Density-based Pattern Discovery in Time Series

Josenildo C. da Silva[1], Gustavo H. B. Oliveira[1], Omar A. C. Cortez[1], and Matthias Klusch[2]

[1] Instituto Federal do Maranhão (IFMA)
Depto. Acad. de Informática
Av. Getúlio Vargas, 04, Monte Castelo
65.030-005 – São Luís – MA – Brazil
[2] Detsches Forschungszentrum für Künstiliche Intelligenz (DFKI GmbH)
Forschungsbereich Agenten und Simulierte Realität
Campus Gebäude D 3 2, D-66123 Saarbrücken, Germany

Abstract. Time series data is a very common kind of data in many different fields. Despite of that, time series has not received a lot of attention in the data mining community. Unknown frequent pattern discovery is one of the core activities em many time series mining algorithms. Patterns can be used for clustering or rule mining in applications ranging from bioinformatics to network intrusion detection. Several solutions to pattern discovery have been proposed so far. However, all solutions assume centralized dataset. With increasingly development of network technology distributed data analysis has become popular, raising issues like scalability and cost minimization. Additionally, some scenarios such as mining distributed medical or financial data involves the question of how to preserve data privacy. In this paper we present a density based pattern discovery algorithm for time series, which is shown to be scalable, communication efficient and privacy-preserving. Our results are based on theoretical analysis and experiments as well.

Keywords: privacy-preservation, time series, pattern discovery

1 Introduction

Time series data is one of the most common type of data generated in real world scenarios. Scientific experiments, space telemetry, medical care, financial and business applications generate an enormous amount of time series data. Several time series mining algorithms have been proposed in the data mining literature, such as time series classification or rule discovery [5, 6, 16, 11, 9, 7]. However there are still a number of challenges to be addressed, such as data distribution and data privacy.

In real world scenarios datasets are normally split among different sites. To work in such distributed scenarios any mining algorithm has to deal with questions related to *scalability* (both in dataset size and number of participants), minimization of *communications* costs and data *quality*. Classic approach to mine distributed data is to gather all data in a single central repository and apply a

mining algorithm to find patterns in the centralized data [18]. Although appealing for most cenarios, this approach may be unfeasible when network bandwidth is restricted or there is a cost associated to newtork usage. Moreover, comercial or scientific databases are normally huge, being measured in Terabytes or even larger. Therefore, centralization in such cases is not an option. Distributed data mining addresses these questions providing a number of techniques based on ideas such as meta learning [18], distributed function learning [12], to name a few.

Another crucial question in distributed scenarios is how to preserve privacy of sensitive data. Since distributed data normally is owned by different instutions, or corporations, data may represent a invaluable asset and the parties may not want to join a mining group if there is no guarantees that privacy is being preserved. In fact, many countries have privacy regulations which controls which data can be disclosed beyond its original purposes, e.g. medical information, or credit card transactions. However, these data present a huge potential for knowledge discovery and should not be simply set aside. Privacy-preserving data mining is a research field that address the general problem of providing good data mining results without violating data privacy in the process [22, 1, 19, 10]. Many solutions to specific data mining tasks have been proposed, such as association rules [4, 20], classification [8, 3, 17], and data clustering (*clustering*) [21], among others.

In this paper we investigate privacy preserving time series mining. We focus on unknown pattern discovery, since this is one important step in many time series mining algorithms [13, 15]. Frequent patterns can be used for clustering or rule mining in applications ranging from bioinformatics to network intrusion detection. Our approach is based on the assumption that time series patterns can be represented as multidimensional points in a $\mathbb{R}^n$ data space. Therefore, we can reduce the problem of searching for frequent patterns, to the search for densely populated regions in this new data space. Our results shows that this approach is time and space efficient and misses no positive. Moreover, this approach allow us to cope with privacy requirements since no patterns are transmited among the parties.

Assumptions. We assume that data is horizontally distributed among a set of parties. Each party holds a mining agent and a data agent which are responsible for taking part in a mining session with agents from other parties. Notice that only local data agents are granted access to local datasets. Datasets are sensitive and should not be disclosed to other parties. However, each party may set a local privacy threshold denoting the minimum amount of privacy it requires to join in a specific mining session. Agents are organized in a pure peer to peer network. We also assume that agents are semi-honest, meaning that they follow the protocols, but are curious enough to try to discover any sensitive data from other parties whenever possible.

Contributions. (i) We propose DPD-TS, an algorithm based on the idea that frequent patterns discovery can be reduced to finding density regions in an ap-

propriate data space; (ii) we show how privacy is given by data transformation approach; and (iii) We propose a privacy metric which is suitable for time series data.

Outline. The remainder of this paper is organized as follows. Section 2 presents background on our notation and definitions. DPD-FS algorithm is presented in section **??** and its privacy properties are discussed in section 3. Experiments are presented in section 4. We discuss related work and conclusion in sections 5 and 6, respectively.

TODO: Explain how data is partitioned

2 Definitions and Background

TODO: what are the roles of the participants

TODO: what are the privacy issues here

TODO: what is the attacker model (curious or malicious)?

In this section we present background on our notation and formulate the time series pattern discovery problem more formally.

Let $f : \mathbb{N} \to \mathbb{R}$ be a function from time stamps to reals. We define a time series $T = \{x_t \mid x_t = f(t)\}$, with $1 \leq t < m$, as ordered set of reals x_t coming from some measurement function f. The ordering is with respect to the time stamp t. So, given a total order $\prec$ we have $x_t \prec x_{t+1}$ for all $t, 1 \leq t < m$. Further, we denote the length of the time series T by $|T|$.

A subsequence of T is denoted $\langle x_t, \ldots, x_{t+v} \rangle$, for given integers $1 \leq t < m$ and $1 \leq v < t - m$. A frequent pattern in time series is a subsequence of the time series that reoccurs at different points of T.

