# Mining Frequent Itemsets through Progressive Sampling with Rademacher Averages

Matteo Riondato
Dept. of Computer Science
Brown University
Providence, RI 02912
matteo@cs.brown.edu

Eli Upfal
Dept. of Computer Science
Brown University
Providence, RI 02912
eli@cs.brown.edu

## ABSTRACT

We present an algorithm to extract an high-quality approximation of the (top-$k$) Frequent itemsets (FIs) from random samples of a transactional dataset. With high probability the approximation is a superset of the FIs, and no itemset with frequency much lower than the threshold is included in it. The algorithm employs progressive sampling, with a stopping condition based on bounds to the empirical Rademacher average, a key concept from statistical learning theory. The computation of the bounds uses characteristic quantities that can be obtained efficiently with a single scan of the sample. Therefore, evaluating the stopping condition is fast, and does not require an expensive mining of each sample. Our experimental evaluation confirms the practicality of our approach on real datasets, outperforming approaches based on one-shot static sampling.

## Categories and Subject Descriptors

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

## General Terms

Algorithms, Theory, Performance, Experimentation

## Keywords

Frequent Itemsets; Pattern Mining; Rademacher Averages; Sampling; Statistical Learning Theory

## 1. INTRODUCTION

The task of Frequent Itemsets (FIs) mining is to extract all sets of items that appear in at least a fraction $\theta$ of a transactional dataset $\mathcal{D}$, or the $k$ most frequent set of items [2]. It is a fundamental primitive of knowledge discovery and is useful, among the others, for market basket analysis, inference, classification, and network management [13]. Exact algorithms to mine FIs have since long been available but their practicality is hindered by the need to scan the dataset multiple times [1, 12]. When the dataset is too large to fit into main memory, as it is the case for many modern datasets, the running time of exact FIs mining algorithms may be too high to be practical. A natural way to reduce the dependency on the dataset size is to only analyze a small random sample of the dataset that can reside in main memory. The collection of FIs obtained from the sample will be an *approximation* to the exact collection, due to the fact that only a subset of the dataset is analyzed. Approximate collections of FIs are nevertheless acceptable in most cases due to the *exploratory nature* of the FIs mining step in the knowledge discovery process. There is an intrinsic trade-off between the size of the sample (number of transactions in the sample) and the accuracy of the estimation, but a loose analysis of this trade-off may result in sample sizes much larger than what is needed to obtain and approximation with the desired level of accuracy and confidence. It is therefore necessary, although challenging, to develop algorithms that leverage on tight bounds to the trade-off between sample size and accuracy in order to fully exploit the power of sampling.

**Contributions.** In this work we study the trade-off between approximation quality and sample size using concepts and results from *statistical learning theory* [29]. We present a randomized algorithm to mine a high-quality approximation of the collection of FIs w.r.t. a minimum frequency threshold $\theta$ (and of the top-$k$ most frequent itemsets) from random samples of the dataset $\mathcal{D}$. With probability at least $1-\delta$, for some user-specified $\delta \in (0,1)$, the returned approximation is a *superset* of the exact collection of FIs and no itemset in the approximation may have frequency less than $\theta - \varepsilon$, for some user-specified $\varepsilon \in (0,1)$. Moreover, the estimation of the frequency of all itemsets in the output is within $\varepsilon/2$ of their exact value. The algorithm uses *progressive sampling*, i.e., it starts from a small sample and enlarges it until a suitable *stopping condition* is verified, meaning that an high-quality approximation can be obtained from the sample. The stopping condition is based on bounds to the *empirical Rademacher average* of the problem at hand, a key concept from statistical learning theory [5]. In particular we prove that we can bound the empirical Rademacher average and therefore the maximum deviation between the frequency of an itemset in the dataset and the frequency of that itemset in the sample using a function of the sample size, of $\delta$, and of a partitioning of the set of Closed Itemsets (CIs) [19] in the sample. We also give a bound, which is of independent interest, to the number of CIs in the sample. To our knowledge this is the first algorithm that uses


*KDD'15,* August 10-13, 2015, Sydney, NSW, Australia.

DOI: http://dx.doi.org/10.1145/2783258.2783265 .

bounds on the empirical Rademacher average in the domain of pattern mining, and one of the first to adapt these highly theoretical concepts to develop an efficient algorithm for an important practical task. We conducted an extensive experimental evaluation to test our algorithm and assess its performances in terms of the quality of the returned collection of itemsets and of the runtime, comparing it with standard baselines.

**Outline.** We start by reviewing related works in Sect. 2. We then formalize the problem of FIs mining and formally define the concept of approximation in Sect. 1. Our algorithm and its analysis are presented in Sect. 4. The goals, methodology, and results of the experimental evaluation can be found in Sect. 5. Finally, we draw some conclusions and suggest some future directions in Sect. 6.

## 2. RELATED WORK

The idea of using random samples to speed up the extraction of FIs has been studied since shortly after the first efficient exact algorithms had been presented [28]. Many works focused on deriving bounds for the size of a single sample to obtain high-quality approximation. Riondato and Upfal [24] present what is currently the best available bound. We refer the interested reader to their extensive discussion of previous results on fixed sample sizes and we focus here on the works that examined progressive sampling [7, 8, 14, 18, 20, 21, 26].

The use of progressive sampling, in contrast with a sample of fixed size, can contribute to an even greater speed up of the extraction of FIs, especially when combined with an appropriate schedule and starting sampling size [3, 14, 21]. Developing a stopping rule that allows to obtain approximations of guaranteed quality is a challenging task. Chen et al. [7], Parthasarathy [18], and Chuang et al. [8] propose progressive-sampling-based algorithms that use *heuristics* based on self-similarity or the frequency of single items to determine the stopping sampling size. Because of the use of heuristics, these approaches offer *no guarantee* on the quality of the obtained collection. In contrast, our algorithm returns, with high probability, a collection of itemsets with strong approximation guarantees.

Pietracaprina et al. [20] and Scheffer and Wrobel [26] focuses on extracting the top-$k$ most frequent itemsets using progressive sampling. The stopping condition suggested by Scheffer and Wrobel [26] employs progressive filtering of the set of candidate FIs based on Chernoff bounds until only $k$ itemsets are left, but offers no guarantee on whether the returned collection contains *any* of the actual top-$k$ FIs, rather a much weaker guarantee is offered. Our algorithm instead guarantees that the returned collection of itemsets is a *superset* of the top-$k$ FIs. The algorithm by Pietracaprina et al. [20] uses a stopping condition based on the frequency of *all* itemsets in the sample. The analysis is based on traditional Chernoff and union bounds and limited to itemsets up to a fixed length. Our algorithm does not suffer from this limitation and our analysis uses powerful deviation bounds based on Rademacher averages.

Another major point of difference between our work and the ones previously presented is the fact that checking the stopping condition of our algorithm does *not* require to run an exact FI mining algorithm on the sample. As a consequence, our stopping condition is much more efficient to evaluate, resulting in lower running time.

We use bounds to the Rademacher averages [4, 16], an important concept from statistical learning theory. We only introduce the necessary notation and results, and we refer the reader to the book by Shalev-Shwartz and Ben-David [27] for an in-depth presentation of these topics.

To the best of our knowledge, the only previous use of bounds or estimates of the Rademacher averages in a progressive sampling setting is the work by Elomaa and Kääriäinen [9] on learning two-level decision trees, whose settings and problem are very different from the ones we study.

## 3. DEFINITIONS AND PRELIMINARIES

Let $\mathcal{I}$ be a *set of items* with an arbitrary fixed total order "$<$". A *transaction* is a subset of $\mathcal{I}$, and a *transactional dataset* is a collection of transactions. An *itemset* is a set of items that appear together in a transaction (i.e., a subset of a transaction). Given an itemset $A$ and a transaction $\tau$ s.t. $A \subseteq \tau$, we say that $A$ *appears* or *is contained* in $\tau$ and that $\tau$ *contains* $A$. The *support set* $T_{\mathcal{D}}(A)$ *of* $A$ *in* $\mathcal{D}$ is the subset of transactions in $\mathcal{D}$ that contain the itemset $A$, and the *frequency* of itemset $A$ in dataset $\mathcal{D}$ is the *fraction* of transactions of $\mathcal{D}$ that contain $A$:

$$f_{\mathcal{D}}(A) = |T_{\mathcal{D}}(A)|/|\mathcal{D}| \ .$$

Given a *frequency threshold* $\theta \in (0, 1]$, the set $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ of *Frequent Itemsets (FIs) in* $\mathcal{D}$ *w.r.t.* $\theta$ is the collection of all itemsets with frequency at least $\theta$ in $\mathcal{D}$:

$$\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta) = \{(A, f_{\mathcal{D}}(A)) \ : \ A \subseteq \mathcal{I} \wedge f_{\mathcal{D}}(A) \geq \theta\} \ .$$

Similarly, let $f_{\mathcal{D}}^{(k)}$ be the maximum frequency such that at least $k$ itemsets have frequency at least $f_{\mathcal{D}}^{(k)}$ in $\mathcal{D}$, then the set of the top-$k$ FIs is

$$\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, k) = \mathsf{FI}(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(k)}) \ .$$

Note that $|\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, k)| \geq k$.

