Temporal Skeletonization on Sequential Data: Patterns, Categorization, and Visualization

Chuanren Liu
Rutgers, The State University of New Jersey
New Jersey, USA
chuanren.liu@rutgers.edu

Kai Zhang
Autonomic Management, NEC Laboratories America, Inc.
New Jersey, USA
kzhang@nec-labs.com

Hui Xiong[*]
Rutgers, The State University of New Jersey
New Jersey, USA
hxiong@rutgers.edu

Geoff Jiang
Autonomic Management, NEC Laboratories America, Inc.
New Jersey, USA
gfj@nec-labs.com

Qiang Yang
Hong Kong University of Science and Technology
Kowloon, Hong Kong
qyang@cse.ust.hk

ABSTRACT

Sequential pattern analysis targets on finding statistically relevant temporal structures where the values are delivered in a sequence. With the growing complexity of real-world dynamic scenarios, more and more symbols are often needed to encode a meaningful sequence. This is so-called "curse of cardinality", which can impose significant challenges to the design of sequential analysis methods in terms of computational efficiency and practical use. Indeed, given the overwhelming scale and the heterogeneous nature of the sequential data, new visions and strategies are needed to face the challenges. To this end, in this paper, we propose a "temporal skeletonization" approach to proactively reduce the representation of sequences to uncover significant, hidden temporal structures. The key idea is to summarize the temporal correlations in an undirected graph. Then, the "skeleton" of the graph serves as a higher granularity on which hidden temporal patterns are more likely to be identified. In the meantime, the embedding topology of the graph allows us to translate the rich temporal content into a metric space. This opens up new possibilities to explore, quantify, and visualize sequential data. Our approach has shown to greatly alleviate the curse of cardinality in challenging tasks of sequential pattern mining and clustering. Evaluation on a Business-to-Business (B2B) marketing application demonstrates that our approach can effectively discover critical buying paths from noisy customer event data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*

[*]Contact Author.

KDD'14, August 24-27 2014, New York, NY, USA
Copyright 2014 ACM 978-1-4503-2956-9/14/08$15.00.
http://dx.doi.org/10.1145/2623330.2623741

General Terms

Algorithms, Application

Keywords

Temporal Skeletonization, Curse of Cardinality, Sequential Pattern Mining, Network Embedding

1. INTRODUCTION

Unraveling meaningful and significant temporal structures from large-scale sequential data is a fundamental problem in data mining with diversified applications, such as mining the customer purchasing sequences, motion gesture/video sequence recognition, and biological sequence analysis [11]. While there have been a large amount of research efforts devoted to this topic and its variants [1, 2, 7, 10, 25], we are still facing significant emerging challenges. Indeed, with the growing complexity of real-world dynamic scenarios, it often requires more and more symbols to encode a meaningful sequence. For example, in Business to Business (B2B) marketing analytics, we are interested in finding critical buying paths of B2B customers from historical customer event sequences. Due to the complexity of the B2B marketing processes, as well as the difficulty of manually annotating the great variety of customer activities, a large number of symbols is often needed to represent the sequential data. This is known as the "curse of cardinality", which can impose significant challenges to the design of sequential analysis methods from the following perspectives.

- **Complexity**. The computational complexity of finding frequent sequential patterns is huge for large symbol sets. Many existing algorithms have a time complexity that grows exponentially with decreasing pattern supports.
- **Rareness**. In general, the support of a specific sequential pattern decreases significantly with the growing cardinality. To see this, let us consider k symbols that appear with uniform probability in a sequence. The possibility of locating a particular pattern of length ℓ is ℓ^{-k}. In other words, the higher the cardinality, the rarer the patterns are. Since the number of unique subsequences grows with the cardinality, the number of sequences required to identify significant patterns also tends to grow drastically.

- **Granularity**. A large number of symbols in a sequence can "dilute" useful patterns which themselves exist at a different level of granularity. As we will discuss in more detail later, semantically meaningful patterns can exist at a higher granularity level, therefore pattern mining on the original, huge set of symbols may provide little clues on interesting temporal structures.
- **Noise**. Due to the stochastic nature of many practical sequential events, or the multi-modality of events, useful patterns do not always repeat exactly but instead can happen in many permutations. For example, the customers may accidentally download some trial products by mistake when they are looking for the desired information. Without dealing with such irregular perturbations, we may fail to discover some meaningful patterns.

In the literatures, there have been some related works on how to reduce the cardinality in pattern mining by performing a grouping operation on the original symbols. A commonality of these approaches is that they all exploit extra knowledge associated with the symbols as a guidance to perform clustering. For example, a taxonomy of the items may already exist in the form of domain knowledge [18] or can be derived from the structured description of the product features [9]. In Giannotti et al. [7], the 2-dimensional coordinates of spatial points are used to group them into regions to further facilitate the finding of the trajectory patterns. Generally speaking, these approaches first apply clustering on the items whose features are relatively easy to extract, and then search the patterns in different clustering levels.

While these methods have been successfully applied in some application scenarios, there are some emerging issues to be addressed when we face the overwhelming scale and the heterogeneous nature of the sequential data. First, in some applications, it might be difficult to obtain the knowledge of symbols. For example, many sequential data simply use an arbitrary coding of events either for simplicity or security reasons. Second, there are circumstances where it is difficult to define distance among symbols, and therefore clustering becomes impractical. For example, it is unclear how to define the distance between actions customers have taken in their purchasing process. Finally, the biggest concern is that the grouping in these methods is performed irrespective of the temporal content. As a result, these methods may not be able to find statistically relevant temporal structures in sequential data. Therefore, there is a need to develop a new vision and strategy for sequential pattern mining.

To this end, this paper proposes a temporal skeletonization approach to proactively reduce the representation of sequences, so as to expose their hidden temporal structures. Our goal is to make temporal structures of the sequences more concise and clarified, and thus more prone to discovery. Our basic assumption is the existence of symbolic events that tend to aggregate temporally. Then, by identifying temporal clusters and mapping each symbol to the cluster it belongs to, we can reduce not only the cardinality of sequences but also their temporal variations. This allows us to find interesting hidden temporal structures which are otherwise obscured in the original representation.

Exploring temporal clusters from a large number of sequences can be challenging. To achieve this, we have resorted to graph-based manifold learning. The basic idea is to summarize the temporal correlations in the data in an undirected graph. The "skeleton" of the graph (i.e., the temporal clusters) can then be extracted through the graph Lapacian, which serves as a higher granularity where hidden temporal patterns are more likely to be identified. A nice interpretation of such temporal grouping is that when individual symbols are replaced by their cluster labels, the averaged smoothness of all sequences is maximized. Intuitively, this can greatly improve the possibility of finding significant sequential patterns, as we shall observe empirically. In addition, the embedding topology of the graph allows us to translate the rich temporal content of symbolic sequences into a metric space for further analysis and visualization. Compared with existing methods that attempt to reduce the cardinality via clustering, our approach does not require specific knowledge about the items. Instead, it caters directly to the temporal contents of given data sequences. To the best of our knowledge, using the temporal correlations to perform clustering and reduction of representation is a novel approach in sequential pattern mining.

