MCRapper: Monte-Carlo Rademacher Averages for Poset Families and Approximate Pattern Mining

Leonardo Pellegrina
Dept. of Information Engineering
Università di Padova
Padova, Italy
pellegri@dei.unipd.it

Cyrus Cousins
Dept. of Computer Science
Brown University
Providence, RI, USA
ccousins@cs.brown.edu

Fabio Vandin
Dept. of Information Engineering
Università di Padova
Padova, Italy
fabio.vandin@unipd.it

Matteo Riondato
Dept. of Computer Science
Amherst College
Amherst, MA, USA
mriondato@amherst.edu

"I'm an MC still as honest" – Eminem, *Rap God*

ABSTRACT

We present MCRapper, an algorithm for efficient computation of Monte-Carlo Empirical Rademacher Averages (MCERA) for families of functions exhibiting poset (e.g., lattice) structure, such as those that arise in many pattern mining tasks. The MCERA allows us to compute upper bounds to the maximum deviation of sample means from their expectations, thus it can be used to find both statistically-significant functions (i.e., patterns) when the available data is seen as a sample from an unknown distribution, and approximations of collections of high-expectation functions (e.g., frequent patterns) when the available data is a small sample from a large dataset. This feature is a strong improvement over previously proposed solutions that could only achieve one of the two. MCRapper uses upper bounds to the discrepancy of the functions to efficiently explore and prune the search space, a technique borrowed from pattern mining itself. To show the practical use of MCRapper, we employ it to develop an algorithm TFP-R for the task of True Frequent Pattern (TFP) mining. TFP-R gives guarantees on the probability of including any false positives (precision) and exhibits higher statistical power (recall) than existing methods offering the same guarantees. We evaluate MCRapper and TFP-R and show that they outperform the state-of-the-art for their respective tasks.

CCS CONCEPTS

• **Information systems → Data mining**; • **Mathematics of computing → Probabilistic algorithms**; • **Theory of computation → Sketching and sampling**.

ACM Reference Format:
Leonardo Pellegrina, Cyrus Cousins, Fabio Vandin, and Matteo Riondato. 2020. MCRapper: Monte-Carlo Rademacher Averages for Poset Families and Approximate Pattern Mining. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '20), August 23–27, 2020, Virtual Event, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3394486.3403267

KDD '20, August 23–27, 2020, Virtual Event, CA, USA

1 INTRODUCTION

Pattern mining is a key sub-area of knowledge discovery from data, with a large number of variants (from itemsets mining [1] to subgroup discovery [13], to sequential patterns [2], to graphlets [3]) tailored to applications ranging from market basket analysis to spam detection to recommendation systems. Ingenuous algorithms have been proposed over the years, and pattern mining is both extremely used in practice and a very vibrant area of research.

In this work we are interested in the analysis of *samples* for pattern mining. There are two meanings of "sample" in this context, but, as we now argue, they are really two sides of the same coin, and our methods work for both sides.

The first meaning is *sample* as a *small random sample of a large dataset*: since mining patterns becomes more expensive as the dataset grows, it is reasonable to mine only a small random sample that fits into the main memory of the machine. Recently, this meaning of sample as "sample-of-the-dataset" has been used also to enable interactive data exploration using progressive algorithms for pattern mining [22]. The patterns obtained from the sample are an *approximation* of the exact collection, due to the noise introduced by the sampling process. To obtain desirable probabilistic guarantees on the quality of the approximation, one must study the *trade-off between the size of the sample and the quality of the approximation*. Many works have progressively obtained better characterizations of the trade-off using advanced probabilistic concepts [7, 17, 18, 20, 22, 26]. Recent methods [17, 18, 20, 22] use VC-dimension, pseudodimension, and Rademacher averages [4, 14], key concepts from statistical learning theory [28] (see also Sect. 2 and Sect. 3.2), because they allow to obtain uniform (i.e., simultaneous) probabilistic guarantees on the deviations of all sample means (e.g., sample frequencies, or other measure of interestingness, of all patterns) from their expectations (the exact interestingness of the patterns in the dataset).

The second meaning is *sample* as a *sample from an unknown data generating distribution*: the whole dataset is seen as a collection of samples from an unknown distribution, and the goal of mining patterns from the available dataset is to gain approximate information (or better, discover knowledge) about the distribution. This area is known as *statistically-sound pattern discovery* [11], and there are many different flavors of it, from significant pattern mining [25] from transactional datasets [12, 15], sequences [27], or graphs [24], to true frequent itemset mining [19], to, at least in part, contrast

pattern mining [5]. Many works in this area also use concepts from statistical learning theory such as empirical VC-dimension [19] or Rademacher averages [15], because, once again, these concepts allow to get very sharp bounds on the maximum difference between the observed interestingness on the sample and the unknown interestingness according to the distribution.

The two meanings of "sample" are really two sides of the same coin, because also in the first case the goal is to approximate an unknown distribution from a sample, thus falling back into the second case. Despite this similarity, previous contributions have been extremely point-of-view-specific and pattern-specific. In part, these limitations are due to the techniques used to study the trade-off between sample size and quality of the approximation obtained from the sample. Our work instead proposes a *unifying solution* for mining approximate collections of patterns from samples, while giving guarantees on the quality of the approximation: our proposed method can easily be adapted to approximate collections of frequent itemsets, frequent sequences, true frequent patterns, significant patterns, and many other tasks, even outside of pattern mining.

At the core of our approach is the *n-Samples Monte-Carlo (Empirical) Rademacher Average (n-MCERA)* [4] (see (4)), which has the flexibility and the power needed to achieve our goals, as it gives much sharper bounds to the deviation than other approaches. The challenge in using the n-MCERA, like other quantities from statistical learning theory, is how to compute it efficiently.

Contributions. We present MCRapper, an algorithm for the fast computation of the n-MCERA of families of functions with a poset structure, which often arise in pattern mining tasks (Sect. 3.1).

- MCRapper is the first algorithm to compute the n-MCERA efficiently. It achieves this goal by using sharp upper bounds to the discrepancy of each function in the family (Sect. 4.1) to quickly prune large parts of the function search space during the exploration necessary to compute the n-MCERA, in a branch-and-bound fashion. We also develop a novel sharper upper bound to the supremum deviation using the 1-MCERA (Thm. 4.6). It holds for any family of functions, and is of independent interest.
- To showcase the practical strength of MCRapper, we develop TFP-R (Sect. 5), a novel algorithm for the extraction of the True Frequent Patterns (TFP) [19]. TFP-R gives probabilistic guarantees on the quality of its output: with probability at least $1 - \delta$ (over the choice of the sample and the randomness used in the algorithm), for user-supplied $\delta \in (0, 1)$, the output is guaranteed to not contain any false positives. That is, TFP-R controls the Family-Wise Error Rate (FWER) at level δ while achieving high statistical power, thanks to the use of the n-MCERA and of novel *variance-aware* tail bounds (Thm. 3.2). We also discuss other applications of MCRapper, to remark on its flexibility as a general-purpose algorithm.
- We conduct an extensive experimental evaluation of MCRapper and TFP-R on real datasets (Sect. 6), and compare their performance with that of state-of-the-art algorithms for their respective tasks. MCRapper, thanks to the n-MCERA, computes much sharper (i.e, lower) upper bounds to the supremum deviation than algorithms using the looser Massart's lemma [23, Lemma 26.8]. TFP-R extracts many more TFPs (i.e., has higher statistical power) than existing algorithms with the same guarantees.

2 RELATED WORK

Our work applies to both the "small-random-sample-from-large-dataset" and the "dataset-as-a-sample" settings, so we now discuss the relationship of our work to prior art in both settings. We do not study the important but different task of output sampling in pattern mining [6, 9]. We focus on works that use concepts from statistical learning theory: these are the most related to our work, and most often the state of the art in their areas. More details are available in surveys [11, 17].

The idea of mining a small random sample of a large dataset to speed up the pattern extraction step was proposed for the case of itemsets by Toivonen [26] shortly after the first algorithm for the task had been introduced. The trade-off between the sample size and the quality of the approximation obtained from the sample has been progressively better characterized [7, 17, 18], with large improvements due to the use of concepts from statistical learning theory. Riondato and Upfal [17] study the VC-dimension of the itemsets mining task, which results in a worst-case *dataset-dependent* but *sample- and distribution-agnostic characterization* of the trade-off. The major advantage of using Rademacher averages [14], as we do in MCRapper is that the characterization is now *sample-and-distribution-dependent*, which gives much better upper bounds to the maximum deviation of sample means from their expectations. Rademacher averages were also used by Riondato and Upfal [18], but they used worst-case upper bounds (based on Massart's lemma [23, Lemma 26.2]) to the empirical Rademacher average of the task, resulting in excessively large bounds. MCRapper instead computes the *exact* n-MCERA of the family of interest on the observed sample, without having to consider the worst case. For other kinds of patterns, Riondato and Vandin [20] studied the pseudodimension of subgroups, while Servan-Schreiber et al. [22] and Santoro et al. [21] considered the (empirical) VC-dimension and Rademacher averages for sequential patterns. MCRapper can be applied in all these cases, and obtains better bounds because it uses the *sample-and-distribution-dependent* n-MCERA, rather than a worst case dataset-dependent bound.

Significant pattern mining considers the dataset as a sample from an unknown distribution. Many variants and algorithms are described in the survey by Hämäläinen and Webb [11]. We discuss only the two most related to our work. Riondato and Vandin [19] introduce the problem of finding the true frequent itemsets, i.e., the itemsets that are frequent w.r.t. the unknown distribution. They propose a method based on empirical VC-dimension to compute the frequency threshold to use to obtain a collection of true frequent patterns with no false positives (see also Sect. 5). Our algorithm TFP-R uses the n-MCERA, and as we show in Sect. 6, it greatly outperforms the state-of-the-art (a modified version of the algorithm by Riondato and Upfal [18] for approximate frequent itemsets mining). Pellegrina et al. [15] use empirical Rademacher averages in their work for significant pattern mining. As their work uses the bound by Riondato and Upfal [18], the same comments about the n-MCERA being a superior approach hold.

Our approach to bounding the supremum deviation by computing the n-MCERA with efficient search space exploration techniques is novel, not just in knowledge discovery, as the n-MCERA has received scant attention. De Stefani and Upfal [8] use it to control the

generalization error in a sequential and adaptive setting, but do not discuss efficient computation. We believe that the lack of attention to the n-MCERA can be be explained by the fact that there were no efficient algorithms for it, a gap now filled by MCRapper.

