Anytime Algorithm for Frequent Pattern Outlier Detection

Arnaud Giacometti, Arnaud Soulet

Abstract Outlier detection consists in detecting anomalous observations from data. During the past decade, pattern-based outlier detection methods have proposed to mine all frequent patterns in order to compute the outlier factor of each transaction. This approach remains too expensive despite recent progress in pattern mining field. In this paper, we provide exact and approximate methods for calculating the frequent pattern outlier factor (FPOF) without extracting any pattern or by extracting a small sample. We propose an algorithm that returns the exact FPOF without mining any pattern. Surprisingly, it works in polynomial time on the size of the dataset. We also present an approximate method where the end-user controls the maximum error on the estimated FPOF. Experiments show the interest of both methods for very large datasets where exhaustive mining fails to provide the exact solution. The accuracy of our approximate method outperforms the baseline approach for a same budget in time or number of patterns.

1 Introduction

Outlier detection consists in detecting anomalous observations from data [Hawkins, 1980]. The outlier detection problem has important applications, such as detection of credit card fraud or network intrusions. Recently, outlier detection methods were proposed for categorical data using the concept of frequent patterns [He et al., 2005, Otey et al., 2006, Koufakou et al., 2011]. The key idea of such approaches is to consider the number of frequent patterns supported by each data observation. A data observation is unlikely to be an outlier if

Université François Rabelais Tours, LI EA 6300
3 place Jean Jaurès, F-41029 Blois, France
firstname.lastname@univ-tours.fr

it supports many frequent patterns since frequent patterns correspond to the "common features" of the dataset. Frequent pattern outlier detection methods first extract all frequent itemsets from the data and then assign an outlier score to each data observation based on the frequent itemsets it contains. These outlier detection methods follow the schema of pattern-based two-step methods.

Pattern-based two-step methods [Knobbe et al., 2008] aim at exhaustively mine all patterns (first step) in order to build models (second step) like pattern sets (e.g., classifier [Liu et al., 1998]) or pattern-based measures (e.g., FPOF [He et al., 2005] or CPCQ index [Liu and Dong, 2012]). The completeness of pattern mining is often considered as a crucial advantage for constructing accurate models or measures. However, it also leads to three important issues that hinders the user interaction with the system:

1. **Threshold issue:** The completeness of the first step requires to adjust thresholds which is recognized as being very difficult. Typically, if the minimal support threshold is too low, the extraction becomes unfeasible. If it is too high, some essential patterns are missed.
2. **Accuracy issue:** Completeness leads to huge pattern volumes without guaranteeing not missing important patterns. For a smaller budget (in time or number of patterns), we claim that non-exhaustive methods can produce collections of patterns better adapted to the task of the second step. Interestingly, a non-exhaustive method can even guarantee a certain quality on the second step.
3. **Anytime issue:** The exhaustive mining of all patterns requires to explore the search space in a certain fashion that extracts either very general patterns first (breadth-first search) or very similar patterns to each other (depth-first search). For having patterns regularly covering the search space, it is necessary to wait for finishing the extraction step before starting the model construction. As this first step is very time consuming, it prevents the user to have an immediate answer.

In order to develop an effective and anytime approach for pattern-based outlier detection, we propose not to have an exhaustive mining.
[modifier les contributions]

This paper revisits the calculation of the Frequent Pattern Outlier Factor (FPOF) by benefiting from recent pattern sampling techniques. We first propose a method for calculating the exact FPOF of each transaction. Surprisingly, our method is non-enumerative in the sense that no pattern is generated (a fortiori, this is also a non-exhaustive method). For this, we reformulate the FPOF by operating directly on transaction pairs. This method calculates the FPOF in polynomial time on the number of transactions and items of the dataset. Experiments show that this method manages to calculate the exact FPOF where the usual approach fails. We also propose a non-exhaustive approximate method that exploits a pattern sample instead

of the complete collection of frequent patterns. Using Bennett's inequality, this method selects the sample size so as to guarantee a maximum error for a given confidence. Experiments show its efficiency with reasonable error.

The outline of this paper is as follows. Section 2 reviews some related work about outlier detection, pattern sampling and anytime algorithms. Section 3 introduces the basic definitions about the FPOF and states the problem of its exact and anytime approximate calculation. In Section **??**, we propose our exact non-enumerative method for calculating FPOF. We introduce our approximate method based on sampling in Section 5. Section 6 provides experimental results. We conclude in Section 7.

