Privacy-Preserving Pattern Mining on Online Density Estimates

Michael Geilke and Stefan Kramer

Johannes Gutenberg University Mainz
Email: {geilke,kramer}@informatik.uni-mainz.de

***Abstract*—Traditional pattern mining algorithms require access to the data, either in the form of a complete set of data, as in batch data mining, or in the form of a window of recent data, as in stream mining. In the case of stream mining, this comes with a number of disadvantages, such as the possibly unbounded growth of relevant instances, drift, possibly changing data mining tasks, and issues with privacy, to name a few. Therefore, an approach has been recently proposed that extracts patterns just from statistical information of the stream – more precisely, an online density estimate that is inferred from it. As this approach is mainly based on sampling from the density estimates, it still struggles with itemsets having a medium to low frequency. To resolve this issue, we pursue an alternative strategy in this paper and directly exploit the structure of the density estimates to extract frequent itemsets. Additionally, we address the important matter of privacy-preserving data mining by ensuring that the density estimate fulfills privacy-related properties. To show the effectiveness of the proposed methods, we provide proofs and evaluate the performance on synthetic and real-world data.**

I. Introduction

Traditional data mining algorithms operate on the data itself, which poses problems in data stream settings, where the amount of data is often too large to be kept in memory. Established solutions try to avoid memory limitations by pursuing window-based approaches, but they implicitly assume that collecting the data and performing the data mining task is either happening simultaneously or with a small temporal delay. Hence, the user has to know in advance, i.e., before collecting the data, which data mining task she wants to perform – something which is unrealistic in interactive environments where data streams may need to be analyzed after days or even weeks. For example, after performing pattern mining, the user discovers that she is only interested in patterns contained in a particular subset of the data. A window-based solution would require to initiate another pattern mining process and to wait for new data instances to complete the task. Hence, knowledge can only be extracted at the time of collecting the data.

Therefore, the MiDEO framework [1] has been recently proposed, in which knowledge is available in form of an online density estimate representing the joint density of the data stream and can be accessed using inference operations. This way, frequent itemsets can be derived without accessing the original data, while also supporting different user roles and providing a clear separation of concerns [1]. For example, a company not having any expertise in the area of data mining could use a density estimator to estimate the joint density of its data stream $\hat{f}$. Subsequently, it modifies $\hat{f}$ to meet certain privacy concerns and sends the result to a data scientist, who can use her expertise to provide an in-depth analysis.

Although the privacy aspect of this framework has been pointed out before, there is still no algorithm providing any privacy-related guarantees. There is only a certain level of privacy-preservation, because the density estimate solely contains statistical information. Another problem is the mining process itself, as the corresponding pattern mining algorithm, called POEt (Pattern mining on Online density esTimates) [1], produces itemsets by constantly transforming the estimate and by sampling attribute-value combinations. Therefore, it struggles with itemsets having a medium to low frequency, since they require a large number of samples. To address these issues, we propose a novel approach for effectively performing privacy-preserving itemset mining on data streams: (1) Unlike POEt, the presented algorithm does not pursue a sampling-based approach but exploits the structure of the estimate to construct the required itemsets. To show its effectiveness, we will prove that, under certain assumptions, it is able discover every frequent itemset. (2) We propose an algorithm to achieve t-closeness [2] for online density estimates, which protects small entity groups by bringing local distributions of sensitive variables closer to the global counterparts.

In the remainder of this paper, we formally introduce the problem (Section II), define relevant concepts (Section III), and present related work (Section IV). Subsequently, we present an algorithm to modify an online density estimator to ensure t-closeness (Section V) and an algorithm that mines itemsets from the structure of online density estimates (Section VI). The paper concludes with the evaluation and a conclusion.

II. Problem Statement

Traditionally, an itemset specifies the items that are included in a given transaction. In this paper, we consider a more general setting where an item is a variable taking several values. Let $\mathcal{X} := \{X_1, X_2, \ldots, X_m\}$ be a set of discrete variables with finite domains $values(X_i)$, i.e., X_i takes the values $v \in values(X_i)$. Further, let $S := \vec{x}_1, \vec{x}_2, \ldots$ be a stream of instances coming from the same probability distribution, where $\vec{x}_i = \{(X_j, v_j) \mid 1 \leq j \leq m\}$ is a set of variable-value combinations. An itemset is a subset $I \subseteq \vec{x}$ and its support is the relative frequency of I in S.

TABLE I: The table summarizes the notation of this paper.

Notation	Meaning
$values(X)$ with $X \in \mathcal{X}$	domain of X
$\{(X_j, v_j) \mid j \in [1;m],\ v_j \in values(X_j)\}$	an instance $\vec{x}$
$itemset \subseteq \vec{x}$	an itemset
$freq(itemset)$	support of $itemset$
θ	minimum support threshold
$\mathcal{F}_S = \{i \subseteq \vec{x} \mid \vec{x} \in S \wedge freq(i) \geq \theta\}$	frequent itemsets
$L := [l_1, \ldots, l_k]$	a list of elements
$L[i] := l_i$	i-th element of L
$L[i:j] := [l_i, \ldots, l_j]$	sublist
$node \in T$	node of tree T
$node.isRoot()$	true iff is $node$ a root
$node.isLeaf()$	true iff is $node$ a leaf
$node.children()$	all direct successors
$node.distribution()$	distribution of children
$node.classDistribution()$	distribution of class

The goal is to find $\mathcal{F}_S = \{I \subseteq \vec{x} \mid \vec{x} \in S \wedge freq(I) \geq \theta\}$, while also preserving privacy-preserving properties such as t-closeness [3]. In contrast to other approaches, which compute $\mathcal{F}(S)$ directly from S, we first compute a density estimate f from S, modify f to meet privacy-preserving properties, and then compute $\mathcal{F}(S)$ from f.