3 PRELIMINARIES

We now define the most important concepts and results that we use throughout this work. Let $\mathcal{F}$ be a class of real valued functions from a domain $\mathcal{X}$ to the interval $[a, b] \subset \mathbb{R}$. We use c to denote $|b - a|$ and z to denote $\max\{|a|, |b|\}$. In this work, we focus on a specific class of families (see Sect. 3.1). In pattern mining from transactional datasets, $\mathcal{X}$ is the set of all possible transactions (or, e.g., sequences). Let μ be an *unknown* probability distribution over $\mathcal{X}$ and the *sample* $\mathcal{S} = \{s_1, \ldots, s_m\}$ be a bag of m i.i.d. random samples from $\mathcal{X}$ drawn according to μ. We discussed in Sect. 1 how in the pattern mining case, the sample may either be the whole dataset (sampled according to an unknown distribution) or a random sample of a large dataset (more details in Sect. 3.1). For each $f \in \mathcal{F}$, we define its *empirical sample average (or sample mean)* $\hat{\mathbb{E}}_{\mathcal{S}}[f]$ *on* $\mathcal{S}$ and its *expectation* $\mathbb{E}[f]$ respectively as

$$\hat{\underset{\mathcal{S}}{\mathbb{E}}}[f] \doteq \frac{1}{m} \sum_{s_i \in \mathcal{S}} f(s_i) \text{ and } \mathbb{E}[f] \doteq \underset{\mu}{\mathbb{E}} \left[\frac{1}{m} \sum_{s_i \in \mathcal{S}} f(s_i) \right] .$$

In the pattern mining case, the sample mean is the observed interestingness of a pattern, e.g., its frequency (but other measures of interestingness can be modeled as above, as discussed for subgroups by Riondato and Vandin [20]), while the expectation is the unknown exact interestingness that we are interested in approximating, that is, either in the large datasets or w.r.t. the unknown data generating distribution. We are interested in developing tight and fast-to-compute upper bounds to the *supremum deviation (SD)* $\mathsf{D}(\mathcal{F}, \mathcal{S}, \mu)$ *of* $\mathcal{F}$ *on* $\mathcal{S}$ between the empirical sample average and the expectation *simultaneously* for all $f \in \mathcal{F}$, defined as

$$\mathsf{D}(\mathcal{F}, \mathcal{S}, \mu) = \sup_{f \in \mathcal{F}} \left| \hat{\underset{\mathcal{S}}{\mathbb{E}}}[f] - \underset{\mu}{\mathbb{E}}[f] \right| . \tag{1}$$

The supremum deviation allows to quantify how good the estimates obtained from the samples are. Because μ is unknown, it is not possible to compute $\mathsf{D}(\mathcal{F}, \mathcal{S}, \mu)$ exactly. We introduce concepts such as Monte-Carlo Rademacher Average and results to compute such bounds in Sect. 3.2, but first we elaborate on the specific class of families that we are interested in.

3.1 Poset families and patterns

A partially-ordered set, or *poset* is a pair $(A, \preceq)$ where A is a set and $\preceq$ is a binary relation between elements of A that is reflexive, antisymmetric, and transitive. Examples of posets include the $A = \mathbb{N}$ and the obvious "less-than-or-equal-to" ($\leq$) relation, and the powerset of a set of elements and the "subset-or-equal" ($\subseteq$) relation. For any element $y \in A$, we call an element $w \in A$, $w \neq y$ a *descendant* of y (and call y an *ancestor* of w) if $y \preceq w$. Additionally, if $y \preceq w$ and there is no $q \in A$, $q \neq y$, $q \neq w$ such that $y \preceq q \preceq w$, then we say that w is a *child* of y and that y is a *parent* of w. For example, the set $\{0, 2\}$ is a parent of the set $\{0, 2, 5\}$ and an ancestor of the set $\{0, 1, 2, 7\}$, when considering A to be all possible subsets of integers and the $\subseteq$ relation.

In this work we are interested in posets where A is a family $\mathcal{F}$ of functions as in Sect. 3.2, and the relation $\preceq$ is the following: for any $f, g \in \mathcal{F}$

$$f \preceq g \text{ iff } \begin{cases} f(x) \geq g(x) & \text{for every } x \in \mathcal{X} \text{ s.t. } f(x) \geq 0 \\ f(x) \leq g(x) & \text{for every } x \in \mathcal{X} \text{ s.t. } f(x) < 0 \end{cases} . \tag{2}$$

The very general but a bit complicated requirement often collapses to much simpler ones as we discuss below. We aim for generality, as our goal is to develop a unifying approach for many pattern mining tasks, for both meanings of "sample", as discussed in Sect. 1. For now, consider for example that requiring $|f(x)| \geq |g(x)|$ for every $x \in \mathcal{X}$ is a specialization of the above more general requirement. We assume to have access to a blackbox function children that, given any function $f \in \mathcal{F}$, returns the list of children of f according to $\preceq$, and to a blackbox function minimals that, given $\mathcal{F}$, returns the *minimal elements w.r.t.* $\preceq$, i.e., all the functions $f \in \mathcal{F}$ without any parents. We refer to families that satisfy these conditions as *poset families*, even if the conditions are more about the relation $\preceq$ than about the family. We now discuss how poset families arise in many pattern mining tasks.

In pattern mining, it is assumed to have a language $\mathcal{L}$ containing the patterns of interest. For example, in itemsets mining [1], $\mathcal{L}$ is the set of all possible *itemsets*, i.e., all non-empty subsets of an alphabet $\mathcal{I}$ of *items*, while in sequential pattern mining [2], $\mathcal{L}$ is the set of sequences, and in subgroup discovery [13], $\mathcal{L}$ is set by the user as the set of patterns of interest. In all these cases, for each pattern $P \in \mathcal{L}$, it is possible to define a function f_P from the domain $\mathcal{X}$, which is the set of all possible *transactions*, i.e., elementary components of the dataset or of the sample, to an appropriate codomain $[a, b]$, such that $f_P(x)$ denotes the "value" of the pattern P on the transaction x. For example, for itemsets mining, $\mathcal{X}$ is all the subsets of $\mathcal{I}$ and f_P maps $\mathcal{X}$ to $\{0, 1\}$ so that $f_P(x) = 1$ iff $P \subseteq x$ and 0 otherwise. A consequence of this definition is that $\hat{\mathbb{E}}_{\mathcal{S}}[f_P]$ is the *frequency* of P in $\mathcal{S}$, i.e., the fraction of transaction of $\mathcal{S}$ that contain the pattern P. A more complex (due to the nature of the patterns) but similar definition would hold for sequential patterns. For the case of *high-utility itemset mining* [10], the value of $f_P(x)$ would be the utility of P in the transaction x. The family $\mathcal{F}$ is the set of the functions f_P for every pattern $P \in \mathcal{L}$. Similar reasoning also applies to patterns on graphs, such as graphlets [3].

Now that we have defined the set that we are interested in, let's comment on the relation $\preceq$ that, together with the set, forms the poset. In the itemsets case, for any two patterns P' and $P'' \in \mathcal{L}$, i.e., for any two functions $f = f_{P'}$ and $g = f_{P''} \in \mathcal{F}$, it holds $f \preceq g$ iff $P' \subseteq P''$. For sequences, the *subsequence relation* $\sqsubseteq$ defines $\preceq$ instead. In all pattern mining tasks, the only minimal element of $\mathcal{F}$ w.r.t. $\preceq$ is the empty itemset (or sequence) $\emptyset$. Our assumption to have access to the blackboxes children and minimals is therefore very reasonable, because computing these collections is extremely straightforward in all the pattern mining cases we just mentioned and many others.

3.2 Rademacher Averages

Here we present Rademacher averages [4, 14] and related results at the core of statistical learning theory [28]. Our presentation uses the most recent and sharper results, and we also introduce new

results (Thm. 3.2, and later Thm. 4.6) that may be of independent interest. For an introduction to statistical learning theory and more details about Rademacher averages, we refer the interested reader to the textbook by Shalev-Shwartz and Ben-David [23]. In this section we consider a generic family $\mathcal{F}$, not necessarily a poset family.

A key quantity to study the supremum deviation (SD) from (1) is the *empirical Rademacher average (ERA)* $\hat{\mathrm{R}}(\mathcal{F}, \mathcal{S})$ *of* $\mathcal{F}$ *on* $\mathcal{S}$ [4, 14], defined as follows. Let $\boldsymbol{\sigma} = \langle \sigma_1, \ldots, \sigma_m \rangle$ be a collection of m i.i.d. Rademacher random variables, i.e., each taking value in $\{-1, 1\}$ with equal probability. The ERA of $\mathcal{F}$ on $\mathcal{S}$ is the quantity

$$\hat{\mathrm{R}}(\mathcal{F}, \mathcal{S}) \doteq \mathop{\mathbb{E}}_{\boldsymbol{\sigma}} \left[\sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{i=1}^{m} \sigma_i f(s_i) \right] . \tag{3}$$

Computing the ERA $\hat{\mathrm{R}}(\mathcal{F}, \mathcal{S})$ exactly is often intractable, due to the *expectation* over 2^m possible assignments for $\boldsymbol{\sigma}$, and the need to compute a *supremum* for each of these assignments, which precludes many standard techniques for computing expectations. Bounds to the SD are then obtained through efficiently-computable *upper bounds* to the ERA. Massart's lemma [23, Lemma 26.2] gives a *deterministic* upper bound to the ERA that is often very loose. Monte-Carlo estimation allows to obtain an often sharper *probabilistic* upper bound to the ERA. For $n \geq 1$, let $\boldsymbol{\sigma} \in \{-1, 1\}^{n \times m}$ be a $n \times m$ matrix of i.i.d. Rademacher random variables. The *n-Samples Monte-Carlo Empirical Rademacher Average (n-MCERA)* $\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma})$ *of* $\mathcal{F}$ *on* $\mathcal{S}$ *using* $\boldsymbol{\sigma}$ is [4]

$$\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}) \doteq \frac{1}{n} \sum_{j=1}^{n} \sup_{f \in \mathcal{F}} \frac{1}{m} \sum_{s_i \in \mathcal{S}} \boldsymbol{\sigma}_{j,i} f(s_i) . \tag{4}$$

The n-MCERA allows to obtain probabilistic upper bounds to the SD as follows (proof in App. A.1). In Sect. 4.3 we show a novel improved bound for the special case $n = 1$ (Thm. 4.6).

Theorem 3.1. *Let* $\eta \in (0, 1)$. *For ease of notation let*

$$\tilde{\mathrm{R}} \doteq \hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}) + 2z\sqrt{\frac{\ln \frac{4}{\eta}}{2nm}} . \tag{5}$$

With probability at least $1 - \eta$ *over the choice of* $\mathcal{S}$ *and* $\boldsymbol{\sigma}$, *it holds*

$$\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu) \leq 2\tilde{\mathrm{R}} + \frac{\sqrt{c(4m\tilde{\mathrm{R}} + c\ln\frac{4}{\eta})\ln\frac{4}{\eta}}}{m} + \frac{c\ln\frac{4}{\eta}}{m} + c\sqrt{\frac{\ln\frac{4}{\eta}}{2m}} . \tag{6}$$

Sharper upper bounds to $\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu)$ can be obtained with the n-MCERA when more information about $\mathcal{F}$ is available. The proof is in App. A.1. We use this result for a specific pattern mining task in Sect. 5.

Theorem 3.2. *Let* v *be an upper bound to the variance of* every *function in* $\mathcal{F}$, *and let* $\eta \in (0, 1)$. *Define the following quantities*

$$\rho \doteq \mathrm{R}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}) + 2z\sqrt{\frac{\ln\frac{4}{\eta}}{2nm}}, \tag{7}$$

$$r \doteq \rho + \frac{1}{2m}\left(\sqrt{c\left(4m\rho + c\ln\frac{4}{\eta}\right)\ln\frac{4}{\eta}} + c\ln\frac{4}{\eta}\right),$$

$$\varepsilon \doteq 2r + \sqrt{\frac{2\ln\frac{4}{\eta}(v + 8cr)}{m}} + \frac{2c\ln\frac{4}{\eta}}{3m} . \tag{8}$$

Then, with probability at least $1 - \eta$ *over the choice of* $\mathcal{S}$ *and* $\boldsymbol{\sigma}$, *it holds*

$$\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu) \leq \varepsilon .$$

Due to the dependency on z in Thms. 3.1 and 3.2, it is often convenient to use $\hat{\mathrm{R}}^n_m(\mathcal{F} - \frac{c}{2}, \mathcal{S}, \boldsymbol{\sigma})$ in place of $\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma})$ in the above theorems, where $\mathcal{F} - \frac{c}{2}$ denotes the *range-centralized* family of functions obtained by shifting every function in $\mathcal{F}$ by $-\frac{c}{2}$. The results still hold for $\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu)$ because the SD is invariant to shifting, but the bounds to the SD usually improve since the corresponding z for the range-centralized family is smaller.