2 Related Work

2.1 Pattern-based outlier detection

In this paper, we focus on the outlier detection methods based on frequent patterns [He et al., 2005, Otey et al., 2006, Koufakou et al., 2011]. A broader view of outlier detection is provided by surveys including [Hawkins, 1980]. Pattern-based methods benefit from the progress of pattern mining made over the past two decades. Such methods have a double interest. On the one hand, they are well suited to handle categorical data unlike most other methods dedicated to numerical data. In addition, they also remain efficient for high-dimensional spaces. The first approach [He et al., 2005] introduced the frequent pattern outlier factor that exploits the complete collection of frequent itemsets (while [Otey et al., 2006] uses an opposite approach by considering non-frequent itemsets). More recently, [Koufakou et al., 2011] replaces the collection of frequent itemsets by the condensed representation of Non-Derivable Itemsets (NDI) which is more compact and less expensive to mine. We would go further by showing that the frequent pattern outlier factor proposed in [He et al., 2005] can be calculated without extracting any pattern or by extracting a small sample.

[mettre ici un tableau avec la plupart des complexitÃ©s des autres approches pour montrer que celle-ci n'est pas moins efficace que les autres]

2.2 Pattern sampling

Recently, there has been a resurgence in pattern mining for non-exhaustive methods through pattern sampling [Chaoji et al., 2008, Boley et al., 2011].

Pattern sampling aims at accessing the pattern space $\mathcal{L}$ by an efficient sampling procedure simulating a distribution $\pi : \mathcal{L} \rightarrow [0, 1]$ that is defined with respect to some interestingness measure m: $\pi(.) = m(.)/Z$ where Z is a normalizing constant (formal framework and algorithms are detailed in [Boley et al., 2011]). In this way, the user has a fast and direct access to the entire pattern language and with no parameter (except possibly the sample size). Pattern sampling has been introduced to facilitate interactive data exploration [van Leeuwen, 2014]. In this paper, we investigate the use of pattern sampling for assigning an outlier score to each transaction. With a lower (pattern or time) budget than that of an exhaustive method, we obtain a higher quality with a bounded error.
[ajouter ici les mÃ©thodes utilisant le sampling pour construire des modÃ¨les (tiling)]

2.3 Anytime algorithm for pattern mining

3 Frequent Pattern Based Outlier Detection

3.1 Basic definitions

Let $\mathcal{I}$ be a set of distinct literals called *items*, an itemset (or a pattern) is a subset of $\mathcal{I}$. The language of itemsets corresponds to $\mathcal{L} = 2^{\mathcal{I}}$. A transactional dataset is a multi-set of itemsets of $\mathcal{L}$. Each itemset, usually called *transaction*, is a data observation. For instance, Table 1 gives three transactional datasets with 4 or 5 transactions t_i described by until 4 items A, B, C and D.

$\mathcal{D}$

Trans.	Items
t_1	A B
t_2	A B
t_3	A B
t_4	C

$\mathcal{D}'$

Trans.	Items
t_1	A B
t_2	A B
t_3	A B
t_4	C
t_5	**A** **B**

$\mathcal{D}''$

Trans.	Items
t_1	A B **D**
t_2	A B **D**
t_3	A B **D**
t_4	C

Table 1 Three toy datasets with slight variations

Pattern discovery takes advantage of interestingness measures to evaluate the relevancy of a pattern. The *support* of a pattern X in the dataset $\mathcal{D}$ is the proportion of transactions covered by X [Agrawal et al., 1994]: $supp(X, \mathcal{D}) = |\{t \in \mathcal{D} : X \subseteq t\}|/|\mathcal{D}|$. A pattern is said to be *frequent* when

its support exceeds a user-specified minimal threshold. The set of all frequent patterns for σ as minimal threshold in $\mathcal{D}$ is denoted by $\mathcal{F}_\sigma(\mathcal{D})$: $\mathcal{F}_\sigma(\mathcal{D}) = \{X \in \mathcal{L} : supp(X, \mathcal{D}) \geq \sigma\}$.

In the following, we manipulate pattern multisets which are collections of patterns admitting several occurrences of the same pattern. The representativeness of a pattern multiset $\mathcal{P}$, denoted by $Supp(\mathcal{P}, \mathcal{D})$, is the sum of the support of each pattern in $\mathcal{P}$: $Supp(\mathcal{P}, \mathcal{D}) = \sum_{X \in \mathcal{P}} supp(X, \mathcal{D})$. The range of $Supp(\mathcal{P}, \mathcal{D})$ is $[0, |\mathcal{P}|]$. Given a cardinality, high representativeness means the multiset contains very common patterns of the dataset. For comparing the content of two pattern multisets, we use the semi-join, denoted by $\mathcal{P}_2 \triangleright \mathcal{P}_1$, that returns all the patterns of $\mathcal{P}_2$ occurring in $\mathcal{P}_1$: $\mathcal{P}_2 \triangleright \mathcal{P}_1 = \{X \in \mathcal{P}_2 : X \in \mathcal{P}_1\}$. For instance, $\{A, AB, A, D\} \triangleright \{C, A, B\} = \{A, A\}$.