**Goal.** We aim at approximating the collection of (top-$k$) FIs by *mining* (i.e., extracting the FIs from) *random samples* of $\mathcal{D}$ (i.e., random collections of transactions from $\mathcal{D}$).

Definition 1. *For $\varepsilon, \delta \in (0, 1)$, a $(\varepsilon, \delta)$-approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ is a collection $\mathcal{C} = \{(A, f_A) \ : \ A \subseteq \mathcal{I}, f_A \in (0, 1]\}$ such that, with probability at least $1 - \delta$:*

1. *for any $(A, f_{\mathcal{D}}(A)) \in \mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ there is a pair $(A, f_A) \in \mathcal{C}$; and*
2. *for any $(A, f_A) \in \mathcal{C}$, it holds $f_{\mathcal{D}}(A) \geq \theta - \varepsilon$; and*
3. *for any $(A, f_A) \in \mathcal{C}$, it holds $|f_{\mathcal{D}}(A) - f_A| \leq \varepsilon/2$.*

*An $(\varepsilon, \delta)$-approximation of $\mathsf{FI}(\mathcal{D}, \mathcal{I}, f_{\mathcal{D}}^{(k)})$ is an $(\varepsilon, \delta)$-approximation of $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, k)$.*

## 4. A PROGRESSIVE SAMPLING ALGORITHM WITH GUARANTEES

We want to compute an $(\varepsilon, \delta)$-approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$ (or to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, k)$) from random samples of $\mathcal{D}$ of progressively increasing size (i.e., through *progressive sampling*). In the rest of this section we focus on $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$, while the case for top-$k$ FIs is presented in Sect. 4.5.

The basic steps of the *iterative* progressive sampling process are:

1. at iteration $i$, create a random sample $\mathcal{S}_i$ of some predefined size $|\mathcal{S}_i|$ by drawing transaction uniformly and independently at random (with replacement) from $\mathcal{D}$;

2. check a *stopping condition* to determine if an $(\varepsilon,\delta)$-approximation of $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ can be extracted from $\mathcal{S}_i$;
3. if the stopping condition is satisfied, return the collection $\mathsf{FI}(\mathcal{S}_i,\mathcal{D},\gamma)$ for an appropriate value of $\gamma$ and exit, otherwise increase $i$ and return to step 1.

In order to obtain an algorithm from this high-level description, it is necessary to formally specify the following components:

1. a *sampling schedule* $(|\mathcal{S}_i|)_{i\geq 1}$ of sample sizes.
2. a *stopping condition* involving the sample $\mathcal{S}_i$, and an *efficient* procedure to check this condition.
3. a *revised minimum frequency threshold* $\gamma$.

Any non-decreasing sequence $(|\mathcal{S}_i|)_{i\geq 1}$ can act as a sample schedule, giving complete freedom to the algorithm designer and the user in this sense. In Sect. 4.6 we show how to compute the next sample size using information from the current sample.

The choice of $\gamma$ and of the stopping condition are intertwined. We choose $\gamma=\theta-\varepsilon/2$, as motivated by the following lemma, and this choice defines rigorous requirements for the stopping condition.

LEMMA 1. *Let $\mathcal{S}$ be a sample of $\mathcal{D}$, and consider the event*

$$\mathcal{E}_{\mathcal{S}} : \text{“}|f_{\mathcal{D}}(A)-f_{\mathcal{S}}(A)|\leq\frac{\varepsilon}{2}\ \textit{for all}\ A\subseteq\mathcal{I}\text{”}\ . \qquad (1)$$

*If*

$$\Pr(\mathcal{E}_{\mathcal{S}})\geq 1-\delta, \qquad (2)$$

*then the collection $\mathsf{FI}(\mathcal{S},\mathcal{I},\theta-\varepsilon/2)$ is a $(\varepsilon,\delta)$-approximation to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$.*

PROOF. Assume that the event $\mathcal{E}_{\mathcal{S}}$ in (1) is verified, which happens by hypothesis with probability at least $1-\delta$. Then for no itemset $B\in\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ we may have $f_{\mathcal{S}}(B)<\theta-\varepsilon/2$, hence $B\in\mathsf{FI}(\mathcal{S},\mathcal{I},\theta-\varepsilon/2)$, as required by property 1 from Def. 1. Let $C$ be any itemset with $f_{\mathcal{D}}(C)<\theta-\varepsilon$. We have that $f_{\mathcal{S}}(C)<\theta-\varepsilon/2$, so $C\notin\mathsf{FI}(\mathcal{S},\mathcal{I},\theta-\varepsilon/2)$, which is the condition specified by property 2 from Def. 1. Property 3 from Def. 1 follows from the fact that the event $\mathcal{E}_{\mathcal{S}}$ is verified. □

This lemma gives the intuition behind the stopping condition of our algorithm: we can stop when (2) holds for the sample $\mathcal{S}$ under consideration, as we can then use $\gamma=\theta-\varepsilon/2$ to extract $\mathsf{FI}(\mathcal{D},\mathcal{I},\gamma)$, which is an $(\varepsilon,\delta)$-approximation to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$.

The rest of this section is devoted to formalize this condition and derive a procedure to check whether (2) holds for a sample $\mathcal{S}$.

Checking whether (2) holds is equivalent to checking whether, with probability at least $1-\delta$,

$$\sup_{A\subseteq\mathcal{I}}|f_{\mathcal{D}}(A)-f_{\mathcal{S}}(A)|\leq\frac{\varepsilon}{2},$$

hence we focus on bounding this quantity.

## 4.1 Rademacher averages

For each itemset $A\subseteq\mathcal{I}$, we define the *indicator function* $\phi_A\colon 2^{\mathcal{I}}\to\{0,1\}$ as:

$$\phi_A(B)=\begin{cases}1 & \text{if } A\subseteq B\\ 0 & \text{otherwise}\end{cases}\quad\text{for any } A\subseteq\mathcal{I}, B\subseteq\mathcal{I}\ .$$

When $B$ is a transaction, $\phi_A(B)=1$ if the itemset $A$ appears in the transaction $B$. Hence, we have

$$f_{\mathcal{D}}(A)=\frac{1}{|\mathcal{D}|}\sum_{\tau\in\mathcal{D}}\phi_A(\tau)$$

类似地

and analogously for the frequency $f_{\mathcal{S}}(A)$ of $A$ in a sample $\mathcal{S}$.

Assume that the sample $\mathcal{S}$ has size $|\mathcal{S}|=n$. For each $\tau_i\in\mathcal{S}$, $1\leq i\leq n$, let $\sigma_i$ be a Rademacher random variable, i.e., a random variable taking value $-1$ or 1, each with probability 1/2. The random variables $\sigma_i$ are independent. The *(sample) conditional Rademacher average* is the quantity

$$\mathcal{R}_{\mathcal{S}}=\mathbb{E}_{\sigma}\left[\sup_{A\subseteq\mathcal{I}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\phi_A(\tau_i)\right],$$

where $\mathbb{E}_{\sigma}$ denotes the expectation taken only w.r.t. the random variables $\sigma_i$, $1\leq i\leq n$ (i.e., *conditionally* on the sample) [5, 16]. An important result from statistical learning theory bounds the supremum of the deviations with the conditional Rademacher average.

THEOREM 1 (THM. 3.2 [5]). *With probability at least $1-\delta$,*

$$\sup_{A\subseteq\mathcal{I}}|f_{\mathcal{D}}(A)-f_{\mathcal{S}}(A)|\leq 2\mathcal{R}_{\mathcal{S}}+\sqrt{\frac{2\ln(2/\delta)}{n}}\ .$$

Note that the bound in the above theorem depends only on properties of the sample.