Temporal skeletonization can be deemed as a transformation that maps the temporal structures of sequences into the topologies of a graph. Such a dual perspective provides not only more insights on pattern mining, but also brings powerful new tools for analysis and visualization. For example, many techniques in graph theories can be used to analyze symbolic sequences, which appear as random walks on the created graph. On the other hand, due to the explicit embedding, symbolic sequences are represented as numerical sequences or point clouds in the Euclidean space, for which visualization becomes much more convenient.

Experimental results on real-world data have shown that the proposed approach can greatly alleviate the problem of curse of cardinality for the challenging tasks of sequential pattern mining and clustering. Also, we show that it is convenient to visualize sequential patterns in the Euclidean space by temporal skeletonization. In addition, the case study on a Business-to-Business (B2B) marketing application demonstrates that our approach can effectively identify critical buying paths from noisy marketing data.

2. TEMPORAL SKELETONIZATION

In this section, we introduce the detail of the proposed method. The key concept is the "temporal cluster", namely group of symbols which tend to aggregate more closely together in the sequences. By transforming the sequential data into graphs, we can identify such temporal clusters to greatly simplify the representation of the sequences.

2.1 Temporal Clusters

We believe that temporal clusters often exist in practical sequence data. Otherwise, if there is no such "preferential" structures and everything becomes uniform, we may not find anything interesting. In the following, we discuss two typical scenarios. One involves stage-wise patterns where each stage can be deemed as a temporal cluster; another scenario involves frequent associative patterns.

2.1.1 Case I: Stage-Wise Patterns

First, some sequential processes exhibit stage-wise behaviors; that is, the process typically goes through a number of stages before reaching the final goal, with each stage marked by a collection of representative events. For example, in B2B markets, the business customer will go through stages such as "Investigating more product information", "Trial experience and evaluation", "Contacting customer service for specific information", and "Contacting sales to finalize the purchase". Here, each stage includes a number of events, and the global structure of the underlying process is shaped by the stages as backbones. Note that the order of stages can vary with regard to different customers. Also, events within a stage may or may not have a dominant ordering. However, collectively, we can observe that each stage forms temporally compact clusters. It will be very useful to find such clusters for understanding the global patterns of sequences.

In case of stage-like sequences, it is obviously more meaningful to detect patterns at the stage level. However, the stages are unknown and typically cannot be determined by grouping the symbols based on their features. Therefore, few existing methods could handle such situations. In the following we use one simple example to show that the large number of symbols in stage-like sequences can "dilute" useful patterns which themselves exist at a different level of granularity, posing a big challenge on existing methods.

$$\begin{array}{l} 1:\texttt{m},\texttt{h},\texttt{j},\texttt{f},\texttt{d},\texttt{a},\texttt{i},\texttt{k},\texttt{b} \\ 2:\texttt{j},\texttt{l},\texttt{m},\texttt{a},\texttt{n},\texttt{f},\texttt{b},\texttt{o},\texttt{g} \\ 3:\texttt{e},\texttt{h},\texttt{l},\texttt{c},\texttt{f},\texttt{n},\texttt{i},\texttt{b},\texttt{o} \\ 4:\texttt{h},\texttt{l},\texttt{e},\texttt{c},\texttt{a},\texttt{f},\texttt{k},\texttt{o},\texttt{i} \end{array}$$

For the four sequences above, if we apply pattern mining on the original level of symbols, we would be unable to find any frequent pattern. However, if we cleverly group the symbols in the following way

$$\begin{array}{l} \texttt{A} = \{\texttt{m},\texttt{h},\texttt{j},\texttt{e},\texttt{l}\} \\ \texttt{B} = \{\texttt{a},\texttt{f},\texttt{c},\texttt{d},\texttt{n}\} \\ \texttt{C} = \{\texttt{k},\texttt{g},\texttt{o},\texttt{b},\texttt{i}\} \end{array}$$

then all the four sequences read as

$$\texttt{A},\texttt{A},\texttt{A},\texttt{B},\texttt{B},\texttt{B},\texttt{C},\texttt{C},\texttt{C}$$

It is obvious that $\texttt{A},\texttt{B},\texttt{C}$ is a frequent (stage) pattern with 100% support.

2.1.2 Case II: Frequent Associative Patterns

Temporal cluster also has an interesting connection with frequent associative patterns. Since associative items tend to occur closely to each other, they are temporally more coherent and can likely form temporal clusters. In other words, there must exist temporal clusters if there are significant frequent patterns. However, temporal clusters can be more general than frequent patterns. Another challenge for finding frequent patterns is the noise in the data. If frequent sub-sequences are somewhat perturbed, special cares have to be taken in finding exact patterns. In comparison, the temporal clusters we try to discover are identified via the temporal distribution of all existing event pairs, thus our approach is inherently more resistant to noise.

2.2 Temporal Graph and Skeletonization

Since large cardinality hampers pattern mining, we propose to find meaningful temporal clusters to alleviate it. The key role of temporal clusters is that they can be used to re-encode the original sequences. Since temporal clusters are composed of symbols that are temporally more coherent, the newly encoded sequences will be temporally smoother than the original sequences. By doing this, we can greatly reduce not only the cardinality but also the temporal variations of the sequences. The latter makes it much easier to find semantically useful patterns. Specifically, we put this under the following optimization framework.

Suppose we have a set of symbols $\mathcal{S} = \{e_1, e_2, \cdots, e_{|\mathcal{S}|}\}$. The n-th sequence is denoted by $S_n = (s_1^n, s_2^n, \cdots, s_{T_n}^n)$, where $s_t^n \in \mathcal{S}$ for $t = 1, 2, \cdots, T_n$ and T_n is the length of the nth sequence.

Problem 1 (Temporal skeletonization). *Given a set of sequences $\{S_n | n = 1, 2, \cdots, N\}$, we want to find a new encoding scheme of the symbols $e \in \mathcal{S}$, denoted by the mapping $y = f(e) \in \{1, 2, \cdots, K\}$, such that when encoded with f, the temporal variation of resultant sequences is minimized*

$$\min_{y \in \{1,2,\ldots,K\}} \frac{1}{N} \sum_{n=1}^{N} \sum_{\substack{1 \le p,q \le T_n \\ |p-q| \le r}} \left(f(s_p^n) - f(s_q^n)\right)^2. \tag{1}$$

Here r is a pre-defined integer that controls the range that local sequence variations are computed, and the cardinality of the encoding scheme, K, is a pre-defined integer that is much smaller than that of the original representation $|\mathcal{S}|$.