4 MCRAPPER

We now describe and analyze our algorithm MCRapper to efficiently compute the n-MCERA (see (4)) for a family $\mathcal{F}$ with the binary relation $\preceq$ defined in (2) and the blackbox functions children and minimals described in Sect. 3.1.

4.1 Discrepancy bounds

For $j \in \{1, \ldots, n\}$, we denote as the *j-discrepancy* $\Delta_j(f)$ *of* $f \in \mathcal{F}$ *on* $\mathcal{S}$ *w.r.t.* $\boldsymbol{\sigma}$ the quantity

$$\Delta_j(f) \doteq \sum_{s_i \in \mathcal{S}} \boldsymbol{\sigma}_{j,i} f(s_i) .$$

The j-discrepancy is not an *anti-monotonic function*, in the sense that it does not necessarily hold that $\Delta_j(f) \geq \Delta_j(g)$ for every descendant g of $f \in \mathcal{F}$. Clearly, it holds

$$\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}) = \frac{1}{nm} \sum_{j=1}^{n} \sup_{f \in \mathcal{F}} \Delta_j(f) . \tag{9}$$

A naïve computation of the n-MCERA would require enumerating *all* the functions in $\mathcal{F}$ and computing their j-discrepancies, $1 \leq j \leq n$, in order to find each of the n suprema. We now present novel easy-to-compute *upper bounds* $\widetilde{\Psi}(f)$ and $\Psi_j(f)$ to $\Delta_j(f)$ such that $\widetilde{\Psi}(f) \geq \Delta_j(g)$ and $\Psi_j(f) \geq \Delta_j(g)$ for every $g \in \mathrm{d}(f)$, where $\mathrm{d}(f)$ denote the set of the *descendants* of f w.r.t. $\preceq$. This key property (which is a generalization of *anti-monotonicity* to posets) allows us to derive efficient algorithms for computing the n-MCERA *exactly* without enumerating *all* the functions in $\mathcal{F}$. Such algorithms take a branch-and-bound approach using the upper bounds to $\Delta_j(f)$ to prune large portions of the search space (see Sect. 4.2).

For every $j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, m\}$, let

$$\boldsymbol{\sigma}^+_{j,i} \doteq \mathbb{1}(\boldsymbol{\sigma}_{j,i} = 1), \text{ and } \boldsymbol{\sigma}^-_{j,i} \doteq \mathbb{1}(\boldsymbol{\sigma}_{j,i} = -1)$$

and for every $f \in F$ and $x \in \mathcal{X}$, define the functions

$$f^+(x) \doteq f(x)\mathbb{1}(f(x) \geq 0), \text{ and } f^-(x) \doteq f(x)\mathbb{1}(f(x) < 0) .$$

It holds $f^+(x) \geq 0$ and $f^-(x) \leq 0$ for every $f \in \mathcal{F}$ and $x \in \mathcal{X}$. For every $j \in \{1, \ldots, n\}$ and $f \in \mathcal{F}$, define

$$\widetilde{\Psi}(f) \doteq \sum_{s_i \in \mathcal{S}} |f(s_i)| \quad \text{and}$$

$$\Psi_j(f) \doteq \sum_{s_i \in \mathcal{S}} \boldsymbol{\sigma}^+_{j,i} f^+(s_i) - \sum_{s_i \in \mathcal{S}} \boldsymbol{\sigma}^-_{j,i} f^-(s_i) . \tag{10}$$

Computationally, these quantities are extremely straightforward to obtain. Both $\widetilde{\Psi}(f)$ and $\Psi_j(f)$ are upper bounds to $\Delta_j(f)$ and to $\Delta_j(g)$ for all $g \in \mathrm{d}(f)$ (proof in App. A.1).

Theorem 4.1. *For any $f \in \mathcal{F}$ and $j \in \{1, \dots, n\}$, it holds*

$$\max\left\{\Delta_j(g) \ : \ g \in \mathsf{d}(f) \cup \{f\}\right\} \le \Psi_j(f) \le \widetilde{\Psi}(f) \ .$$

The bounds we derived in this section are *deterministic*. An interesting direction for future research is how to obtain sharper *probabilistic* bounds.

4.2 Algorithms

We now use the discrepancy bounds $\widetilde{\Psi}(\cdot)$ and $\Psi.(\cdot)$ from Sect. 4.1 in our algorithm MCRapper for computing the exact n-MCERA. As the real problem is usually not to only compute the n-MCERA but to actually compute an upper bound to the SD, our description of MCRapper includes this final step, this also enables fair comparison with existing algorithms that use *deterministic* bounds to the ERA to compute an upper bound to the SD (see also Sect. 6).

MCRapper offers probabilistic guarantees on the quality of the bound it computes (proof deferred to after the presentation).

Theorem 4.2. *Let $\delta \in (0, 1)$. With probability at least $1 - \delta$ over the choice of $\mathcal{S}$ and of $\boldsymbol{\sigma}$, the value ε returned by* MCRapper *is such that* $\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu) \le \varepsilon$.

The pseudocode of MCRapper is presented in Alg. 1. The division in functions is useful for reusing parts of the algorithm in later sections (e.g., Alg. 3). After having sampled the $n \times m$ matrix of i.i.d. Rademacher random variables (line 1), the algorithm calls the function `getSupDevBound` with appropriate parameters, which in turn calls the function `getNMCERA`, the real heart of the algorithm. This function computes the n-MCERA $\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma})$ by exploring and pruning the search space (i.e., $\mathcal{F}$) in according to the order of the elements in the priority queue Q (line 8). One possibility is to explore the space in Breadth-First-Search order (so Q is just a FIFO queue), while another is to use the upper bound $\widetilde{\Psi}(f)$ as the priority, with the top element in the queue being the one with maximum priority among those in the queue. Other orders are possible, but we assume that the order is such that all parents of a function are explored before the function, which is reasonable to ensure maximum pruning, and is satisfied by the two mentioned orders. We assume that the priority queue also has a method `delete(e)` to delete an element e in the queue. This requirement could be avoided with some additional book-keeping, but it simplifies the presentation of the algorithm.

The algorithm keeps in the quantities v_j, $j \in \{1, \dots, n\}$, the currently best available lower bound to the quantity $\sup_{f \in \mathcal{F}} \Delta_j(f)$ (see (9)), which initially are all $-zm$ (the lowest possible value of a discrepancy). MCRapper also maintains a dictionary $\mathcal{J}$ (line 10), initially empty, whose keys will be elements of $\mathcal{F}$ and the values are subsets of $\{1, \dots, n\}$. The value associated to a key f in the dictionary is a superset of the set of values $j \in \{1, \dots, n\}$ for which $\widetilde{\Psi}(f) \ge v_i$, i.e., for which f or one of its descendants *may* be the function attaining the supremum j-discrepancy among all the functions in $\mathcal{F}$ (see (9)). A function and all its descendants are pruned when this set is the empty set. The set of keys of the dictionary $\mathcal{J}$ is, at all times, the set of all and only the functions in $\mathcal{F}$ that have ever been added to Q. The last data structure is the set H (line 11), initially empty, which will contain pruned elements of $\mathcal{F}$, in order to avoid visiting either them or their descendants.

Algorithm 1: MCRapper

Input: Poset family $\mathcal{F}$, sample $\mathcal{S}$ of size m, $\delta \in (0, 1)$, $n \ge 1$
Output: Upper bound to $\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu)$ with probability $\ge 1 - \delta$.

1 $\boldsymbol{\sigma} \leftarrow$ `draw(`m, n`)`
2 $\varepsilon \leftarrow$ `getSupDevBound(`$\mathcal{F}$, $\mathcal{S}$, δ, $\boldsymbol{\sigma}$`)`
3 **return** ε
4 **Function** `getSupDevBound(`$\mathcal{F}$, $\mathcal{S}$, δ, $\boldsymbol{\sigma}$`)`**:**
5 $\quad \tilde{\mathrm{R}} \leftarrow$ `getNMCERA(`$\mathcal{F}$, $\mathcal{S}$, $\boldsymbol{\sigma}$`)` $+ 2z\sqrt{\frac{\ln(4/\delta)}{2nm}}$
6 $\quad$ **return** *r.h.s. of* (6) *using* $\eta = \delta$
7 **Function** `getNMCERA(`$\mathcal{F}$, $\mathcal{S}$, $\boldsymbol{\sigma}$`)`**:**
8 $\quad Q \leftarrow$ empty priority queue
9 $\quad$ **foreach** $j \in \{1, \dots, n\}$ **do** $v_j \leftarrow -zm$
10 $\quad \mathcal{J} \leftarrow$ empty dictionary from $\mathcal{F}$ to subsets of $\{1, \dots, n\}$
11 $\quad H \leftarrow \emptyset$
12 $\quad$ **foreach** $f \in$ `minimals(`$\mathcal{F}$`)` **do**
13 $\quad\quad Q$`.push(`f`)`
14 $\quad\quad \mathcal{J}[f] \leftarrow \{1, \dots, n\}$
15 $\quad$ **while** Q *is not empty* **do**
16 $\quad\quad f \leftarrow Q$`.pop()`
17 $\quad\quad Y \leftarrow \emptyset$
18 $\quad\quad$ **foreach** $j \in \mathcal{J}[f]$ *s.t.* $\widetilde{\Psi}(f) \ge v_j$ **do**
19 $\quad\quad\quad$ **if** $\Psi_j(f) \ge v_j$ **then**
20 $\quad\quad\quad\quad v_j \leftarrow \max\{v_j, \Delta_j(f)\}$
21 $\quad\quad\quad\quad Y \leftarrow Y \cup \{j\}$
22 $\quad\quad$ **foreach** $g \in$ `children(`f`)` $\setminus H$ **do**
23 $\quad\quad\quad$ **if** $g \in \mathcal{J}$ **then** $N \leftarrow \mathcal{J}[g] \cap Y$ **else** $N \leftarrow Y$
24 $\quad\quad\quad$ **if** $N = \emptyset$ **then**
25 $\quad\quad\quad\quad H \leftarrow H \cup \{g\}$
26 $\quad\quad\quad\quad$ **if** $g \in \mathcal{J}$ **then** Q`.delete(`g`)`
27 $\quad\quad\quad$ **else**
28 $\quad\quad\quad\quad$ **if** $g \notin \mathcal{J}$ **then** Q`.push(`g`)`
29 $\quad\quad\quad\quad \mathcal{J}[g] \leftarrow N$
30 $\quad$ **return** $\frac{1}{nm}\sum_{j=1}^{n} v_j$

MCRapper populates Q and $\mathcal{J}$ by inserting in them the minimal elements of $\mathcal{F}$ w.r.t. $\preceq$ (line 12), using the set $\{1, \dots, n\}$ as the value for these keys in the dictionary. It then enters a loop that keeps iterating as long as there are elements in Q (line 15). The top element f of Q is extracted at the beginning of each iteration (line 16). A set Y, initially empty, is created to maintain a superset of the set of values $j \in \{1, \dots, n\}$ for which a child of f *may* be the function attaining the supremum j-discrepancy among all the functions in $\mathcal{F}$ (see (9)). The algorithm then iterates over the elements $j \in \mathcal{J}[f]$ s.t. $\widetilde{\Psi}(f)$ is greater than v_j (line 18). The elements for which $\widetilde{\Psi}(f) < v_j$ can be ignored because f and its descendants can not attain the supremum of the j-discrepancy in this case, thanks to Thm. 4.1. Computing $\widetilde{\Psi}(f)$ is straightforward and can be done even faster if one keeps a frequent-pattern tree or a similar data structure to avoid having to scan $\mathcal{S}$ all the times, but we do not discuss this case for ease of presentation. For the values j that satisfy the condition on line 18, the algorithm computes $\Delta_j(f)$ and updates v_j to this value if larger than the current value of v_j (line 20), to maintain the

invariant that v_j stores the highest value of j-discrepancy seen so far (this invariant, together with the one maintained by the pruning strategy, is at the basis of the correctness of MCRapper). Finally, j is added to the set Y (line 21), as it may still be the case that a descendant of f has j-discrepancy higher than v_j. The algorithm then iterates over the children of f that have not been pruned, i.e., those not in H (line 22). If the child g is such that there is a key g in $\mathcal{J}$ (because before f we visited another parent of g), then let N be $\mathcal{J}[g] \cap Y$, otherwise, let N be Y. The set N is a superset of the indices j s.t. g may attain the supremum j-discrepancy. Indeed for a value j to have this property, it is *necessary* that $\Psi_j(f) \geq v_j$ *for every parent* f of j (where the value of v_j in this expression is the one that v_j had when f was visited). If $N = \emptyset$, then g and all its descendants can be pruned, which is achieved by adding g to H (line 25) and removing g from Q if it is a key $\mathcal{J}$ (line 26). When $N \neq \emptyset$, first g is added to Q (with the appropriate priority depending on the ordering of Q) if it did not belong to $\mathcal{J}$ yet (line 28), and then $\mathcal{J}[g]$ is set to N (line 29). This operation completes the current loop iteration starting at line 15.