Computing $\mathcal{R}_{\mathcal{S}}$ directly is not easy. It would first require to mine *all* itemsets from $\mathcal{S}$ (i.e., extracting $\mathsf{FI}(\mathcal{S},\mathcal{I},1/|\mathcal{S}|)$, which is excessively expensive, and then to find the expectation over the $\sigma_i$ variables. Given that no analytical methods are currently available to compute this expectation in general, this second step would require an expensive Monte-Carlo simulation [5]. Nevertheless a different result from statistical learning theory allows us to bound $\mathcal{R}_{\mathcal{S}}$ using combinatorial properties of the sample. For any itemset $A\subseteq\mathcal{I}$, let $\mathbf{v}_{\mathcal{S}}(A)$ be the $n$-dimensional vector

$$\mathbf{v}_{\mathcal{S}}(A)=(\phi_A(\tau_1),\ldots,\phi_A(\tau_n)),$$

and let $V_{\mathcal{S}}=\{\mathbf{v}_{\mathcal{S}}(A), A\subseteq\mathcal{I}\}$. Since $V_{\mathcal{S}}$ is a *set* rather than a *bag*, we have $|V_{\mathcal{S}}|\leq 2^{|\mathcal{I}|}$ (and potentially $|V_{\mathcal{S}}|\ll 2^{|\mathcal{I}|}$).

THEOREM 2 (MASSART'S LEMMA, THM. 3.3 [5]).

$$\mathcal{R}_{\mathcal{S}}\leq\max_{A\subseteq\mathcal{I}}\|\mathbf{v}_{\mathcal{S}}(A)\|\frac{\sqrt{2\ln|V_{\mathcal{S}}|}}{n},$$

*where $\|\cdot\|$ denotes the Euclidean norm.*

Although the above is the form in which the Theorem is usually stated, a careful reading of its proof allows us to state the following stronger version.

THEOREM 3. *Let $w:\mathbb{R}^+\to\mathbb{R}^+$ be the function*

$$w(s)=\frac{1}{s}\ln\sum_{\mathbf{v}\in V_{\mathcal{S}}}\exp(s^2\|\mathbf{v}\|^2/(2n^2))\ . \qquad (3)$$

*Then*

$$\mathcal{R}_{\mathcal{S}}\leq\min_{s\in\mathbb{R}^+}w(s)\ .$$

PROOF. As in the proof for [5, Thm. 3.3] we can use the independence of the $\sigma_i$'s and the Hoeffding's inequality to show that, for any $s > 0$ and for any itemset $A \subseteq \mathcal{I}$, we have

$$\mathbb{E}_\sigma\left[\exp\left(s\frac{1}{n}\sum_{i=1}^{n}\sigma_i\phi_A(\tau_i)\right)\right] \leq \exp\left(\frac{s^2\|\mathbf{v}_\mathcal{S}(A)\|^2}{2n^2}\right) \ .$$

We can use this inequality to write

$$\begin{aligned} e^{s\mathcal{R}_\mathcal{S}} &= \exp\left(s\mathbb{E}_\sigma\left[\max_{A\subseteq\mathcal{I}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\phi_A(\tau_i)\right]\right) \\ &\leq \mathbb{E}_\sigma\left[\exp\left(s\max_{A\subseteq\mathcal{I}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\phi_A(\tau_i)\right)\right] \\ &\leq \sum_{A\subseteq\mathcal{I}}\mathbb{E}_\sigma\left[\exp\left(s\frac{1}{n}\sum_{i=1}^{n}\sigma_i\phi_A(\tau_i)\right)\right] \\ &\leq \sum_{A\subseteq\mathcal{I}}\exp\left(\frac{s^2\|\mathbf{v}_\mathcal{S}(A)\|^2}{2n^2}\right) \ . \end{aligned}$$

We can now take the logarithm on both sides and divide by $s$ (which is strictly positive) and we obtain $w$. Since the above inequalities are true for any $s > 0$, we can choose the one that minimizes the r.h.s. to obtain the thesis. □

As we show later, computing the function $w$ is too expensive for our purposes as it requires the computation of the set $V_\mathcal{S}$, therefore, in the following, we develop an upper bound to $w$ that is easy and fast to compute.

## 4.2 Connection with Closed Itemsets

We now show a connection between the set $V_\mathcal{S}$ and the collection of Closed Itemsets [19].

We recall that a *Closed Itemset* (CI) is an itemset $A \subseteq \mathcal{I}$ such that none of its proper supersets has the same frequency of $A$ (i.e., there is no $B \supsetneq A$ s.t. $f_\mathcal{S}(B) = f_\mathcal{S}(A)$) [19]. Let $\mathsf{CI}(\mathcal{S})$ be the set of CIs in the sample.

LEMMA 2. *The set $V_\mathcal{S}$ contains all and only the vectors $\mathbf{v}_\mathcal{S}(A)$ for all $A \in \mathsf{CI}(\mathcal{S})$:*

$$V_\mathcal{S} = \{\mathbf{v}_\mathcal{S}(A), A \in \mathsf{CI}(\mathcal{S})\}, \ \textit{and} \ |V_\mathcal{S}| = |\mathsf{CI}(\mathcal{S})| \ .$$

To prove Lemma 2, we need the following result.

LEMMA 3. *Let $S \subseteq \mathcal{S}$. There is at most one CI $A$ in $\mathcal{S}$ whose support set in $\mathcal{S}$ is $T_\mathcal{S}(A) = S$.*

PROOF. Suppose that there could be more than one CI in $\mathcal{S}$ with support set $S$, for example, w.l.o.g., two itemsets $C$ and $D$. Then the support set of $C \cup D$ in $\mathcal{S}$ would be exactly $S$, so $C$ and $D$ can not be closed, as there is a superset of them with the same support set. We reached a contradiction, so the thesis is true. □

We can now prove Lemma 2.

PROOF OF LEMMA 2. Let $A$ be a CI in $\mathcal{S}$, and let $S_A$ be the set of subsets of $A$ with the same frequency in $\mathcal{S}$ as $A$:

$$S_A = \{B \subseteq A \ : \ f_\mathcal{S}(B) = f_\mathcal{S}(A)\} \ .$$

The elements of $S_A$ are the itemsets that appear in all and only the transactions of $\mathcal{S}$ where $A$ appears. This means that, for all $B \in S_A$, $\mathbf{v}_\mathcal{S}(B) = \mathbf{v}_\mathcal{S}(A)$. To conclude the proof it is sufficient to show that there can not be two CIs $C$ and $D$ in $\mathcal{S}$ s.t. $\mathbf{v}_\mathcal{S}(C) = \mathbf{v}_\mathcal{S}(D)$. This is an immediate consequence of Lemma 3 and the proof is complete. □

This result explains why computing the function $w$ from (3) is expensive: we would need to extract the set $\mathsf{CI}(\mathcal{S})$ of *all* CIs in the sample (i.e., mine the sample at frequency $1/|\mathcal{S}|$). In the following we develop an upper bound to $w$ that can be computed efficiently with a single scan of the sample.

## 4.3 Bounding the Rademacher Average

In this section we show how to efficiently bound the conditional Rademacher average $\mathcal{R}_\mathcal{S}$. To do so, we define a function $\tilde{w}$ which is an upper bound to $w$ from (3) in every point of $\mathbb{R}^+$. The advantage of $\tilde{w}$ is that it can be computed using just the frequencies in the sample of the items in $\mathcal{I}$ and some additional information that can be obtained with a single scan of the sample. To define $\tilde{w}$ we need a partitioning of $\mathsf{CI}(\mathcal{S})$ that we now introduce.

Assume to sort the items in $\mathcal{I}_\mathcal{S}$ in increasing order by their frequency in $\mathcal{S}$, ties broken arbitrarily (e.g., according to the order $<$ on $\mathcal{I}$). Let $<_\mathrm{i}$ denote the resulting ordering. Given an item $a$, assume to sort the transactions of $T_\mathcal{S}(\{a\})$ in increasing order by the number of items they contain that come after $a$ in the ordering $<_\mathrm{i}$, ties broken arbitrarily (e.g., using unique transaction identifiers). Let $<_a$ denote the resulting ordering.