Here, in each of the N sequences, we only consider pairs of events s_p^n and s_q^n that are within r intervals to each other, such that when they are re-encoded with $f(s_p^n)$ and $f(s_q^n)$, they are similar to each other.

This is an integer programming problem which has shown to be NP-hard. Therefore, we propose to relax the integer constraint to real numbers. In addition, we will define the so-called "temporal graph" to re-phrase the problem as a graph optimization one.

Definition 1 (Temporal graph). *Let G be a weighted graph $G = \langle V, E \rangle$ with vertex set $V = \mathcal{S}$ and edges E. The i-th node of G corresponds to the i-th symbol e_i in the symbol set $\mathcal{S}$. The weight of the edge between node i and node j is defined as the ij-th entry of an $|\mathcal{S}| \times |\mathcal{S}|$ matrix W, where*

$$W_{ij} = \frac{1}{N} \sum_{\substack{1 \le n \le N \\ e_i, e_j \in S_n}} \delta\left(|\ell(e_i, S_n) - \ell(e_j, S_n)| \le r\right). \tag{2}$$

Here, we say $e \in S_n$ if the symbol e can be observed in the sequence S_n and $\ell(e, S_n) \in \{1, 2, \cdots, T_n\}$ is the corresponding location of e in the sequence S_n.

We call G "temporal graph", because the edge weight of the graph captures the averaged temporal closeness between any pair of symbols/events across all the input sequences. With this definition, and let $\mathbf{y} \in \mathbb{R}^{|\mathcal{S}|}$ where $\mathbf{y}_i = f(e_i)$, Problem 1 can be written in the following compact form

$$\min_{\mathbf{y} \in \mathbb{R}^{|\mathcal{S}|}} \sum_{i,j} W_{ij} (\mathbf{y}_i - \mathbf{y}_j)^2. \tag{3}$$

Problem (3) has a standard form of graph-based optimization. Let us define the graph Laplacian of G as $L = D - W$, where D is a diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$. Then, Equation 3 can be formalized as

$$\begin{aligned} \min_{\mathbf{y} \in \mathbb{R}^{|\mathcal{S}|}} \quad & \mathbf{y}'(D - W)\mathbf{y} \\ s.t. \quad & \mathbf{1}' D \mathbf{y} = 0 \\ & \mathbf{y}' D \mathbf{y} = 1 \end{aligned} \tag{4}$$

where $\mathbf{1}$ is a vector of all 1's. Here, the translation and scale constraints are added to avoid trivial solutions. This is also known in the literatures as Laplacian eigenmap [3], which has also been applied in spectral clustering [14]. To the best of our knowledge, it is a novel application to use graph-based algorithm to extract interesting temporal structures from multiple sequences.

Note that the more often symbol e_i and symbol e_j appear close to each other in the sequences, the higher the W_{ij} is and the larger the penalty it induces on the objective (Equation 4), and as a result, the closer $\mathbf{y}_i$ and $\mathbf{y}_j$ should be. This equivalently achieves a grouping of the symbols, which are the temporal clusters we try to extract. As can be expected, by re-encoding the sequence with the label of the temporal clusters, we can improve the temporal smoothness which is beneficial to subsequent pattern mining.

The level of smoothness can be adjusted effectively by the order parameter r. The order parameter controls the resolution on which clusters are extracted. A larger r captures the similarities among events in a longer temporal range, which potentially lead to fewer clusters, while a small r only considers directly adjacent symbols as similar, which lead to more clusters. In the extreme case when r approaches infinity, W becomes a constant matrix, and all the events will be categorized into one cluster. In practice, instead of using the delta function ($\delta\left(|\ell(e_i, S_n) - \ell(e_j, S_n)| \leq r\right)$) to measure the similarity, one can also use a smoother function such as

$$W_{ij} = \frac{1}{N} \sum_{\substack{1 \leq n \leq N \\ e_i, e_j \in S_n}} \kappa_h\left(|\ell(e_i, S_n) - \ell(e_j, S_n)|\right), \qquad (5)$$

where κ_h is a non-increasing function parametrized by h. For example, we can use the exceedance of the Exponential distribution $\kappa_h(d) = \exp(-hd)$.

2.3 Embedding and Visualization

The optimal solution of Equation 4 is the eigenvector of the graph Lapacian corresponding to the second smallest eigenvalue. In practice, one usually computes several eigenvectors and applies some simple clustering algorithms such as K-means or GMM (Gaussian mixture model) in this low-dimensional space. The useful eigenvectors of the graph Lapacian not only provide a relaxed solution for finding temporal clusters, but also more interestingly, naturally connect to the manifold embedding of the graph.

Note that the eigenvectors of the graph can be deemed as a low-dimensional embedding, in which the proximity relation among objects preserves that in the original space [24]. Since the similarity measurements in W_{ij} of the graph reflect the temporal closeness of the events, the embedding eigenvectors of the graph will also inherit this configuration. Namely, if two symbols, e_i and e_j are temporally more related, their distance will also be small in the embedded space. In other words, our approach provides a direct platform for visualizing the temporal structures of sequential data. We believe that such visualization can provide interesting insights allowing domain experts to draw useful conclusions.

To provide more intuition on the temporal embedding results, in Figure 1, we give several examples. Figure 1a is the embedding of a collection of random sequences. As can be seen, the embedded symbols (each represented by one point) are distributed uniformly and there is hardly any interesting structure. Figure 1b is a simulated data containing 5 stages of events (more details in Section 4). As can be seen, there are clear clusters in the embedding, each representing exactly events belonging to one stage. In Figure 1c we used a real-world data set composed of thousands of B2B customer event sequences. As can be seen, the cluster structures are complicated: some clusters are well separated while others are diffusing. This has to do the complicated relationships between practical events in the data set. From these examples, we can see that our temporal skeletonization approach can translate the temporal structures in the sequential data into their topological counterparts. The resultant visualization can bring useful insights.

In the literatures, there are many algorithms for manifold learning. Many of these approaches rely on the eigenvalue decomposition of a similarity matrix to obtain the manifold embedding. For example, Isomap [20] is another popular method that embed a graph into an Euclidean space. In our experiments, we also use Isomap to visualize the data, and find that it can provide spatially more unfolded embedding. The details of using Isomap are provided in the Appendix.

2.4 Post-Temporal-Smoothing

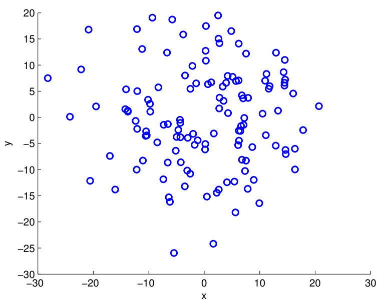
By finding temporal clusters in the embedded Euclidean space, and use it to re-encode the sequence, we can obtain temporally smoother representation. For example, we can transform the original customer event sequences to sequences of stages, with each stage being defined as the groups of symbols (marketing campaigns) in the temporal clusters identified. However, although the embedded graph is estimated robustly with the integrated data from all the sequential observations, the individual sequence might still be noisy in some cases, for instance, the order parameter r is chosen too small. Thus, one might want to further smooth away the irregularities in the sequences. To this end, we propose a multi-series post-smoothing using fused lasso [21].