Once Q is empty, the loop ends and the function `getNMCERA()` returns the sum of the values v_j divided by $n \cdot m$. The returned value is summed to an appropriate term to obtain $\tilde{\mathrm{R}}$ (line 5), which is used to compute the return value of the function `getSupDevBound()` using (6) with $\eta = \delta$ (line 6). This value ε is returned in output by MCRapper when it terminates (line 2).

The following result is at the core of the correctness of MCRapper (proof in App. A.1.)

Lemma 4.3. `getNMCERA`$(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma})$ *returns the value* $\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma})$.

The proof of Thm. 4.2 is then just an application of Lemma 4.3 and Thm. 3.1 (with $\eta = \delta$), as the value ε returned by MCRapper is computed according to (6).

4.2.1 Limiting the exploration of the search space. Despite the very efficient pruning strategy made possible by the upper bounds to the j-discrepancy, MCRapper may still need to explore a large fraction of the search space, with negative impact on the running time. We now present a "hybrid" approach that limits this exploration, while still ensuring the guarantees from Thm. 4.2.

Let β be any positive value and define

$$\mathcal{G}(\mathcal{S}, \beta) \doteq \left\{ f \in \mathcal{F} : \frac{1}{m} \sum_{i=1}^{m} (f(s_i))^2 \geq \beta \right\},$$

and $\mathcal{K}(\mathcal{S}, \beta) = \mathcal{F} \setminus \mathcal{G}(\mathcal{S}, \beta)$. In the case of itemsets mining, $\mathcal{G}(\mathcal{S}, \beta)$ would be the set of frequent itemsets w.r.t. $\beta \in [0, 1]$.

The following result is a consequence of Hoeffding's inequality and a union bound over $n \cdot |\mathcal{K}(\mathcal{S}, \beta)|$ events.

Lemma 4.4. *Let* $\eta \in (0, 1)$. *Then, with probability at least* $1-\eta$ *over the choice of* $\boldsymbol{\sigma}$, *it holds that* simultaneously for all $j \in \{1, \ldots, n\}$,

$$\hat{\mathrm{R}}^1_m(\mathcal{K}(\mathcal{S}, \beta), \mathcal{S}, \boldsymbol{\sigma}_j) \leq \sqrt{\frac{2\beta \log\left(\frac{n|\mathcal{K}(\mathcal{S},\beta)|}{\eta}\right)}{m}} \,. \tag{11}$$

The following is an immediate consequence of the above and the definition of n-MCERA.

Theorem 4.5. *Let* $\eta \in (0, 1)$. *Then with probability* $\geq 1 - \eta$ *over the choice of* $\boldsymbol{\sigma}$, *it holds*

$$\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}) = \frac{1}{n} \sum_{j=1}^{n} \max\left\{ \hat{\mathrm{R}}^1_m(\mathcal{G}(\mathcal{S}, \beta), \mathcal{S}, \boldsymbol{\sigma}_j), \hat{\mathrm{R}}^1_m(\mathcal{K}(\mathcal{S}, \beta), \mathcal{S}, \boldsymbol{\sigma}_j) \right\}$$

$$\leq \frac{1}{n} \sum_{j=1}^{n} \max\left\{ \hat{\mathrm{R}}^1_m(\mathcal{G}(\mathcal{S}, \beta), \mathcal{S}, \boldsymbol{\sigma}_j), \sqrt{\frac{2\beta \log\left(\frac{n|\mathcal{K}(\mathcal{S},\beta)|}{\eta}\right)}{m}} \right\} .$$

The result of Thm. 4.5 is especially useful in situations when it is possible to compute efficiently reasonable upper bounds on the cardinality of $\mathcal{K}(\mathcal{S}, \beta)$, possibly using information from $\mathcal{S}$ (but not $\boldsymbol{\sigma}$). For the case of pattern mining, these bounds are often easy to obtain: e.g., in the case of itemsets, it holds $|\mathcal{K}(\mathcal{S}, \beta)| \leq \sum_{s_i \in \mathcal{S}} 2^{|s_i|}$, where $|s_i|$ is the number of items in the transaction s_i. Much better bounds are possible, and in many other cases, but we cannot discuss them here due to space limitations.

Combining the above with MCRapper may lead to a significant speed-up thanks to the fact that MCRapper would be exploring only (a subset of) $\mathcal{G}(\mathcal{S}, \beta)$ instead of (a subset of) the entire search space $\mathcal{F}$, at the cost of computing an *upper bound* to $\hat{\mathrm{R}}^n_m(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}_j)$, rather than its exact value. We study this trade-off, which is governed by the choice of β, experimentally in Sect. 6.3. The correctness follows from Thms. 3.1, 4.2 and 4.5, and an application of the union bound.

We now describe this variant MCRapper-H of MCRapper, presented in Alg. 2. MCRapper-H accepts in input the same parameters of MCRapper, but also the parameters β and $\gamma < \delta$, which controls the confidence of the probabilistic bound from Thm. 4.5. After having drawn $\boldsymbol{\sigma}$, MCRapper-H computes the upper bound to $|\mathcal{K}(\mathcal{S}, \beta)|$ (line 3), and calls the function `getNMCERA`$(\mathcal{G}(\mathcal{S}, \beta), \mathcal{S}, \boldsymbol{\sigma})$ (line 2), slightly modified w.r.t. the one on line 30 of Alg. 1 so it returns the set of n values $\{v_1, \ldots, v_n\}$ instead of their average. Then, it computes $\tilde{\mathrm{R}}$ using the r.h.s. of (11) and returns the bound to the SD obtained from the r.h.s. of (6) with $\eta = \delta - \gamma$.

Algorithm 2: MCRapper-H

Input: Poset family $\mathcal{F}$, sample $\mathcal{S}$ of size m, $\delta \in (0, 1)$, $\beta \in [0, z^2]$, $\gamma \in (0, \delta)$
Output: Upper bound to $\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu)$ with prob. $\geq 1 - \delta$.

1 $\boldsymbol{\sigma} \leftarrow$ `draw(`m, n`)`
2 $\{v_1, \ldots, v_n\} \leftarrow$ `getNMCERA(`$\mathcal{G}(\mathcal{S}, \beta), \mathcal{S}, \boldsymbol{\sigma}$`)`
3 $\omega \leftarrow$ upper bound to $|\mathcal{K}(\mathcal{S}, \beta)|$
4 $\tilde{\mathrm{R}} \leftarrow \frac{1}{n} \sum_{j=1}^{n} \max\left\{ \frac{v_j}{m}, \sqrt{\frac{2\beta \log\left(\frac{n\omega}{\gamma}\right)}{m}} \right\} + 2z\sqrt{\frac{\ln\left(\frac{4}{\delta-\gamma}\right)}{2nm}}$
5 **return** *r.h.s. of* (6) *using* $\eta = \delta - \gamma$

It is not necessary to choose β a-priori, as long as it is chosen without using any information that depends on $\boldsymbol{\sigma}$. In situations where deciding β a-priori is not simple, one may define instead, for a given value of k set by the user, the quantity β_k defined as

$$\beta_k \doteq \min\{\beta : |\mathcal{G}(\mathcal{S}, \beta)| \leq k\} .$$

When the queue Q (line 8 of Alg. 1) is sorted by decreasing value of $\sum_{i=1}^{n} (f(s_i))^2$, the value k is the maximum number of nodes the

branch-and-bound search in `getNMCERA()` may enumerate. We are investigating more refined bounds than Thm. 4.5.

4.3 Improved bounds for $n = 1$

For the special case of $n = 1$, it is possible to derive a better bound to the SD than the one presented in Thm. 3.1. This result is new and of independent interest because it holds for *any* family $\mathcal{F}$. The proof is in App. A.1.

Theorem 4.6. *Let $\eta \in (0, 1)$. With probability at least $1 - \eta$ over the choice of $\mathcal{S}$ and σ, it holds that*

$$\mathsf{D}(\mathcal{F}, \mathcal{S}, \mu) \leq 2\hat{\mathsf{R}}^1_m\left(\mathcal{F} - \frac{c}{2}, \mathcal{S}, \boldsymbol{\sigma}\right) + 3c\sqrt{\frac{\ln\frac{2}{\eta}}{2m}} \quad . \tag{12}$$

The advantage of (12) over (6) (with $n = 1$) is in the smaller "tail bounds" terms that arise thanks to a single application of a probabilistic tail bound, rather than three such applications. To use this result in MCRapper, line 2 must be replaced with

$$\varepsilon \leftarrow \texttt{getNMCERA}(\mathcal{F}, \mathcal{S}, \boldsymbol{\sigma}) + 3c\sqrt{\frac{\ln\frac{2}{\delta}}{2m}};$$

so the upper bound to the SD is computed according to (12). The same guarantees as in Thm. 4.2 hold for this modified algorithm.

5 APPLICATIONS

To showcase MCRapper's practical strengths, we now discuss applications to various pattern mining tasks. The value ε computed by MCRapper can be used, for example, to compute, from a random sample $\mathcal{S}$, a high-quality approximation of the collection of frequent itemsets in a dataset w.r.t. a frequency threshold $\theta \in (0, 1)$, by mining the sample at frequency $\theta - \varepsilon$ [17]. Also, it can be used in the algorithm by Pellegrina et al. [15] to achieve statistical power in significant pattern mining, or in the progressive algorithm by Servan-Schreiber et al. [22] to enable even more accurate interactive data exploration. Essentially any of the tasks we mentioned in Sect. 1 and 2 would benefit from the improved bound to the SD computed by MCRapper. To support this claim, we now discuss in depth one specific application.

Mining True Frequent Patterns. We now show how to use MCRapper together with sharp variance-aware bounds to the SD (Thm. 3.2) for the specific application of identifying the True Frequent Patterns (TFPs) [19]. The original work considered the problem only for itemsets, but we solve the problem for a general poset family of functions, thus for many other pattern classes, such as sequences.