3.2 Frequent Pattern Outlier Factor

Intuitively, a transaction is more representative when it contains many patterns which are very frequent within the dataset. In contrast, an outlier contains only few patterns and these patterns are not very frequent. The frequent pattern outlier factor [He et al., 2005] formalizes this intuition:

Definition 1 (FPOF). The frequent pattern outlier factor of a transaction t in $\mathcal{D}$ is defined as follows:

$$fpof(t, \mathcal{D}) = \frac{Supp(2^t, \mathcal{D})}{\max_{u \in \mathcal{D}} Supp(2^u, \mathcal{D})}$$

The range of *fpof* is $[0, 1]$ where 1 means that the transaction is the most representative transaction of the dataset while a value near 0 means that the transaction is an outlier. Other normalizations (denominator) are possible like $Supp(\mathcal{L}, \mathcal{D})$ or $\sum_{t \in \mathcal{D}} Supp(2^t, \mathcal{D})$. Whatever the normalization method, two transactions remain ordered in the same way (so it does not affect the Kendall's tau that we use to evaluate our method). Under a certain Markov model, the score $fpof(t, \mathcal{D})$ is also the proportion of time that an analyst would dedicate to study the transaction t considering the collection of frequent itemsets [Giacometti et al., 2014].

In the first dataset provided by Table 1, t_1 is covered by $\emptyset$ ($supp(\emptyset, \mathcal{D}) = 1$) and, A, B and AB whose support equals to 0.75 ($Supp(\{\emptyset, A, B, AB\}, \mathcal{D}) = 3.25$) while t_4 is only covered by $\emptyset$ and C ($Supp(\{\emptyset, C\}, \mathcal{D}) = 1.25$). Consequently, $fpof(t_1, \mathcal{D}_1) = 3.25/3.25$ and $fpof(t_4, \mathcal{D}_1) = 1.25/3.25$. In this example, t_4 appears to be an outlier. It is easy to see that increasing the frequency of the patterns covering the first transactions (e.g., dataset $\mathcal{D}'$) decreases the FPOF of t_4. Similarly, increasing the number of patterns covering the first transactions also decreases the FPOF factor of t_4 (e.g., dataset $\mathcal{D}''$).

4 Problem Formulation

4.1 Exact FPOF computation

Given a dataset $\mathcal{D}$, the outlier detection problem consists in computing the FPOF for each transaction $t \in \mathcal{D}$. In practice, this exact calculation of frequent pattern outlier factor was performed by mining all patterns appearing at least once in the dataset (i.e., with $\sigma = 1/|\mathcal{D}|$). Of course, this expensive task is not possible for very large datasets. Recently, it has been demonstrated that the FPOF can be reformulated in order to calculate the exact FPOF in polynomial time.

To calculate the FPOF of a transaction t, Definition 1 formulates the problem in terms of frequent patterns appearing in t. The idea is to reformulate this factor by considering what each transaction u brings to the transaction t. For instance, in dataset $\mathcal{D}$, the FPOF of the first transaction relies on $Supp(\{\emptyset, A, B, AB\}, \mathcal{D})$ which is equal to $|\{\emptyset, A, B, AB, \quad \emptyset, A, B, AB, \quad \emptyset, A, B, AB, \quad \emptyset\}|/4$. Each subset $\{\emptyset, A, B, AB\}$ or $\{\emptyset\}$ results from the intersection of patterns covering t_1 with those covering another transaction $u \in \mathcal{D}$. Thereby, $Supp(\{\emptyset, A, B, AB\}, \mathcal{D}) = |\{\bigcup_{u \in \mathcal{D}} 2^{t_1} \cap 2^u\}|/|\mathcal{D}| = |\{\bigcup_{u \in \mathcal{D}} 2^{t_1 \cap u}\}|/|\mathcal{D}|$. Given a dataset $\mathcal{D}$, this observation leads to reformulate the frequent pattern outlier factor as follows for all transaction $t \in \mathcal{D}$:

$$fpof(t, \mathcal{D}) = \frac{\sum_{u \in \mathcal{D}} 2^{|t \cap u|}}{\max_{v \in \mathcal{D}} \sum_{u \in \mathcal{D}} 2^{|v \cap u|}}$$

From a conceptual point of view, it is interesting to note that ultimately, the FPOF of a transaction is just the sum of its similarity with each of transactions (where similarity between t and u is $2^{|t \cap u|}$). This measure is therefore very close to traditional methods relying on pair-wise distance among data observations.