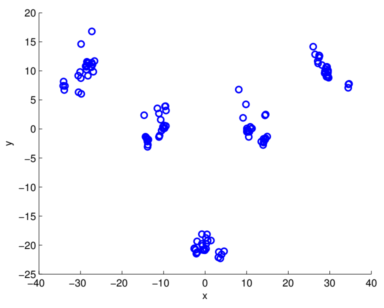
Specifically, we use the Gaussian mixture models to perform a soft clustering on the embedded symbols. We denote this by a partition matrix $Y \in \mathbb{R}^{|\mathcal{S}| \times K}$ where Y_{sk} is the probability that the symbol $s \in \mathcal{S}$ belongs to the k-th temporal cluster. Then we can use this partition matrix to transform each individual sequence $S_n = (s_1^n, s_2^n, \cdots, s_{T_n}^n)$ into a multiple of K sequences, denoted by $Y^n \in \mathbb{R}^{T_n \times K}$ where Y_{tk}^n is the probability that the t-th interaction s_t^n in S_n belongs to the k-th cluster. Then we try to find a smoother version of the multiple sequences Y^n, denoted by X^n. To achieve this, we encourage sparsity of the differences between the successive rows in X^n, i.e., $\sum_{t=1}^{T_n-1} \|X_t^n - X_{t+1}^n\|_1$ where $\|X_t^n - X_{t+1}^n\|_1 = \sum_{k=1}^{K} |X_{tk}^n - X_{(t+1)k}^n|$. We optimize the approximation of X^n by maximizing the alignment $\sum_{t=1}^{T_n} \sum_{k=1}^{K} X_{tk}^n Y_{tk}^n$ and with probability constraints. Thus, we would like to maximize $\sum_{t=1}^{T_n} \sum_{k=1}^{K} X_{tk}^n Y_{tk}^n$, subject to

$$\frac{1}{T_n - 1} \sum_{t=1}^{T_n - 1} \|X_t^n - X_{t+1}^n\|_1 \leq \lambda,$$

in addition to $\sum_{k=1}^{K} X_{tk}^n = 1$ and $X_{tk}^n \geq 0$. Here λ is a tuning parameter controlling smoothness of X^n.

To solve the approximation problem with the smooth constraint, we let $X_{tk}^n - X_{(t+1)k}^n = \alpha_{tk}^n - \beta_{tk}^n$ for $t = 1, \cdots, T_n - 1$ and $k = 1, \cdots, K$ where $\alpha_{tk}^n, \beta_{tk}^n \geq 0$. Let $A^n \in \mathbb{R}^{(T_n-1) \times T_n}$ with $A_{tt}^n = 1$, $A_{t(t+1)}^n = -1$ and $A_{tt'}^n = 0$ otherwise so that $A^n X^n = \alpha^n - \beta^n$. We can rewrite the smooth approxima-


(a) Random sequences.


(b) Stage-wise sequences.

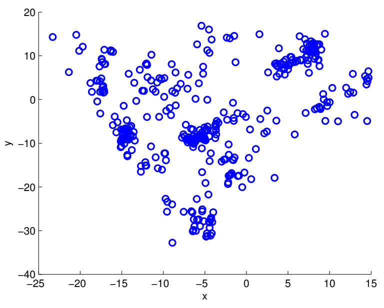

(c) B2B customer event sequences.

Figure 1: The embedding of symbols in different types of sequence data.

tion as a linear programming problem:

$$\begin{aligned} \max \quad & \sum_{t=1}^{T_n} \sum_{k=1}^{K} X_{tk}^n Y_{tk}^n, \\ s.t. \quad & \sum_{k=1}^{K} X_{tk}^n = 1, \forall t = 1, \cdots, T_n, \\ & A^n X^n = \alpha^n - \beta^n, \\ & \frac{1}{T_n - 1} \sum_{t=1}^{T_n - 1} \sum_{k=1}^{K} (\alpha_{tk}^n + \beta_{tk}^n) \leq \lambda, \end{aligned}$$

in addition to the non-negative constraints $X^n, \alpha^n, \beta^n \geq 0$.

3. APPLICATIONS

In this section, we discuss the applications of the proposed temporal skeletonization method in several interesting problems. The reduced representation makes it much easier to perform these tasks than on the original sequences.

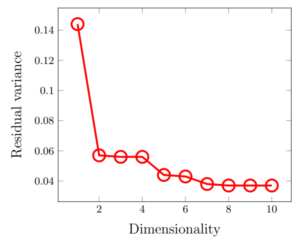
3.1 Sequence Visualization

Our framework embeds symbolic events in sequence data into an Euclidean space, which allows to visualize each sequence as a trajectory. Such visualization can provide direct insights on the relationship between events, which, when subject to examination of domain expert, can greatly facilitate them making analysis and decisions. In Section 5, we will report how temporal clusters in the B2B customer event data can help to understand typical purchase patterns.

In choosing the embedding dimensions, we typically choose two or three dimensions, which correspond to the dominant components of the temporal graph. Usually, these few dimensions can encode a sufficient amount of the temporal relations. To see this, we investigate the residual variance in the Isomap with regard to the number of selected embedding dimensions. As shown in Figure 2, in both the simulation and real-world data sets, the residual variance drops most significantly with the first few dimensions.

3.2 Sequential Pattern Mining

By identifying meaningful temporal clusters, our method transforms the original sequence of events to the sequence of temporal clusters (the cluster labels are used as a new set of symbols to encode events). This helps to reduce the cardinality of the sequence representation, and the supports


(a) Simulation data.

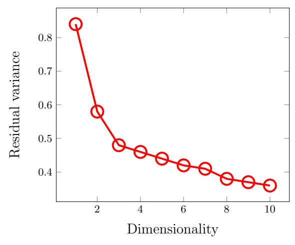

(b) B2B customer data.

Figure 2: The residual variance vs. dimensionality.

of the sequential patterns are increased. As a result, consequent sequential pattern mining is able to discover significant knowledge which otherwise would be diluted in the raw data. Indeed, the patterns discovered in this way are defined with a higher level of granularity, i.e., the temporal clusters. Therefore, to interpret the patterns, we can first annotate the temporal clusters with domain knowledge.

For example, in the B2B customer event data, the temporal event clusters can be semantically labelled as Webinar, Tradeshow, Direct Marketing Mail, Web Marketing Ads, Trial Product Download, and Unsubscribe, etc. As we shall see, these temporal clusters correspond to stages in the purchasing route of the customers, which are much easier to understand and interpret compared with the raw sequences.