The task of TFP mining is, given a pattern language $\mathcal{L}$ (i.e., a poset family) and a threshold $\theta \in [0, 1]$, to output the set

$$\mathsf{TFP}(\theta, \mathcal{L}) = \left\{f \in \mathcal{L} : \underset{\mu}{\mathbb{E}}[f] \geq \theta\right\} \quad .$$

Computing $\mathsf{TFP}(\theta, \mathcal{L})$ *exactly* requires to know $\mathbb{E}_\mu[f]$ for all f; since this is almost never the case (and in such case the task is trivial), it is only possible to compute an *approximation* of $\mathsf{TFP}(\theta, \mathcal{L})$ using information available from a random bag $\mathcal{S}$ of m i.i.d. samples from μ. In this work, mimicking the guarantees given in significant pattern mining [11] and in multiple hypothesis testing settings, we are interested in approximations that are a *subset* of $\mathsf{TFP}(\theta, \mathcal{L})$, i.e., we do not want false *positives* in our approximation, but we accept false *negatives*. A variant that returns a superset of $\mathsf{TFP}(\theta, \mathcal{L})$ is possible and only requires minimal modifications ot the algorithm. Due to the randomness in the generation of $\mathcal{S}$, no algorithm can guarantee to be able to compute a (non-trivial) subset of $\mathsf{TFP}(\theta, \mathcal{L})$ from *every* possible $\mathcal{S}$. Thus, one has to accept that there is a probability over the choice of $\mathcal{S}$ *and other random choices made by the algorithm* to obtain a set of patterns that is not a subset of $\mathsf{TFP}(\theta, \mathcal{L})$. We now present an algorithm TFP-R with the following guarantee (proof in App. A.1).

Theorem 5.1. *Given $\mathcal{L}$, $\mathcal{S}$, $\theta \in [0, 1]$, $\delta \in (0, 1)$, and a number $n \geq 1$ of Monte-Carlo trials,* TFP-R *returns a set Y such that*

$$\Pr_{\mathcal{S},\boldsymbol{\sigma}}(Y \subseteq \mathsf{TFP}(\theta, \mathcal{L})) \geq 1 - \delta,$$

where the probability is over the choice of both $\mathcal{S}$ and the randomness in TFP-R*, i.e., an $n \times m$ matrix of i.i.d. Rademacher variables $\boldsymbol{\sigma}$.*

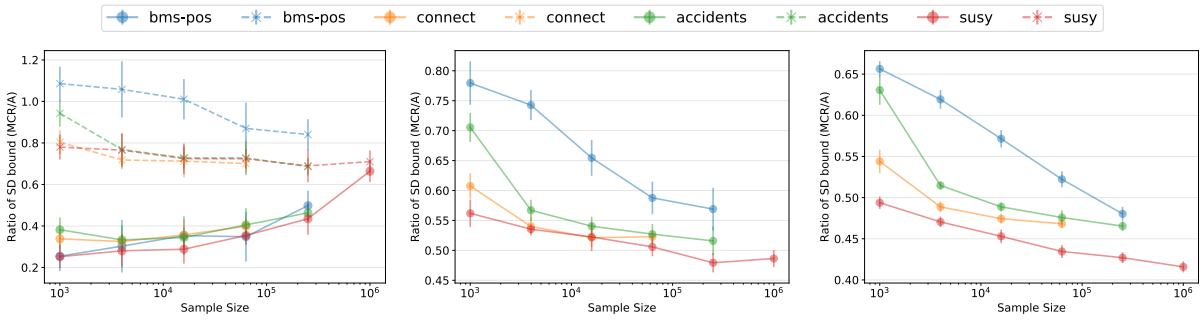
The intuition for TFP-R is the following. Let $\mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L}))$ be the *negative border* of $\mathsf{TFP}(\theta, \mathcal{L})$, that is, the set of functions in $\mathcal{L} \setminus \mathsf{TFP}(\theta, \mathcal{L})$ such that every parent w.r.t. $\preceq$ of f is in $\mathsf{TFP}(\theta, F)$. If we can compute an $\hat{\varepsilon} \in (0, 1)$ such that, for every $f \in \mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L}))$, it holds $\hat{\mathbb{E}}_\mathcal{S}[f] \leq \theta + \hat{\varepsilon}$, then we can be sure that any $g \in \mathcal{L}$ such that $\hat{\mathbb{E}}_\mathcal{S}[g] > \theta + \hat{\varepsilon}$ belongs to $\mathsf{TFP}(\theta, \mathcal{L})$. This guarantee will naturally be probabilistic, for the reasons we already discussed. Since $\mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L}))$ is unknown, TFP-R approximates it by *progressively* refining a *superset* C of it, starting from $\mathcal{L}$. The correctness of TFP-R is based on the fact that at every point in the execution, it holds $\mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L})) \subseteq C$, as we show in the proof of Thm. 5.1.

Algorithm 3: TFP-R

Input: Poset family $\mathcal{L}$, sample $\mathcal{S}$ of size m, $\theta \in [0, 1]$, $\delta \in (0, 1)$, $n \geq 1$.
Output: A set Y of patterns

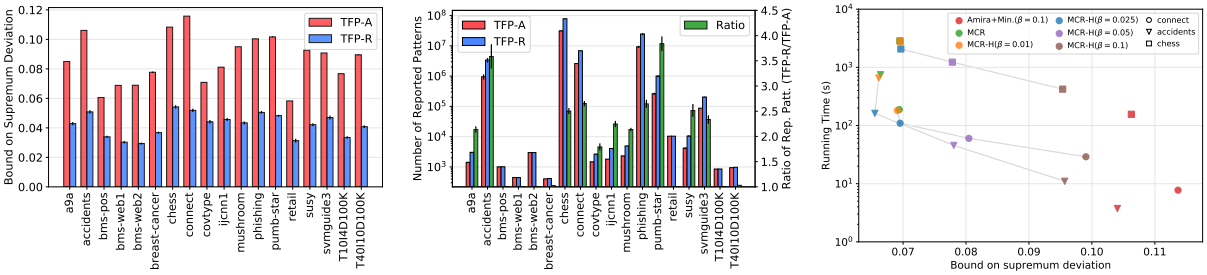
1 $Y \leftarrow \emptyset$
2 $\boldsymbol{\sigma} \leftarrow \texttt{draw}(m, n)$
3 **if** $\theta \geq \frac{1}{2}$ **then** $v \leftarrow \frac{1}{4}$ **else** $v \leftarrow \theta(1 - \theta)$
4 $C \leftarrow \mathcal{L}$
5 **repeat**
6 $\quad\hat{\varepsilon} \leftarrow \texttt{getSupDevBoundVar}(C, \mathcal{S}, \delta, \boldsymbol{\sigma}, v)$
7 $\quad C' \leftarrow C$
8 $\quad C \leftarrow \{f \in C' : \hat{\mathbb{E}}_\mathcal{S}[f] < \theta + \hat{\varepsilon}\}$
9 $\quad Y \leftarrow Y \cup (C' \setminus C)$
10 **until** $C = C'$
11 **return** Y

The pseudocode of TFP-R is presented in Alg. 3. The algorithm first draws the matrix $\boldsymbol{\sigma}$ (line 2), and then computes an upper bound v to the variances of the the frequencies in $\mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L}))$ (line 3). It then initializes, as discussed above, the set C to $\mathcal{L}$ (line 4) and enters a loop. At each iteration of the loop, TFP-R calls the function `getSupDevBoundVar` which returns a value $\hat{\varepsilon}$ computed as in (8) using $\mathcal{F} = C$, and $\eta = \delta$. The function `getNMCERA` from Alg. 1 is used inside of `getSupDevBoundVar` (with parameters C, $\mathcal{S}$, and $\boldsymbol{\sigma}$) to compute the n-MCERA in the value ρ from (7). The properties of $\hat{\varepsilon}$ are discussed in the proof for Thm. 5.1.



(a) $n = 1$. (b) $n = 10$. (c) $n = 10^2$.

Figure 1: Ratios of the SD Bound obtained by MCRapper ($n \in \{1, 10, 10^2\}$) and Amira for the entire $\mathcal{F}$, for 4 of the datasets we analyzed. For $n = 1$, dashed lines use the tail bound from Thm. 3.1 instead of the one from Thm. 4.6.



(a) (b) (c)

Figure 2: (a) Bound on the Supremum Deviation obtained by TFP-R and TFP-A. (b) Number of reported patterns (left y-axis) and ratios (right y-axis) by TFP-R and TFP-A. (c) Running times of MCRapper, MCRapper-H and Amira vs corresponding upper bound on SD of the entire $\mathcal{F}$. For MCRapper-H we use different values of β. Each marker shape corresponds to one of the datasets we considered (other 3 shown in the Appendix). For Amira we also show the time for mining the TFPs (Amira+Min.), with freq. $\geq \beta = 0.1$, as needed after processing the sample.

TFP-R uses $\hat{\varepsilon}$ to refine the set C with the goal of obtaining a better approximation of $\mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L}))$. The set C' stores the current value of C, and the new value of C is obtained by keeping all and only the patterns $f \in C'$ such that $\hat{\mathbb{E}}_{\mathcal{S}}[f] < \theta + \hat{\varepsilon}$ (line 8). All the patterns that have been filtered out, i.e., the patterns in $C' \setminus C$, or in other words, all the patterns $f \in C'$ such that $\hat{\mathbb{E}}_{\mathcal{S}}[f] \geq \theta + \hat{\varepsilon}$, are added to the output set Y (line 9). TFP-R keeps iterating until the value of C does not change from the previous iteration (condition on line 10), and finally the set Y is returned in output. While we focused on the a conceptually high-level description of TFP-R, we note that an efficient implementation only requires *one* exploration of $\mathcal{F}$, such that Y can be provided in output *as $\mathcal{F}$ is explored*, therefore without executing either multiple instances of MCRapper or, at the end of TFP-R, a frequent pattern mining algorithm to compute Y.

6 EXPERIMENTS

In this section we present the results of our experimental evaluation for MCRapper. We compare MCRapper to Amira [18], an algorithm that bounds the Supremum Deviation by computing a deterministic upper bound to the ERA with one pass on the random sample. The goal of our experimental evaluation is to compare MCRapper to Amira in terms of the upper bound to the SD they compute. We also assess the impact of the difference in the SD bound provided by MCRapper and Amira for the application of mining true frequent patterns, by comparing our algorithm TFP-R with TFP-A, a simplified variant of TFP-R that uses Amira to compute a bound ε on the SD for all functions in $\mathcal{L}$, and returns as candidate true frequent patterns the set $\mathcal{G}(\theta + \varepsilon, \mathcal{S})$. It is easy to prove that the output of TFP-A is a subset of true frequent patterns with probability $\geq 1 - \delta$. We also evaluate the running time of MCRapper and of its variant MCRapper-H.

Datasets and implementation. We implemented MCRapper and MCRapper-H in C, by modifying TopKWY [16]. Our implementations are available at https://github.com/VandinLab/MCRapper. The implementation of Amira [18] has been provided by the authors. We test both methods on 18 datasets (see Table 1 in the Appendix for their statistics), widely used for the benchmark of frequent itemset mining algorithms. To compare MCRapper to Amira in terms of the upper bound to the SD, we draw, from every dataset, random samples of increasing size m; we considered 6 values equally spaced in the logarithmic space in the interval $[10^3, 10^6]$. We only consider

values of m smaller than the dataset size $|\mathcal{D}|$. For both algorithms we fix $\delta = 0.1$. For MCRapper we use $n \in \{1, 10, 100\}$.

To compare TFP-R to TFP-A, we analyze synthetic datasets of size $m = 10^4$ obtained by random sampling transactions from each dataset: the true frequency of a pattern corresponds to its frequency in the original dataset, which we use as the ground truth. We use $n = 10$ for TFP-R, and $\delta = 0.1$. We report the results for $\theta = 0.05$ (other values of θ and n produced similar results).

For all experiments and parameters combinations we perform 10 runs (i.e., we create 10 random samples of the same size from the same dataset). In all the figures we report the averages and avg ± standard deviations of these runs.