Let $\mathcal{C}_1 = \mathsf{CI}(\mathcal{S}) \cap \mathcal{I}_\mathcal{S}$ and $\mathcal{C}_{2+}$ be the subset of $\mathsf{CI}(\mathcal{S})$ containing only the CIs of size at least two. We partition $\mathcal{C}_{2+}$ as follows. Let $A \in \mathcal{C}_{2+}$ and let $a \in A$ be the item in $A$ that comes before any other item in $A$ according to the order "$<_\mathrm{i}$". Let $\tau$ be transaction containing $A$ that comes before any other transaction containing $A$ in the order "$<_a$". Clearly $a \in \tau$. We assign $A$ to the set $\mathcal{C}_{a,\tau}$.

Consider now a transaction $\tau \in T_\mathcal{S}(\{a\})$, and assume that it contains *exactly* $k_{a,\tau}$ items that come after $a$ in the ordering $<_\mathrm{i}$. In the ordering $<_a$, the transaction $\tau$ comes

1. *before* all transactions with more than $k_{a,\tau}$ items that come after $a$ in the ordering $<_\mathrm{i}$ and
2. *before* zero or more of the transactions with exactly $k_{a,\tau}$ items that come after $a$ in the ordering $<_\mathrm{i}$ (the exact number of such transactions depends on the tie-breaking criteria).

For each $r \geq 1$, let $g_{a,r}$ be the number of transactions in $T_\mathcal{S}(\{a\})$ containing exactly $r$ items that come after $a$ in the ordering $<_\mathrm{i}$. Let $\chi_a = \max\{r \ : \ g_{a,r} > 0\}$ and let

$$h_{a,r} = \sum_{j\geq r}^{\chi_a} g_{a,j} \ .$$

The value $\chi_a$ is the maximum $r$ such that there exists at least one transaction in $T_\mathcal{S}(\{a\})$ containing exactly $r$ items that come after $a$ in the order $<_\mathrm{i}$. Each value $h_{a,r}$ is the number of transactions in $T_\mathcal{S}(\{a\})$ that contain *at least* $r$ items that come after $a$ in the order $<_\mathrm{i}$.

Now, assume that $\tau$ is the $\ell_{a,\tau}$-th transaction in the ordering $<_a$ that contains exactly $k_{a,\tau}$ items that come after $a$ in the ordering $<_\mathrm{i}$. In other words, if we consider only the transactions containing exactly $k_{a,\tau}$ items that come after $a$ in the ordering $<_\mathrm{i}$, then $\tau$ is the $\ell_{a,\tau}$-th of such transactions in the ordering $<_a$. We have the following result on the size of $\mathcal{C}_{a,\tau}$.

LEMMA 4. *We have*

$$|\mathcal{C}_{a,\tau}| \leq 2^{\min\{k_{a,\tau}, h_{a,k_{a,\tau}} - \ell_{a,\tau}\}} \ .$$

PROOF. The quantity $2^{k_{a,\tau}}$ is the number of subsets $B$ of $\tau$ such that $B = \{a\} \cup C$ where $C$ is any non-empty subset of $\tau$ containing only items that come after $a$ in the order $<_{\mathrm{i}}$. Since $\mathcal{C}_{a,\tau}$ contains only itemsets that appear in $\tau$ and are in the form of $B$, then $|\mathcal{C}_{a,\tau}| \leq 2^{k_{a,\tau}}$.

Consider now an itemset $A \in \mathcal{C}_{a,\tau}$. Apart from $\tau$, $A$ can only appear in transactions $\tau' \in T_{\mathcal{S}}(\{a\})$ such that $\tau <_a \tau'$, as $A = \{a\} \cup C$, for $C$ as above. This is true for any itemset $A \in \mathcal{C}_{a,\tau}$. Let $\mathcal{T}$ denote the set of such transactions, then $|\mathcal{T}| = h_{a,k_{a,\tau}} - \ell_{a,\tau}$. From Lemma 3 we have that there is at most one CI for each set $D = \{\tau\} \cup F$ of transactions, where $F \subseteq \mathcal{T}$, so there at most $2^{h_{a,k_{a,\tau}} - \ell_{a,\tau}}$ CIs in $\mathcal{C}_{a,\tau}$. □

From Lemma 4 and the fact that

$$\mathsf{CI}(\mathcal{S}) = \mathcal{C}_1 \cup \mathcal{C}_{2+} = \mathcal{C}_1 \cup \left( \bigcup_{a \in \mathcal{I}_{\mathcal{S}}} \bigcup_{\tau \in T_{\mathcal{S}}(\{a\})} \mathcal{C}_{a,\tau} \right) \quad (4)$$

we have the following result on the number of Closed Itemsets in $\mathcal{S}$, which is of independent interest.

COROLLARY 1.

$$|\mathsf{CI}(\mathcal{S})| \leq |\mathcal{I}_{\mathcal{S}}| + \sum_{a \in \mathcal{I}_{\mathcal{S}}} \sum_{\tau \in T_{\mathcal{S}}(\{a\})} 2^{\min\{k_{a,\tau}, h_{a,k_{a,\tau}} - \ell_{a,\tau}\}} \ .$$

The following lemma puts together the above results to obtain an upper bound to $\mathcal{R}_{\mathcal{S}}$.

LEMMA 5. *Let $\tilde{w} : \mathbb{R}^+ \to \mathbb{R}^+$ be the function*

$$\tilde{w}(s) = \frac{1}{s} \ln \sum_{a \in \mathcal{I}_{\mathcal{S}}} \left( \left( 1 + \sum_{r=1}^{\chi_a} \sum_{j=1}^{g_{a,r}} 2^{\min\{r, h_{a,r} - i\}} \right) e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}} \right) \ .$$

*Then*

$$\mathcal{R}_{\mathcal{S}} \leq \min_{s \in \mathbb{R}^+} \tilde{w}(s) \ .$$

PROOF. Consider the function $w$ from (3). From the definition of Euclidean norm, we have that, for any $A \subseteq \mathcal{I}$, $\|\mathbf{v}_{\mathcal{S}}(A)\| = \sqrt{n f_{\mathcal{S}}(A)}$. Using this fact and combining Lemma 2 and the equality from (4), we can rewrite $w$ as

$$w(s) = \frac{1}{s} \ln \left( \sum_{a \in \mathcal{C}_1} e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}} + \sum_{a \in \mathcal{I}_{\mathcal{S}}} \sum_{\tau \in T_{\mathcal{S}}(a)} \sum_{A \in \mathcal{C}_{a,\tau}} e^{\frac{s^2 f_{\mathcal{S}}(A)}{2n}} \right) \ .$$

We now show that $w(s) \leq \tilde{w}(s)$ for any $s \in \mathbb{R}^+$. The thesis will then follow from Thm. 3.

First of all, since $\mathcal{C}_1 \subseteq \mathcal{I}_{\mathcal{S}}$, we have

$$\sum_{a \in \mathcal{C}_1} e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}} \leq \sum_{a \in \mathcal{I}_{\mathcal{S}}} e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}} \ .$$

Then, for any $a \in \mathcal{I}_{\mathcal{S}}$,

$$\sum_{\tau \in T_{\mathcal{S}}(\{a\})} \sum_{A \in \mathcal{C}_{a,\tau}} e^{\frac{s^2 f_{\mathcal{S}}(A)}{2n}} \leq \sum_{\tau \in T_{\mathcal{S}}(\{a\})} 2^{\min\{k_{a,\tau}, h_{a,k_{a,\tau}} - \ell_{a,\tau}\}} e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}},$$

where we used Lemma 4 to bound the size of $\mathcal{C}_{a,\tau}$ and the fact that for any $A \in \mathcal{C}_{a,\tau}$, $f_{\mathcal{S}}(A) \leq f_{\mathcal{S}}(a)$, given the anti-monotonicity property of the frequency.

Finally, we can rewrite the right-hand side of this last equation as

$$\sum_{r=1}^{\chi_a} \sum_{j=1}^{g_{a,r}} 2^{\min\{r, h_{a,r} - j\}} e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}} \ .$$

By combining these equations we have that $w(s) \leq \tilde{w}(s)$ for any $s \in \mathbb{R}^+$, and the thesis follows from Thm. 3. □

We are now ready to formally state our stopping condition that guarantees that an $(\varepsilon, \delta)$-approximation can be computed when the condition is satisfied.