III. Background

The online density estimator used in this paper, called EDDO, represents joint densities using classifiers [4]. It builds on the product rule and expresses a joint density $f(X_1, \ldots, X_m \mid Y_1, \ldots, Y_l)$ as products of conditional densities, such that $f(X_1, \ldots, X_m \mid Y_1, \ldots, Y_l) = f_1(X_1 \mid Y_1, \ldots, Y_l) \cdot \prod_{i=2}^{m} f_i(X_i \mid Y_1, \ldots, Y_l, X_1, \ldots, X_{i-1})$. To model the density f, it is sufficient to model the density $f_1(X_1 \mid Y_1, \ldots, Y_l)$ and the densities $f_i(X_i \mid Y_1, \ldots, Y_l, X_1, \ldots, X_{i-1})$, $i \in \{2, \ldots, m\}$. For the individual densities f_i, $1 \leq i \leq m$, it employs classifiers that return class probability estimates and, in particular, Hoeffding trees [5]. To increase the robustness of the estimate, the authors also considered an ensemble of classifier chains, where each chain used a different variable ordering.

As it is assumed that users are not only interested in the full density but also specific parts, EDDO provides infrastructure to pose queries such as drawing instances, incorporating hard evidence, incorporating soft evidence, marginalizing out variables, and determining the density value of an instance (with respect to the given evidence) [4]. Density estimators supporting these inference operations constitute a *probabilistic condensed representation of data* on which data mining tasks such as pattern mining can be performed [1].

IV. Related Work

In this section, we present related work from the area of pattern mining and privacy-preserving data mining.

A. Pattern Mining

When applying itemset mining to real-world domains, users are often confronted with large volumes of frequent itemsets, easily outnumbering the data items themselves. Scanning these volumes is in many cases neither possible nor desired, so that a lot of research focused on finding so-called *condensed representation*s. Such a representation is supposed to consist of substantially fewer itemsets from which all frequent itemsets can be derived. First steps in this direction have been undertaken by Mannila and Toivonen [6], who proposed *positive* and *negative borders*. Although all frequent itemsets can be derived from them, the frequencies of non-border elements are lost. To solve this issue, other types of representations have been proposed [7], [8], [9], [10], [7], but it can still happen that too many itemsets are presented to the user. Therefore, researchers relaxed the definitions by introducing a parameter δ to control the size of the condensed representations while still allowing to derive most of the frequent itemsets [11], [12], [10]. This, however, comes at the cost of no longer having exact support values for the itemsets. Only if δ equals 0, exact support values can be computed. For $\delta > 0$, the support can only be computed with an error depending on δ [10], [11]. Deriving the support of other itemsets has also been the main focus of other types of condensed representations [8], [13], [14]. However, all of these approaches used itemsets to derive other itemsets, which is different from our approach.

With the increasing importance of data streams, researchers also addressed the problem of finding frequent itemsets from data streams. Most approaches either introduce some kind of decay rate to diminish the influence of old transactions or use a window [15]. But common to all of them is the underlying procedure of extracting itemsets when scanning the data stream. A popular method using a sliding window is *Moment* [16], which uses an in-memory prefix-tree-based data structure to represent four types of nodes indicating whether an itemset is infrequent, a potential candidate of becoming a frequent closed itemset (unpromising or promising), or a frequent closed itemset. Deviating from this approach of directly scanning the data stream, Geilke *et al.* [1] recently proposed the MiDEO framework, which is centered around a *probabilistic condensed representation of the data* estimating the joint density of the data stream. Due to inference operations, it provides sufficient information about the data to enable pattern mining, which the authors demonstrated with POEt [1].

B. Privacy-Preserving

Privacy-preserved association rule mining can be classified into two main directions: association rule mining on perturbed data and association rule hiding. Most of the work, however, is more concerned with hiding rules.

The objective of association rule hiding is to return only non-sensitive association rules to the user. To accomplish this task, most approaches either sanitize the transactions of a database or sanitize the rules returned by mining algorithm. The latter usually means deleting association rules until no sensitive rules can be inferred from the remaining ones [17], whereas the former requires to change the support or the confidence of a rule [18]. However, performing data sanitization to hide association rules is an NP-hard problem [19]. Therefore, many pursued heuristic approaches to hide association rule by sanitization. Atallah *et al.* [19] and Oliveira *et al.* [17] suggested a heuristic method that removed sensitive association rules based on the structure of the itemset graph. It continues until no sensitive rule is returned and none of the rules can be used to infer sensitive rules. Other approaches directly changed the transactions to influence the supports of

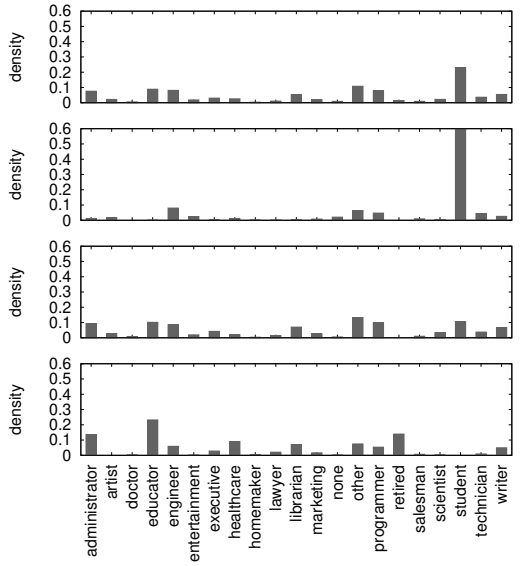


Fig. 1: The figure illustrates how the probability distributions of the movielens dataset differ for certain entity groups. At the top is the global distribution of the sensitive attribute $occupation$. Below are the distribution of the same attribute for entities of age $0-24$, $25-49$, and $50-75$, respectively.

the itemsets [18], [20], [18], [20]. Jain *et al.* [21], on the other hand, suggested to directly change the left- and the right-hand side. In the context of frequent itemset mining, Sun and Yu [22] used the border to monitor the effect of changing transactions in the database. Gkoulalas-Divanis and Verykios [23] also pursued a border-based approach, but they formulate the hiding of rules as a constraint satisfication problem, which they solved by binary integer programming.