6.1 Bounds on the SD

Figure 1 shows the ratio between the upper bound on the SD obtained by MCRapper and the one obtained by Amira for different values of n. The bound provided by MCRapper is always better (i.e., lower) than the bound provided by Amira (e.g., for $n = 100$ the bound from MCRapper is always at least 34% smaller than the bound from Amira). For $n = 1$ one can see that the *novel* improved bound from Thm. 4.6 should really be preferred over the "standard" one (dashed lines). Similar results hold for all other datasets. These results highlight the effectiveness of MCRapper in providing a much tighter bound to the SD than currently available approaches.

6.2 Mining True Frequent Patterns

We compare the *final* SD computed by MCRapper with the one computed by TFP-A. The results are shown in Fig. 2a. Similarly to what we observed in Sect. 6.1, MCRapper provides much tighter bounds being, in most cases, less than 50% of the bound reported by Amira. We then assessed the impact of such difference in the mining of TFP, by comparing the number of patterns reported by TFP-R and by TFP-A. Since for both algorithms the output is a subset of the true frequent patterns with probability $\geq 1 - \delta$, reporting a higher number of patterns corresponds to identifying more true frequent patterns, i.e., to higher power. Figure 2b shows the number of patterns reported by TFP-R and by TFP-A (left y-axis) and the ratio between such quantities (right y-axis). The SD bound from MCRapper is always lower than the SD bound from Amira, so TFP-R always reports at least as many patterns as TFP-A, and for 10 out of 18 datasets, it reports at least *twice* as many patterns as TFP-A. These results show that the SD bound computed by TFP-R provides a great improvement in terms of power for mining TFPs w.r.t. current state-of-the-art methods for SD bound computation.

6.3 Running time

For these experiments we take 10 random samples of size 10^4 of the 6 most demanding datasets (`accidents`, `chess`, `connect`, `phishing`, `pumb-star`, `susy`; for the other datasets MCRapper takes much less time than the ones shown) and use the hybrid approach MCRapper-H (Sect. 4.2.1) with different values of β (and $n = 1$, which gives a good trade-off between the bounds and the running time, $\gamma = 0.01$, $\delta = 0.1$). We naïvely upper bound $|\mathcal{K}(\mathcal{S}, \beta)|$ with $\sum_{s_i \in \mathcal{S}} 2^{|s_i|}$, where $|s_i|$ is the length of the transaction s_i, a *very loose* bound that could be improved using more information from $\mathcal{S}$. Figures 2c and 3 (in the Appendix) show the running time of MCRapper and Amira vs. the obtained upper bound on the SD (different colors correspond to different values of β). With Amira one can quickly obtain a fairly loose bound on the SD, by using MCRapper and MCRapper-H one can trade-off the running time for smaller bounds on the SD.

7 CONCLUSION

We present MCRapper, an algorithm for computing a bound to the supremum deviation of the sample means from their expectations for families of functions with poset structure, such as those that arise in pattern mining tasks. At the core of MCRapper there is a novel efficient approach to compute the n-sample Monte-Carlo Empirical Rademacher Average based on fast search space exploration and pruning techniques. MCRapper returns a much better (i.e., smaller) bound to the supremum deviation than existing techniques. We use MCRapper to extract true frequent patterns and show that it finds many more patterns than the state of the art.

ACKNOWLEDGMENTS

Part of this work was conducted while L.P. was visiting the Department of Computer Science of Brown University, supported by a "Fondazione Ing. Aldo Gini" fellowship. Part of this work is supported by the National Science Foundation grant RI-1813444, by the MIUR of Italy under PRIN Project n. 20174LF3T8 AHeAD (Efficient Algorithms for HArnessing Networked Data), and by the University of Padova grant STARS 2018.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *SIGMOD Rec.* 22 (June 1993), 207–216. Issue 2. https://doi.org/10.1145/170036.170072

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering, (ICDE'95)*. IEEE, 3–14.

[3] N. K. Ahmed, J. Neville, R. A. Rossi, and Duffield N. 2015. Efficient Graphlet Counting for Large Networks. In *2015 IEEE International Conference on Data Mining*. 1–10. https://doi.org/10.1109/ICDM.2015.141

[4] Peter L. Bartlett and Shahar Mendelson. 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research* 3, Nov (2002), 463–482.

[5] Stephen D. Bay and Michael J. Pazzani. 2001. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery* 5, 3 (2001), 213–246.

[6] Mario Boley, Claudio Lucchese, Daniel Paurat, and Thomas Gärtner. 2011. Direct local pattern sampling by efficient two-step random procedures. *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11* (2011). https://doi.org/10.1145/2020408.2020500

[7] Venkatesan T. Chakaravarthy, Vinayaka Pandit, and Yogish Sabharwal. 2009. Analysis of sampling techniques for association rule mining. In *Proc. 12th Int. Conf. Database Theory* (St. Petersburg, Russia) *(ICDT '09)*. ACM, New York, NY, USA, 276–283. https://doi.org/10.1145/1514894.1514927

[8] L. De Stefani and E. Upfal. 2019. A Rademacher Complexity Based Method for Controlling Power and Confidence Level in Adaptive Statistical Analysis. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 71–80. https://doi.org/10.1109/DSAA.2019.00021

[9] Vladimir Dzyuba, Matthijs van Leeuwen, and Luc De Raedt. 2017. Flexible constrained sampling with guarantees for pattern mining. *Data Mining and Knowledge Discovery* 31, 5 (Mar 2017), 1266–1293. https://doi.org/10.1007/s10618-017-0501-6

[10] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Tin Truong-Chi, and Roger Nkambou. 2019. A Survey of High Utility Itemset Mining. In *High-Utility Pattern Mining*. Springer International Publishing.

[11] Wilhelmiina Hämäläinen and Geoffrey I. Webb. 2018. A Tutorial on Statistically Sound Pattern Discovery. *Data Mining and Knowledge Discovery* (Dec 2018). https://doi.org/10.1007/s10618-018-0590-x

[12] Adam Kirsch, Michael Mitzenmacher, Andrea Pietracaprina, Geppino Pucci, Eli Upfal, and Fabio Vandin. 2012. An efficient rigorous approach for identifying statistically significant frequent itemsets. *Journal of the ACM (JACM)* 59, 3 (2012), 1–22.

[13] Willi Klösgen. 1992. Problems for knowledge discovery in databases and their treatment in the Statistics Interpreter Explora. *International Journal of Intelligent Systems* 7 (1992), 649–673.

[14] Vladimir Koltchinskii and Dmitriy Panchenko. 2000. Rademacher processes and bounding the risk of function learning. In *High dimensional probability II*. Springer, 443–457.

[15] Leonardo Pellegrina, Matteo Riondato, and Fabio Vandin. 2019. SPuManTE: Significant Pattern Mining with Unconditional Testing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Anchorage, AK, USA) *(KDD '19)*. ACM, New York, NY, USA, 1528–1538. https://doi.org/10.1145/3292500.3330978

[16] Leonardo Pellegrina and Fabio Vandin. 2020. Efficient mining of the most significant patterns with permutation testing. *Data Mining and Knowledge Discovery* (2020).

[17] Matteo Riondato and Eli Upfal. 2014. Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees. *ACM Trans. Knowl. Disc. from Data* 8, 4 (2014), 20. https://doi.org/10.1145/2629586

[18] Matteo Riondato and Eli Upfal. 2015. Mining Frequent Itemsets through Progressive Sampling with Rademacher Averages. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, 1005–1014.

[19] Matteo Riondato and Fabio Vandin. 2014. Finding the true frequent itemsets. In *Proceedings of the 2014 SIAM international conference on data mining*. SIAM, 497–505.

[20] Matteo Riondato and Fabio Vandin. 2018. MiSoSouP: Mining Interesting Subgroups with Sampling and Pseudodimension. In *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. and Data Mining (KDD '18)*. ACM, 2130–2139.

[21] Diego Santoro, Andrea Tonon, and Fabio Vandin. 2020. Mining Sequential Patterns with VC-Dimension and Rademacher Complexity. *Algorithms* 13, 5 (2020), 123.

[22] Sacha Servan-Schreiber, Matteo Riondato, and Emanuel Zgraggen. 2018. ProSecCo: Progressive Sequence Mining with Convergence Guarantees. In *Proceedings of the 18th IEEE International Conference on Data Mining*. 417–426.

[23] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press.

[24] Mahito Sugiyama, Felipe Llinares-López, Niklas Kasenburg, and Karsten M Borgwardt. 2015. Significant subgraph mining with multiple testing correction. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 37–45.

[25] Aika Terada, Mariko Okada-Hatakeyama, Koji Tsuda, and Jun Sese. 2013. Statistical significance of combinatorial regulations. *Proceedings of the National Academy of Sciences* 110, 32 (2013), 12996–13001.

[26] Hannu Toivonen. 1996. Sampling Large Databases for Association Rules. In *Proc. 22nd Int. Conf. Very Large Data Bases (VLDB '96)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 134–145.

[27] Andrea Tonon and Fabio Vandin. 2019. Permutation Strategies for Mining Significant Sequential Patterns. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1330–1335.

[28] Vladimir N. Vapnik. 1998. *Statistical learning theory*. Wiley.

A APPENDIX

A.1 Missing Proofs

Theorem A.1 (Symmetrization inequality [1]). *For any family $\mathcal{F}$ it holds* $\mathbb{E}_{\mathcal{S}}\left[\sup_{f\in\mathcal{F}}\left(\hat{\mathbb{E}}_{\mathcal{S}}[f]-\mathbb{E}_{\mu}[f]\right)-2\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})\right]\leq 0$

Theorem A.2 ([3, Thm. 2.2]). *Let* $Z=\sup_{f\in\mathcal{F}}\left(\hat{\mathbb{E}}_{\mathcal{S}}[f]-\mathbb{E}_{\mu}[f]\right)$. *Let $\eta\in(0,1)$. Then, with probability at least $1-\eta$ over the choice of $\mathcal{S}$, it holds*

$$Z\leq \mathop{\mathbb{E}}_{\mu}[Z]+\sqrt{\frac{2\ln\frac{1}{\eta}\left(v+4c\,\mathbb{E}_{\mu}[Z]\right)}{m}}+\frac{2c\ln\frac{1}{\eta}}{3m}\quad . \tag{13}$$

Proof of Thm. 3.2. Consider the following events

$$\mathsf{E}_1 \doteq \rho\geq\hat{\mathrm{R}}(\mathcal{F},\mathcal{S}),$$
$$\mathsf{E}_2 \doteq E_{\mu}[\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})]\leq\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})+\frac{1}{2m}\left(\sqrt{c\left(4m\rho+c\ln\frac{4}{\delta}\right)\ln\frac{4}{\delta}}+c\ln\frac{4}{\delta}\right)\quad .$$

From Lemma A.4, we know that E_1 holds with probability at least $1-\frac{\delta}{4}$ over the choice of $\mathcal{S}$ and $\boldsymbol{\sigma}$. E_2 is guaranteed to with probability at least $1-\frac{\delta}{4}$ over the choice of $\mathcal{S}$ [2, (generalization of) Thm. 3.11]. Define the event E_3 as the event in (13) for $\eta=\frac{\delta}{4}$ and the event E_4 as the event in (13) for $\eta=\frac{\delta}{4}$ and for $\mathcal{F}=-\mathcal{F}$. [3, Thm. 2.2] tells us that events E_3 and E_4 hold each with probability at least $1-\frac{d}{4}$ over the choice of $\mathcal{S}$. Thus from the union bound we have that the event $\mathsf{E}=\mathsf{E}_1\cap\mathsf{E}_2\cap\mathsf{E}_3$ holds with probability at least $1-\delta$ over the choice of $\mathcal{S}$ and $\boldsymbol{\sigma}$. Assume from now on that the event E holds.