THEOREM 4 (STOPPING CONDITION). *Let $i$ be the minimum index for which it holds that*

$$2\tilde{w}(s^*) + \sqrt{\frac{2\ln(2/\delta)}{|\mathcal{S}_i|}} \leq \frac{\varepsilon}{2} \ . \quad (5)$$

*Then $\mathsf{FI}(\mathcal{S}_i, \mathcal{I}, \theta - \varepsilon/2)$ is an $(\varepsilon, \delta)$-approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$.*

PROOF. The proof follows by combining Lemma 1, Thm. 1, Thm. 3, and Lemma 5. □

## 4.4 Computing the bound

We now discuss how it is possible to check the stopping condition with a single scan of the sample. In particular, it is possible to obtain the expression for $\tilde{w}$ with a single scan, then its minimum of $\tilde{w}$ can be found by computing the value $s^*$ which minimizes $\tilde{w}$.

***Computing $\tilde{w}$.*** To compute the expression for $\tilde{w}$ we only need the quantities $g_{a,k}$ and $h_{a,r}$ for any $a \in \mathcal{I}_{\mathcal{S}}$ and for all $r$, $1 \leq r \leq \chi_a$. These can be computed with a single scan of the sample. Indeed, the order $<_{\mathrm{i}}$ can be obtained from the frequencies of the items in the sample, which we assumed to have been computed during the sample creation. Then, it is sufficient to look at each transaction $\tau$ once, sort its items according to the order $<_{\mathrm{i}}$ and, for any item $a \in \tau$, increment $g_{a,k_{a,\tau}}$ by one and increase by one all counters $h_{a,r}$ for $1 \leq r \leq k_{a,\tau}$.

***Minimizing $\tilde{w}$.*** The function $\tilde{w}$ has first and second derivatives w.r.t. $s$ everywhere in $\mathbb{R}^+$ and it is *convex*, so it has a *global* minimum which can be found efficiently using a non-linear optimization solver like NLopt [15].

Algorithm 1 presents the pseudocode of our progressive sampling algorithm to compute an $(\varepsilon, \delta)$-approximation to $\mathsf{FI}(\mathcal{D}, \mathcal{I}, \theta)$. The function `random_sample`$(\mathcal{D}, m)$ returns $m$ transactions sampled at random with replacement from $\mathcal{D}$.

## 4.5 Top-$k$ Frequent Itemsets

Only minor modifications are needed to obtain an algorithm for computing $(\varepsilon, \delta)$-approximations to the set of top-$k$ FIs. The main differences from the algorithm presented in the previous section are: *1.* a stricter stopping condition; and *2.* the need to run an exact mining algorithm on the final sample *twice*, one to find the top-$k$-th highest frequency $f_{\mathcal{S}}^{(k)}$ in the sample and one to extract the approximation at a lowered frequency threshold that depends on $f_{\mathcal{S}}^{(k)}$.

THEOREM 5. *Let $i$ be the minimum index for which it holds that*

$$2\tilde{w}(s^*) + \sqrt{\frac{2\ln(2/\delta)}{|\mathcal{S}_i|}} \leq \frac{\varepsilon}{4},$$

*and let $f_{\mathcal{S}_i}^{(k)}$ be the frequency in $\mathcal{S}_i$ of the $k$-th most frequent itemset in $\mathcal{S}_i$. Then $\mathsf{FI}(\mathcal{S}_i, \mathcal{I}, f_{\mathcal{S}_i}^{(k)} - \varepsilon/2)$ is an $(\varepsilon, \delta)$-approximation to $\mathsf{TOPK}(\mathcal{D}, \mathcal{I}, k)$.*

The proof leverages on Thm. 4, following the same steps as the proof for [24, Lemma 5.3].

**Algorithm 1:** Progressive sampling algorithm

**input** : a dataset $\mathcal{D}$ built on alphabet $\mathcal{I}$, parameters $\theta, \varepsilon, \delta \in (0,1)$, a sampling schedule $(|\mathcal{S}_i|)_{i\geq 1}$.
**output**: An $(\varepsilon,\delta)$-approximation to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$
$i \leftarrow 0$, $\mathcal{S}_0 \leftarrow \emptyset$, $|\mathcal{S}_0| \leftarrow 0$
**repeat**
  $i \leftarrow i+1$
  $\mathcal{S}^* \leftarrow$ `random_sample` $(\mathcal{D}, |\mathcal{S}_i| - |\mathcal{S}_{i-1}|)$
  $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \cup \mathcal{S}^*$
  //We assume that the frequencies of the items have been computed while creating the sample.
  $g_{a,r} \leftarrow 0, \forall a \in \mathcal{I}_{\mathcal{S}_i}, r \in \mathbb{N}$
  $h_{a,r} \leftarrow 0, \forall a \in \mathcal{I}_{\mathcal{S}_i}, r \in \mathbb{N}$
  **for** $\tau \in \mathcal{S}_i$ **do**
    **for** $a \in \tau$ **do**
      $k_{a,\tau} \leftarrow$ number of items in $\tau$ that come after $a$ in the order $<_a$
      $g_{a,k_{a,\tau}} \leftarrow g_{a,k_{a,\tau}} + 1$
      **for** $j \leftarrow 1,\ldots,k_{a,\tau}$ **do**
        $h_{a,j} \leftarrow h_{a,j} + 1$
      **end**
    **end**
    $\chi_a \leftarrow \max\{r \ : \ h_{a,r} > 0\}$
  **end**
  //In the following expression, $s$ is a symbol.
  $\tilde{w}(s) \leftarrow$
  $$\ln \sum_{a\in\mathcal{I}_{\mathcal{S}}} \left( \left( 1 + \sum_{r=1}^{\chi_a} \sum_{j=1}^{g_{a,r}} 2^{\min\{r,h_{a,r}-j\}} \right) e^{\frac{s^2 f_{\mathcal{S}}(a)}{2n}} \right)$$
  $s^* \leftarrow \arg\min_{s\in\mathbb{R}^+} \tilde{w}(s)$
  $\eta \leftarrow 2\tilde{w}(s^*) + \sqrt{\frac{2\ln(2/\delta)}{|\mathcal{S}_i|}}$
**until** $\eta \leq \varepsilon/2$
**return** $\mathsf{FI}(\mathcal{S}_i,\mathcal{I},\theta-\varepsilon/2)$

## 4.6 Selecting the sampling schedule

Any non-decreasing sequence $(|\mathcal{S}_i|)_{i\geq 1}$ can act as a sampling schedule and Provost et al. [21] showed that a *geometric sampling schedule* (i.e., a schedule where $S_i = \alpha^i S_0$ for some constant $\alpha$) is asymptotically optimal when checking the stopping condition takes time $O(|\mathcal{S}_i|)$. Nevertheless, even such a schedule requires the user to specify two parameters: an initial sample size $S_0$, and a "growth rate" $\alpha > 1$.

In our case it is possible to avoid forcing the choices of these parameters to the user, and instead allow the algorithm to select an initial sample size and then choose successive sample sizes based on the quality of the current sample. This has the net result of removing two parameters from the algorithm.

*Choosing the initial sample size.* We ask whether it is possible to choose the initial sample size wisely so that the algorithm does not waste time in creating and analyzing samples that are just too small for the stopping condition to be satisfied (exceeding in the other direction, i.e., having an initial sample size that is a bit too large, is not a significant issue). In our case it is possible to compute the "necessary" initial sample size $S_0^*$, i.e., the minimum sample size which makes it possible for the stopping condition to be satisfied. In other words, for sample size smaller than $S_0^*$ it is deterministically impossible that the stopping condition is satisfied, and therefore it is useless to create and analyze samples smaller than $S_0^*$.

LEMMA 6. *Let*

$$S_0^* = \frac{8\ln(2/\delta)}{\varepsilon^2} \tag{6}$$

*The stopping condition* (5) *from Thm. 4 can not be satisfied on samples with size smaller than* $S_0^*$.

PROOF. Assume that there exists a sample $\mathcal{S}$ of size smaller than $S_0^*$ for which the stopping condition (5) in Thm. 4 can be satisfied. For such a sample, we have

$$\sqrt{\frac{2\ln(2/\delta)}{|\mathcal{S}|}} > \frac{\varepsilon}{2} \ .$$

From this and the fact that $\tilde{w}(s) \geq 0$, we have that the stopping condition can not be satisfied, so we reached a contradiction and the thesis holds. □

*Computing the size of the next sample.* We can exploit the information obtained from mining the current sample to compute a meaningful sample size for the next iteration.