Definition 1 (Match). *Let T be a time series with size m. Given a subsequence $Q = \langle x_q, \ldots, x_{q+v} \rangle$ with $x_q \in \mathbb{R}, 1 \leq v < q - m$ be a query subsequence, a* match *is a subsequence $S = \langle x_s \ldots, x_{s+v} \rangle$ of T which satisfies $d(Q, S) \leq \delta$, for a given distance function d and threshold δ.*

The definition of match helps us to capture the notion of reoccurring subsequences.

Definition 2 (Reoccurring subsequence). *Given a time series T, a subsequence Q is reoccurring if*

(i) there is more than one match for this query in T; and
(ii) for every pair of matching subsequences R and S from T, it holds that R and S do not overlap.

The non-overlapping requirement intends to avoid *trivial matches*. For example, if we find a match sequence at position t=1 to t=10, all subsequent matches found which includes any point from up to t=10 will be considered trivial. Trivial matches may falsely indicate a frequent subsequence if the sequence has a high degree of self-similarity. This constraint helps us to eliminate these false positives.

Definition 3 (k-frequent subsequence). *Given a real-valued time series T, a reoccurring subsequence Q is said to be k-frequent if Q is in the top-k list of reoccurring subsequences from T.*

Following this definition a *1-frequent* subsequence is the most frequent one. Similarly, a *3-frequent* subsequence is one of the three most frequent, not necessarily the most frequent one.

Throughout this paper we use the term *pattern* as a synonym to *k-frequent subsequence*. In the figure 1 we show three matches for a given pattern. Each match can be seen as an instance of a prototypical pattern summarizing the subsequences matched.

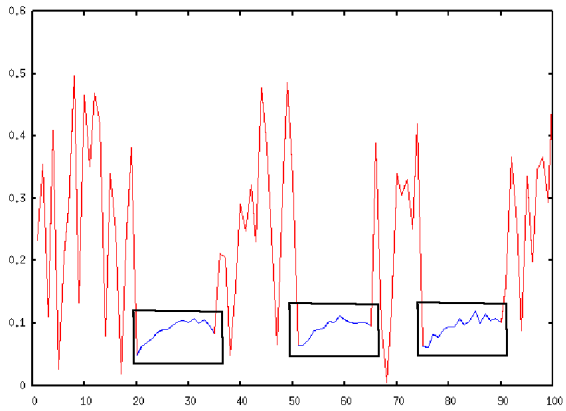


Fig. 1. Time series with three occurrences of the same *pattern*.

Having introduced the basic notation and definition we move forward to define the privacy preserving pattern discovery problem.

Problem 1 (PP-DPDTS). Given a real-valued time series T, an integer k, and a set of sites $\mathcal{L} = \{L_i\}_{1 \leq i \leq P}$, each of them with a local time series T_i, find the set $\mathcal{P}$ of the k-most frequent patterns occurring in $T = \bigcup_{i=1}^{P} T_i$, such that:

1. The total communication cost is minimized;
2. The result using the distributed data T_i is the same if the algorithm runs using $T = \bigcup_{i=1}^{P} T_i$;
3. No party learns about specific values stored in other parties, up to a user defined privacy threshold.

An important assumption in this work is that the time series data collected at different sites refers to the same variable and has the same time spacing.

For example, all parties might work with time series about daily sales volumes about the same product. In this case the common variable is sales volume and the time spacing is 24h.

3 Privacy Metric for Time Series Mining

Privacy can be informally defined as the right to keep some information hidden from other people. It can be further refined to privacy of datasets (*data privacy*) and privacy of information about the data collector (*organization privacy*). Privacy-preserving data mining address the general problem of providing useful mining results while providing data privacy. Many solutions have been proposed to specific distributed data mining tasks, for instance association rules [4, 20], classification [8, 3, 17], or data clustering [21].

Time dimension in data introduces several aspects which may describe a process evolution through a period of time [23]. *Amplitude* is the value of time series at a particular time point and can be compared to the data value in non-time series data. Amplitude in time series must be protected if it represents a measurement of a sensitive variable, such as sales volume, purchase history, etc. Further, *peaks* are extreme values assumed in the series and may indicate a sudden change of normal behavior, money flow problems, etc. *Predictions*, or trends, is another aspect that may be considered sensitive since it allows the attacker to anticipate a given value in the future, with a given statistical confidence level. Predictions depends on the accuracy of the prediction model that is available to the attacker. If predictions are too accurate, it can represent a privacy breach.

The basic piece of information to all time series aspects is the amplitude, from which all other aspects can be derived (peaks, trends and predictions). If a particular data point is not known, or only known to be in a given interval, all other aspects will have less accuracy than if the point was known with exact precision. Therefore, we focus on the amplitude, i.e. the raw value at a particular data point.

As discussed in Sections **??** and **??**, several different metrics have been proposed in the privacy-preserving data mining literature. Since none of the proposed metrics are developed for the time series the question is whether we can pick one of the proposed metrics, or if we need a new metric. In some sense, each metric focuses on a specific aspect of privacy. Information theoretical approaches focus on the aspect of how much confident the attacker can be, given the entropy of the variable, using the privacy as the reciprocal of entropy. In more database-oriented metrics, like the ideas used in k-anonymization, the focus in on how many points can be identified. In time series, there is no identification issue, therefore we need to provide some probability bound on the reconstruction quality. Therefore, we chose to work with an information theoretical based metric.

We follow an information-theoretical model of privacy. It means that our measure of privacy is given by the uncertainty an attacker has on where a given original point is located in the amplitude. Other privacy models involve a proof of infeasibility of attacks or showing the computational power necessary to learn something from the available data. We see these approaches as complementary efforts.

Our model is an extension of the entropy-based metric introduced by Agrawal and Aggarwal [2]. We include an additional element in our model to capture the fact that parties may collude. The following definition gives the details.

