Mining Sequential Patterns on Data Streams: A Near-Optimal Statistical Approach

P.A. Laur[1], J.E. Symphor[1], R. Nock[1], and P. Poncelet[2]

[1] Univ. Antilles-Guyanne/DSI/GRIMAAG, B.P. 7209
97275 Schoelcher Cedex, France
Email: {palaur,je.symphor,rnock}martinique.univ-ag.fr
[2] EMA-LGI2P/Site EERIE, Parc Scientifique Georges Besse
30035 Nîmes cedex 1 - France
Email: Pascal.Poncelet@ema.fr

Abstract. When we mine for sequential patterns on a whole data stream it's necessary to cope with uncertainty as only a part of the stream is stored. Even if different approaches exist for mining sequential patterns in static databases they are not suitable for this context. We evaluate a statistical technique which biases the estimation of the support of sequential patterns, so as to maximize either the precision or the recall, as chosen by the user, and limits the degradation of the other criterion. Experiments with databases demonstrate the potential of such technique.

1 Introduction

A data stream is an ordered sequence of itemsets that arrives in timely manner. Such data (e.g. stock tickers, telecom records, network traffic measurements, click streams, sensor networks) are very useful for many emerging and real applications. Different efficient approaches have been proposed to mine sequential patterns on static databases [1, 6, 20]. They are usually based on Generating Pruning techniques which are irrelevant when considering streaming data [5]. Furthermore, mining either itemsets or sequential patterns when the database is subject to be updated regularly, maintaining frequent patterns has been addressed by various incremental algorithms, in order to mine frequent itemsets without having to build everything from scratch [21]. Nevertheless, they suffer the same drawbacks as traditional approaches. Recently, in order to handle efficiently such streaming data, new data mining approaches have been proposed. They mainly focus on finding all frequent items or itemsets over the entire history of a streaming data. The first approach was proposed by [10] where they define the first single-pass algorithm. Li et al. [9] use a top-down frequent itemset discovery scheme. A regression-based algorithm is proposed in [16] to find frequent itemsets in sliding windows. Chi et al. [2] consider closed frequent itemsets. In [5], they propose a FP-tree-based algorithm [6] to mine frequent itemsets at multiple time granularities by a novel tilted-time windows technique.
All these approaches consider inter-transaction associations, i.e. there is no limitation on order of events. In this paper we consider that itemsets are really

ordered into the streams, *i.e.* there is a strict order of events. Therefore, we are interested in mining sequential patterns rather than itemsets. So such approaches are not suitable for this task. The first problem we are facing is algorithmic, as in static database the search space for sequential patterns is much more bigger than the one of itemsets.

The second problem, we are facing is typically statistical. As we can mine information from only a fragment of the stream, we propose an approach to forecast the stream future with some guarantees on this prediction. For example, it is not enough to observe some sequential patterns as frequent (similarly infrequent) in the data stored; it is much more important to *predict* if it is really frequent (similarly infrequent) in the whole data stream. Our guarantees are typically the maximization (with high probability) of some user-fixed criteria; they do not rely on optimizing *the estimation* of these criteria (utility functions), like for example in [4], [13]. The bounds we obtain are also finer and more applicable than previous ones that could be used in our case [17].

The remainder of the paper is organized as follows. Section 2 goes deeper into presenting the problem statement. Section 3 presents our solution to this problem. Section 5 reports the result of our experiments. In Section 6, we summarize our findings and conclude the paper with future avenues for research.