Assume to be at iteration $i \geq 0$ of the algorithm, and let $\eta_i$ be the value of the l.h.s. of (5) at the end of the current iteration, and let $|\mathcal{S}_i|$ be the size of the sample used in iteration $i$. Then at the next iteration $i+1$ we use a sample of size $|\mathcal{S}_{i+1}|$, with

$$|\mathcal{S}_{i+1}| = \left(\frac{2\eta_i}{\varepsilon}\right)^2 |\mathcal{S}_i| \ . \tag{7}$$

The intuition behind the above formula is that $\eta_i$ is, through Thm. 1, an upper bound to the maximum deviation between the frequency in $\mathcal{S}_i$ of any itemset and the frequency of that itemset in the original dataset. There is a necessary quadratic dependency between this measure and the sample size [17], hence we can use $\varepsilon_i$ and $|\mathcal{S}_i|$ to compute a sample size $|\mathcal{S}_{i+1}|$ for which, *everything else unvaried*, the error allowed in a sample of that size (i.e., the l.h.s. of (5)) would be at most $\varepsilon/2$, as required by the stopping condition of our algorithm.

Although the method we just presented does not give any guarantee on the optimality of the schedule, our experimental evaluation results (Sect. 5) show that is highly effective and results in a faster execution of the algorithm than using a geometric sample schedule, thanks to the fact that intermediate sample sizes that are probably not sufficient for computing an $(\varepsilon,\delta)$-approximation are skipped.

## 4.7 Discussion

To the best of our knowledge, our algorithm improves over all progressive-sampling approaches previously presented in the literature [7, 8, 14, 18, 20, 21, 26] because it does not require the execution of any expensive Frequent Itemsets mining algorithm on each sample to check the stopping condition. Indeed the computation of the stopping condition only requires one scan of the sample. More straightforward stopping conditions with the same requirements are possible: we explored a number of them, both empirically and theoretically, and found them substantially looser (i.e., satisfied only at larger sample size) than the one presented in

this work. We plan to include a presentation of these alternative sub-par stopping conditions, and the comparison of their performances with the one from our algorithm in the extended version of this work. It is indeed necessary to strike a delicate balance between the speed of checking the stopping condition and its strictness (i.e., how early it becomes satisfied), otherwise the advantages of using sampling rather than analyzing the entire dataset are lost.

As the stopping condition does not depend in any way on $\theta$, this parameter can be fixed at a later stage. This is again a consequence of the fact that we do not need to run a mining algorithm on the sample to check the stopping condition.

We remark that the dependency on $1/\varepsilon^2$ of the sample size can not be improved, as shown by Liberty et al. [17].

## 4.8 Static-sampling variant

A variant of the approach presented in previous sections can be used in a static-sampling fashion. Consider the following scenario: rather than having access to the entire dataset $\mathcal{D}$ and being able to sample from it as much as desired, we are given a single random sample $\mathcal{S}$ of the dataset of some size $n$, a fixed parameter $\delta \in (0,1)$, and a minimum frequency threshold $\theta \in (0,1]$. The task requires to compute an $(\varepsilon,\delta)$-approximation to $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ for the best possible $\varepsilon$. No access to the dataset is possible and no other information about the dataset is available. This scenario is realistic and actually common, as it may be easy to create (and maintain) one single random sample of the dataset of a specific size while the dataset is created, while obtaining access to the entire dataset may not be feasible. Previous approaches like those presented by Riondato and Upfal [24] and Chakaravarthy et al. [6] rely on characteristic quantities *of the dataset* (e.g., the d-index [24], or the longest transaction in the dataset [6]) to compute a single sample size that allows to obtain the desired quality guarantees. Computing such quantities require scanning the *entire* dataset. Not only this may be extremely expensive for modern datasets, but it is not even possible in the scenario we just described. These approaches would then be useless in this scenario, as they have no information on the characteristic quantities they need. On the other hand, our approach only uses *sample-dependent quantities* (namely, the distribution of the sample frequencies of single items and related quantities), and can therefore compute the best (i.e., smallest) $\varepsilon$ obtainable from the given sample $\mathcal{S}$. Indeed, it follows from Thm. 4 that such value is

$$\varepsilon = 2\left(2\tilde{w}(s^*) + \sqrt{\frac{2\ln(2/\delta)}{|\mathcal{S}_i|}}\right) \quad . \tag{8}$$

Similarly, to approximate $\mathsf{TOPK}(\mathcal{D},\mathcal{I},k)$: from Thm. 5 we get

$$\varepsilon = 4\left(2\tilde{w}(s^*) + \sqrt{\frac{2\ln(2/\delta)}{|\mathcal{S}_i|}}\right) \quad .$$

Even if we relax the scenario and assume that the algorithm by Riondato and Upfal [24] (currently the best available for static sampling) has knowledge of an upper bound to the d-index of the dataset, the results of our experimental evaluation (Sect. 5) show that the value for $\varepsilon$ computed by our approach using (8) is consistently (although not always) better (i.e., smaller) than the one computed by the algorithm in [24].

Riondato et al. [22] presented PARMA, a MapReduce algorithm for mining approximate collections of FIs. The variant presented in this section can be used in PARMA to obtain even higher-quality approximation or even smaller samples.

# 5. EXPERIMENTAL EVALUATION

We evaluate the performances of our algorithm by assessing the accuracy of the returned collection of FIs and by evaluating the algorithm runtime, comparing it with the time needed to extract the exact collection of FIs and to extract an approximate collection with the same guarantees using the algorithm by Riondato and Upfal [24] (from now on denoted as VC, as it is based on VC-dimension). We choose to compare to this static sampling approach rather than to other existing progressive sampling approaches due to the fact that no existing progressive sampling approach offers the same guarantees of our algorithm. Due to space limitations, we only report here a subset of the results. More results are available in the Appendix of the extended version [23].

*Implementation, datasets, and parameters.* We implemented our algorithm in C++11 and used the C implementation by Grahne and Zhu [11] for the mining step. Our implementation is publicly available at `http://cs.brown.edu/~matteo/radeprogrfi.tar.bz2`. We use NLopt [15] to compute the minimum of $\tilde{w}$ for the stopping condition (Thm. 4). We run the experiments on a machine with a quad-core AMD Phenom$^{\text{TM}}$ II X4 955 processor and 16GB of RAM, running GNU/Linux 3.2.0. We used datasets from the FIMI'03 repository (`http://fimi.ua.ac.be/data/`) [10]. The characteristics of the datasets are reported in Table 1. Each dataset is replicated a number of times (between 200 and 1000) w.r.t. its FIMI'03 version, so that its size is representative of modern datasets and the real-life distributions of the frequencies of the itemsets and of the transaction length are preserved. The d-bound $d$ is a quantity used by VC to compute the sample size $n$ as

$$n = \frac{4}{\varepsilon^2}\left(d + \ln\frac{1}{\delta}\right) \quad .$$

It is, informally, the maximum index $d$ for which the dataset contains at least $d$ different transactions of length at least $d$ [24, Sect. 4.1], and can be computed with a scan of the whole dataset.

| Name | Repl. factor | Size ($\lvert\mathcal{D}\rvert$) | $\lvert\mathcal{I}\rvert$ | d-bound [24] |
|---|---|---|---|---|
| accidents | 200 | 68036601 | 468 | 46 |
| connect | 1000 | 67557000 | 129 | 43 |
| BMS-POS | 200 | 103119400 | 1657 | 81 |
| kosarak | 200 | 1980001400 | 41270 | 443 |
| pumsb_star | 1000 | 49046000 | 2088 | 59 |
| retail | 400 | 35264804 | 16470 | 58 |

Table 1: Dataset characteristics

In all our experiments we fixed $\delta = 0.1$. The initial sample sizes are computed according to (6). Except when otherwise specified, we used the "automatic" sampling schedule described in Sect. 4.6, i.e., we used (7) to compute the size of the sample to analyze at the next iteration. The value for $\varepsilon$ ranged in the set $\{0.01, 0.012, 0.015, 0.017, 0.02\}$. We

run our algorithm five times for each combination of parameters, in order to evaluate fluctuations in accuracy and running time due to the randomized nature of our algorithm. Unless otherwise specified, the reported quantities are the averages over the runs.

*Accuracy.* We evaluate the accuracy of our algorithm in terms of the recall, precision, and error in frequency estimation for the output collection.

**Recall.** The first result of our experimental evaluation is that in *all* the hundreds of runs of our algorithm, for all datasets and combinations of parameters, the returned collection of itemsets *always* has the three properties from Def. 1, not just with probability $1-\delta$. This holds in particular for the first property (all itemsets in $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ are in the output collection), which means that the *recall* of our algorithm is always 100%.