The method proposed in this paper can be seen as a combination of data perturbation and rule sanitization. Similarly to typical data perturbation techniques, it hides information about individual entities by only maintaining a statistical model. At the same time, however, it also guarantees that no sensitive pattern is disclosed that can be associated with a small group of entities, which is similar to rule sanitization techniques.

V. Privacy-Preserving EDDO Estimates

In this section, we discuss how EDDO estimates can be modified to preserve the privacy of the entities described by the data. Due to space constraints, we only focus on t-closeness and do not present our algorithm for k-anonymity.

Let $\mathcal{Q}$ be a set of sensitive attributes that could be used to identify entities. *t-closeness* basically demands that, for a certain variable-value combination, the distribution of a sensitive variable does not deviate more than a threshold t from the corresponding global distribution [2]. The attributes in $\mathcal{Q}$ are called quasi-identifiers and are typically something like age, gender, or occupation. Figure 1 gives an example of the effect of the t-closeness property. To ensure t-closeness for an EDDO estimator, it suffices to perform two types of actions:

Algorithm 1: t-closeness

Input: $EDDO_{ECC}$ f_{ecc}, sensitive attributes $\mathcal{Q}$, the accepted deviation from the global distribution δ, the percentaged difference by which a distribution is changed $step$
Output: f' preserving t-closeness

```
1  for Q ∈ 𝒬 do
2      ĝ ← f̂(Q)
3      𝒩 ← collect nodes associated with Q
4      for node ∈ 𝒩 do
           // determine node distribution
5          if node.isLeaf() then
6              d ← node.getClassDistribution()
7          else
8              d ← node.getEdgeDistribution()
           // reduce distance ‖ĝ − d‖
9          while ‖ĝ − d‖ ≥ δ do
10             for 1 ≤ i ≤ values(q) do
11                 c ← ĝ[i] − d[i]
12                 d[i] ← d[i] + c · step/100
```

a) Adapting the edge weights: Between a node $node$ and its children are edges. Each edge has a weight representing the probability of following this edge to the corresponding child. Hence, the successor of $node$ is described by a probability distribution d over its children. If $node \in \mathcal{Q}$ and d deviates by more than t from the global distribution g, d can be adapted accordingly.

b) Adapting the class distributions:: Some Hoeffding trees have a sensitive variable as a class attribute. If the class distribution d deviates by more than t from the global distribution g, d is adapted accordingly.

The global distribution of a variable $Q \in \mathcal{Q}$ can be easily estimated by computing $\hat{g} = \hat{f}(X)$, i.e., the variables $\mathcal{X} \setminus \{Q\}$ have been marginalized out. To compute the distance between the estimate of the global distribution $\hat{g}$ and $\hat{f}$, there are several well-known distance measures on discrete probability distributions such as the variational distance or the KL-divergence. However, the authors who proposed the notion of t-closeness [2], suggested to use the *Earth Mover's Distance* instead (also known as the Wasserstein metric or Kantorovich metric [24]) In this paper, we do not assume a specific distance measure and simply write $\|\cdot\|$ for the distance.

Algorithm 1 formalizes these ideas. It corrects the probability distributions for every quasi identifier $Q \in \mathcal{Q}$ (line 1). In particular, it determines the global distribution $\hat{g}$ of Q (line 2) and then considers every node that is associated with Q (line 2-4). If it is a leaf, Q is the class attribute of the Hoeffding tree and it takes the class probabilities (line 6). If it is an inner node, it considers the distribution of its outgoing edges (line 8). Subsequently, it measures the distance between the global distribution $\hat{g}$ and the node distribution d (line 10). If it is larger than the given threshold δ, it corrects d until $\|\hat{g} - d\|$ is below δ. To do so, it changes each component of d towards $\hat{g}$ by iteratively correcting it by a factor of $\frac{step}{100}$ (lines 10-12). Theorem 1 states that Algorithm 1 turns a given EDDO estimator into an estimator preserving t-closeness.

Theorem 1. *Let $\hat{f}_{ECC}$ be an EDDO estimate, $\mathcal{Q}$ be a set of quasi identifiers, and δ be a threshold for the accepted deviation from the global distribution. Algorithm 1 ensures that $\hat{f}_{ECC}$ preserves t-closeness for all $Q \in \mathcal{Q}$.*

Proof: (proof by contradiction) Let $\mathcal{Q}$ be the set of quasi

identifiers and let G be a group of entities that is given by some evidence $(Z_1, v_1), \ldots, (Z_l, v_l)$ with $Z_i \in \mathcal{X}$ and $v_i \in values(Z_i)$, $1 \leq i \leq l$. Assume that the probability distribution of a random variable $Q \in \mathcal{Q}$ deviates more than δ from $\hat{f}'_{ECC}(Q \mid Z_1 = v_1, \ldots, Z_l = v_l)$, i.e., $\|\hat{f}'_{ECC}(Q) - \hat{f}'_{ECC}(Q \mid Z_1 = v_1, \ldots, Z_l = v_l)\| \leq \delta$. Algorithm 1 guarantees that the corresponding deviation for every node in $\hat{f}'$ is always smaller than or equal to δ. Hence, in order to enforce a deviation of more than δ, one needs to combine paths from several trees. Let $(Z_1, v_1), \ldots, (Z_l, v_l)$ be a combination for which $\hat{f}'_{ECC}(Q \mid Z_1 = v_1, \ldots, Z_l = v_l)$ deviates more than δ from $\hat{f}'_{ECC}(Q)$. Relevant paths are those that start at the root, end at a leaf, and contain all elements from $\mathcal{P} := \{(Z_1, v_1), \ldots, (Z_l, v_l)\}$. Relevant nodes $\mathcal{N}$ are the ones contained in the largest suffix not containing any element from $\mathcal{P}$. Hence, for each Hoeffding tree ht in $\hat{f}'_{ECC}$, the conditional probability $\hat{f}'_{ECC}(Q \mid Z_1 = v_1, \ldots, Z_l = v_l)$ results from $\sum_{C \in \mathcal{N}} weight(C) \cdot weight(ht) \cdot distribution(node)$, where $weight(C)$ is the weight induced by soft evidence and $weight(ht)$ is the weight of classifier chain to which ht belongs. Hence, $\|\hat{f}'_{ECC}(Q) - \hat{f}'_{ECC}(Q \mid Z_1 = v_1, \ldots, Z_l = v_l)\| = \sum_{C \in \mathcal{N}} w_C \cdot \|\hat{f}'_{ECC}(Q) - distribution(C)\|$, where $w_C = \frac{weight(C) \cdot weight(ht)}{\sum_{C \in \mathcal{N}} weight(C) \cdot weight(ht)}$. Since $\|\hat{f}'_{ECC}(Q) - distribution(C)\| \leq \delta$ for all $C \in \mathcal{N}$ and $w_c \leq 1$, it follows that $\|\hat{f}_{ECC}(Q) - \hat{f}'_{ECC}(Q \mid Z_1 = v_1, \ldots, Z_l = v_l)\| \leq \delta$. Since all choices were arbitrary, this contradicts the initial assumption. ■