2 Problem Statement

The problem of mining frequent sequential patterns was previously introduced in [15] and extended in [14]: Let $I = \{i_1, i_2 ... i_m\}$ be a set of literals called items. An itemset is a non-empty set of items. A sequence s is a set of itemsets ordered according to their time stamp. It is denoted by $< s_1\ s_2\ ...s_n >$, where $s_j, j \in 1...n$, is an itemset. A k-sequence is a sequence of k itemsets. Let DB be a set of transactions where each transaction T consists of customer-id, transaction time and an itemset. A sequence $S' =< s'_1\ s'_2\ ...\ s'_n >$ is a subsequence of another sequence $S =< s_1\ s_2\ ...\ s_m >$if there exist integers $i_1 < i_2 < ... i_j ... < i_n$ such that $s'_1 \subseteq s_{i_1}$, $s'_2 \subseteq si_2$, ... $s'_n \subseteq si_n$. All transactions from the same customer are grouped together, sorted by increasing timestamp order and called a *data sequence*. A data sequence contains a sequence S if it is a subsequence of the data sequence. A support value (denoted $supp(S)$) for a sequence gives its number of actual data sequence that contain it in DB. A sequence is *frequent* if $supp(S) \geq \theta \times |DB|$ where $\theta \in (0, 1)$ is a user-specified minimum support. The problem of mining frequent sequences is to mine all sequences whose support is greater than, or equal to $\theta \times |DB|$. Each of them is called a frequent *sequential pattern*. Meanwhile in the case of data stream we have a partial storage of DB. Let us now assume that we are provided with a *data stream*. Let $DS = B_0, B_1$, ..., B_n, ..., be an infinite sequence of batches, where each batch is associated with a timestamp t, i.e. B_t, and n is the identifier of the "latest" batch B_n. Each batch B_i consists of a set of customer data sequences; that is, $B_i = [S_1, S_2, S_3, ..., S_k]$. The length ($L$) of the data stream is defined as $L = |B_0| + |B_1| + ... + |B_n|$

where $|B_i|$ stands for the cardinality of the set B_i. In this context, the support is defined as follows.

Definition 1. *The support of a sequence S at a specific time t is the ratio of the number of customers data sequence containing S in the current time window to the total number of customers data sequences.*

Therefore, the problem of sequential patterns in data streams is to find frequent patterns S, i.e. verifying $\sum_{t=0}^{L} \text{support}_t(S) >= \theta \times L$. Given the nature of the streaming data, there are two sources of error when estimating frequent sequential patterns:

1. it is possible that some sequences observed as frequent might in fact not be frequent anymore from a longer observation of the data stream;
2. on the other hand, some sequences observed as not frequent may well in fact be frequent from a longer history of the data stream.

Should it rely on frequencies estimations, any loss due to the imperfection of the information stored is incurred by at least one of these two sources of error. It is statistically hard to nullify both of them from a subset, even very large, of the whole data stream [18]. It is also generally impossible to capture the missing information from the data stream to make a fully accurate prediction. The problem we address in this paper is how to obtain a solution to the following problem while mining a data stream for sequential patterns:

(a) the user chooses a source of error, and fixes some related parameters;
(b) the source of error chosen is nullified with high probability, while the other one incurs a limited loss.

3 Our approach

The data stream is supposed to be obtained from the repetitive sampling of a potentially huge *domain* X which contains all possible data sequences, see figure 1 (a). Each sequence is sampled independently through a distribution $\mathcal{D}$, see figure 1 (b), for which we make absolutely *no* assumption, except that it remains fixed: there is no distribution drift through time. The reader may find relevant empirical studies on concept drift for supervised mining in [19]. The user specifies a real $0 < \theta < 1$, the *theoretical* support, and ideally wishes to recover all the *true* θ-frequent sequential patterns of X. This set is called X_θ.

Definition 2.

$$\forall 0 \leq \theta \leq 1, X_\theta = \{T \in X : \rho_X(T) \geq \theta\} \ , \tag{1}$$

with $\rho_X(T) = \sum_{T' \in X : T \leq_t T'} \mathcal{D}(T')$, and $T \leq_t T'$ means that T generalizes T'.

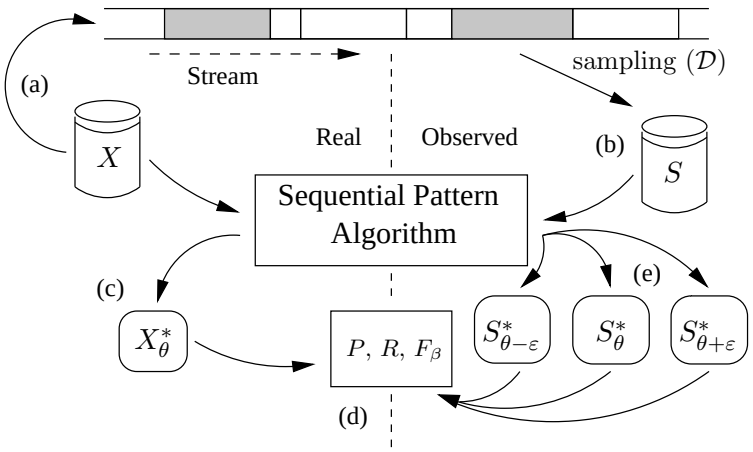

Fig. 1. Our framework.