**Precision.** As for the *precision* of our algorithm, i.e., the ratio between the size of $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ and the output size, we remark that our algorithm gives no guarantee in this sense, as any itemsets with frequency in $[\theta-\varepsilon,\theta)$ may be in the output collection. Hence the precision depends on the distribution of the dataset frequencies in this interval. In our experiments, it varied between 15% and 92%, depending on the parameters and on the dataset. We also measure the fraction of the itemsets with frequencies in $[\theta-\varepsilon,\theta)$ that are included in the output. This quantity ranges from 49% to a vanishingly small quantity when $\varepsilon \geq \theta$ (indeed in this case any itemset appearing in the dataset may be included in the output). We report the behavior of this quantity for various values of $\varepsilon$ in Figure 1, for the connect dataset at $\theta = 0.72$. In this figure we report the size of $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ ("FI" line), the number of possible "False Positives" (i.e., the number of itemsets with frequencies in $[\theta-\varepsilon,\theta)$ in the dataset), the average number of FP in the output collection, and the ratio between this latter quantity to the former (aligned to the right vertical axis). We can see that the number of included FP grows slower than the number of possible FP, and therefore the ratio goes down. These False Positives are the price to pay when mining a sample of the dataset, and, by setting $\varepsilon$, the user understands that such False Positives are possible. In any case, our algorithm still returns a relatively compact collection of itemsets, rather than including any itemset that could theoretically be included (i.e., all the itemsets with frequencies in $[\theta-\varepsilon,\theta)$). Indeed the collection can still be used, in all cases, as a set of candidates from which to compute efficiently the exact collection of FIs with a single linear scan of the dataset. The cost of this operation is almost always negligible. We remark once more that no itemset with frequency less than $\theta-\varepsilon$ was ever included in the output nor any itemset with frequency at least $\theta$ was ever missed.

**Frequency Estimation.** For every itemset $A$ in the output collection, we measure the absolute frequency error $\mathsf{err}_{\mathcal{S}}(A) = |f_{\mathcal{S}}(A) - f_{\mathcal{D}}(A)|$, where $\mathcal{S}$ is the last sample analyzed. The third property from Def. 1 requires $\mathsf{err}_{\mathcal{S}}(A)$ to be at most $\varepsilon/2$. Figure 2 shows the behavior of $\mathsf{err}_{\mathcal{S}}$ on the retail dataset, with $\theta = 0.015$. The behavior for other datasets and combination of parameters was similar and can be found in the extended online version [23]. We can see that the absolute error is almost an order of magnitude less than $\varepsilon/2$, both for the average and for the maximum error, and that it is very concentrated around the average. These low numbers

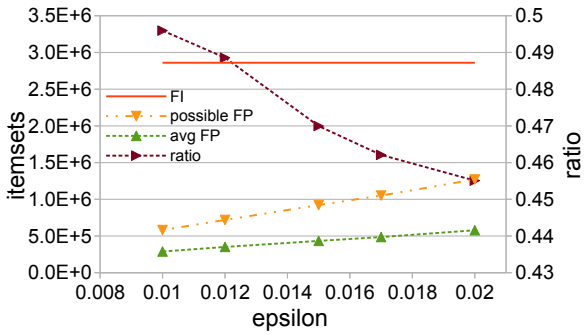


Figure 1: Precision for connect, $\theta = 0.72$.

are not due to the fact that many itemsets in the output collection have a low frequency in the dataset: we also measure the *relative* frequency error, defined as $100\mathsf{err}_{\mathcal{S}}(A)/f_{\mathcal{D}}(A)$, and we report it in Figure 2, aligned to the right vertical axis. As we can see, this quantity was always less than 1.4%. In the future, we plan to develop an algorithm that gives guarantee on the relative frequency error, rather than on the absolute error.

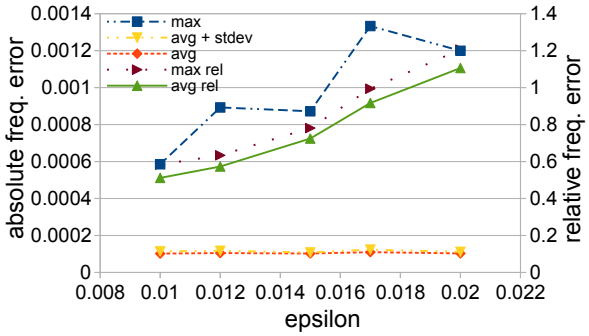


Figure 2: Frequency error for retail, $\theta = 0.015$.

**Discussion.** The results of the accuracy experiments allow us to state that the algorithm performs in practice even better than what the theory guarantees. This is due to the fact that the theoretical analysis uses upper bounds that are developed for the worst case which almost never corresponds to naturally-arising datasets. The results also suggest that there is room for further improvements in the derivation of these bounds and their use in pattern mining.

*Runtime.* The main motivation of our work is that a FI mining algorithm based on progressive can be faster than one based on static sampling as it avoids the need to compute (or assume as known) characteristic quantities of the dataset which would require access to the entire dataset, and it can use properties of the sample to stop at smaller sample sizes. We compare the running time of our algorithm to that of VC, to that of an exact algorithm for mining $\mathsf{FI}(\mathcal{D},\mathcal{I},\theta)$ from the whole dataset [11], and to the running time of our algorithm using a geometric sampling schedule $|\mathcal{S}_i| = \alpha^i|\mathcal{S}_0|$ for different values of $\alpha$ (in these cases, the initial sample size $|\mathcal{S}_0|$ was still computed using (6) as in all our experiments). The results are reported for BMS-POS, $\theta = 0.015$ in Figure 3. Results for other datasets are sim-

ilar and are not reported due to space limitations, but can be found in the extended version [23]. From the plot, it is possible to appreciate that our algorithm vastly outperforms the exact algorithm and also VC. While the first fact should be expected, the latter is due to VC having to scan the dataset in order to compute the d-bound, which can be relative expensive compared to our algorithm which needs no such computation. Moreover, as we discuss later, the sample size computed by VC is in most cases larger than the final sample size used by our algorithm. We also report the running time for our algorithm using a geometric sample schedule with different values (2.0, 2.5, 3.0) for the scaling parameter $\alpha$. This allows us to evaluate the performances of the "automatic" sampling schedule described in Sect. 4.6. We can see that the automatic sampling schedule is more efficient as it allows our algorithm to run faster than with a geometric sample schedule by avoiding the creation and analysis of samples whose size is probably not sufficient for the stopping condition to be satisfied, based on information obtained from the current sample. In almost all the runs of our algorithm, for all combinations of parameters and datasets, our algorithm stops after only two iterations (the only exception (3 iterations) happens for larger values of $\epsilon$ on the kosarak dataset). This means that the information obtained at the minimum reasonable sample size (as computed by (6)) is extremely useful to compute a sufficient sample size using (7). Instead, the runs using the geometric sample schedule stops after a variable number of iterations, which was not possible to predict in advance, and does not behave monotonically, as can be seen from Fig. 3. Hence, the use of the automatic sampling schedule is highly recommended, as it allows faster or comparable execution times and the removal of the parameter $\alpha$), whose impact on the algorithm performances may not be clear a priori to the user.

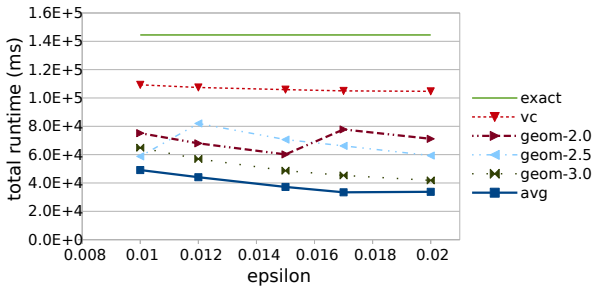


Figure 3: Running time for BMS-POS, $\theta = 0.015$.