3.3 Sequence Clustering

Sequence clustering is an important task, however, it is not always easy to extract appropriate features or define distances among sequences so that clustering can be performed properly. This is particularly true when sequences are represented by a huge number of symbols. The temporal skeletonization method we have proposed can be used to tackle these difficulties. This is because the temporal skeletonization can remove noises in the sequences based on their collectively temporal behaviours. More importantly, it re-summarizes the events in the form of groups of events, therefore we will observe much more repeated subsequence on which sequential features can be more meaningful.

For example, when there is only a reasonably small number of symbols in the sequences, we can extract the following useful features, such as the counts of each temporal cluster passed by the sequence, or how many times one symbol appears in precedence of another, and so on. It turns out such a straightforward approach can effectively cluster the sequences. To incorporate more temporal information, we can also leverage the frequent sequential patterns discovered by the aforementioned sequential pattern mining, as suggested by Lee et al. [13].

4. EMPIRICAL EVALUATION

In this section, we evaluate the performances of our approach in comparison with several state-of-the-art methods. All the experiments are performed on a GNU/Linux system with 8 CPUs (Intel i7 2.93GHz) and 8G RAM.

4.1 Synthetic Data

We have simulated symbolic sequential data composed of stages of events. We define 5 stages $\{A, B, C, D, E\}$, where each contains 25 symbols. Then, we create 5000 sequences that are of two patterns. The first 2500 sequences mainly follow stage pattern $A \to B \to C \to D$; the other 2500 sequences follow $B \to E \to C$. The simulation proceeds as follows. After deciding which stage to sample from based on the two patterns, we randomly pick d symbols from that stage, where d is a random integer. Then, we inject the selected symbols into the sequence, and continue to the next stage in the pattern. Indeed, such a simulation process is equivalent to a standard Hidden Markov Model (HMM), where 5 stages correspond to 5 hidden states and symbols within each stage correspond to observations. Let the transition probability from each stage to itself be p, and that to the next stage (as specified in the two patterns) be $1 - p$. Then, the stage duration d follows a geometric distribution $d \sim (1-p)p^{d-1}$, with the expected value $\mathbb{E}[d] = \frac{1}{1-p}$. To have significant stage-wise patterns in the produced sequences, we have used a large probability $p = \frac{14}{15}$, leading to $\mathbb{E}[d] = 15$. In other words, on average, we randomly pick 15 symbols for each stage in the sequences.

4.2 Baselines

First, we apply state-of-the-art Frequent Sequence Mining (FSM) algorithms, including GSP [19], SPADE [25], PrefixSpan [10], SPAM [2]. The results in Figure 3 show that, when desired pattern support drops, the time consumption of these algorithms grow super-exponentially, indicating the difficulties introduced by the large numbers of symbols. The number of detected patterns also becomes explosive, most of which are non-informative and provide no clear insight of the underlying sequence generating processes (as shown in Table 1). In comparison, using the temporal clusters identified via our approach (more details in Section 4.3), the mining process succeeds quickly in one second.

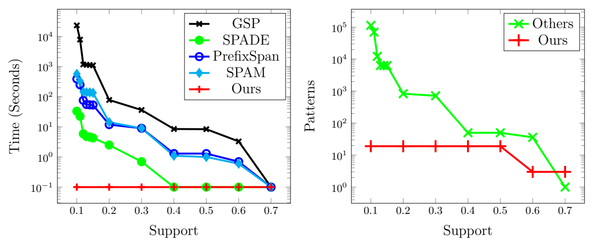


Figure 3: FSM algorithms on the simulated data.

In addition to the improvement on efficiency, we also compare the pattern mining results on the original and the re-encoded sequences via our method in Table 1. For the task of pattern mining, we compute the precision (fraction of discovered patterns that are relevant) and recall (fraction of the relevant patterns that are discovered) of the discovered patterns against the ground truth. The results show that when working on the raw data, FSM performs poorly with an F-measure around 0.281. In contrast, after re-encoding using our approach, it can lead to an 100% accuracy.

Task	Pattern Mining		Sequence Clustering		Stage Recovery	
Method	FSM	Ours	HMM	Ours	HMM	Ours
Precision	0.725	**1.0**	0.997	**1.0**	0.488	**1.0**
Recall	0.174	**1.0**	0.997	**1.0**	0.448	**1.0**

Table 1: Utility comparison on the simulated data.

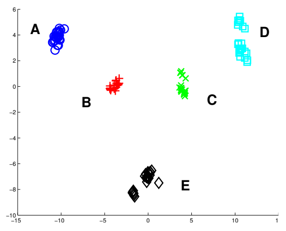
Since data simulation process follows the Markov property, the second baseline approach we have experimented with is the classical HMM. In our data, there are two hidden patterns, thus we adopt the HMM based clustering (HMMC) [15] to simultaneously cluster the sequences and estimate the HMM parameters for each cluster. Specifically, to group sequences into M clusters, the HMMC randomly allocates all sequences to M disjoint subsets as initial clusters, then the following two procedures are iterated until convergence. First, for all sequences in cluster C_m, we estimate a transition matrix ϕ_m and a emission matrix θ_m; Second, we reallocate each sequence S_n to the cluster C_m on whose transition and emission matrices it has the highest probability of being produced, i.e., $m = \arg\max_m \Pr(S_n|\phi_m, \theta_m)$.

We have provided the HMMC method with some ground truth parameters, i.e., the number of clusters $M = 2$, and the number of hidden states (stages) for each cluster. Table 1 shows the accuracy of the HMM based method for the task of sequence clustering and stage recovery. To be specific, for these two tasks, we first compute the so-called confusion matrix C, where C_{ij} is the number of instances in resulted group i and ground truth class j. Then, the precision is computed as $\frac{1}{N}\max_\sigma \sum_j C_{\sigma(j)j}$ where $N = \sum_{ij} C_{ij}$ and σ maps classes to different groups; the recall is computed as $\frac{1}{N}\sum_i \max_j C_{ij}$, which is also termed as clustering purity. Again, our method gives perfect results on both tasks. The HMMC only works well for sequence clustering, while its performance is almost random for the task of stage recovery.

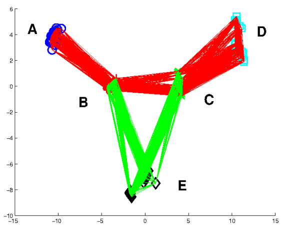
4.3 Our Results

Here, we show that our approach can successfully recover patterns hidden in the data. In Figure 4a, we see that our approach embeds altogether 125 symbols in such a way that 5 dominant clusters emerge. This is in perfect consistency with the ground truth structure we have used in the simulation. In Figure 4b, we show that, by extracting simple features as discussed in Section 3.3, we can correctly group the 5000 sequences into two clusters (in red and green).