The recovery of X_θ faces two problems. Apart from our statistical estimation problem, there is a combinatorial problem which comes from the fact that X is typically huge. The set of observed data sequences which we have sampled from X in the data stream (S) has a size $|S| = m$ ($|S| << |X|$). In our framework, we usually reduce this difference with some algorithm returning a superset S^* of S, having size $|S^*| = m^* > m$. Typically, S^* contains additional generalizations of the elements of S. The key point is that S^* is usually still not large enough to cover X_θ, regardless of the way it is built, so that the pregnancy of our statistical estimation problem remains the same.

Our statistical estimation problem can be formalized as follows:

- approximate as best as possible the set $X^*_\theta = X_\theta \cap S^*$, for any S and S^* (see figures 1 (c) and 2).

Remark that $\forall T \in S^*$, we cannot compute exactly $\rho_X(T)$, since we do not know X and $\mathcal{D}$. Rather, we have access to its best unbiased estimator, $\rho_S(T) = \sum_{T' \in S : T \leq_t T'} w(T')$ ($\forall T \in S^*$). Here, $w(T')$ is the weight (observed frequency) of T' in S. We adopt the following approach to solve our problem:

- find some $0 < \theta' < 1$ and approximate the set X^*_θ by the set of *observed* θ'-frequent of S^*, that is:

$$S^*_{\theta'} = \{T \in S^* : \rho_S(T) \geq \theta'\} \ . \tag{2}$$

Before computing θ', we first turn to the formal criteria appreciating the goodness-of-fit of $S^*_{\theta'}$, see figure 1 (d). The two sources of error, committed with respect to X^*_θ, come from the two subsets of the symmetric difference with $S^*_{\theta'}$, as presented in Figure 2. To quantify them, let us define:

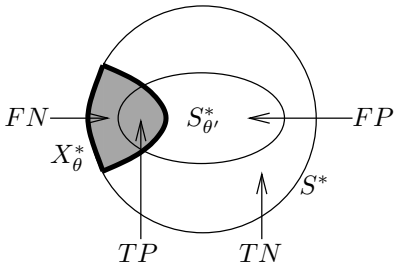

Fig. 2. The error estimation.

$$TP = \sum_{T \in S^*_{\theta'} \cap X^*_\theta} \mathcal{D}(T) \qquad (3) \qquad FN = \sum_{T \in X^*_\theta \setminus S^*_{\theta'}} \mathcal{D}(T) \qquad (5)$$

$$FP = \sum_{T \in S^*_{\theta'} \setminus X^*_\theta} \mathcal{D}(T) \qquad (4) \qquad TN = \sum_{T \in S^* \setminus (S^*_{\theta'} \cup X^*_\theta)} \mathcal{D}(T) \qquad (6)$$

The *precision* allows to quantify the proportion of estimated θ-frequent that are in fact not true θ-frequents, out of $S^*_{\theta'}$:

$$\mathtt{P} = TP/(TP + FP) \ . \qquad (7)$$

Maximizing P leads to minimize our first source of error. Symmetrically, the *recall* allows to quantify the proportion of true θ-frequent that are missed in $S^*_{\theta'}$:

$$\mathtt{R} = TP/(TP + FN) \ . \qquad (8)$$

Maximizing R leads to minimize our second source of error. We also make use of a well known quantity in information retrieval, which is a weighted harmonic average of precision and recall, the $\mathtt{F}_\beta$-measure. Thus, we can adjust the importance of one source of error against the other by adjusting the β value:

$$\mathtt{F}_\beta = (1 + \beta^2)\mathtt{PR}/(\mathtt{R} + \beta^2\mathtt{P}) \ , \qquad (9)$$

A naive approach to approximate X^*_θ would typically be to fix $\theta' = \theta$. Unfortunately, the main and only interesting property of $S^*_{\theta'}$ is that it converges with probability 1 to X^*_θ as $m \to \infty$ from Borel-Cantelli Lemma [3]. Glivenko-Cantelli's Theorem gives a rate of convergence as a function of m, but it does not give the possibility to maximize P or R.

4 Choosing θ'

Informally, our approach boils down to picking a θ' different from θ, so as to maximize either P or R. Clearly, extremal values for θ' would do the job, but they would yield very poor values for F_β, and also be completely useless for data

mining purposes. For example, we could choose $\theta' = 0$, and would obtain $S^*_0 = S^*$, and thus $\mathtt{R} = 1$. However, in this case, we would also have $\mathtt{P} = |X^*_\theta|/|S^*|$, a too small value for many domains and values of θ, and we would also keep all elements of S^* as true θ-frequents sequential patterns, a clearly huge drawback for mining issues. We could also choose $\theta' = 1$, so as to be sure to maximize $\mathtt{P}$ this time; however, we would also have $\mathtt{R} = 0$, and would keep *no* element of S^* as θ-frequent sequential patterns.