Due to the nature of the modifications, the representation is possibly less accurate. Regarding data privacy, however, this is not only acceptable but actually desired – at least from the viewpoint of the entities described by the data.

VI. Privacy-Preserving Itemset Mining

Given an EDDO density estimate, Geilke *et al.* [1] showed that they contain sufficient information to discover the frequent itemsets of the data using inference algorithms. In this section, we show that that their approach can be improved by exploiting the structure of EDDO estimates.

Frequent itemsets are variable-value combinations exceeding a probability threshold θ. The probabilities of these combinations are contained in the classifier chains of EDDO estimates and can be computed by following the correct paths in its Hoeffding trees – from the roots to the leaves. Although each tree is only responsible for predicting the conditional probability of a given variable, it implicitly provides information about other variables through its inner nodes: Let $f(X_i \mid X_1, \ldots, X_{i-1})$ be a conditional density that is represented by a Hoeffding tree ht. Further, let $path := node_1, edge_1, node_2, edge_2, \ldots, edge_l, node_l$ be a path from the root ($node_1$) to a leaf ($node_l$), where each $node_i$ corresponds to a variable and each $edge_i$ corresponds to a value that $node_i$ can take. Then $f(X_i \mid X_1, \ldots, X_{i-1})$ is the density of the values of X_i given the variable values of $path$. By construction of ht, each node on the path tries to make $f(X_i \mid X_1, \ldots, X_{i-1})$ as discriminating as possible, i.e., choosing variables that allow to distinguish between values with low probability and those with high probability. The inner nodes of the resulting tree yield a partitioning of the data

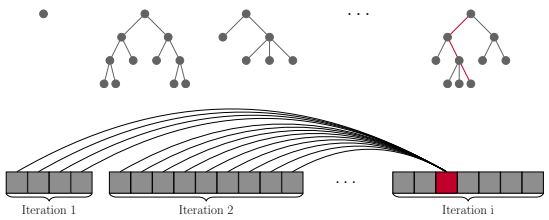


Fig. 2: At the top are the Hoeffding trees of a classifier chain. The red path shows one path that is turned into a itemset candidate. At the bottom is the merging, where the itemsets for iteration $i+1$ are constructed by combining the itemsets of iteration i with the ones from the previous iterations.

instances and the paths specify which values the variables should take to obtain certain probabilities. Hence, combining the paths from all trees would enable a direct construction of itemsets that is guided by the structure of the density estimate.

Pursuing this idea, we propose ISON (Itemset Mining on the Structure of Online deNsity estimates) in this section, which exploits this structure to guide the construction of itemsets (illustrated by Figure 2 and Algorithm 2): Unlike the well-known Apriori algorithm, which first constructs 1-itemsets, then 2-itemsets, ..., ISON immediately constructs itemsets of different length. It iterates over all Hoeffding trees and turns theirs paths into itemset candidates (lines $4-11$). Candidates for which $filter$ returns false are neither relevant for the output nor for future iterations and are therefore removed as early as possible. Extracting paths from the Hoeffding trees and filtering them is considered as iteration 1. Then it starts merging already found itemsets to larger ones by computing the union over the items of each merge partner (lines $12-20$). In the beginning, when only one iteration exists, the itemsets from the Hoeffding trees are merged pair-wise. For iteration $i+1$, ISON will take every itemset from iteration i and merges it with every itemset from the iterations $j < i$. As in the case of iteration 1, each candidate needs to pass the given filter.

In principle, we could also searched the itemset space the same way as Apriori, but the proposed approach has benefits: (1) We immediately obtain itemsets of different lengths, so that the user can interrupt the process and still obtain itemsets of different length. (2) By constructing the density estimates, a lot of statistical information is already contained in the Hoeffding trees, which is a more informed way of searching the itemset space. (3) Other kinds of itemsets can be mined from the statistical information, which is beyond the scope of this paper.

Algorithm 2 is only the general framework that can be combined with many different filters. For frequent itemsets, we need a filter that estimates the frequencies based on the given density estimate f_{ecc}. An example for such a filter is provided by Algorithm 3. It computes the frequency as $\sum_{f_{cc} \in f_{ecc}} w[f_{cc}] \cdot \hat{f}_{cc}.getDensityValue(candidate)$, where w is a weight vector for the classifier chains. Instead of using the original weights, however, it reweights the chains using the itemset and the chain ordering, because the ordering can have an impact on how accurately it estimates certain density values. The closer a variable is to the end of the ordering, the more variables are considered during split decisions, which possibly

Algorithm 2: $ISON$

Input: $EDDO_{ECC}$ $\hat{f}_{ecc}$, itemset filter $filter$
Output: the itemsets for which $filter$ is true