Definition 4. *The privacy level of a given point x_t in a time series T, modeled by a random variable X, in presence of c colluders, is given by*

$$PR(x_t, c) = 2^{h(X)} \tag{1}$$

where $h(X)$ is the differential entropy of X, i.e. $h(X) = -\int f(x) log_2(f(x))dx$, and $f(x)$ is the probability density function of X modeled in a scenario where c colluders are active.

We compute the privacy of a given point in the time series by modeling it from the point of view of the attacker. Therefore, we model an arbitrary point x_t of the original time series T as a random variable X, what allow us to use the privacy definition above. The probability density function $f(x)$ may be used to model the knowledge the attacker has about the point x_t. If the attacker has no knowledge, we use the uniform distribution, what give us $PR(X) = 2^{log_2(a)} = a$, the size of the interval from where X is drawn. On the other hand, if we know a better model for x_t we can incorporate it in the privacy level naturally. This would be the case, when the time series has good predictability and we compute the privacy with a correct model.

Now, we extend the basic metric to a complete time series.

Definition 5. *The privacy level of a time series T in presence of c colluders is*

$$PR(T, c) = \min\{PR(x_t, c) \mid t = 0, 1, 2, 3, \ldots, |T|\} \tag{2}$$

where $|T|$ denotes the size of T.

Finally, using the previous metric, we can measure the privacy level of a time series mining algorithm.

Definition 6. *Given a time series mining algorithm $\mathcal{A}$, its privacy level given c colluders is:*

$$PR_{\mathcal{A}}(c) = \min\{PR(O, c) \mid O \text{ is an output of } \mathcal{A}\} \tag{3}$$

The privacy of an algorithm is the weakest level measured for any of its outputs, at any stage of the mining process. This includes any model, mining results, intermediate results, etc.

The question now is how to control the privacy level used the above definition. This control should be provided by the algorithm, allowing the local site to decide how much privacy it want, before engaging in a mining session with other sites.

3.1 The DPD-TS Algorithm

In this section we present DPD-TSalgorithm, which is a first step towards a solution to PP-DPDTS problem stated in Section **??**. DPD-TSfollows the general three-step scheme discussed before (cf. Sec. **??**), but differs from other approaches in the third step, as discussed below.

DPD-TSworks as follows. Given a sequence S, it computes the density of subsequences of S and output a list with the top k most dense subsequences. Our approach exploits the fact that a density estimate can be used to find overcrowded regions in a hyperspace. Our main hypothesis is that if we take subsequences of time series and represent it as points in a multidimensional space, we can reduce the search for frequent subsequences of fixed size, to the search for dense regions in a multidimensional space. Consequently, the search for k-most frequent patterns of size w reduces to the search for k-most dense regions on a $\mathbb{R}^w$ space.

DPD-TScomputes the density over a distributed dataset. At each peer a local density estimate is computed and together the peers sum up local densities to produce a global density estimate. With the global density estimate, each local mining agent can perform the discovery step to spot frequent subsequences on the local dataset.

Algorithm 1 DPD-TS: initiator

Input: k, T_i, n, w, Σ, $\mathcal{L}$, r;
Output: $\mathcal{P}$;
Method:
At the initiator party L_1 do:
1: negotiate(k,n,w, Σ,r);
2: $T_1' \leftarrow$ reduceDim(T_1, n, w); // Using Eq. (4)
3: $T_1'' \leftarrow$ discret(T_1', w, Σ); //Local dim. reduction and discretization
4: $\hat{\varphi}_1 \leftarrow$estimateDensity($T_1'', w, r$); //Local density estimation
5: **send** $\hat{\varphi}_1$ **to** L_2; //Cooperative sum
6: **receive** $\hat{\varphi}_p$ **from** L_p
7: $\hat{\varphi} \leftarrow \hat{\varphi}_p$ // Global density estimate
8: $\mathcal{P} \leftarrow$getCenters($\hat{\varphi}$,k, r); //Finding globally Frequent patterns
9: **send** $\mathcal{P}$ **to** all agent $L_j \in \mathcal{L}$;

The pseudocode for DPD-TSis outlined in Algorithms 1 (initiator) and 2 (arbitrary party). Details are discussed in the following.

Detailed Description of DPD-TS DPD-TScomputes a set of k-frequent patterns occurring in the union of local time series T_i owned by peers in the mining group $\mathcal{L}$. Peers are assumed to form a peer-to-peer network. DPD-TSneeds the following parameters: T_i is the local dataset, n is the size used to generate subsequences, w is the number of symbols per string, i.e. the string size, Σ is the

Algorithm 2 DPD-TS: arbitrary party

Input: k, T_i, n, w, Σ, $\mathcal{L}$, r;
Output: $\mathcal{P}$;
At an arbitrary party j do:
1: negotiate(k,n,w, Σ,r);
2: $T'_j \leftarrow$ reduceDim(T_j, n, w); // Using Eq. (4)
3: $T''_j \leftarrow$ discret(T'_j, w, Σ); //Local dim. reduction and discretization
4: $\hat{\varphi}_j \leftarrow$estimateDensity($T''_j, w, r$); //Local density estimation
5: **receive** $\hat{\varphi}_{j-1}$ **from** L_{j-1};
6: $\hat{\varphi}_j \leftarrow \hat{\varphi}_{j-1} + \hat{\varphi}_j$; // Updating with local density
7: **send** $\hat{\varphi}_j$ **to** $L_{(j \mod p)+1}$; // Send to next peer and the last one sends to initiator
8: **receive** $\mathcal{P}$ **from** L_1;

alphabet used to generate strings, and $\mathcal{L}$ is the set of peers forming the mining group. The parameter r define the radius of the density ball to be used in the second step. As output, DPDTS returns a set $\mathcal{P}$ with the *globally* k-most frequent patterns.

Negotiation. The first step in DPD-TSis a negotiation on the parameters values. In this phase, a given peer may decide no to engage in the mining session if the negotiation is not in accordance with its own local policy. The other peers may decide to continue with the mining session even if some of the original members leave the group. All further steps in the algorithm assume that the negotiation was successful.