These extremal examples show the principle of our approach. Should we want to maximize $\mathtt{P}$, we would pick a θ' larger than θ to guarantee with high probability that $\mathtt{P} = 1$, yet while keeping large enough values for $\mathtt{R}$ (or F_β), and a set $S^*_{\theta'}$ not too small to contain significant informations. There is obviously a statistical barrier which prevents θ' to be too close to θ to keep the constraint $\mathtt{P} = 1$. The objective is to be the closest to this barrier, which statistically guarantees the largest $\mathtt{R}$ values under the constraint. The following Theorem holds regardless of the domain, the distribution of the data sequences, the size of S^*, or the user-fixed parameters (support, statistical risk). It relies *only* on independence for sampling data sequences.

Theorem 1. $\forall X, \forall \mathcal{D}, \forall m > 0, \forall 0 \leq \theta \leq 1, \forall 0 < \delta \leq 1$, *suppose we pick* ε *satisfying* $\varepsilon \geq \sqrt{(1/2m)\ln(|S^*|/\delta)}$. *If we fix* $\theta' = \theta + \varepsilon$ *in eq. (2), then* $\mathtt{P} = 1$ *with probability at least* $1 - \delta$. *If we fix* $\theta' = \theta - \varepsilon$ *in eq. (2), then* $\mathtt{R} = 1$ *with probability at least* $1 - \delta$.

(proof uses standard tools on concentration inequalities [12]; omitted due to the lack of space). The main point is that the values of θ', see figure 1 (e), are in fact close (*i.e.* up to negligible factirs) to the statistical barriers [18, 8] that still guarantee maximal values for $\mathtt{P}$ or $\mathtt{R}$. This is where lies the near-optimality of our approach, formalized in the following Theorem (proof uses a careful analysis of the tail of the binomial distribution; omitted due to the lack of space).

Theorem 2. $\exists X, \exists \mathcal{D}, \exists m > 0, \exists \delta > 0$ *s. t.* $\forall 1/2 < \theta \leq 1$, *with probability* $> \delta$, $\mathtt{P} \neq 1$ *and* $\mathtt{R} \neq 1$ *for any* $\varepsilon \leq c\sqrt{(\theta(1-\theta)/m)\ln(|X^*_\theta|/\delta)}$, *with* $c > 0$ *a constant.*

A previous Chernoff-type analysis, due to [17], may be fit to handling data streams as well, but for slightly more restricted problems; in particular while some of the bounds would typically not be applicable for large S^*, the others would be mainly addressed at controlling the precision of the support estimation, and not the maximization of our criteria (precision or recall).

5 Experiments

We focus on evaluating how our statistical support can be helpful to mine sequential patterns on a data stream, given a fragment of this stream. For this purpose, we use the previously defined measures: $\mathtt{P}$ (7), $\mathtt{R}$ (8) and F_β (9).

We have chosen two Web log databases and a sequential pattern algorithm `PSP` [11]. The first database named "Dragons"[3] (132k transactions, 2,54Go),

[3] www.elevezundragon.com.

Database	θ	sampling1	sampling2
Dragons	[.07, .2] / .03	[.02,.1] / .01	[.15, .7] /.05
BuAG	[.08, .2] / .03	[.05,.1] / .01	[.15, .7] /.05

Fig. 3. Range of parameters for the experiments in the form $[a, b]/c$, where a is the starting value, c is the increment, and b is the last value.

holds click-stream data of an online gaming site. The second database "BuAG"[4] (54k transactions, 3,48Go), holds click-stream of users access to the University Library Web site.

On each database, we have evaluated our method with a broad range of values for each free parameters (the exception is δ, which was fixed to be .05). These parameters that vary during our experiments are described in Fig. 3.