Our approach does not require any prior knowledge on the simulated data. We tried different ways (Equation 2 and Equation 5) to construct the temporal graph, and the results were very robust with respect to parameters (e.g., $1 \le r \le 5$ and $1 \le h \le 5$). Also, we can clearly identify the number of stages based on well clustered symbols. Finally, the post-temporal-smoothing procedure works with λ in a wide range, e.g., $1 \le \lambda \le 10$.



(a) The temporal clusters.

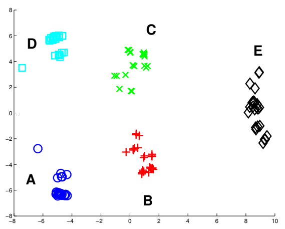


(b) The sequence clusters.

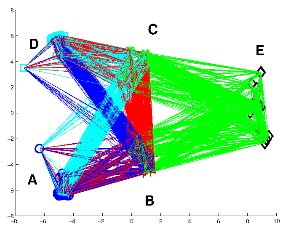
Figure 4: The embedding of simulated data.

4.4 Noisy Cases

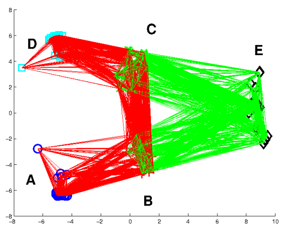
Now we examine the performances of our approach in case of noisy data. We have injected two types of noises. The first is introduced on items of each sequence, such that one stage could contain symbols that belong to other stages. Such a noisy behavior is quite natural in the buying process of customers. For example, customers might occasionally participate events not very relevant to their current buying stages. The second kind of noise is introduced as random sequences not following any certain patterns. In real world, these random sequences might correspond to event sequences of some unintended customers. We have 5% noisy observations for each type of noise. Moreover, we have added another two stage-wise patterns to make the problem more challenging.



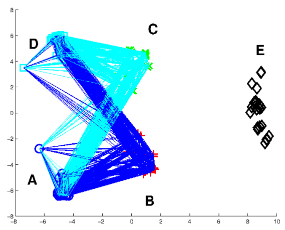
(a)



(b)



(c)



(d)

Figure 5: The noisy simulated data. a: The temporal clusters. b: The sequences clusters. c: Pattern $A \to B \to C \to D$ and $B \to E \to C$. d: Pattern $A \to B \to D$ and $A \to C \to D$.

We can see from Figure 5a that, even in this noisy data, the temporal clusters are still identifiable, which correspond to stages of events. The reason is that, our temporal graph is estimated robustly with integrated temporal content from all the sequences, therefore a small portion of individual noisy observations cannot significantly affect the result. In particular, the post-temporal-smoothing (Section 2.4) can also be very useful in removing the random events. As shown in Figure 5b, once noisy random events are removed, we can discover important patterns (groups of sequences). Indeed, the 4 stage-wise patterns are all discovered by our approach.

We report the results of standard FSM and HMMC in Table 2. As can be seen, their performances both degrade compared with the noiseless case, while our approach still successfully returns the ground truth patterns.

Task	Pattern Mining		Sequence Clustering		Stage Recovery	
Method	FSM	Ours	HMM	Ours	HMM	Ours
Precision	0.526	**1.0**	0.688	**1.0**	0.480	**1.0**
Recall	0.134	**1.0**	0.621	**1.0**	0.416	**1.0**

Table 2: Utility comparison on the noisy cases.

5. B2B PURCHASE PATTERN ANALYSIS

In this section, we apply our method to find critical buying paths of Business-to-Business (B2B) buyers from historical customer event sequences. Since B2B purchases are often involved with strategic development of the company, and as a result, extra cautions and extensive research efforts have to be taken in making such investment, the decision process of customers in purchasing certain products or services is much more complicated than that in our daily purchasing activities. Thus, it is of significant business value if we can discover characteristic and critical buying paths from observations. These can be used to recommend directed advertising campaigns so as to increase potential profits and also reduce the marketing cost. In addition, we would also like to visually display the buying processes of the customers. By doing this, we can better understand the behavior patterns of B2B customers and accordingly develop promising marketing strategies. In the following, we show that our framework is effective for these objectives.

5.1 Data Description

We have collected huge amount of purchasing event data for the customers of a big company. In more detail, we have event sequences from $N = 88040$ customers, with the number of unique events (symbols) $|\mathcal{S}| = 5028$, leading to altogether $T = \sum_n T_n = 248725$ event records. We construct the temporal graph using Equation 5 and $h = 5$, and then embed the graph using Isomap. In the appendix, we describe details on how to apply Isomap embedding on the temporal graph we have constructed.

5.2 Embedding Results and Buying Stages

We plot the embedding of selected 503 events (top 10% nodes with the largest degrees in the temporal graph) in Figure 6, and mark it with the clustering results. For each detected cluster, we are able to extract dominant semantic keywords for the events in that cluster, as shown in Table 3. Note that the semantic information here is only used to summarize each temporal cluster for better understanding of our results, but not for the purpose of grouping the events.

We discuss some interesting observations on the temporal clusters. First, clusters that are close to each other appear logically related, e.g., ‘search engine’ (C_{13}) and ‘trial product download’ (C_3); ‘webinar’ (C_{12}) and ‘trade show’ (C_8). Second, symbols with the same semantic meaning may not be in the same temporal cluster. For example, C_6, C_{11}, C_{12}

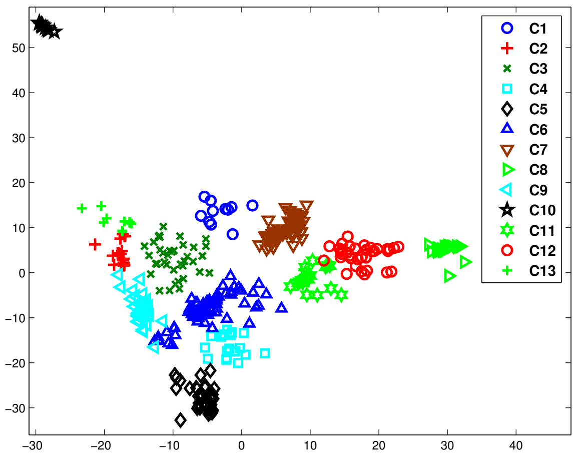


Figure 6: The customer event clusters.