$\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\lvert\mathcal{I}\rvert$	Exh. Time (s)	Non-enum. Time (s)
`chess`	3,196	75	439.5	**1.1**
`connect`	67,557	129	748.5	**577.7**
`mushroom`	8,124	119	**0.4**	5.9
`pumsb`	49,096	7,117	time out	**1,970.5**
`retail`	88,162	16,470	**8.7**	5,969.9
`sick`	2,800	58	0.8	**0.5**

Table 2 Time comparison of the different methods

Table 2 reports the running time required for calculating the exact FPOF using the exhaustive and the non-enumerative methods (respectively the 4th and the 5th column). Note that the exact exhaustive method (as baseline) benefits from LCM which is one of the most recognized frequent itemset mining

algorithm. The non-enumerative method is effective and rivals the exact exhaustive one. Its main advantage is to calculate the exact FPOF with datasets where the exact exhaustive method fails (e.g., `pumsb` where the execution was aborted after 5h).

However, even with a polynomial method, the exact calculation remains time-consuming. It is clear that the exact FPOF calculation cannot be guaranteed in a short time response. Then, it makes sense to approximate the FPOF.

4.2 Anytime approximate FPOF computation

Instead of using the complete collection of patterns, FPOF is usually approximated with a collection of frequent patterns i.e., with a higher minimal support threshold:

Definition 2 (σ-Exhaustive FPOF). Given a minimal support threshold σ, the σ-exhaustive FPOF of a transaction t in $\mathcal{D}$ is defined as follows:

$$fpof_\sigma(t,\mathcal{D}) = \frac{Supp(\mathcal{F}_\sigma(\mathcal{D}) \triangleright 2^t, \mathcal{D})}{\max_{u\in\mathcal{D}} Supp(\mathcal{F}_\sigma(\mathcal{D}) \triangleright 2^u, \mathcal{D})}$$

[Variante: prendre les k motifs les plus frÃ©quents pour calculer l'approximation]

The approximation becomes accurate with very low minimal support thresholds. Figure 1 plots on the left-hand side, the Kendall's tau of $fpof_\sigma$ in comparison with *fpof* for some benchmarks[1]. Unfortunately, when the minimal support threshold becomes very low, the number of patterns (see the right plot of Figure 1) and the extraction time explode. Furthermore, the approximation error is not estimated. With a smaller (time or pattern) budget, we claim that it is possible to approximate more precisely FPOF while having a bound on the error.

The Kendall's tau varies significantly depending on the dataset for a same minimal support threshold. This threshold is not easy to fix for obtaining a good compromise between efficiency and quality. Therefore, it seems interesting that the user sets the maximum error that he/she tolerates on the result rather than a threshold related to the method:

Problem 1 (Approximate Problem). Given a dataset $\mathcal{D}$, two reals δ and ϵ, find a function approximating the frequent pattern outlier factor such that for each transaction $t \in \mathcal{D}$ $|fpof(t,\mathcal{D}) - \widetilde{fpof}(t,\mathcal{D})| \leq \epsilon$ with confidence $1-\delta$.

[1] It is the proportion of pairs of transactions which would be ranked similarly with the approximate FPOF and with the true FPOF (see Section 6 for a formal definition).

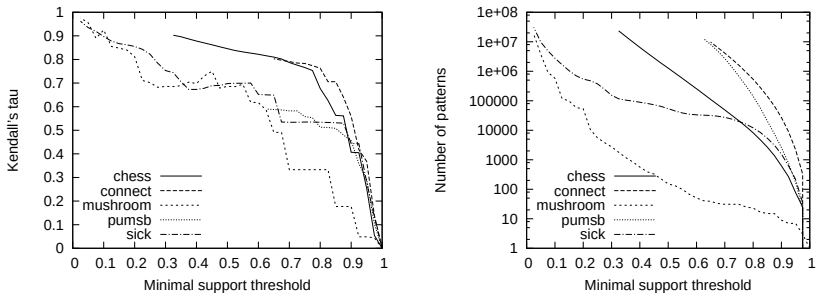

Fig. 1 Kendall's tau and number of patterns with minimal support threshold

Algorithm 1 Support-based Sampling [Boley et al., 2011]

Input: A dataset $\mathcal{D}$
Output: A random itemset $X \sim supp(\mathcal{L}, \mathcal{D})$
1: Let weights w be defined by $w(t) = 2^{|t|}$ for all $t \in \mathcal{D}$
2: Draw a transaction $t \sim w(\mathcal{D})$
3: **return** an itemset $X \sim u(2^t)$

This problem aims at assigning an approximate FPOF to each transaction with the probability $1 - \delta$ that the error is less than ϵ. Problem **??** is a particular case of Problem 1 by fixing $\epsilon = 0$ and $\delta = 0$.