Dimension Reduction. Function reduceDim() splits the original time series T_i in various subsequences S of size n. For each non-overlaping subsequence $S \sqsubseteq T_i$ a reduced subsequence $\overline{S}$ is computed. Each point of $\overline{S}$ is the average of a small subsequence of S of size $\frac{n}{w}$. This operation (proposed elsewhere [14]) is known as piecewise aggregate approximation (PAA):

$$\overline{s}_j = \frac{w}{n}\left(\sum_{k=\frac{n}{w}(j-1)+1}^{\frac{n}{w}j} s_k\right) \tag{4}$$

where s_k is a single point in the subsequence S. This transformation reduces the dimensionality of a given subsequence S from n to w, where n is the size of S and w is the size of the resulting subsequence $\overline{S}$. The resulting time series T' is a concatenation of all reduced subsequences $\overline{S}$ computed from from T. Figures 2 and 3 illustrate the dimension reduction of a single subsequence.

The following example illustrates the dimension reduction process.

Example 1. Let $T = \langle 1.0, 2.0, 3.0, 5.0, 3.0, 4.0, 8.0, 9.0, 5.0, 6.0, 7.0, 2.0\rangle$, subsequence size $n = 6$, and word size $w = 3$. With these parameters T can be split in two subsequences $S_1 = \langle 1, 2, 3, 5, 3, 4\rangle$ and $S_2 = \langle 8, 9, 5, 6, 7, 2\rangle$. Since we want to reduce the size from 6 to 3 we compute for each subsequence, S_1 and S_2, three

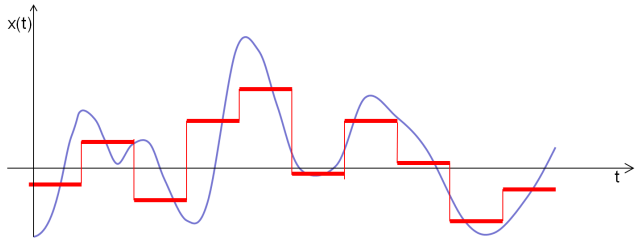


Fig. 2. A subsequence S of T is split into n/w subsequences of size w and the average value is computed for each one of these subsequences of S.

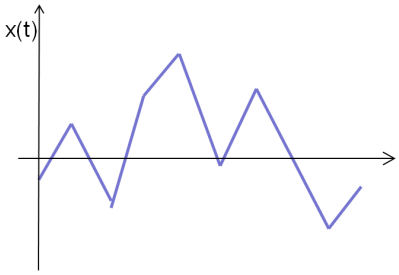


Fig. 3. Reduced subsequence is composed of n/w average values.

averages using two values at time. Threfore we have the reduced subsequences $\overline{S}_1 = \langle 1.5, 4.0, 3.5 \rangle$ and $\overline{S}_2 = \langle 8.5, 5.5, 4.5 \rangle$. The resulting reduced time series is a concatenation of all reduced subsequences,i.e. $T' = \langle 1.5, 4.0, 3.5, 8.5, 5.5, 4.5 \rangle$.

Discretization. Function discret() produces a discretized version of T', which is a sequence os symbols in a given alphabet. We refer to the discretized version as T''. For each element x of T', the corresponding string T'' will have a symbol $\sigma_a \in \Sigma$. The substution procedure is accomplished by choosing break points $\{\beta_a\}$ in the values dimension of a given time series T, such that $|\{\beta_a\}| = |\Sigma| - 1$, and such that each occurrence of a given value x of T'' has the same probability [15], assuming they are normally distributed. For example, considering a 4-symbol alphabet, we need 3 break points, where each region will have probability 0.25 of appearing in the time series (cf. Fig. 4).

Then, the substitution rule is applied:

$$x_j = \begin{cases} \sigma_1 & \text{if } \overline{s_j} \leq \beta_1 \\ \sigma_a & \text{if } \beta_{a-1} < \overline{s_j} \leq \beta_a, \text{with } 1 < a < m \\ \sigma_m & \text{otherwise.} \end{cases} \quad (5)$$

where $1 < m \leq |\Sigma|$ and $x_j \in T''$.

Example 2. Figure 5 illustrates the discretization process. Each value in the reduced time series is mapped to a symbol in the alphabet. Considering the

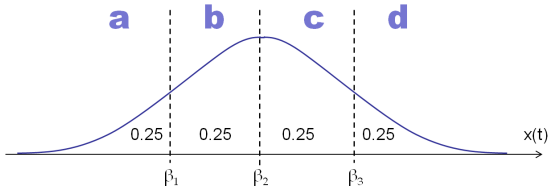


Fig. 4. Break points are defined in the value dimension of time series, i.e. x(t), such that each symbol of the alphabet is mapped to a region with same probability, considering that the values of the time series follows a normal distribution [15].

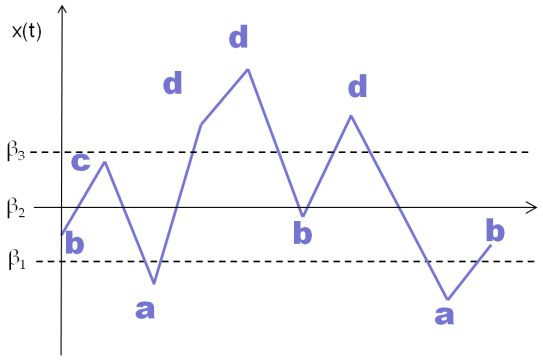


Fig. 5. Each average value is substituted by a symbol, according to the regions defined by break points (Adapted from [15]). In this case a 4-symbol alphabet is used to discretize a time series, generating the word "bcaddbdab".

alphabet $\Sigma = \{a, b, c, d\}$ and the reduced time series in Fig. 3, the discretized sequence of symbols is $T'' =$ "*bcaddbdab*". Notice that breakpoints are chosen in the value dimension of time series.