Cluster	Top keywords	Size
C_1	`Official Website`	12
C_2	`Corporate Event, Direct Marketing Mail`	20
C_3	`Trial Product Download`	45
C_4	`Conference`	27
C_5	`Unsubscribe`	38
C_6	`Webinar`	101
C_7	`Trial Product Download`	70
C_8	`Tradeshow`	37
C_9	`Corporate Event, Direct Marketing Mail`	65
C_{10}	`Web Marketing Ads`	13
C_{11}	`Webinar`	21
C_{12}	`Webinar`	42
C_{13}	`Search Engine`	12

Table 3: The semantic annotation of event clusters.

are all marked with 'webinar' but they form 3 separate clusters. Note that these three clusters are close to 'direct marketing mail', 'trial product download', 'trade show', respectively, indicating that they have different levels of maturity towards final purchase. Thus, it is reasonable to have them separated. Nevertheless, the three clusters are still neighbors, after all, since they have the same nature ('webinar'). In other words, temporal clusters could be partially consistent with attribute-based clusters, while meanwhile revealing more fine-grained structure by exploiting the temporal correlations. This is where the extra value comes from.

5.3 Critical Buying Paths

With the detected temporal clusters, we apply the post-temporal-smoothing, since the data set we collected is very noisy and can be subject to human errors. Then, we can transform the original event sequences to sequences of temporal clusters, and apply the FSM algorithms on the re-encoded sequences. Figure 7 reports some results of pattern mining on the original and re-encoded sequences respectively. The pattern supports are generally chosen much smaller than those used in the simulation study, since we have much more symbols here and more complicated patterns. As can be observed, on the original sequences, the number of identified patterns again grows super-exponentially with decreasing support. This would indicate that it is very hard to have a conclusive and comprehensive understandings of the customers' purchasing patterns. In contrast, using the transformed sequences via temporal skeletonization, the number of detected patterns is much more reasonable.

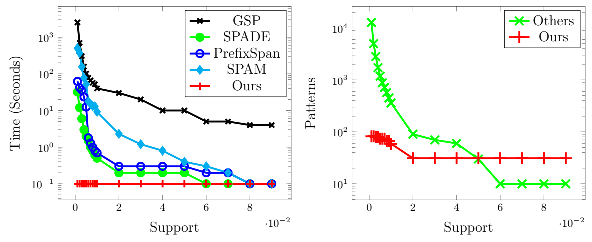


Figure 7: Sequential patterns in B2B customer event data.

We then perform clustering on the transformed sequences, using the frequent patterns detected to extract features as discussed in Section 3.3. We focus on a few dominant clusters in which the customers have relatively longer event sequences. The remaining customers only participated events in one or two buying stages and their behaviors are almost random. In Figure 8, we plot the sequences corresponding to some dominant clusters covering 3501 customers, by connecting each event of the sequence embedded in the two-dimensional plane. Here, each cluster corresponds to one type of customers with a unique buying path. We summarized these buying paths in Table 4.

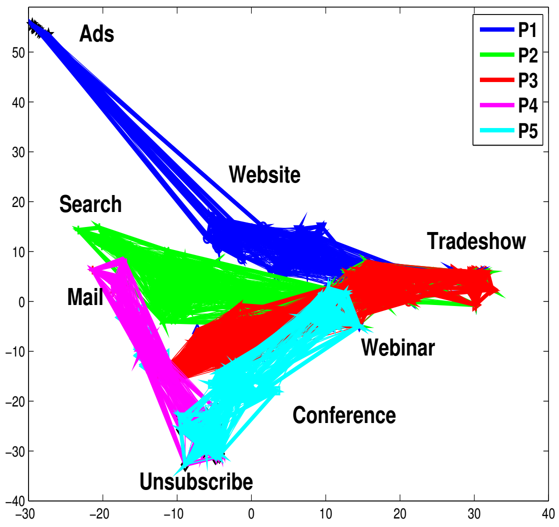


Figure 8: The customer buying paths.

With the semantic annotation in Table 3, we can see that the temporal clusters can be used to reveal several interesting buying paths. For example, the blue path P_1 passes through 5 clusters (or stages), as $C_{10} \to C_1 \to C_7 \to C_{12} \to C_8$. These customers were attracted by 'Web Marketing Ads', then they went to 'Official Website' and found the 'Trial Product Download'. When customers needed more information to make decisions, they continued to attend 'Webinar' and finally went to 'Tradeshow'. The green path P_2 also ends with 'Trade Show', but starts with 'Search Engine', indicating that these customers started from their own effort in acquiring information of the product suiting their needs. In comparison, the pattern in the red path P_3 is simpler, starting with 'Webinar' and ending with 'Tradeshow'. All these three paths can be grouped into the 'Successful' class, which leads to the higher maturity of the customers. The remaining two paths, P_4 and P_5, which end with 'Unsubscribe', indicating these customers do not want to participate further events any more.

These buying paths reveal different psychologies of B2B customers. Among the successful class, P_1 and P_2 represent customers that are comfortable with self-motivated/directed actions, e.g., searching product information themselves or browsing advertisements. In comparison, in the unsuccessful class, P_4 represents customers passively involved via 'direct marketing mail' but finally choose to give up. For paths P_3 and P_5, although they both start from webinar, they branch to opposite routes. We speculate that P_3 are easy customers; while P_5 are customers that are relatively more difficult to persuade. These uncovered patterns can be helpful in guiding the marketing campaigns, such as identifying customer groups, initiating more customer-friendly and less commercialized advertisements, and so on.

Class	Path	Path/Keyword	Size
Successful	P_1	$C_{10} \rightarrow C_1 \rightarrow C_7 \rightarrow C_{12} \rightarrow C_8$ `Ads→Website→Download→Webinar→Tradeshow`	933
	P_2	$C_{13} \rightarrow C_3 \rightarrow C_{11} \rightarrow C_{12} \rightarrow C_8$ `Search→Download→Webinar→Webinar→Tradeshow`	1110
	P_3	$C_6 \rightarrow C_{11} \rightarrow C_{12} \rightarrow C_8$ `Webinar→Webinar→Webinar→Tradeshow`	702
Unsuccessful	P_4	$C_2 \rightarrow C_9 \rightarrow C_5$ `Mail→Corporate Event→Unsubscribe`	423
	P_5	$C_{11} \rightarrow C_4 \rightarrow C_5$ `Webinar→Conference→Unsubscribe`	333

Table 4: The annotation of sequence clusters/buying paths.

From these observations, we can see that temporal clusters do represent meaningful buying stages and they greatly facilitate the description of typical buying paths. Identifying the buying paths/stages for B2B customers are critical to many marketing applications. This clearly demonstrates the advantages of our approach.

6. RELATED WORK

Sequential pattern mining [1] is an important topic in data mining. Given a database of customer transactions, where each transaction consists of customer-id, transaction time, and the items bought, sequential pattern mining finds frequent sequences of item sets. Some research efforts [25, 10, 2] focused on the computing efficiency. In addition to the customer behavior analysis, sequential data from other domains have also been exploited. For example, Giannotti et al. [7] proposed to find trajectory patterns from the location traces of moving objects to study their movement behaviors.