1 $iterations \leftarrow \emptyset$ // generated candidates
// iteration 0
2 $C' \leftarrow \emptyset$
3 **for** $ht \in \hat{f}_{cc}$ *and* $\hat{f}_{cc} \in \hat{f}_{ecc}$ **do**
4 $\mathcal{P} \leftarrow \{node_1, \ldots, node_l \in ht \mid node_1.isRoot() \wedge node_l.isLeaf()\}$
5 **for** $path \in \mathcal{P}$ **do**
6 $c \leftarrow$ turn $path$ into an itemset
// iterate over itemsets by size and prune candidates using filter
7 $it \leftarrow subsets(c,$ **pruning = true**$)$
8 $C' \leftarrow C' \cup \{s \mid s \in it\}$
9 $iterations.append(C')$
// iteration > 0
10 **while** $|C'| > 0$ **do**
11 $C_{curr} \leftarrow C', C' \leftarrow \emptyset$
12 **if** $|iterations| = 1$ **then**
13 $mergePartners \leftarrow iterations[1]$
14 **else**
15 $mergePartners \leftarrow iterations[1:i]$
16 **for** $c_1 \in mergePartners$ *and* $c_2 \in C_{curr}$ **do**
17 **if** $filter(merge(c_1, c_2))$ **then**
18 $C' \leftarrow C' \cup \{merge(c_1, c_2)\}$
19 $iterations.append(C')$
20 **return** $\bigcup_{1 \leq i \leq |iterations|} iterations[i]$

Algorithm 3: $isFrequent$

Input: $EDDO_{ECC}$ $\hat{f}_{ecc}$, threshold $\theta \in [0;1]$, itemset C
Output: true iff C is frequent

1 $w \leftarrow []$ // heuristic: weight chains
2 **for** $\hat{f}_{cc} \in \hat{f}_{ecc}$ **do**
// determine the indices of the attributes in the chain ordering
3 $O \leftarrow [argmin_{j=1,\ldots,m}(\hat{f}_{cc}[j] = X_i) \mid (X_i, v_i) \in C]$
// weight $\hat{f}_{cc}$ according to these indices
4 $w.append(\sum_{i=1}^{|O|} |O| - O[i] + \sum_{i=2}^{|O|} |O| - O[i] - O[i-1])$
5 $w \leftarrow normalize(w)$
6 **return** $\sum_{cc \in f_{ecc}} w[cc] \cdot cc(candidate) < \theta$

yields less accurate results for itemsets consisting of only a couple of variables. To take this into account, the $isFrequent$ filter reweights the chains of f_{ecc} for each itemset candidate in dependence of the variables ordering (lines $1-5$).

If $\hat{f}$ is an accurate representation of f, i.e., the density values correspond to the true density values, then $ISON(\hat{f}, isFrequent(\hat{f}, \theta))$, denoted as $ISON \circ isFrequent$, is able to discover all frequent itemsets:

Theorem 2. *Let f_{cc} be the joint density of the data stream S, and let $\theta \in [0;1]$ be the threshold for frequent itemsets. If $\hat{f}$ is an accurate representation of f, then $ISON(\hat{f}, isFrequent(\hat{f}, \theta)) = \mathcal{F}_S$.*

Proof: $\mathcal{F}_S \subseteq ISON \circ isFrequent$: (proof by induction) There is a Hoeffding tree in f_{cc} with target variable X for all variables $X \in \mathcal{X}$. Hence, we have the discrete probability distribution of all $X \in \mathcal{X}$. This implies that $(X, v) \in \mathcal{F}(S)$ for all $X \in \mathcal{X}$ and for all $v \in values(X)$, since $\hat{f}$ is an accurate representation of f.

Now, assume that $ISON \circ isFrequent$ contains all frequent itemsets of length k. Further assume that $itemset \in \mathcal{F}(S)$ be an arbitrary frequent itemset of length $k+1$ and that $itemset \notin ISON \circ isFrequent$. By the induction assumption, there is an iteration i in ISON where every proper subset of $itemset$ has been constructed by ISON. Hence, at latest in iteration $i+1$, $itemset$ will be constructed using one of its possible decomposition into subsets. Since $\hat{f}$ is an accurate representation of f, $itemset \in ISON \circ isFrequent$, which contradicts our assumptions and, therefore, implies that $\mathcal{F}_S \subseteq ISON \circ isFrequent$.

$ISON \circ isFrequent \subseteq \mathcal{F}_S$: Follows immediately, because $\hat{f}$ is an accurate representation of f. ∎

Theoretically, a single chain is sufficient. In practice, however, one chain could easily miss important variables interdependencies due the underlying ordering. By using an ensemble of chains, the probability of missing important attribute interdependencies is reduced, thereby increasing its robustness and accuracy. Theorem 2 can be directly applied to f_{ecc}.