Estimating Density of Strings. Function estimateDensity() computes the density estimates of strings S'' from discretized time series T''_j. An important requirement is that the density estimate function $\hat{\varphi}$ builds a non-negative monotonic function over $\mathbb{R}$ and that the local maxima represent the most dense regions in the feature space. In practice, the function does not even need to be a true estimation; an approximation will do it.

A general approach to compute data density function is kernel-based density estimation (already discussed in previous chapter, Sec. **??**). For a given kernel function K such that $\int_{\infty}^{+\infty} K(x)dx = 1$, an estimate of the density, for a specific dataset D, is given by:

$$\hat{\varphi}[D, r](x) = \frac{1}{Nh} \sum_{x_i \in Neigh(x,r)} K\left(\frac{d(x, x_i)}{h}\right) \tag{6}$$

where N is the total number of points, d is a distance function, h is a bandwidth parameter and $Neigh(x, r)$ is the neighborhood of point x in a given dataset D, considering only neighbors inside a ball radius r computed with distance d.

We use the triangle kernel $K(u) = (1-|u|)I\{|u| \leq 1\}$, where I is the indicator function[3]. We choose this kernel for its simplicity, but any other kernel can be used instead. We set the kernel bandwidth parameter equals to the neighborhood radius, i.e. $h = r$. We use the Euclidean distance, denoted d, and assume that the alphabet has a total order. An arbitrary point is a string S'' from T''_j, i.e. $S'' \sqsubseteq T''_j$. For a given discretized time series T'', there are $|T''|/w$ strings to consider. In the Kernel expression we use the substitution $u = d(S'', S''_i)/r$. Putting all together, we have:

$$\hat{\varphi}(S'') = \frac{w}{|T''|r} \sum_{S''_i \in Neigh(\sigma)} \left(1 - \left(\frac{d(S'', S''_i)}{r}\right)\right) I\{(d(S'', S''_i)/r) \leq 1\} \quad (7)$$

Computing Global Density. The mining group cooperatively compute the set of globally k-most frequent patterns by summing up all local tables. Initially, peer L_1 sends its local density to L_2. After that, each peer L_j receives partial density estimate $\hat{\varphi}_i$ from its neighbor L_i, with $i = j - 1$ and $j > 1$. L_j adds its own local density to partial global density estimate, i.e. $\hat{\varphi}_j = \hat{\varphi}_i + \hat{\varphi}_j$. Then, site L_j sends partial global density $\hat{\varphi}_j$ to the next neighbor L_{j+1} in the mining group. This protocol continues until the partial sum is sent to L_1, which broadcasts the global density estimate $\hat{\varphi}$ to all members of the mining group.

Finding Patterns by Locating Centers. To find the patterns, each peer uses the getCenters(), which works as follows. Choose a set of strings, each of them representing the points that are local maxima in the global density estimate, i.e. centers of the top k most dense regions, and call it $\mathcal{P}$. The regions are constrained to define a ball radius r. More formally, for a given density estimate $\hat{\varphi}$, distance function d, and a string S, we have:

$$\mathcal{P} = \{S \mid \forall\ R \in \Sigma^w :\ (d(S, R) \leq r \rightarrow \hat{\varphi}(S) > \hat{\varphi}(R))\} \quad (8)$$

In general, we can say that the higher populated the neighborhood, the higher the estimated density of a given point. Of course, by just looking at the density, we would identify several candidate patterns, which are even not present on the string T''. The number of these candidate pattern is controlled by the parameter r. It can be thought as kind of error rate. If we set a high r, e.g. $r = 2$, for a given pattern, we expect to find all variations of it in a ball radius r, even if it is not present in the current string T''. Setting $r < 1$ admits no error and only the patterns explicitly found in T'' are reported.

Lemma 1. *Let $S_1, S_2 \in \Sigma^w$ be two sequences of size w. Let $\hat{\varphi}$ be a density estimate computed using Eq. 7 and denote $Neigh(S)$ the neighborhood of a string*

[3] The indicator function, also known as characteristic function, returns 1 if the expression in curly brackets holds and 0 otherwise.

S, i.e. $Neigh(S) \subset \Sigma^w$. *Then*

$$\hat{\varphi}(S_1) > \hat{\varphi}(S_2) \leftrightarrow |Neigh(S_1)| > |Neigh(S_2)|$$

Proof. ($\rightarrow$) Assume that $\hat{\varphi}(S_1) > \hat{\varphi}(S_2)$. By Eq. (7) it means that

$$\hat{\varphi}(S_1) = \sum_{S_i \in Neigh(S_1)} \left(1 - \left(\frac{d(S, S_i)}{r}\right)\right) I\left(\frac{d(S_1, S_i)}{r} \leq 1\right) >$$
$$\hat{\varphi}(S_2) = \sum_{S_j \in Neigh(S_2)} \left(1 - \left(\frac{d(S_2, S_j)}{r}\right)\right) I\left(\frac{d(S_2, S_j)}{r} \leq 1\right)$$

In general, the inequality only holds if $Neigh(S_1) > Neigh(S_2)$, because the sum of indicator function and the distance function are proportional to the size of the neighborhood. Therefore, it holds that $\hat{\varphi}(S_1) > \hat{\varphi}(S_2) \rightarrow |Neigh(S_1)| > |Neigh(S_2)|$.

($\leftarrow$) By similar argumentation, we verify that the assertion holds on the other direction, i.e. $|Neigh(S_1)| > |Neigh(S_2)| \rightarrow \hat{\varphi}(S_1) > \hat{\varphi}(S_2)$.

By lemma 1, the local maxima in the pattern space correspond to strings that reoccur more frequently than others do.