Better than using a real data stream, we have chosen to simulate data streams assuming the complete knowledge of the domains, thus allowing to compute exact values for the performance measurements. More precisely, we simulate data streams by sampling each database into fragments. For example, we could consider that data arrive in a timely manner from the "Dragons" database, and that only 20% of the whole data is stored. So we pick 20% of the transactions of this database, we consider that it is the data stored, and check the $\mathtt{P}, \mathtt{R}$ and F_β values with the whole database. We have chosen to sample the database on a broad range of percentages using two scales. The first allows a fine sampling of the database, for values ranging from 2% to 10% by steps of 1% ("sampling1" in Fig. 3), and typically gives an idea of what may happens for very large, fast data streams.We have completed this first range with a coarse range of samplings, from 15% to 70% by steps of 5% ("sampling2" in Fig. 3), which gives an idea of the average and limit behaviors of our method. Due to the very large number of experiments and the lack of space to report them all, we have chosen to report some plots we consider as representative, and synthesize the whole results.

Figure 4 shows result from experiments on choosen databases with $\delta = .05$. Each plot describes for one database and one support value, either $\mathtt{P}$ or $\mathtt{R}$ of the three methods which consist in keeping $S^*_{\theta-\varepsilon}, S^*_\theta$, and $S^*_{\theta+\varepsilon}$.

A first glance at these plots reveals that their behavior is almost always the same. Namely:

- the $\mathtt{P}$ increases with θ' (eq. 2), while the $\mathtt{R}$ decreases with θ',
- the $\mathtt{P}$ equals or approaches 1 for mostly storing sizes when $\theta' = \theta + \varepsilon$,
- the $\mathtt{R}$ equals or approaches 1 for mostly storing sizes when $\theta' = \theta - \varepsilon$.

These observations are in accordance with the theoretical results of Section 4. There is another phenomenon we may observe: the $\mathtt{R}$ associated to $\theta' = \theta + \varepsilon$ is not that far from the $\mathtt{R}$ of $\theta' = \theta$. Similarly, the $\mathtt{P}$ associated to $\theta' = \theta - \varepsilon$ is not that far from the $\mathtt{P}$ of $\theta' = \theta$. This shows that the maximization of $\mathtt{P}$ or $\mathtt{R}$ is

[4] www.univ-ag.fr/buag/.

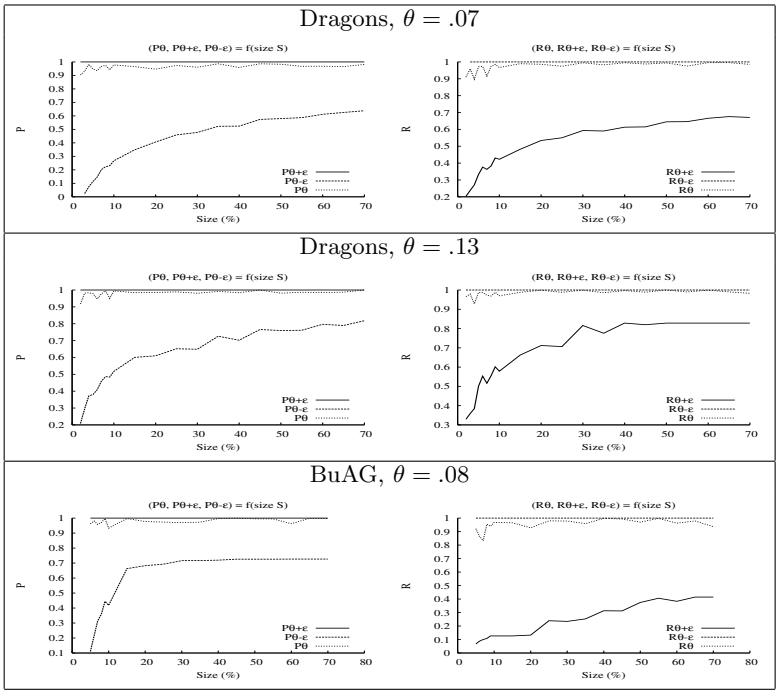

Fig. 4. Examples of plots with $\delta = .05$ and three θ values. For theses values we give the `P` (left plot) and `R` (right plot) for the three methods consisting in picking $S^*_{\theta-\varepsilon}, S^*_{\theta}, S^*_{\theta+\varepsilon}$.

obtained at a reduced degradation of the other parameter. We also remark that `P` plots tend to be better than `R` plots. This is not really surprising, as advocated in Section 4, since the range of values for `P` is smaller than that of `R`.

A close look at small storing sizes of the streams (before 10%) also reveals a more erratic behavior without straight convergence to maximal `P` or `R`. This behavior is not linked to the statistical support, but to the databases used. Indeed, small databases lead to even smaller storing sizes, and frequent sequential patterns kept out of small databases are in fact trickier to predict than for bigger ones. This point is important as, from a real-world standpoint, we tend to store very large databases, so we may expect this phenomenon to be reduced.