5 Anytime Sampling Method

This section addresses Problem 1 by using pattern sampling. First, we propose a method for approximating FPOF from a pattern sample drawn according to frequency. Then we show how to choose the sample size to control the error.

5.1 Pattern Sampling for FPOF

In Section 4, we showed that the use of the most frequent patterns is insufficient to approximate accurately FPOF. The most frequent patterns do not measure the singularity of each transaction that also relies on more specific patterns (whose frequency varies from small to average). Conversely do not considering frequent patterns would also be a mistake because they contribute significantly to FPOF. A reasonable approach is to select patterns randomly with a probability proportional to their weight in the calculation of

FPOF. Typically, in the dataset $\mathcal{D}$ of Table 1, the itemset AB is 3 times more important than itemset C in the calculation of FPOF due to their frequency.

In recent years pattern sampling techniques have been proposed to randomly draw patterns in proportion to their frequency [Boley et al., 2011] (see Algorithm 1). Such approaches are ideal to bring us a well-adapted collection of patterns. Of course, it remains the non-trivial task of approximating FPOF starting from this collection. This is what provides the following definition:

Definition 3 (k-Sampling FPOF). Given an integer $k > 0$, a k-sampling frequent pattern outlier factor of a transaction t in $\mathcal{D}$ is defined as follows:

$$fpof_k(t, \mathcal{D}) = \frac{|\mathcal{S}_k(\mathcal{D}) \triangleright 2^t|}{\max_{u \in \mathcal{D}} |\mathcal{S}_k(\mathcal{D}) \triangleright 2^u|}$$

where $\mathcal{S}_k(\mathcal{D})$ is a sample of k patterns drawn from $\mathcal{D}$ according to support: $\mathcal{S}_k(\mathcal{D}) \sim supp(\mathcal{L}, \mathcal{D})$.

It is important to note that $|.|$ is used here instead of $Supp(., \mathcal{D})$ as in Definition 1. As the sampling technique already takes into account the frequency when it draws patterns, it is not necessary to involve the support here. Indeed, the draw is with replacement for the correct approximation of FPOF (without this replacement the most frequent patterns would be disadvantaged). It induces that the same pattern can have multiple occurrences within the sample $\mathcal{S}_k(\mathcal{D})$.

For the same sample size k and for the same transaction t, it is possible to calculate different values of a k-sampling FPOF due to $\mathcal{S}_k(\mathcal{D})$. But, the higher the threshold k, the less the difference between values stemming from two samples is high. Furthermore, the greater the sample size k, the better the approximation:

Property 1 (Convergence). Given a dataset $\mathcal{D}$, a k-sampling FPOF converges to the FPOF for all transaction $t \in \mathcal{D}$.

Proof. $\mathcal{S}_k(\mathcal{D}) \sim supp(\mathcal{L}, \mathcal{D})$ means that there exists a constant $\alpha > 0$ such that $\forall X \in \mathcal{L}$, $\lim_{k \to \infty} |\mathcal{S}_k(\mathcal{D}) \triangleright \{X\}| = \alpha supp(X, \mathcal{D})$. Then, for each transaction t, we obtain that: $\lim_{k \to \infty} |\mathcal{S}_k(\mathcal{D}) \triangleright 2^t| = \alpha \sum_{X \in 2^t} supp(X, \mathcal{D}) = \alpha Supp(2^t, \mathcal{D})$. By injecting this result into Definition 3, we conclude that Property 1 is right. □

Beyond convergence, the interest of this approach is the speed of convergence far superior to that of the σ-exhaustive frequent pattern outlier factor as shown in the experimental study (see Section 6). This speed is accompanied by a good efficiency due to a reasonable complexity of pattern sampling:

Property 2 (Complexity). A k-sampling FPOF of all transactions can be calculated in time $O(k \times |\mathcal{I}| \times |\mathcal{D}|)$.

Algorithm 2 Anytime FPOF Computation

Input: A dataset $\mathcal{D}$, a confidence $1-\delta$
Output: A k-sampling frequent pattern outlier factor of all transactions in $\mathcal{D}$ with an error bounded by ϵ for a confidence $1-\delta$