However, limited efforts [18, 9, 7] have been focused on the "curse of cardinality" problem. As we discussed in Section 1, these methods typically reduce the cardinality by performing a grouping of symbols, for which either a taxonomy already exists [18], or extracted from domain knowledge [9], or through clustering on the features associated with the symbols [7]. These grouping are irrespective of the temporal content in the sequences, while our approach achieves the grouping of symbols based on their temporal relations. It is worthy to note that combining the two types of clustering is a very interesting topic we shall pursue in the future.

Instead of reducing the cardinality of original symbols/items, one can also compress the discovered patterns for more concise interpretation. Pei et al. [16] computed so-called condensed frequent pattern bases to approximate the support of any frequent pattern. Xin et al. [22] proposed to compress frequent patterns with representative patterns via clustering. In these methods, an initial set of frequent patterns has to be identified first, which could suffer from the large cardinality. In comparison, our approach can be deemed as compressing the original set of symbols.

Another category of related work is rank aggregation [17], which tries to find a unified ranking of a set of elements that is "closest to" a given set of input (partial) rankings. For example, each customer event sequence can be deemed as a ranking of the participated events. Methods include position based statistics [5] and permutation optimization [6, 8], etc. However, rank aggregation is suited only when there is a dominant ordering in the data. When there are different patterns of the ordering, rank aggregation fails to give a valid result. In comparison, our approach can identify different types of orders as discussed in Section 3.3.

The HMM is another widely used method for sequence analysis. Most algorithms for HMM estimation are supervised; that is, hidden states in the training data need to be provided for model estimation. However, in the B2B customer event sequence analysis, it is very difficult to obtain labels for the buying stages corresponding to individual events. For unsupervised estimation of HMM, Expectation Maximization (EM) is often used, which could suffer from the local optimum problem. In our simulated study (Section 4), we observed that unsupervised HMM estimation often fails to recover the ground truth. In the literature, there have been convex approaches for the learning of HMM [23, 12]. However, this is achieved by avoiding explicit estimation of model parameters (transition matrix, emission matrix, etc.), hence not applicable if the model parameters are needed. Markov models were also used for clustering sequential data [4], as we also exploited in our empirical study [15]. Simultaneous clustering the sequential data and computing the temporal skeletonization shall be an interesting future work.

7. CONCLUDING REMARKS

In this paper, we proposed a novel approach of temporal skeletonization to address the problem of "curse of cardinality" in sequential data analysis. The key idea is to map the temporal structures of sequences into the topologies of a graph in a way that the temporal contents of the sequential data are preserved in the so-called temporal graph. Indeed, the embedding topology of the graph can allow to translate the rich temporal content into the metric space. Such a transformation enables not only sequential pattern mining at a more informative level of granularity, but also enables new possibilities to explore, quantify, and visualize statistically relevant temporal structures in the metric space. Finally, the experimental results showed the advantages of temporal skeletonization over existing methods. Also, the case study showed the effectiveness of the proposed method in terms of finding interesting buying paths from real-world B2B marketing data, which otherwise would be hidden.

APPENDIX

Isomap [20] is a method to embed a graph into a Euclidean space. It works on graphs weighed by a distance matrix $\mathbf{G}$, where $\mathbf{G}_{ij}$ is the distance between nodes i and j. Isomap first sparsifies the distance matrix by only preserving edges between K-nearest neighbours. Then, it computes the geodesic distance $\mathbf{D}_{ij}$'s using the Dijkstra's algorithm. The matrix $\mathbf{D}$ is (entry-wise) squared and converted to an inner product matrix via $\mathbf{K} = -\frac{1}{2}\mathbf{H}(\mathbf{D} \circ \mathbf{D})\mathbf{H}$, where $\mathbf{H}$ is the centering matrix. Finally, the eigenvectors of $\mathbf{K}$ with dominant eigenvalues are used to construct the embedding $\mathbf{X}$ that maximally preserves the manifold structure. We used the code from `http://isomap.stanford.edu`.

To apply Isomap on the temporal graph (Section 2.2), we convert temporal similarities $\mathbf{W}_{ij}$'s to distances. Since we mainly used radial basis function $\kappa_h(d) = \exp(-h \cdot d)$, which gives the relation $\mathbf{W}_{ij} = \exp(-h \cdot \mathbf{G}_{ij})$, we can transform the similarity to distance via $\mathbf{G}_{ij} = -\frac{1}{h}\log(\mathbf{W}_{ij})$.

Acknowledgements

This research was partially supported by National Science Foundation (NSF) via grant numbers CCF-1018151 and IIS-1256016.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE*, 1995.

[2] Jay Ayres, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Sequential pattern mining using a bitmap representation. In *SIGKDD*, 2002.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, 2001.

[4] Igor Cadez, David Heckerman, Christopher Meek, Padhraic Smyth, and Steven White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7 (4):399–424, 2003.

[5] Don Coppersmith, Lisa Fleischer, and Atri Rudra. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *Symposium on Discrete algorithm*, 2006.

[6] Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *WWW*, 2001.

[7] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *SIGKDD*, 2007.

[8] Aristides Gionis, Heikki Mannila, Kai Puolamäki, and Antti Ukkonen. Algorithms for discovering bucket orders from data. In *SIGKDD*, 2006.

[9] Jiawei Han and Yongjian Fu. Mining multiple-level association rules in large databases. *TKDE*, 11(5): 798–805, 1999.

[10] Jiawei Han, Jian Pei, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and MC Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 2001.

[11] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15 (1):55–86, 2007.

[12] Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5): 1460–1480, 2012.

[13] J-G Lee, Jiawei Han, Xiaolei Li, and Hong Cheng. Mining discriminative patterns for classifying trajectories on road networks. *TKDE*, 23(5):713–726, 2011.

[14] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *NIPS*, 2002.

[15] Lane MD Owsley, Les E Atlas, and Gary D Bernard. Automatic clustering of vector time-series for manufacturing machine monitoring. In *ICASSP*, 1997.

[16] Jian Pei, Guozhu Dong, Wei Zou, and Jiawei Han. On computing condensed frequent pattern bases. In *ICDM*, 2002.

[17] Frans Schalekamp and Anke van Zuylen. Rank aggregation: Together we're strong. In *ALENEX*, 2009.

[18] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. In *VLDB*, 1995.

[19] Ramakrishnan Srikant and Rakesh Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *EDBT*, 1996.

[20] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500): 2319–2323, 2000.

[21] Tibshirani et al. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

[22] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In *VLDB*, 2005.

[23] Linli Xu, Li Cheng, Tao Wang, and Dale Schuurmans. Convex hidden markov models. In *NIPS Workshop on Advances in Structured Learning for Text and Speech Processing*, 2005.

[24] Shuicheng Yan, Dong Xu, Benyu Zhang, Hong-Jiang Zhang, Qiang Yang, and Stephen Lin. Graph embedding and extensions: a general framework for dimensionality reduction. *TPAMI*, 29(1):40–51, 2007.

[25] Mohammed J Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine learning*, 42(1-2): 31–60, 2001.