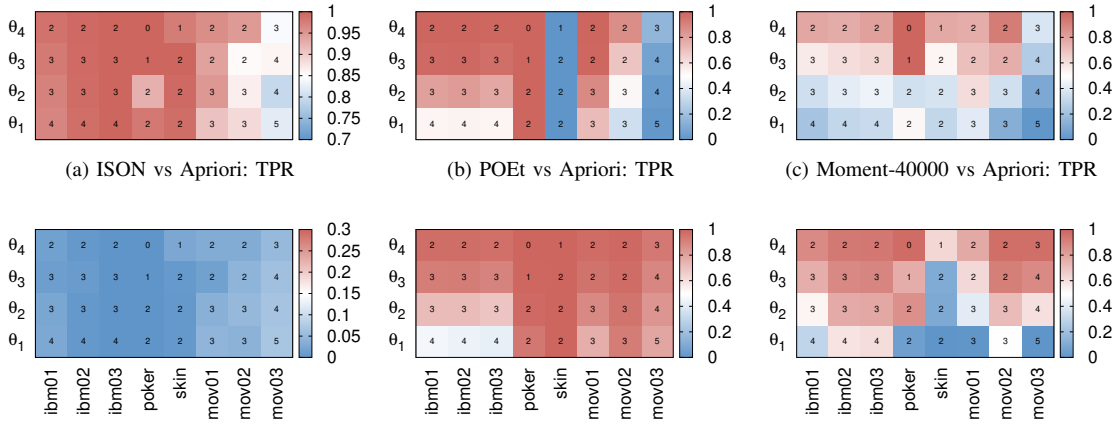
VII. Evaluation

In this section, we evaluate the capabilities of ISON to discover frequent itemsets and the effects of applying the t-closeness property to a density estimate. We compare to Apriori, as an algorithm delivering the ground truth, to Moment, as one of the few stream mining algorithms for which a functioning implementation is available, and to POEt. ISON has been implemented as part of MiDEO[1]. For Apriori we used the implementation by Christian Borgelt[2] and, for Moment, the MOA extension provided on the MOA website[3]. Since the implementation of Moment is not able to deal with non-Boolean variables and only produces closed itemsets, we added a wrapper that flattened the data stream and computed the frequent itemsets based on the closed frequent itemsets. As window sizes, we considered $1 \cdot 10^2$, $1 \cdot 10^3$, $1 \cdot 10^4$, $4 \cdot 10^4$, according to the dataset with the fewest instances.

To generate synthetic datasets, we used the well-known IBM dataset generator [25], where we set (1) the number of patterns to 5000, (2) the average transaction size to 4, (3) the average itemset lengths to 2 or 4, and (4) the number of instances to a value between $5 \cdot 10^4$ and $5 \cdot 10^7$. The real-world datasets are based on three publicly available datasets (http://archive.ics.uci.edu/ml/): skin, pokerhand, and movielens. To obtain more meaningful itemsets and to make some trends more visible, we modified movielens slightly: Instead of having the age of the users as attribute in the movielens dataset, we introduce a binning and grouped users into the four groups 0-25, 26-50, 51-75, 76-100. Moreover, as the dataset has relatively few instances compared to the number of possible combinations, we consider two variants of the movielens dataset having fewer variables, which will help to determine how the accuracy of the representation affects its ability to discover frequent itemsets.

The accuracy of a repesentation is mainly determined by two factors: the number of δ-tolerant closed frequent itemsets Δ and the number of instances N. For small δ, Δ basically characterizes the number of variable-value combinations having differing probabilities. As each of them need to be captured

[1] https://github.com/kramerlab/mideo
[2] http://www.borgelt.net/pyfim.html
[3] http://moa.cms.waikato.ac.nz/moa-extensions/



(a) ISON vs Apriori: TPR (b) POEt vs Apriori: TPR (c) Moment-40000 vs Apriori: TPR

(d) ISON vs Apriori: FPR (e) POEt vs Apriori: FPR (f) Moment-40000 vs Apriori: FPR

Fig. 3: The plots present the TPR and FPR values with respect to several datasets and support thresholds. For ibm01, ibm02, ibm03, poker, skin, and mov01, $\theta_1 = 0.05$, $\theta_2 = 0.10$, $\theta_3 = 0.20$, $\theta_4 = 0.30$. For mov02 and mov03, $\theta_1 = 0.50$, $\theta_2 = 0.60$, $\theta_3 = 0.70$, $\theta_4 = 0.80$. At the top are the TPR values (red is better, i.e., larger values); at the bottom are the FPR values (blue is better, i.e., smaller values). The numbers in the heatmap are logarithms to base 10 of the number of itemsets. Notice that we set the window size of Moment equally large for all datasets. This is, however, beneficial for the TPR, if no drift occurs. Also notice that POEt did not finish on the skin dataset within 24 hours, so that we set the TPR to 0 and the FPR to 100.

by the density estimate, it cannot treat them the same way without compromising its accuracy. If $|\Delta|$ is large and only few instances are available, an accurate description is difficult. The more instances are available, the easier it gets. To describe this dependence on Δ and N, we define the ratio of Δ and N as *combination ratio* and use it as a complexity measure.

A. Itemset Discovery

To assess the performance of the algorithms, we measure the true positive rate (TPR), i.e., $\frac{|\mathcal{F}_{ISON}(S) \subseteq \mathcal{F}_{Apriori}(S)|}{|\mathcal{F}_{Apriori}(S)|}$, and the false positive rate (FPR), i.e., $\frac{|\{I \in \mathcal{F}_{ISON}(S) | I \notin \mathcal{F}_{Apriori}(S)\}|}{|\mathcal{F}_{Apriori}(S)|}$.

How many itemsets are discovered by ISON is illustrated in Figure 3. The TPR is very high on datasets with a low combination ratio (ibm01, ibm02, ibm03, pokerhand, mov01) and high on datasets with a higher combination ratio (mov02 and mov03). Compared to POEt, ISON is showing an overall better TPR, but on datasets with fewer frequent itemsets (i.e., ibm01, ibm02, ibm03, skin, and mov01 with $\theta \geq 0.20$) both algorithms are roughly on par. Regarding the FPR, ISON exhibits very low values on all datasets, whereas POEt exhibits high values. This due to the few assumptions that POEt imposes on the density estimate. POEt constantly retrains density estimates and draws a fixed number of itemsets from them. Many of these itemsets are not frequent, which causes the low FPR values. These itemsets can obviously be filtered out using the original density estimate, as in the case of ISON, but it shows that POEt has to generate a multiple of itemsets to achieve high TPR values. Moreover, many of the itemsets are also duplicates, which increases the number of required itemsets even further. This is not a problem for datasets with a low combination ratio, but leads to a substantially increased running time for datasets with higher ratios. Hence, POEt can be successful in discovering extremely frequent itemsets but struggles with itemsets having a medium to low frequency.