Because E holds, it must be $r\geq\mathbb{E}_{\mu}[\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})]$. From this result and Thm. A.1 we have that

$$\mathop{\mathbb{E}}_{\mu}[\sup_{f\in\mathcal{F}}\left(\hat{\mathbb{E}}_{\mathcal{S}}[f]-\mathbb{E}_{\mu}[f]\right)]\leq 2\mathop{\mathbb{E}}_{\mu}[\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})]\leq 2r\quad .$$

From here, and again because E, by plugging $2r$ in place of $E[Z]$ into (13) (for $\eta=\frac{\delta}{4}$), we obtain that $\sup_{f\in\mathcal{F}}\left(\hat{\mathbb{E}}_{\mathcal{S}}[f]-\mathbb{E}_{\mu}[f]\right)\leq\varepsilon$. To show that it also holds

$$\sup_{f\in\mathcal{F}}\left(\hat{\mathop{\mathbb{E}}_{\mathcal{S}}}[f]-\mathop{\mathbb{E}}_{\mu}[f]\right)\leq\varepsilon$$

(which allows us to conclude that $\mathrm{D}(\mathcal{F},\mathcal{S},\mu)\leq\varepsilon$), we repeat the reasoning above for $-\mathcal{F}$ and use the fact that $\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})=\hat{\mathrm{R}}(-\mathcal{F},\mathcal{S})$, a known property of the ERA, thus

$$\rho\geq\hat{\mathrm{R}}(-\mathcal{F},\mathcal{S})\text{ and }r\geq E_{\mu}[\hat{\mathrm{R}}(-\mathcal{F},\mathcal{S})]\text{ and}$$
$$\varepsilon\geq\mathrm{D}(-\mathcal{F},\mathcal{S})=\sup_{f\in\mathcal{F}}\left(\hat{\mathop{\mathbb{E}}_{\mathcal{S}}}[f]-\mathop{\mathbb{E}}_{\mu}[f]\right)\quad .$$ □

Theorem A.3 (McDiarmid's inequality [4]). *Let $\mathcal{Y}\subseteq\mathbb{R}^{\ell}$, and let $g:\mathcal{Y}\to\mathbb{R}$ be a function such that, for each i, $1\leq i\leq\ell$, there is a nonnegative constant c_i such that:*

$$\sup_{\substack{x_1,\ldots,x_{\ell}\\ x_i'\in\mathcal{X}}}|g(x_1,\ldots,x_{\ell})-g(x_1,\ldots,x_{i-1},x_i',x_{i+1},\ldots,x_{\ell})|\leq c_i\quad . \tag{14}$$

Let $x_1,\ldots,x_{\ell}$ be ℓ independent random variables taking value in $\mathbb{R}^{\ell}$ such that $\langle x_1,\ldots,x_{\ell}\rangle\in\mathcal{Y}$. Then it holds

$$\Pr\left(g(x_1,\ldots,x_{\ell})-\mathop{\mathbb{E}}_{\mu}[g]>t\right)\leq e^{-2t^2/C},$$

where $C=\sum_{i=1}^{\ell}c_i^2$.

The following result is an application of McDiarmid's inequality to the n-MCERA, with constants $c_i=\frac{2z}{nm}$.

Lemma A.4. *Let $\eta\in(0,1)$. Then, with probability at least $1-\eta$ over the choice of $\boldsymbol{\sigma}$, it holds*

$$\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})=\mathop{\mathbb{E}}_{\boldsymbol{\sigma}}\left[\hat{\mathrm{R}}^n_m(\mathcal{F},\mathcal{S},\boldsymbol{\sigma})\right]\leq\hat{\mathrm{R}}^n_m(\mathcal{F},\mathcal{S},\boldsymbol{\sigma})+2z\sqrt{\frac{\ln\frac{1}{\eta}}{2nm}}\quad .$$

The following result gives a probabilistic upper bound to the supremum deviation using the RA and the ERA [2, Thm. 3.11].

Theorem A.5. *Let $\eta\in(0,1)$. Then, with probability at least $1-\eta$ over the choice of $\mathcal{S}$, it holds*

$$\mathrm{D}(\mathcal{F},\mathcal{S},\mu)\leq 2\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})+\frac{\sqrt{c\left(4m\hat{\mathrm{R}}(\mathcal{F},\mathcal{S})+c\ln\frac{3}{\eta}\right)\ln\frac{3}{\eta}}}{m}+\frac{c\ln\frac{3}{\eta}}{m}+c\sqrt{\frac{\ln\frac{3}{\eta}}{2m}}\quad .^{1} \tag{15}$$

Proof of Thm. 3.1. Through Lemma A.4 (using η there equal to $\frac{\eta}{4}$), Thm. A.5 (using η there equal to $\frac{3\eta}{4}$), and an application of the union bound. □

Proof of Thm. 4.1. It is immediate from the definitions of $\widetilde{\Psi}(f)$ and $\Psi_j(f)$ in (10) that $\Psi_j(f)\leq\widetilde{\Psi}(f)$, so we can focus on $\Psi_j(f)$. We start by showing that $\Delta_j(f)\leq\Psi_j(f)$. It holds

$$\Delta_j(f)=$$
$$\sum_{s_i\in\mathcal{S}}\sigma^+_{j,i}f^+(s_i)-\sum_{s_i\in\mathcal{S}}\sigma^-_{j,i}f^-(s_i)-\sum_{s_i\in\mathcal{S}}\sigma^-_{j,i}f^+(s_i)+\sum_{s_i\in\mathcal{S}}\sigma^+_{j,i}f^-(s_i)$$
$$\leq\sum_{s_i\in\mathcal{S}}\sigma^+_{j,i}f^+(s_i)-\sum_{s_i\in\mathcal{S}}\sigma^-_{j,i}f^-(s_i)=\Psi_j(f)$$

where the inequality comes from the fact that $\sum_{s_i\in\mathcal{S}}\sigma^-_{j,i}f^+(s_i)\geq 0$, and $\sum_{s_i\in\mathcal{S}}\sigma^+_{j,i}f^-(s_i)\leq 0$.

To prove that $\Delta_j(g)\leq\Psi_j(f)$ for every $g\in\mathrm{d}(f)$ it is sufficient to show that $\Psi_j(g)\leq\Psi_j(f)$ hold for every such g, since we just showed that $\Delta_j(g)\leq\Psi_j(g)$ is true for any $f\in\mathcal{F}$. It holds $f\preceq g$, so from the definition of the relation $\preceq$ in (2), we get

$$\Psi_j(g)=\sum_{s_i\in\mathcal{S}}\sigma^+_{j,i}g^+(s_i)-\sum_{s_i\in\mathcal{S}}\sigma^-_{j,i}g^-(s_i)$$
$$\leq\sum_{s_i\in\mathcal{S}}\sigma^+_{j,i}f^+(s_i)-\sum_{s_i\in\mathcal{S}}\sigma^-_{j,i}f^-(s_i)=\Psi_j(f)$$

which completes our proof. □

[1] Slightly sharper bounds are possible at the expense of an increased complexity of the terms.

Proof of Lemma 4.3. For $j \in \{1, \dots, n\}$, let h_j be any of the functions attaining the supremum in $\sup_{f \in \mathcal{F}} \Delta_j(f)$. We need to show that the algorithm updates v_j on line 20 of Alg. 1 using $\Delta_j(h_j)$ at some point during its execution. We focus on a single j, as the proof is the same for any value of j.

It is evident from the description of the algorithm that v_j is always only set to values of $\Delta_j(g)$, and since h_j has the maximum of these values, v_j will be, at any point in the execution of the algorithm less than or equal to $\Delta_j(h_j)$. Let's call this fact F_1. Thus, if the algorithm ever hits line 20 with $f = h_j$, then we can be sure that the value stored in v_j will be $\Delta_j(h_j)$, and this variable will never take an higher value. From fact F_1 and Thm. 4.1 we also have that at any point in time it must be $v_j \le \Psi_j(h_j) \le \widetilde{\Psi}(h_j)$, so the conditions on lines 19 and 18 are definitively satisfied, so the question is now whether $j \in \mathcal{J}[h_j]$ and whether there is an iteration of the **while** loop of line 15 for which $f = h_j$.

It holds from Thm. 4.1 that it must be $\Delta_j(h_j) \le \Psi_j(g) \le \widetilde{\Psi}(g)$ for every ancestor g of h_j. From this fact and from fact A then it holds that at any point in time it must hold $v_j \Psi_j(g) \le \widetilde{\Psi}(g)$ for every such ancestor g of h_j. Thus, the value j is always added to the set Y at every iteration of the **while** loop for which f is an ancestor of h_j. Let's call this fact F_2. Thus, as long as no ancestor of h_j is pruned or h_j itself is pruned, j is guaranteed to be in $\mathcal{J}[h_j]$. But from fact F_2 and from the fact that j belongs to $\mathcal{J}[f]$ for all the ancestors of h_j that are in `minimals(`f`)` (line 14), then j must belong to the set N computed on line 23 for all ancestors of h_j, thus N is never empty and therefore no ancestor of h_j is ever pruned and neither is f and we are guaranteed that h_j is added to Q on line 28 when the first of its parents is visited. Thus, there is an iteration of the **while** loop that has $f = h_j$, and because of what we discussed above, then it will be the case that $v_j = \Delta_j(h_j)$ and our proof is complete. □

Proof of Thm. 4.6. For ease of notation, let $\mathcal{G} = \mathcal{F} - \frac{c}{2}$. Consider the event

$$\mathsf{E}_1 \doteq \sup_{g \in \mathcal{G}} \left(\hat{\mathbb{E}}_{\mathcal{S}}[g] - \mathop{\mathbb{E}}_{\mu}[g] \right) \le 2\hat{\mathrm{R}}^1_m(\mathcal{G}, \mathcal{S}, \boldsymbol{\sigma}) + 3c\sqrt{\frac{\ln \frac{2}{\eta}}{2m}} \quad . \tag{16}$$

We now show that this event holds with probability at least $1 - \frac{\eta}{2}$ over the choices of $\mathcal{S}$ and $\boldsymbol{\sigma}$, and then we use this fact to obtain the thesis with some additional steps.

Using linearity of expectation and the fact that the n-MCERA is an unbiased estimator for the ERA (i.e., its expectation is the ERA), we can rewrite the symmetrization inequality (Thm. A.1) as

$$\mathop{\mathbb{E}}_{\mathcal{S}, \boldsymbol{\sigma}} \left[\sup_{g \in \mathcal{G}} \left(\hat{\mathbb{E}}_{\mathcal{S}}[g] - \mathop{\mathbb{E}}_{\mu}[g] \right) - 2\hat{\mathrm{R}}^1_m(\mathcal{G}, \mathcal{S}, \boldsymbol{\sigma}) \right] \le 0 \quad .$$

The argument of the (outmost) expectation on the l.h.s. can be seen as a function h of the m pairs of r.v.'s $(\boldsymbol{\sigma}_{1,1}, s_1), \dots, (\boldsymbol{\sigma}_{1,m}, s_m)$. Fix any possible assignment v' of values to these pairs. Consider now a second assignment v'' obtained from v' by changing the value of *any of the pairs* with any other value in $\{-1, 1\} \times \mathcal{X}$. We want to show that it holds $|h(v') - h(v'')| \le 3\frac{c}{m}$.

We separately handle the SD and the 1-MCERA, as both depend on the values of the assignment of values to the pairs. The SD does not depend on $\boldsymbol{\sigma}_{1,\cdot}$, and in the argument of the supremum, changing any s_j changes a single summand of the empirical mean $\hat{\mathbb{E}}_{\mathcal{S}}[f]$, with maximal change when $f(s_j)$ changes from a to b (or viceversa), thus the SD itself changes by no more than $\frac{c}{m}$.

