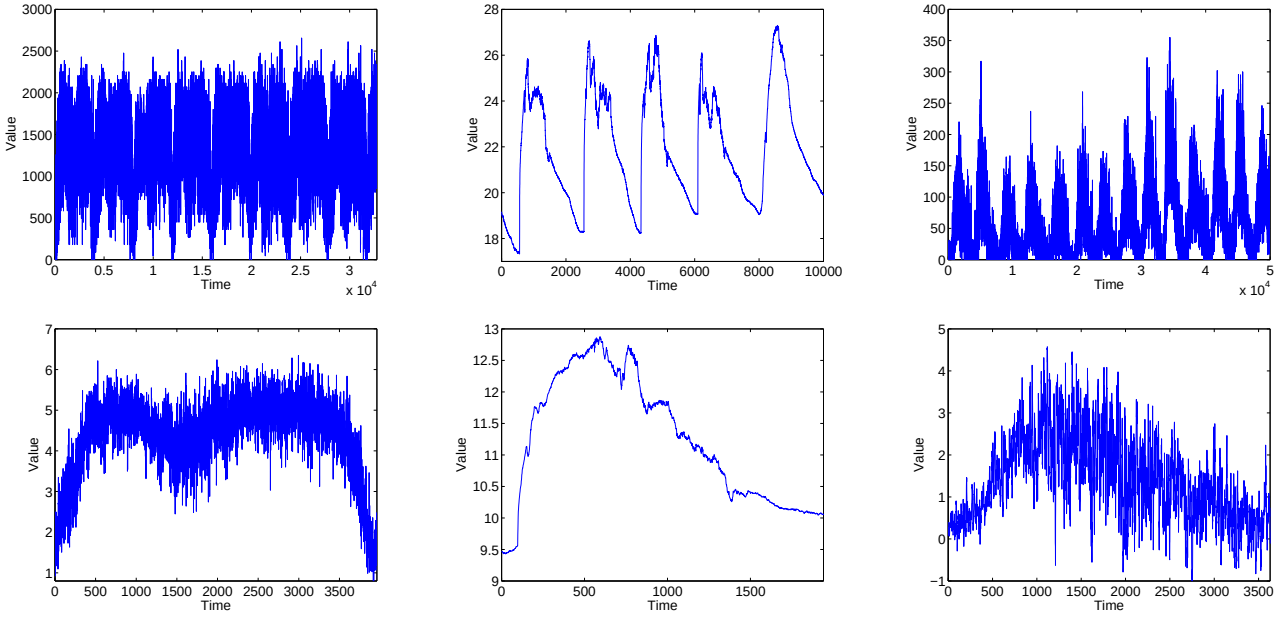


(a) Dictionary (b) Kids (c) Baby



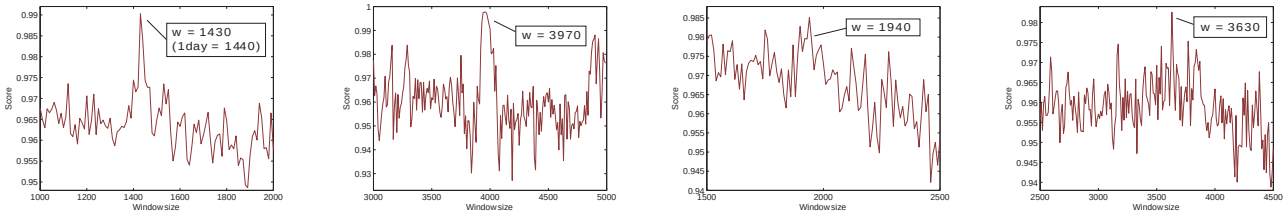
(d) Weather news (e) Health (f) Diet

Figure 8: Daily components for the dictionary, kids, baby, weather news, health and diet sites of *WebClick*. *WindMine* discovers daily trends according to various lifestyles.



(a) *Automobile* (b) *Temperature* (c) *Sunspot*

Figure 9: Detection of the common trends for *Automobile*, *Temperature* and *Sunspot*. The top and bottom rows show the original data sequences and the captured trends, respectively.