ISON also exhibits a higher TPR and lower FPR than Moment on most datasets. If the window size is large enough and θ is large, we observe that ISON and Moment are on a similar level with respect to the TPR. These datasets have a rather low combination ratio, so that a fraction of the instances is already sufficient to determine the frequent itemsets precisely – at least for sufficiently large minimum support thresholds. Since Moment measures the frequency using a window, this helps to achieve a good performance. If the combination ratio is high, as in the case of the movielens dataset, the window size needs to be large to capture the frequencies of the itemsets properly. Surprisingly, however, increasing the window size on movielens seems to decrease the coverage. We assume that this is due to small distribution drifts in the data that cause Moment to correct the frequency of itemsets that are frequent within the window but not frequent overall. Although this leads to a low coverage, this behavior is actually intended – because Moment focuses on the currently frequent itemsets.

B. Number of Classifier Chains

In a separate experiment, we also analyzed how the number of classifier chains of the density estimate afffect the discovery of frequent itemsets. The results are summarized in Figure 4. It shows that the TPR does not change much, whereas the FPR can reduced substantially. When having several classifier chains to extract itemsets, the algorithm has a higher confidence on the itemsets that are represented in all classifier chains, whereas itemsets only available in one or two classifier chains are more or less ignored.

C. Running Time

To analyze ISON's running time behavior, we conducted an experiment on IBM datasets having equal properties and between $5 \cdot 10^3$ and $5 \cdot 10^7$ instances. The experiment has been

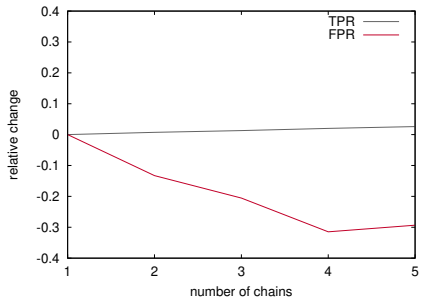


Fig. 4: The plot shows how much the TPR and FPR values of ISON are affected, if more than one classifier chain is used.

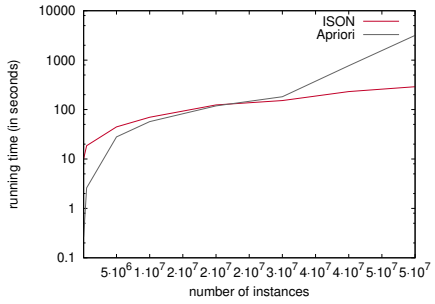


Fig. 5: This plot compares the running time of Apriori and ISON, if the number of instances is increased.

conducted on a standalone computer with an Intel® Core™ i7-4770K CPU (4 cores, 3.50 GHz, 8 MB cache) and 32 GB RAM. As baseline, we used Apriori instead of Moment, since Moment only produces the currently closed frequent itemsets.

The results are summarized in Figure 5. In the beginning, when the dataset consists of fewer than 2 million instances, Apriori extracts the frequent itemsets faster than ISON – especially, when only few instances are available. But, for increasing numbers of instances, ISON's running time only increases moderately, whereas Apriori's increases substantially. Considering the approach taken by ISON, this was to be expected, because ISON first estimates the joint density of the dataset and then extracts the itemsets from the estimate. Hence, the itemset discovery is completely independent of the size of the dataset and ISON only requires longer running times for constructing the density estimate, which is in general fast. Apriori, on the other hand, has to scan the dataset to discovery frequent itemsets, which takes longer, the larger the dataset.

D. Privacy

In addition to Theorem 1, we also evaluated the effects of the t-closeness property on real-world data. The plot in Figure 6 illustrates how many and how strongly itemsets are changed, if one allows no deviation from the global distribution for the attributes *age* and *occupation* of the movielens dataset. It shows that most of the density values are modified compared to the initial density estimator and is in line with our expectation, because the t-closeness algorithm makes sure that more extreme distributions are brought closer to the global distribution. This can also affect variable-value

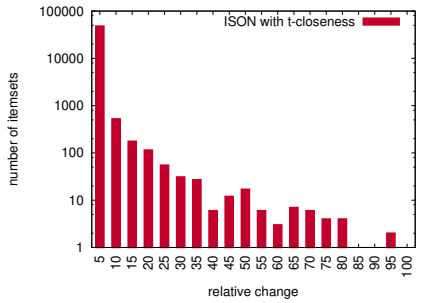


Fig. 6: This histogram shows how the t-closeness of $\delta = 0$ affects the supports of itemsets. On the x-axis is the percentaged deviation of the support, i.e., $x_{i-1} \leq \frac{f.getDensityValue(itemset)}{f'.getDensityValue(itemset)} \cdot 100 \leq x_i$, where f is the original estimate, f' is the modified estimate. On the y-axis is the number of itemsets matching this percentaged deviation.

TABLE II: When training an EDDO estimate on the movielens dataset, ISON extracts itemsets that can be associated with a single entity: a student of age 50-74. Without our t-closeness algorithm, the preferences, the itemsets are still visible, but the supports suggest that at least several hundreds of entities are associated with them, making it less likely that certain itemsets are associated with individual persons. Notice that '¬' indicates that this person did not choose this genre.

before	after	itemset
0.00002	0.02406	$\neg film{-}noir$
0.00002	0.02291	$\neg mystery$
0.00002	0.01871	$\neg thriller$
0.00002	0.01854	$\neg film{-}noir, \neg thriller$
0.00002	0.01843	$\neg mystery, \neg thriller$
0.00002	0.01816	$\neg action$
0.00001	0.01782	$\neg action, \neg film{-}noir$
0.00001	0.01695	$\neg action, \neg mystery$
0.00001	0.01665	$\neg comedy$
0.00001	0.01472	$\neg action, \neg thriller$
0.00001	0.01333	$\neg film{-}noir, \neg thriller$
0.00001	0.01136	$\neg comedy, \neg thriller$
0.00001	0.01132	$\neg action, \neg comedy$
0.00001	0.01116	$\neg comedy, \neg film{-}noir, \neg mystery$
0.00001	0.01112	$\neg comedy, \neg mystery, \neg thriller$
0.00001	0.01089	$\neg action, \neg comedy, \neg film{-}noir$
0.00001	0.01028	$\neg action, \neg comedy, \neg mystery$

combinations not containing any sensitive attributes, since their density values is computed by marginalizing out variables from $\mathcal{Q}$, which influences the weighting of Hoeffding tree branches. These changes are obviously intended, as information about the age and occupation of entity groups are no longer different from other groups and therefore protects smaller groups.