Performance Analysis of DPD-TS

Time. The time complexity of DPDTS at a local peer is $O(|T_i|)$, where $|T_i|$ means the size of the time series at peer L_i. There are $\lfloor \frac{|T_i|}{n} \rfloor$ subsequences in T_i. For each subsequence w arithmetical means are computed summing $\lfloor \frac{n}{w} \rfloor$ points for each mean, i.e. n steps. Additionally each arithmetical mean is substituted by a symbol, which takes w steps. The overall time cost is $\frac{|T_i|}{n}(n+w) = T_i(\frac{w}{n}+1)$ steps. Note that normally $w < n \ll |T_i|$. The discovery step, which is the search for the k-most dense regions, is independent of the size of T_i and is $O(|\Sigma|^w)$.

Communication. Each peer sends 1 message to a neighbor peer and receives 1 message from another neighbor. There are only 2 rounds of messages, one of which informs the mining results. Each message has size $O(|\Sigma|^w)$, for given global w and Σ.

Correctness. One important property of this algorithm is that it produces no false positives, although it may miss some less frequent patterns. It is not a problem because we are trying to find just the most frequent patterns.

Theorem 1. *Algorithm* DPD-TS*produces no false positives.*

Proof. This is a result from the following facts. First, patterns correspond to local maxima in the density estimate and maxima correspond only to highly populated regions as a consequence of Lemma 1. Therefore, a frequent pattern represents a frequently reoccurring subsequence (within a given range of dissimilarity). Second, given two different local maxima the one with higher density represent the pattern with larger neighborhood and consequently higher frequency rate. Therefore, the ordering of the k-most dense regions corresponds to the ordering of the k-most frequent subsequences.

3.2 Privacy Analysis of DPD-TS

We analyzed the privacy of DPD-TSin various attacks scenarios, as discussed in the following.

Single Initiator Attack. Recall that the initiatior peer knows all parameters, the set of global patterns,its own local density estimates and the global density estimates. It does not know any local data from other peers, since, by construction, DPD-TS scheme does not require the parties to transmit raw data. The only information transmitted by the peers during the mining session is the encrypted partial density estimate. Therefore, a malicious initiator will not receive any piece of original sensitive data from other peers.

The initiator is the only peer that has access to the global density estimate. However, the density estimate has no information on the order of occurrence of each time series subsequence, which is necessary to reconstruct the original time series. Consequently, an attacker can only try to reconstruct the most provable values for x_t, for a given time stamp t.

The initiator has the set of global patterns $\mathcal{P}$ from which it can try to infer the original values T. Since the discretization step is the main responsible for the privacy, we intuitively know that the more symbols in the alphabet Σ, the less privacy we get, because the discretized version tends to get the "shape" of the original time series data. The next result shows how the size of Σ influences the privacy of a single point in T.

Theorem 2. *Let Σ be an alphabet of symbols used by the DPD-TS protocol. Let $\{\beta_j \in \mathbb{R}\}_{j=1}^{|\Sigma|+1}$ be a set of breakpoints which divides the normal curve in $|\ \Sigma\ |$ equiprobable regions. Let T be a time series and $T'' \in \Sigma^w$ be its transform according to the discretization step of Algorithms 1 and 2. For a given point x_t if its transformed counterpart x''_u is known then its privacy level is given by:*

$$PR(x_t) = |\ \beta_{j+1} - \beta_j\ | \tag{9}$$

Proof. This is a consequence of the discretization step. Let $\sigma_j \in \Sigma$ be the symbol at point x''_u. Since we know that the symbol σ_j comes from the substitution rule (cf. Sec. **??**), we know that each point of the subsequence $\langle x_t, \ldots, x_{t+n} \rangle$ lies in the interval (β_j, β_{j+1}). In the absence of further information, the only suitable option is to model x_t as a random variable uniformly distributed in the given interval, i.e. $x_t \sim U(\beta_j, \beta_{j+1})$. Now, using the Equation (1) we have

$$\begin{aligned}\mathbf{PR}(x_t) = 2^{h(x_t)} &= 2^{\int_{\beta_j}^{\beta_{j+1}} p(x) log_2 p(x) dx} \\ &= 2^{log_2(\beta_{j+1} - \beta_j)} \\ &= \mid \beta_{j+1} - \beta_j \mid\end{aligned}$$

Using the above theorem, peers in the mining group can define a minimum amount of privacy they require in order to join the mining session. This is done by setting the maximal size of the alphabet Σ and, consequently, the size of the intervals defined by the breakpoints $\{\beta_j\}$. If the minimal privacy requirement is not met, the peer does not join the mining group. Recall that the patterns in $\mathcal{P}$ are sequences of symbols and not subsequences of the true data T.

Single Arbitrary Party Attack. Each party in a DPD-TSmining session knows all parameters values agreed in the negotiation step, the set of global patterns receive from the initiator, and its own local density estimates. Without collusion an arbitrary party has no access to density estimates of other peers and, as a consequence, may try to infer only the true values x_t associated to symbols in the global patterns. As in the previous attack, the privacy level is provided by the amount of discretization applied to original time series T. Therefore, the privacy level of each point in the time series is the same as in the previous attack.

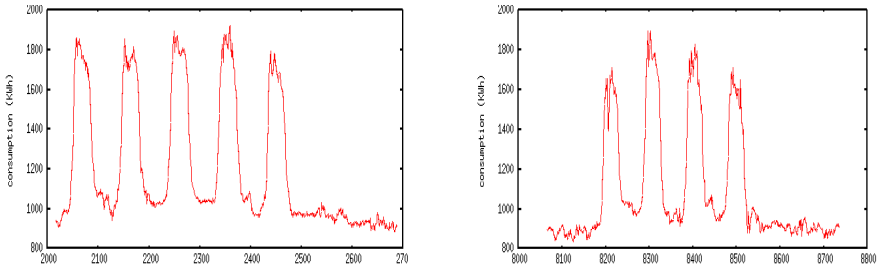
Collusion Attack (including initiator). Any collusion group that includes the initiator peer has information about the parameter values, local density estimates of attacker and the global density estimates from initiator. Thus, any collusion group that includes the initiator has enough information to isolate the partial density estimates of a group of victms. In the extreme case, when $p - 1$ peers colludes against one single victim peer, the attackers can discover the victim density estimate. Nevertheless, the density estimate can only be used to find the most frequent patterns at a given peer, or group of peers. As discussed in the fist attack scenario, the density estimates gives no information on the order of occurrence of each pattern. Consequenctly, the attackers cannot reconstruct the entire local times series of any victim. Again, the privacy of each point in the time series is preserved as in the single initiator attack.