$\tilde{\epsilon} \leftarrow 1$; $\mathcal{S} \leftarrow \emptyset$
repeat
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{X\}$ where $X \sim supp(\mathcal{L}, \mathcal{D})$ *// add a pattern in the sample*
  $m \leftarrow \arg\max_{t \in \mathcal{D}} cov_{\mathcal{S}}(t)$ *// select the most covered transaction*
  // estimate the maximal error on $cov_{\mathcal{S}}$
  $e_t \leftarrow \sqrt{2\overline{\sigma_t}\ln(1/\delta)/|\mathcal{S}|} + \ln(1/\delta)/(3|\mathcal{S}|)$ for each $t \in \mathcal{D}$
  // estimate the maximal error on FPOF
  $\tilde{\epsilon} \leftarrow \max_{t \in \mathcal{D}}\{\min\{1; (cov_{\mathcal{S}}(t) + e_t)/(cov_{\mathcal{S}}(m) - e_m)\} - cov_{\mathcal{S}}(t)/cov_{\mathcal{S}}(m)\}$
  $\tilde{\epsilon} \leftarrow \max_{t \in \mathcal{D}}\{cov_{\mathcal{S}}(t)/cov_{\mathcal{S}}(m) - \max\{0; (cov_{\mathcal{S}_k}(t) - e_t)/(cov_{\mathcal{S}}(m) + e_m)\}; \tilde{\epsilon}\}$
  Print the estimated maximal bound as feedback
until The user stops the process
return $\langle cov_{\mathcal{S}}(t)/\max_{u \in \mathcal{D}} cov_{\mathcal{S}}(u)\rangle_{t \in \mathcal{D}}$

Proof. Pattern sampling according to frequency is performed in time $O(|\mathcal{I}| \times |\mathcal{D}| + k(|\mathcal{I}| + \ln|\mathcal{D}|))$ [Boley et al., 2011] and the FPOF of all transactions consists in computing the transactions containing each sampled pattern. Thus, it is calculated in time $O(k \times |\mathcal{I}| \times |\mathcal{D}|)$. □

Given a number of patterns k, a k-sampling FPOF is therefore effective to calculate. However, the choice of the sample size k is both difficult and essential in order to achieve a desired approximation as suggested by Problem 1. The next section presents an iterative method for fixing this sample size.

5.2 Bounding the Error

This section shows how to determine the right sample size k for computing a k-sampling FPOF satisfying user specified parameters (a maximum error with a given confidence). The idea is to draw a sample and to bound the maximum error of FPOF using a statistical result known as Bennett's inequality. If this error is less than that allowed by the user, the algorithm returns a sampling FPOF based on the current sample. Otherwise, it increases the sample size by drawing more patterns and so on.

We use Bennett's inequality to estimate the current error because it is true irrespective of the probability distribution. After k independent observations of real-valued random variable r with range $[0, 1]$, Bennett's inequality ensures that, with confidence $1-\delta$, the true mean of r is at least $\overline{r} - \epsilon$ where $\overline{r}$ and $\overline{\sigma}$ are respectively the observed mean and variance of the samples and

$$\epsilon = \sqrt{\frac{2\overline{\sigma}\ln(1/\delta)}{k}} + \frac{\ln(1/\delta)}{3k}$$

In our case, the random variable is the average number of patterns within a sample $\mathcal{S}_k \sim supp(\mathcal{L}, \mathcal{D})$ that cover the transaction t. It is denoted by $cov_{\mathcal{S}_k}(t)$ and defined as follows: $cov_{\mathcal{S}_k}(t) = |\mathcal{S}_k \triangleright 2^t|/k$. It is easy to see that a k-sampling FPOF factor can be rewritten using $cov_{\mathcal{S}_k}$: $fpof_k(t, \mathcal{D}) = cov_{\mathcal{S}_k}(t)/\max_{u \in \mathcal{D}} cov_{\mathcal{S}_k}(u)$. Using Bennett's inequality and the above definition enables us to bound FPOF:

Property 3 (FPOF Bound). Given a dataset $\mathcal{D}$ and confidence $1-\delta$, the FPOF of transaction t is bounded as follows:

$$\max\left\{0, \frac{cov_{\mathcal{S}_k}(t) - \epsilon_t}{cov_{\mathcal{S}_k}(u) + \epsilon_u}\right\} \leq fpof(t, \mathcal{D}) \leq \min\left\{\frac{cov_{\mathcal{S}_k}(t) + \epsilon_t}{cov_{\mathcal{S}_k}(u) - \epsilon_u}, 1\right\}$$

where $\mathcal{S}_k \sim supp(\mathcal{L}, \mathcal{D})$, $u = \arg\max_{v \in \mathcal{D}} cov_{\mathcal{S}_k}(v)$ and $\epsilon_t = \sqrt{2\overline{\sigma_t}\ln(1/\delta)/k} + \ln(1/\delta)/(3k)$.