Table II illustrates how smaller groups can benefit from this effect. In this particular example, 17 itemsets with low support thresholds have been discovered from the movielens dataset. Given the number of instances, one can infer that each of these itemsets can be attributed to a single person. Because of the given age and the additional information that each person provided at least 20 ratings, one can also infer that all of them belong to the same person, a student between age 50 and 74. After applying the t-closeness algorithm, the frequency of these itemsets have been changed, so that they appear to originate from more than 400 persons.

VIII. Conclusion

In this paper, we proposed a novel itemset mining algorithm, called ISON, that extracts frequent itemsets from a probabilistic condensed representation. Compared to the current state-of-the-art method POEt, which pursues a sampling-based approach, ISON extracts the itemsets from the structure of the density estimate. We provided a proof that this strategy yields all frequent itemsets, if the representation is a perfect description of the density of the data stream. The experimental results showed that it discovers most of the itemsets and that it performs substantially better than POEt and Moment on datasets with a high combination ratio, a measure we introduced to capture the frequency of occurrence of distinct patterns in the data. Moreover, we discussed a strategy to ensure well-known properties from privacy-preserving data mining. In contrast to existing methods, we did not modify the data itself but its probabilistic condensed representation.

In the future, we would like to explore other types of itemsets, to extend ISON to numeric attributes, and to push or use other frequency-related constraints such as class-correlations.

References

[1] M. Geilke, A. Karwath, and S. Kramer, "A probabilistic condensed representation of data for stream mining," in *Proceedings of the 2014 International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2014, pp. 297–303.

[2] N. Li, T. Li, and S. Venkatasubramanian, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proceedings of the 23rd International Conference on Data Engineering ICDE*, 2007, pp. 106–115.

[3] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 2000, pp. 439–450.

[4] M. Geilke, E. Frank, A. Karwath, and S. Kramer, "Online estimation of discrete densities," in *Proceedings of the 13th International Conference on Data Mining (ICDM)*, 2013, pp. 191–200.

[5] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Knowledge Discovery and Data Mining*, 2000, pp. 71–80.

[6] H. Mannila and H. Toivonen, "Levelwise search and borders of theories in knowledge discovery," *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 241–258, 1997.

[7] T. Calders, C. Rigotti, and J. Boulicaut, "A survey on condensed representations for frequent sets," in *Proceedings of the European Workshop on Inductive Databases and Constraint Based Mining*, 2004, pp. 64–80.

[8] A. Bykowski and C. Rigotti, "A condensed representation to find frequent patterns," in *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2001.

[9] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal, "Mining frequent patterns with counting inference," *SIGKDD Explorations*, vol. 2, no. 2, pp. 66–75, 2000.

[10] J. Boulicaut, A. Bykowski, and C. Rigotti, "Free-sets: A condensed representation of boolean data for the approximation of frequency queries," *Data Mining and Knowledge Discovery*, vol. 7, no. 1, pp. 5–22, 2003.

[11] J. Cheng, Y. Ke, and W. Ng, "delta-tolerance closed frequent itemsets," in *Proceedings of the 6th IEEE International Conference on Data Mining*, 2006, pp. 139–148.

[12] J. Boulicaut and A. Bykowski, "Frequent closures as a concise representation for binary data mining," in *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2000, pp. 62–73.

[13] A. Bykowski and C. Rigotti, "DBC: a condensed representation of frequent patterns for efficient mining," *Information Systems*, vol. 28, no. 8, pp. 949–977, 2003.

[14] T. Calders and B. Goethals, "Mining all non-derivable frequent itemsets," in *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, 2002, pp. 74–85.

[15] J. Cheng, Y. Ke, and W. Ng, "A survey on algorithms for mining frequent itemsets over data streams," *Knowledge and Information Systems*, vol. 16, no. 1, pp. 1–27, 2008.

[16] Y. Chi, H. Wang, P. S. Yu, and R. R. Muntz, "Catch the moment: maintaining closed frequent itemsets over a data stream sliding window," *Knowledge and Information Systems*, vol. 10, no. 3, pp. 265–294, 2006. [Online]. Available: http://dx.doi.org/10.1007/s10115-006-0003-0

[17] S. R. M. Oliveira, O. R. Zaïane, and Y. Saygin, "Secure association rule sharing," in *Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, 2004, pp. 74–85.

[18] V. S. Verykios, A. K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association rule hiding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 434–447, 2004.

[19] M. Atallah, A. Elmagarmid, M. Ibrahim, E. Bertino, and V. Verykios, "Disclosure limitation of sensitive rules," in *Proceedings of the 1999 Workshop on Knowledge and Data Engineering Exchange*. Washington, DC, USA: IEEE Computer Society, 1999, pp. 45–52.

[20] Y. Saygin, V. S. Verykios, and C. Clifton, "Using unknowns to prevent discovery of association rules," *SIGMOD Record*, vol. 30, no. 4, pp. 45–54, 2001.

[21] Y. K. Jain, V. K. Yadav, and G. S. Panday, "An efficient association rule hiding algorithm for privacy preserving data mining," *International Journal on Computer Science and Engineering (IJCSE)*, vol. 3, no. 7, pp. 2792–2798, 2011.

[22] X. Sun and P. S. Yu, "Hiding sensitive frequent itemsets by a border-based approach," *Journal of Computing Science and Engineering JCSE*, vol. 1, no. 1, pp. 74–94, 2007.

[23] A. Gkoulalas-Divanis and V. S. Verykios, "Exact knowledge hiding through database extension," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 699–713, 2009.

[24] A. L. Gibbs and F. E. Su, "On choosing and bounding probability metrics," *International Statistical Review*, pp. 419–435, 2002.

[25] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of 20th International Conference on Very Large Data Bases (VLDB)*, 1994, pp. 487–499.