Collusion Attack (without initiator). Any collution attack without the initatior has no access to the global density estimates. Thus, a collusion without the initatior brings no further information to the collusion group beyond the public information they already have, such as the parameter values and the local density estimates. The privacy level of each point is preserved to the same level as in previous attacks.

4 Experiments

Datasets. For the experiments reported here, we used the *power data* records the electricity consumption from Netherlands Energy Research Foundation (ECN)

for one year, recorded every 15 minutes. There are 35 040 data points corresponding to the year of 1997. Figure **??** shows an excerpt of the power data. This dataset has a pattern structure that can be observed visually.



(a) (b)

Fig. 6. (a) An instance of a *normal week*, the most frequent subsequence in the power data, showing high consumption on workdays and low consumption at weekends. (b) An instance of a *week with holiday* pattern in the power data. In this example, the Monday was a holiday.

Methodology. We set the parameter as follows. Subsequence size $n = 96$, which corresponds to one day (with one measurement every 15 minutes). We choose pattern size $w = 7$, since it represents a week. The alphabet Σ was set to { a, b, c, d}. Symbol 'a' represents lowest values and 'd' represents highest value of consumption. The radius was set to $r = 1$. Larger values of r produces larger neighborhoods. The density landscape becomes smoother what may reduce the number of local maxima and consequently the number of patterns. Choosing smaller values of r produces a more spiky density with more local maxima and more patterns. Therefore, r help us to control the number of patterns.

The *information loss* of the DPD-HE is was defined as

$$IL_{MAE}(T, T') = \frac{\sum_{i=1}^{n} \mid x_i - x'_{\lfloor \frac{w}{n}(i-1) \rfloor + 1} \mid}{n} \tag{10}$$

where T is the original time series, T' is the result of substituting the symbols in the resulting string by its corresponding breakpoint (see Algorithm DPD-TSin Sec. **??**), and n is the size of the PAA sequence and w is the size of the resulting string. The involved index expression assures that each point in the original time series T is compared with the correct element in T', which is smaller than T.

Results. With the aforementioned parameters values we found 2 frequent patterns. The first pattern is "cccccaa" which corresponds to a normal week. The second pattern is "acccaa" which correspond to weeks having a holiday on the first day. Figure 6 shows one instance of each pattern.

Results with different alphabet sizes, with all other parameters set as above, found an increasing number of patterns. This is mainly because larger alphabets produce more accurate discretization, what allows for a more detailed differentiation among the patterns. These additional patterns basically presented refinements of the more general pattern "five day high + two days low".

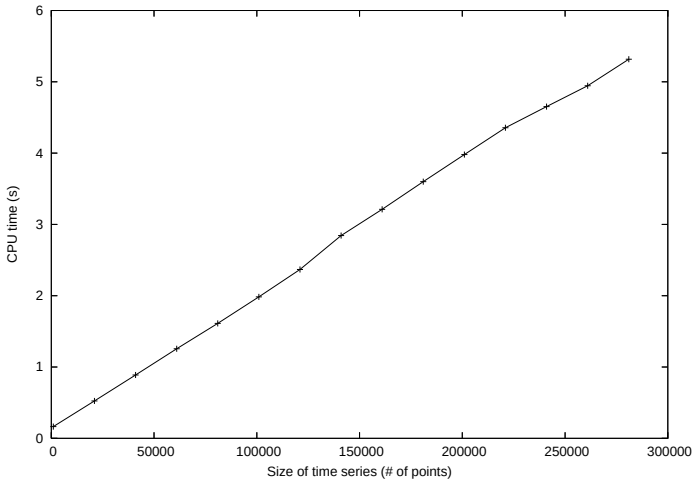


Fig. 7. Time performance (in seconds) with increasing size of time series.

Results of performance are shown in Figures 7 and 8. We performed the experiments with the same parameters values as in the previous experiments. For the performance tests, we created a synthetic time series with 300 000 points, by cloning power data 10 times. As shown in the Figure 7, the CPU time increases linearly with the size of the time series.

The results of privacy vs. size of alphabet are presented in figure 8. To measure the privacy, we used the interval size corresponding to each alphabet symbol in the discretization step. In the initial case, with just one symbol, we used the size of interval from the minimum to the maximum value observed in the time series, which is 1 056 kWh. Assuming that max and min values are public, the attacker can compute the entropy of a random variable X over this interval, and consequently the privacy level $2^{h(X)}$. That is the privacy we get when the sequence consists of symbols from a singleton alphabet. In the figure, we see the decrease of privacy by using more symbols to discretize the time series. With 10 symbols we get a privacy level of 300 kWh, which means that an attacker cannot reconstruct a data point within an interval smaller than 300 kWh. It is up to the user, however, to decide whether or not a given privacy level is enough.

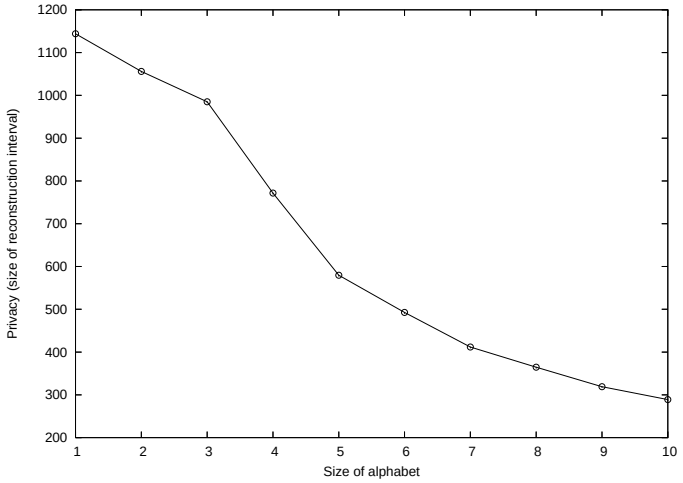


Fig. 8. Privacy level (size of reconstruction interval) with increasing size of alphabet.