Algorithm 2 returns the approximate FPOF of all transactions by guaranteeing a bounded error of ϵ with confidence $1-\delta$. Basically, the main loop is iterated until that the maximal error $\tilde{\epsilon}$ is inferior to the expected bound ϵ. Lines 4-7 calculate the maximal error $\tilde{\epsilon}$ using Property 3. If the maximal error is less than ϵ, Line 9 returns the k-sampling FPOF with the current sampling $\mathcal{S}$. Otherwise, one more pattern is drawn (Line 3).

As desired by Problem 1, Algorithm 2 approximates the FPOF of all transactions:

Property 4 (Correctness). Given a dataset $\mathcal{D}$, a confidence $1-\delta$, a bound ϵ, Algorithm 2 returns a k-sampling frequent pattern outlier factor of a transaction t in $\mathcal{D}$ approximating FPOF with an error bounded by ϵ with confidence $1-\delta$.

6 Experimental Study

This experimental study aims to compare the speed of the non-enumerative method with that of the exact exhaustive method (i.e., $1/|\mathcal{D}|$-exhaustive) and to estimate the error quality of the ϵ-approximate sampling method faced to the σ-exhaustive method. Due to the lack of space, we do not provide new experiments showing the interest of FPOF for detecting outliers as this aspect is already detailed in literature [He et al., 2005, Otey et al., 2006, Koufakou et al., 2011]. Experiments are conducted on datasets coming from the UCI Machine Learning repository (`archive.ics.uci.edu/ml`) and the FIMI repository (`http://fimi.ua.ac.be/`). Table 2 gives the main features of datasets in first columns. All experiments are performed on a 2.5 GHz Xeon processor with the Linux operating system and 2 GB of RAM memory. Each reported evaluation measure is the arithmetic mean of 10 repeated measurements (interval confidence are narrow enough to be omitted).

6.1 Approximate Sampling Method

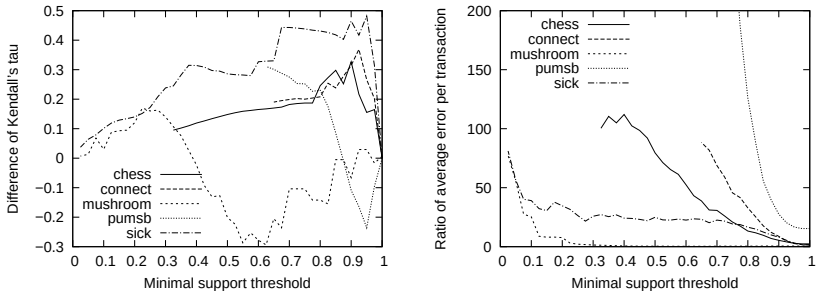

Fig. 2 Error of σ-exhaustive FPOF and k-sampling FPOF with σ

This section compares our sampling method with the traditional heuristic based on frequent patterns as baseline. For this purpose, we use the Kendall's tau for comparing the ranking stemming from an approximate method f with that stemming from the FPOF (calculated with an exact method):

$$\tau(f,\mathcal{D}) = \frac{|\{(t,u) \in \mathcal{D}^2 : sgn(f(t,\mathcal{D}) - f(u,\mathcal{D})) = sgn(fpof(t,\mathcal{D}) - fpof(u,\mathcal{D})\}|}{|\mathcal{D}|^2}$$

and in the same way, we also compute the average error per transaction: $\varepsilon(f,\mathcal{D}) = \sum_{t\in\mathcal{D}} |f(t,\mathcal{D}) - fpof(t,\mathcal{D})|/|\mathcal{D}|$.

The left chart of Figure 2 plots the difference between the Kendall's tau of k-sampling FPOF and that of σ-exhaustive FPOF (when the curve is above 0, it means that the ranking of k-sampling FPOF is better than the ranking of σ-exhaustive FPOF). The right chart reports the average error of σ-sampling FPOF divided by that of k-sampling FPOF (when the curve is above 1, it means that the average error of k-sampling FPOF is smaller than that of σ-exhaustive FPOF). For each point, the minimal support threshold σ is used as parameter of the σ-exhaustive FPOF method. At the same time, the sample size k is fixed with the number of patterns mined with the minimal support threshold σ: $k = |\mathcal{F}_\sigma(\mathcal{D})|$. The k-sampling FPOF is clearly more accurate than the σ-exhaustive FPOF for a same pattern budget. For some datasets (e.g., `chess` or `sick`), the difference is always greater than 0. For other datasets, as soon as the number of patterns in the sample increases, the difference becomes positive. The average error ratio clearly shows that our method gives a better approximation of FPOF (especially, when the number of patterns is high).