(a) *Ondemand TV* (b) *Automobile* (c) *Temperature* (d) *Sunspot*

Figure 10: CEM scores for *Ondemand TV*, *Automobile*, *Temperature* and *Sunspot*.

mine the best window size by using the CEM criterion. As expected, our method indeed suggests that the best window is daily periodicity. It identifies $w = 1430$ as the best window size, which is close to the one-day duration ($w = 1440$). Due to the window size estimation, we can discover the daily pattern for *Ondemand TV* (see Figure 5 (c)).

Figure 10 (b)-(d) show the CEM scores per window element for *Automobile*, *Temperature* and *Sunspot*, respectively. Note that the *Temperature* dataset includes missing values and the *Sunspot* dataset has time-varying periodicity. As shown in these figures, *WindMine* successfully detects the best window size for each dataset, which corresponds to the duration of the main trend (see the figures of the bottom row in Figure 9).

5.3 Performance We conducted experiments to evaluate the efficiency of our method. Figure 11 compares *WindMine-plain* and the scalable version, *WindMine-part*, in terms of computation time for different numbers of subsequences. The wall clock time is the processing time needed to capture the trends of subsequences. Note that the vertical axis is logarithmic. We observed that *WindMine-part* achieves a dramatic reduction in computation time that can be up to 70 times faster than the plain method.

Figure 12 shows the wall clock time as a function of duration n. The plots were generated using *WebClick*. Although the run-time curves are not so smooth due to the convergence of ICA, they reveal the almost linear dependence of the duration n. As we expected, *WindMine-part* identifies the trends of subsequences much faster than *WindMine-plain*.

6 Conclusions

We focused on the problem of fast, scalable pattern extraction and anomaly detection in large web-click sequences. The main contributions of this work are as follows:

1. We proposed *WindMine*, a scalable, parallelizable method for breaking sequences into a few, fundamental ingredients (e.g., spikes, cycles).
2. We described a partitioning version, which has the same accuracy, but scales *linearly* over the sequence duration, and near-linearly on the number of sequences.
3. We proposed a criterion that allows us to choose the best window sizes from the data sequences.
4. We applied *WindMine* to several real sets of sequences (web-click data, sensor measurements) and showed how to derive useful information (spikes, differentiation of weekdays from weekends). *WindMine* is fast and practical, and requires only a few minutes to process 67 GB of data on commodity hardware.

References

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] Y. Fujiwara, Y. Sakurai, and M. Yamamuro. Spiral: Efficient and exact model identification for hidden markov models. In *KDD Conference*, pages 247–255, Las Vegas, Nevada, August 2008.

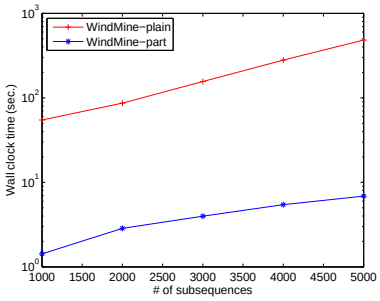
[3] J. Gao, B. Ding, W. Fan, J. Han, and P. S. Yu. Classifying data streams with skewed class distributions and concept drifts. *Internet Computing*, 12(6):37–49, 2008.

[4] A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. Surfing wavelets on streams: One-pass summaries for approximate aggregate queries. In *Proceedings of VLDB*, pages 79–88, Rome, Italy, Sept. 2001.

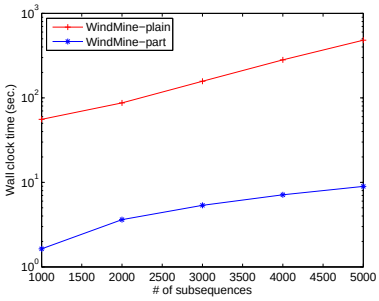
[5] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[6] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4-5):411–430, 2000.