We now consider the 1-MCERA, and assume that the pair changing value is $(\boldsymbol{\sigma}_{1,j}, s_j)$. Then the only term of the 1-MCERA sum that changes is the j-th term. If only the first component of the pair changes value (i.e., $\boldsymbol{\sigma}_{1,j}$ changes from 1 to -1 or viceversa, i.e., from $\boldsymbol{\sigma}_{1,j}$ to $-\boldsymbol{\sigma}_{1,j}$), then the j-th term in the 1-MCERA sum cannot change by more than c, because it holds $\boldsymbol{\sigma}_{1,j} g(s_j) \in [-\frac{c}{2}, \frac{c}{2}]$, thus $-\boldsymbol{\sigma}_{1,j} g(s_j)$ also belongs to this interval, and it must be $|\boldsymbol{\sigma}_{1,j} g(s_j) - (-\boldsymbol{\sigma}_{1,j} g(s_j))| \le c$. If only the second component of the pair changes value (i.e., s_j changes value to $\bar{s}_j$), then the j-th term in the 1-MCERA sum cannot change by more than c, because each function $g \in \mathcal{G}$ goes from $\mathcal{X}$ to $[-\frac{c}{2}, \frac{c}{2}]$, and it must be $|\boldsymbol{\sigma}_{1,j} g(s_j) - \boldsymbol{\sigma}_{i,j} g(\bar{s}_j)| \le c$. Consider now the final case where both $\boldsymbol{\sigma}_{1,j}$ and s_j change value. We have once again $|\boldsymbol{\sigma}_{1,j} g(s_j) - (-\boldsymbol{\sigma}_{1,j} g(\bar{s}_j))| \le c$.

By the adding the maximum change in the SD and the maximum change in the 1-MCERA we can conclude that function h satisfies the requirements of McDiarmid's inequality (Thm. A.3) with constants $3\frac{c}{m}$, and obtain that event E_1 from (16) holds with probability at least $1 - \frac{\eta}{2}$.

Let now $-\mathcal{G}$ represent the family of functions containing $-g$ for each $g \in \mathcal{G}$. Consider the event

$$\mathsf{E}_2 \doteq \sup_{g \in -\mathcal{G}} \left(\hat{\mathbb{E}}_{\mathcal{S}}[g] - \mathop{\mathbb{E}}_{\mu}[g] \right) \le 2\hat{\mathrm{R}}^1_m(-\mathcal{G}, \mathcal{S}, -\boldsymbol{\sigma}) + 3c\sqrt{\frac{\ln \frac{2}{\eta}}{2m}} \quad .$$

Following the same steps as for E_1, we have that E_2 holds with probability at least $1 - \frac{\eta}{2}$, as the fact that we are considering $\hat{\mathrm{R}}^1_m(-\mathcal{G}, \mathcal{S}, -\boldsymbol{\sigma})$ rather than $\hat{\mathrm{R}}^1_m(-\mathcal{G}, \mathcal{S}, \boldsymbol{\sigma})$ is not influential.

It is easy to see that $\hat{\mathrm{R}}^1_m(-\mathcal{G}, \mathcal{S}, -\boldsymbol{\sigma}) = \hat{\mathrm{R}}^1_m(\mathcal{G}, \mathcal{S}, \boldsymbol{\sigma})$, and that

$$\sup_{g \in -\mathcal{G}} \left(\hat{\mathbb{E}}_{\mathcal{S}}[g] - \mathop{\mathbb{E}}_{\mu}[g] \right) = \sup_{g \in \mathcal{G}} \left(\mathop{\mathbb{E}}_{\mu}[g] - \hat{\mathbb{E}}_{\mathcal{S}}[g] \right) \quad .$$

Thus we can rewrite E_2 as

$$\mathsf{E}_2 = \sup_{g \in \mathcal{G}} \left(\mathop{\mathbb{E}}_{\mu}[g] - \hat{\mathbb{E}}_{\mathcal{S}}[g] \right) \le 2\hat{\mathrm{R}}^1_m(\mathcal{G}, \mathcal{S}, \boldsymbol{\sigma}) + 2c\sqrt{\frac{\ln \frac{2}{\eta}}{2m}} \quad .$$

From the union bound, we have that E_1 and E_2 hold simultaneously with probability at least $1 - \eta$, i.e., the following event holds with probability at least $1 - \eta$

$$\mathrm{D}(\mathcal{G}, \mathcal{S}, \mu) \le 2\hat{\mathrm{R}}^1_m(\mathcal{G}, \mathcal{S}, \boldsymbol{\sigma}) + 3c\sqrt{\frac{\ln \frac{2}{\eta}}{2m}} \quad .$$

The thesis then follows from the fact $\mathrm{D}(\mathcal{F}, \mathcal{S}, \mu) = \mathrm{D}(\mathcal{G}, \mathcal{S}, \mu)$. □

Proof of Thm. 5.1. For ease of notation, let $\mathcal{G} = \mathsf{B}^-(\mathsf{TFP}(\theta, \mathcal{L}))$. Let ρ, r, and ε be as in Thm. 3.2 for $\eta = \delta$ and $\mathcal{F} = \mathcal{G}$. Thm. 3.2 tells us that, with probability at least $1 - \delta$, it holds $\mathrm{D}(\mathcal{G}, \mathcal{S}) \le \varepsilon$.[2] Assume from now on that that is the case.

We use this fact to show inductively that, at the end of every iteration of the loop of TFP-R (lines 5–10 of Alg. 3), it holds that $\mathcal{G} \subseteq C$ and $Y \subseteq \mathsf{TFP}(\theta, \mathcal{L})$, and therefore the thesis will hold.

[2] We actually only need a value ε such that $\sup_{f \in \mathcal{G}} \left(\hat{\mathbb{E}}_{\mathcal{S}}[f] - \mathbb{E}_{\mu}[f] \right) < \varepsilon$, but the gain would be minimal and it would make the presentation more complicated.

Consider the first iteration of the loop. We have $C = \mathcal{L} \supseteq \mathcal{G}$. Let $\hat{\rho}$, $\hat{r}$, and $\hat{\varepsilon}$ be the values computed inside the call to the function `getSupDevBoundVar` on line 6 with the parameters mentioned in the description of the algorithm. It holds that $\hat{\rho} \geq \rho$, because the n-MCERA of a superset of a family is not smaller than the n-MCERA of the family. It follows that $\hat{r} \geq r$, which in turn implies that $\hat{\varepsilon} \geq \varepsilon$. Since we assumed that $\mathrm{D}(\mathcal{G}, \mathcal{S}) \leq \varepsilon$, we have $\hat{\varepsilon} \geq \varepsilon \geq \mathrm{D}(\mathcal{G}, \mathcal{S})$ No function $f \in \mathcal{G}$ may then have sample mean $\hat{\mathbb{E}}_{\mathcal{S}}[f]$ greater than or equal to $\theta + \hat{\varepsilon}$, as every such f has $\mathbb{E}_{\mu}[f] < \theta$. Call this fact A. A first consequence of A is that, at the end of the iteration, it holds $\mathcal{G} \subseteq C$. A second consequence of A and of the antimonotonicity property is that *none* of the functions $f \in \mathcal{L}$ such that $\mathbb{E}_{\mu}[f] < \theta$ may have $\hat{\mathbb{E}}_{\mathcal{S}}[f] \geq \theta + \hat{\varepsilon}$. Equivalently, only functions $f \in \mathcal{L}$ such that $\mathbb{E}_{\mu}[f] \geq \theta$, i.e., such that $f \in \mathsf{TFP}(\theta, \mathcal{L})$, may have $\hat{\mathbb{E}}_{\mathcal{S}}[f] \geq \theta + \hat{\varepsilon}$, i.e., $C' \setminus C \subseteq \mathsf{TFP}(\theta, \mathcal{L})$, so $Y \subseteq \mathsf{TFP}(\theta, \mathcal{L})$ at the end of the first iteration. The base case is complete.

Assume now that $\mathcal{G} \subseteq C$ and $Y \subseteq \mathsf{TFP}(\theta, \mathcal{L})$ at the end of all iterations from 1 to i. Following the same reasoning as for the base case, it holds that these facts are true also at the end of iteration $i + 1$ and our proof is complete. □

dataset	$\lvert\mathcal{D}\rvert$	$\lvert I\rvert$	avg. trans. len.
svmguide3	1,243	44	21.9
chess	3,196	75	37
breast cancer	7,325	396	11.7
mushroom	8,124	117	22
phishing	11,055	137	30
a9a	32,561	245	13.9
pumb-star	49,046	7,117	50.9
bms-web1	58,136	60,878	3.51
connect	67,557	129	43.5
bms-web2	77,158	330,285	5.6
retail	87,979	16,470	10.8
ijcnn1	91,701	43	13
T10I4D100K	100,000	1,000	10
T40I10D100K	100,000	1,000	40
accidents	340,183	468	34.9
bms-pos	515,420	1,657	6.9
covtype	581,012	108	12.9
susy	5,000,000	190	19

Table 1: Datasets statistics. For each dataset, we report the number $\lvert\mathcal{D}\rvert$ of transactions; the number $\lvert I\rvert$ of items; the average transaction length.

A.2 Reproducibility

We now describe how to reproduce our experimental results. Code and data are available at https://github.com/VandinLab/MCRapper.

The code of MCRapper, TFP-R, and Amira are in the sub-folders `mcrapper/` and `amira/`. To compile with recent GCC or Clang, use the `make` command inside each sub-folder.

The convenient scripts `run_amira.py` and `run_mcrapper.py` can be used to run the experiments (i.e., run Amira, MCRapper, and TFP-R). They accept many input parameters (described using the flag `-h`). You need to specify a dataset and the size of a random sample to create using the flags `-db` and `-sz`. E.g., to process a

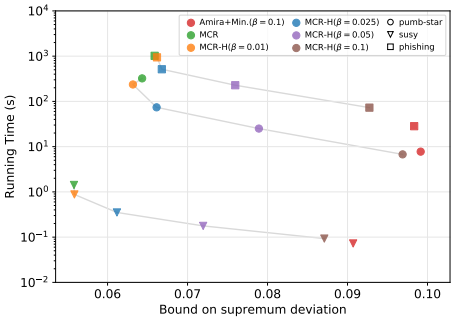


Figure 3: Running times of MCRapper, MCRapper-H and Amira vs corresponding upper bound on supremum deviation of the entire set of functions $\mathcal{F}$. For MCRapper-H we use different values of β. y-axis in log scale but x axis is linear. Each marker shape corresponds to one of the datasets.

random sample of 10^3 transactions from the dataset `mushroom` with $n = 100$, run

```
run_mcrapper.py -db mushroom -sz 1000 -j 100
```

and it automatically executes both Amira and MCRapper. The command line to process with TFP-R a sample of 10^4 transactions from the dataset `retail` with $n = 10$ and $\theta = 0.05$ is

```
run_mcrapper.py -db retail -sz 10000 -j 10 -tfp 0.05
```

The `run_all_datasets.py` script runs all the instances of MCRapper and Amira in parallel, and can be used to reproduce all the experiments described in Sect. 6. The `run_tfp_all_datasets.py` script reproduces the experiments for TFP-R and TFP-A.

All the results are stored in the files `results_mcrapper.csv` and `results_tfp_mcrapper.csv`.

REFERENCES

[1] V. Koltchinskii and D. Panchenko. Rademacher processes and bounding the risk of function learning. In *High dimensional probability II*, 443–457. Springer, 2000.
[2] L. Oneto, A. Ghio, D. Anguita, and S. Ridella. An improved analysis of the Rademacher data-dependent bound using its self bounding property. *Neural Networks*, 44:107–111, 2013.
[3] O. Bousquet. A Bennett concentration inequality and its application to suprema of empirical processes. *Comptes Rendus Mathem.*, 334(6):495–500, 2002.
[4] C. McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.