Figure 3 plots the number of patterns and the average error per transaction of the ϵ-approximate method with bound ϵ (for $\delta = 0.1$). As expected, the smaller the error bound ϵ, the greater the number of patterns in the sample. Therefore, the longer the execution time. It is interesting to note that

the approximate method (with $\epsilon = 0.1$) is regularly faster than the exact methods (see Table 2). Finally, the real average error per transaction of the approximate method is always much lower than the requested bound ϵ (e.g., $\epsilon = 0.1$ actually gives an error less than 0.01). This difference results from the Bennett's inequality that makes no assumption about the distribution.

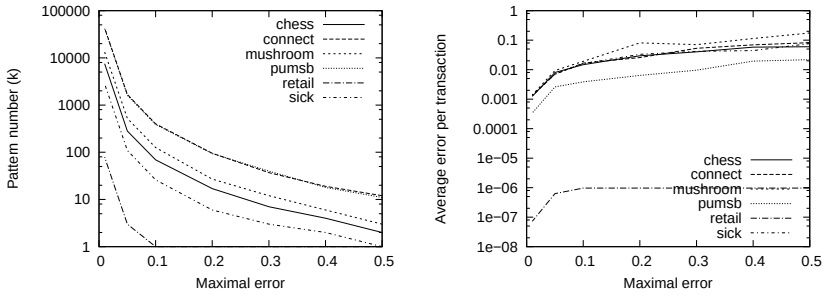

Fig. 3 Number of patterns and average error per transaction with maximum error ϵ

7 Conclusion

We revisited the FPOF calculation in extracting the least possible patterns or no pattern. Despite this constraint, the proposed exact method has a complexity better suited to certain datasets. Our approximate method using a sampling technique provides additional guarantees on the result with a maximum bound on the error. The experiments have shown the interest of these two approaches in terms of speed and accuracy compared to the usual exhaustive approach where all frequent patterns are mined.

Our proposal therefore combines the proven power of pattern-based methods by adding a guarantee on the quality of results without sacrificing speed thanks to sampling techniques. We think it can be generalized to other measures involving patterns or pattern-based models. We would also like to adapt this approach to build anytime algorithms. In the case of FPOF, it consists in extending the pattern sample indefinitely until the end-user wants to stop the process. Then, the algorithm returns the FPOF achieved with the current sample while estimating its error.

Acknowledgements. This work has been partially supported by the Prefute project, PEPS 2015, CNRS.

References

Agrawal et al., 1994. Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *International conference on Very Large Data Bases*, volume 1215, pages 487–499.

Boley et al., 2011. Boley, M., Lucchese, C., Paurat, D., and Gärtner, T. (2011). Direct local pattern sampling by efficient two-step random procedures. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 582–590.

Chaoji et al., 2008. Chaoji, V., Hasan, M. A., Salem, S., Besson, J., and Zaki, M. J. (2008). ORIGAMI: A novel and effective approach for mining representative orthogonal graph patterns. *Statistical Analysis and Data Mining*, 1(2):67–84.

Giacometti et al., 2014. Giacometti, A., Li, D. H., and Soulet, A. (2014). Balancing the analysis of frequent patterns. In *Advances in Knowledge Discovery and Data Mining*, pages 53–64. Springer.

Hawkins, 1980. Hawkins, D. M. (1980). *Identification of outliers*, volume 11. Springer.

He et al., 2005. He, Z., Xu, X., Huang, Z. J., and Deng, S. (2005). FP-outlier: Frequent pattern based outlier detection. *Computer Science and Information Systems*, 2(1):103–118.

Knobbe et al., 2008. Knobbe, A., Crémilleux, B., Fürnkranz, J., and Scholz, M. (2008). From local patterns to global models: The lego approach to data mining. In *From local patterns to global models: proceedings of the ECML PKDD 2008 Workshop*, pages 1–16.

Koufakou et al., 2011. Koufakou, A., Secretan, J., and Georgiopoulos, M. (2011). Non-derivable itemsets for fast outlier detection in large high-dimensional categorical data. *Knowledge and information systems*, 29(3):697–725.

Liu et al., 1998. Liu, B., Hsu, W., and Ma, Y. (1998). Integrating classification and association rule mining. In *International conference on Knowledge Discovery and Data mining*.

Liu and Dong, 2012. Liu, Q. and Dong, G. (2012). CPCQ: contrast pattern based clustering quality index for categorical data. *Pattern Recognition*, 45(4):1739–1748.

Otey et al., 2006. Otey, M. E., Ghoting, A., and Parthasarathy, S. (2006). Fast distributed outlier detection in mixed-attribute data sets. *Data Mining and Knowledge Discovery*, 12(2-3).

van Leeuwen, 2014. van Leeuwen, M. (2014). Interactive data exploration using pattern mining. In *Interactive Knowledge Discovery and Data Mining in Biomedical Informatics*, pages 169–182. Springer.