[7] I. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

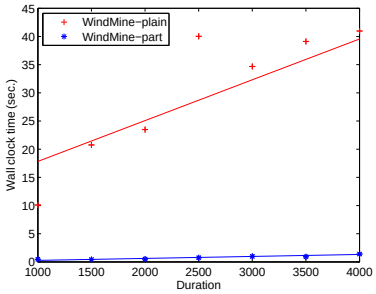


Ondemand TV



WebClick

Figure 11: Scalability: wall clock time vs. # of subsequences.



Ondemand TV

Figure 12: Scalability: wall clock time vs. duration.

[8] E. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417. VLDB Endowment, 2002.

[9] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle. Indexing large human-motion databases. In *VLDB*, pages 780–791. VLDB Endowment, 2004.

[10] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *Proc. Eighth IEEE International Conference on Data Mining ICDM '08*, pages 363–372, 2008.

[11] F. Korn, H. Jagadish, and C. Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *ACM SIGMOD*, pages 289–300, May 13-15 1997.

[12] J.-G. Lee, J. Han, and X. Li. Trajectory outlier detection: A partition-and-detect framework. *IEEE 24th International Conference on Data Engineering*, pages 140–149, April 2008.

[13] L. Li, J. McCann, N. Pollard, and C. Faloutsos. Dynammo: Mining and summarization of coevolving sequences with missing values. In *KDD*, New York, NY, USA, 2009. ACM.

[14] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *DMKD '03: Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, New York, NY, USA, 2003. ACM.

[15] S. Mehta, S. Parthasarathy, and R. Machiraju. On trajectory representation for scientific features. In *ICDM*, pages 997–1001, 2006.

[16] D. Newman, C. Chemudugunta, and P. Smyth. Statistical entity-topic models. In *KDD Conference*, pages 680–686, Philadelphia, PA, August 2006.

[17] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. *VLDB*, 2005.

[18] S. Papadimitriou and P. S. Yu. Optimal multi-scale patterns in time series streams. In *SIGMOD Conference*, pages 647–658, 2006.

[19] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *Proceedings of ICDE*, pages 1046–1055, Istanbul, Turkey, April 2007.

[20] Y. Sakurai, S. Papadimitriou, and C. Faloutsos. Braid: Stream mining through group lag correlations. In *Proceedings of ACM SIGMOD*, pages 599–610, Baltimore, Maryland, June 2005.

[21] Y. Sakurai, M. Yoshikawa, and C. Faloutsos. Ftw: Fast similarity search under the time warping distance. In *Proceedings of PODS*, pages 326–337, Baltimore, Maryland, June 2005.

[22] J. Shieh and E. Keogh. isax: indexing and mining terabyte sized time series. In *KDD*, pages 623–631, New York, NY, USA, 2008. ACM.

[23] J. Sun, S. Papadimitriou, and C. Faloutsos. Distributed pattern discovery in multiple streams. *PAKDD*, pages 713–718, 2006.

[24] J. Sun, S. Papadimitriou, and P. S. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *ICDM*, pages 1076–1080, 2006.

[25] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. *KDD*, pages 374–383, 2006.

[26] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Trans. Knowl. Discov. Data*, 2(3):1–37, 2008.

[27] Y. Tao, C. Faloutsos, D. Papadias, and B. Liu. Prediction and indexing of moving objects with unknown motion patterns. In *SIGMOD*, pages 611–622, New York, NY, USA, 2004. ACM Press.

[28] M. E. Wall, A. Rechtsteiner, and L. M. Rocha. Singular value decomposition and principal component analysis. In D. P. Berrar, W. Dubitzky, and M. Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 91–109, Norwell, MA, Mar 2003. Kluwel.

[29] X. Wei, J. Sun, and X. Wang. Dynamic mixture models for multiple time-series. In *IJCAI*, pages 2909–2914, 2007.

[30] B.-K. Yi, N. Sidiropoulos, T. Johnson, H. Jagadish, C. Faloutsos, and A. Biliris. Online data mining for co-evolving time sequences. *ICDE*, pages 13–22, 2000.