5 Related Work

6 Conclusion

References

1. Aggarwal, C.C., Yu, P.S. (eds.): Privacy-Preserving Data Mining: Models and Algorithms, Advances in Database Systems, vol. 34. Springer-Verlag (2008)
2. Agrawal, D., Aggarwal, C.C.: On the design and quantification of privacy preserving data mining algorithms. In: 20th ACM PODS. pp. 247–255. Santa Barbara, Califonia (May 2001), `citeseer.nj.nec.com/agrawal01design.html`
3. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proc. of the ACM SIGMOD Conference on Management of Data. pp. 439–450. ACM Press (May 2000), `citeseer.nj.nec.com/agrawal00privacypreserving.html`
4. Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, V.: Disclosure limitation of sensitive rules. In: Proceedings of 1999 IEEE Knowledge and Data Engineering Exchange Workshop (KDEX'99). pp. 45–52. Chicago,IL (November 1999), `citeseer.nj.nec.com/atallah99disclosure.html`
5. Caraca-Valente, J.P., Lopez-Chavarrias, I.: Discovering similar patterns in time series. In: ACM KDD ' 00. pp. 497–505. ACM Press, New York, NY, USA (2000)
6. Davies, P.L., Fried, R., Gather, U.: Robust signal extraction for on-line monitoring data. Journal of Statistical Planning and Inference 122, 65–78 (2004)
7. Elfeky, M., Aref, W., Elmagarmid, A.K.: Using convolution to mine obscure periodic patterns in one pass. In: 9th EDBT. LNCS, vol. 2992, pp. 605–620. Springer, Heraklion, Crete, Greece (March 14-18 2004)
8. Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J.: Privacy preserving mining of association rules. In: Proceedings of 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (KDD). Edomonton, Alberta, Canada (2002), `citeseer.nj.nec.com/article/evfimievski02privacy.html`

9. Geurts, P.: Pattern extraction for time series classification. LNCS 2168, 115–127 (2001), citeseer.ist.psu.edu/geurts01pattern.html
10. Jones, C., Hall, J., Hale, J.: Secure distributed database mining: Principle of design. In: Kargupta, H., Chan, P. (eds.) Advances in Distributed and Parallel Knowledge Discovery, pp. 277–294. AAAI Press / MIT Press, Menlo Park, CA / Cambridge, MA (2000), chap. 10, part III
11. Kadous, M.W.: Learning comprehensible descriptions of multivariate time series. In: Proc. 16th International Conf. on Machine Learning. pp. 454–463. Morgan Kaufmann, San Francisco, CA (1999), citeseer.ist.psu.edu/kadous99learning.html
12. Kargupta, H., Park, B.H., Hershberger, D., Johnson, E.: Collective data mining: A new perspective toward distributed data mining. In: Kargupta, H., Chan, P. (eds.) Advances in Distributed and Parallel Knowledge Discovery, pp. 131–174. AAAI Press / MIT Press, Menlo Park, CA / Cambridge, MA (2000), citeseer.nj.nec.com/kargupta99collective.html, chap. 5, part II
13. Keogh, E., Lonardi, S., B.Chiu: Finding surprising patterns in a time series database in linear time and space. In: Proc. of the International Conference on Knowledge Discovery and Data Mining (KDD'02). pp. 550–556. Edmonton, Alberta, Canada (July 2002), http://www.cs.ucr.edu/~stelo/papers/
14. Keogh, E.J., Chakrabarti, K., Pazzani, M.J., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and Information Systems 3(3), 263–286 (2000), citeseer.ist.psu.edu/keogh00dimensionality.html
15. Lin, J., Keogh, E., Lonardi, S., Patel, P.: Finding motifs in time series. In: Proc. of the Second Workshop on Temporal Data Mining. Edmonton, Alberta, Canada (July 2002), citeseer.ist.psu.edu/lin02finding.html
16. Moerchen, F., Ultsch, A.: Discovering temporal knowledge in multivariate time series. In: Proc. GfKl 04. Dortmund, Germany (2004)
17. Pinkas, B.: Cryptographic techniques for privacy-preserving data mining. ACM SIGKDD Explorations Newsletter 4(2), 12–19 (2002)
18. Prodromidis, A.L., Chan, P.K.: Meta-learning in distributed data mining systems: issues and approaches. In: Kargupta, H., Chan, P. (eds.) Advances in Distributed Data Mining. AAA/MIT Press (2000)
19. Provost, F.: Distributed data mining: Scaling up and beyond, pp. 3–27. AAAI Press / MIT Press, Cambridge, MA, USA (2000), citeseer.ist.psu.edu/provost99distributed.html
20. Saygin, Y., Verykios, V.S., Elmagarmid, A.K.: Privacy preserving association rule mining. In: Reseach Issues in Data Engineering (RIDE) (2002), citeseer.nj.nec.com/saygin02privacy.html
21. da Silva, J.C., Klusch, M., Lodi, S., Moro, G.L.: Secure agent-based distributed data clustering. International Journal of Web Intelligence and Agent Systems 4(2) (2006)
22. Terrovitis, M.: Privacy preservation in the dissemination of location data. SIGKDD Explorations 13(1), 6–18 (2011)
23. Zhu, Y., Fu, Y., Fu, H.: On privacy in time series data mining. In: Proceedings of the 12th Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD'08). pp. 479–493. Springer-Verlag, Berlin, Heidelberg (2008), http://dl.acm.org/citation.cfm?id=1786574.1786619