We also analyze the breakdown of the runtime of our algorithm, splitting it between time needed to sample the transaction, time needed to evaluate the stopping condition, and time needed to perform the mining of the sample after the stopping condition is satisfied. The results are reported for the pumsb_star dataset, $\theta = 0.32$ in Figure 4. We can see that the runtime decreases as $\varepsilon$ grows. This is due to the sampling time and the mining time decreasing because the algorithm stops at smaller samples for larger values of $\varepsilon$. The fact that the mining time decreases as $\varepsilon$ increases is particularly interesting: the lowered frequency threshold $\theta - \varepsilon/2$ at which the final sample is mined is smaller for larger values of $\varepsilon$ and, on a sample of the same size, it would imply longer mining time than for lower values of $\varepsilon$. Instead the time saved due to the smaller sample dominates the impact of the lower threshold. It is also clear that at small values of $\varepsilon$, the sampling time accounts for the majority of the running time. As expected, the sampling time depends quadratically on $1/\varepsilon$ while the time needed to evaluate the stopping condition is almost constant. This suggests that it is indeed important to achieve a delicate balance between the cost of evaluating the stopping condition and the possibility that it is satisfied at smaller sample sizes. This was indeed one of our main guiding principles when designing our stopping condition.

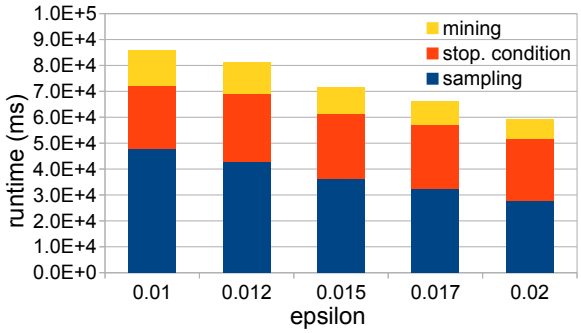


Figure 4: Breakdown of runtime for pumsb_star, $\theta = 0.32$.

*Static sampling.* We also evaluate whether the static-sampling variant presented in Sect. 4.8 could outperform VC. We compared for a given sample size $n$, the value for $\varepsilon$ obtained using (8) on a sample of size $n$, to the value $\varepsilon_{\mathsf{VC}}$ obtained using VC for the same sample size, which is

$$\varepsilon_{\mathsf{VC}} = \sqrt{\frac{d + \ln(1/\delta)}{n}},$$

where $d$ is the *d-bound* of the dataset (values in Table 1). In Figure 5 we show the results for the datasets kosarak and accidents. It is possible to see that $\varepsilon_{\mathsf{VC}}$ is smaller than the one computed by our method at smaller sample size on the accidents dataset, but the $\varepsilon$ computed using (8) decreases faster as $n$ grows and becomes smaller than $\varepsilon_{\mathsf{VC}}$ at larger but reasonable sample sizes. On the other hand, on kosarak our method vastly outperforms VC, with a $\varepsilon$ that is half the one computed by VC. The datasets BMS-POS, pumsb-star, and retail showed results similar to those for kosarak, while the comparison for the dataset connect was similar to that on accidents. Looking at the characteristics of the datasets connect and accidents we noticed that they have a smaller number of items, a smaller d-bound, and more items with very high frequency than the other datasets. Of these characteristics, the last two are intuitively the ones with major impact on the results we see: a low d-bound results in a smaller $\varepsilon_{\mathsf{VC}}$, while high-frequency items will have a high frequency also in the sample, resulting in higher values for $w^*(s)$, which depends on the items frequencies, and therefore in a higher $\varepsilon$. We are currently investigating how to improve our stopping condition in these cases.

We remark again that VC requires access to the *entire* dataset in order to compute $d$, which makes it unusable in some situation, as mentioned in Sect. 4.8. Moreover, computing $d$, as we showed when presenting the runtime results, can be extremely expensive, and the loss in terms of the accuracy parameter $\varepsilon$ may be traded off by the gain in speed.

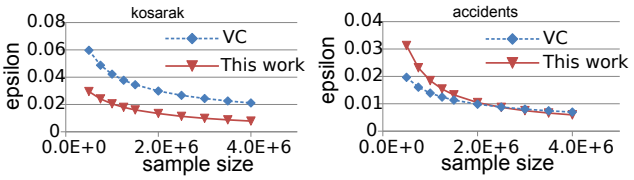


Figure 5: Static sampling evaluation.

The comparison is therefore a slightly unfair to our methods, given that VC is allowed to obtain crucial information by performing additional computation on the entire dataset. For these reasons, we consider our method more flexible and more powerful than VC.

# 6. CONCLUSIONS

We present an algorithm for extracting a high-quality approximation of the collection of FIs with probabilistic guarantees. The algorithm employs progressive sampling with a stopping condition that relies on bounding the conditional Rademacher average of the problem using easy-to-compute characteristic quantities of the sample. The stopping condition can therefore be evaluated very efficiently without the need to perform an expensive in-depth mining of the frequent itemsets in the sample at each step. To our knowledge this is the first work that uses Rademacher averages in a knowledge discovery setting. The experimental results confirm that the algorithm is extremely successful at stopping fast at early iterations, and allows to extract very high-quality approximation of the collection of FIs. Among the possible directions for future work, it would be particularly interesting to better study the trade-off between the computational complexity of the stopping condition and its ability to stop at small sample sizes. We are currently investigating algorithms that give relative/multiplicative approximation guarantees, and extensions of our work to additional significance measures different from frequency [25, 26, 30].

# 7. ACKNOWLEDGMENTS

This work was supported by NSF grant IIS-1247581 and NIH grant R01-CA180776.

# 8. REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. VLDB '94.
[2] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22:207–216, June 1993.
[3] B. Gu, B. Liu, F. Hu, and H. Liu. Efficiently determining the starting sample size for progressive sampling. ECML'01.
[4] P. L. Bartlett, S. Boucheron, and G. Lugosi. Model selection and error estimation. *Mach. Learn.*, 48:85–113, 2002.
[5] S. Boucheron, O. Bousquet, and G. Lugosi. Theory of classification : A survey of some recent advances. *ESAIM: Probability and Statistics*, 9:323–375, 2005.
[6] V. T. Chakaravarthy, V. Pandit, and Y. Sabharwal. Analysis of sampling techniques for association rule mining. ICDT'09.
[7] B. Chen, P. Haas, and P. Scheuermann. A new two-phase sampling based algorithm for discovering association rules. KDD'02.
[8] K.-T. Chuang, M.-S. Chen, and W.-C. Yang. Progressive sampling for association rules based on sampling error estimation. PAKDD'05.
[9] T. Elomaa and M. Kääriäinen. Progressive Rademacher sampling. AAAI'02.
[10] B. Goethals and M. J. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explor. Newsl.*, 6(1):109–117, June 2004.
[11] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. FIMI'03.
[12] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. SIGMOD'00.
[13] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowl. Disc.* , 15:55–86, 2007.
[14] G. H. John and P. Langley. Static versus dynamic sampling for data mining. KDD'96.
[15] S. G. Johnson. The NLopt nonlinear-optimization package. URL `http://ab-initio.mit.edu/nlopt`.
[16] V. Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Trans. Inf. Theory*, 47(5): 1902–1914, July 2001.
[17] E. Liberty, M. Mitzenmacher, J. Thaler, and J. Ullman. Space lower bounds for itemset frequency sketches. *CoRR*, 1407.3740, July 2014.
[18] S. Parthasarathy. Efficient progressive sampling for association rules. ICDM'02.
[19] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. ICDT'99.
[20] A. Pietracaprina, M. Riondato, E. Upfal, and F. Vandin. Mining top-$k$ frequent itemsets through progressive sampling. *Data Mining Knowl. Disc.*, 21(2): 310–326, 2010.
[21] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. KDD'99.
[22] M. Riondato, J. A. DeBrabant, R. Fonseca, and E. Upfal. PARMA: A parallel randomized algorithm for association rules mining in MapReduce. CIKM'12.
[23] M. Riondato and E. Upfal. Mining frequent itemsets through progressive sampling with Rademacher averages. Extended Version. URL `http://cs.brown.edu/%7Ematteo/papers/progrsamplfi-ext.pdf`.
[24] M. Riondato and E. Upfal. Efficient discovery of association rules and frequent itemsets through sampling with tight performance guarantees. *ACM Trans. Knowl. Disc. from Data*, 8(2), 2014.
[25] M. Riondato and F. Vandin. Finding the true frequent itemsets. SDM'14.
[26] T. Scheffer and S. Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *J. Mach. Learn. Res.*, 3:833–862, Dec. 2002.
[27] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.
[28] H. Toivonen. Sampling large databases for association rules. VLDB'96.
[29] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for engineering and information science. Springer-Verlag, New York, NY, USA, 1999.
[30] G. I. Webb. Discovering significant patterns. *Mach. Learn.*, 68(1):1–33, 2007.