On these databases, another phenomenon seems to appear. First of all, because of the small values for θ, some tests have not be performed because $\theta-\varepsilon$ was < 0. Furthermore, the greater difference observed between the curves seems to stem out from the different sizes of databases. For example, the BuAG database is smaller than the Dragons database by a factor 2.4. This, we think, explains the greater differences between the curves: they are mostly a small database phenomenon, and may not be expected from larger databases, or even real-world data streams.

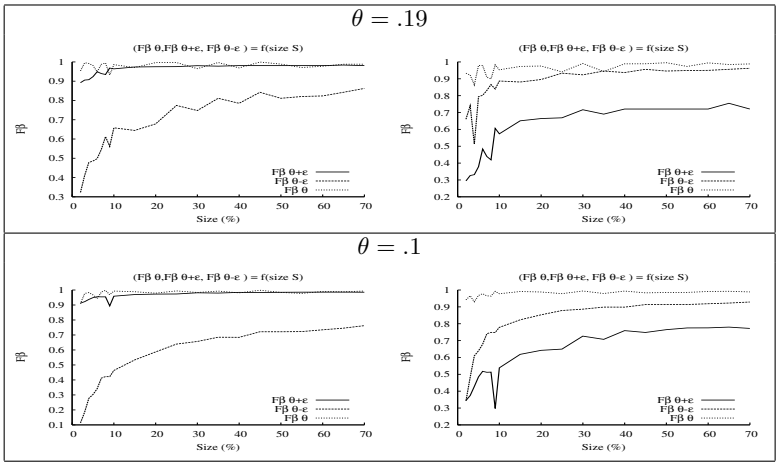

Fig. 5. Two sets of plots of the F_β value from the Dragons database, with $\beta = .2$ for the left plots and $\beta = 1.8$ for the right plots.

In Figure 5, two sets of two plots taken from the Dragons database plot the F_β measure, against the size of the stream used (in %). The values of β have been chosen different from 1, but not too small or too large to yield a reasonable prominence of one criterion (.2 and 1.8, see Figure 5).

Different phenomenons appears on these plot. First of all, as seen on the left plots, the F_β value displays the advantage of choosing $\theta' = \theta + \varepsilon$ against the choice $\theta' = \theta$. Meanwhile, as shown on the right plots choosing $\theta' = \theta - \varepsilon$ against the choice $\theta' = \theta$ is not always a win / win strategy. Moreover, `R` that this is obtained while statistically guaranteeing the *maximal* value for whichever of the `P` or `R` criterion, as chosen by the user.

6 Conclusion

In this paper we address the problem of mining sequential patterns on data streams. We proposed a statistical technique for estimating the support of sequential patterns on data streams, regardless of the size of the data stored. Conducted experiments have shown that this technique is very efficient. Furthermore, one of the main advantage of the proposed approach is that it does not depend on the used mining algorithm. One promising research direction would be to integrate our approach with approaches that consisting in somehow reducing the size of the data stored, so as to keep the property that sequential patterns *observed* as frequent still remain frequent with high probability [7]. In the framework of data streams it would be much more efficient from a statistical standpoint to keep the sequential patterns that are *truly* frequent. This would basically boil down to mixing our approach with them, so as to keep maximal `R`

(or P). Because of the technical machinery used in these papers (*e.g.* Blum filters [7]), mixing the approaches into a global technique for reducing the error in maintaining frequent sequential patterns from data streams may be more than simply interesting: it seems to be very natural.

References

1. J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using bitmap representation. In *Proc. of KDD'02*, 2002.
2. Y. Chi, H. Wang, P.S. Yu, and R.R. Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Proc. of ICDM'04*, 2004.
3. L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition.* Springer, 1996.
4. Carlos Domingo, Ricard Gavaldà, and Osamu Watanabe. Adaptive sampling methods for scaling up knowledge discovery algorithms. *Proc. of DMKD'02*, 6, 2002.
5. G. Giannella, J. Han, J. Pei, X. Yan, and P. Yu. Mining frequent patterns in data streams at multiple time granularities. In *Next Generation Data Mining, MIT Press*, 2003.
6. J. Han, J. Pei, B. Mortazavi-asl, Q. Chen, U. Dayal, and M. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proc. of KDD'00*, 2000.
7. C. Jin, W. Qian, C. Sha, J.-X. Yu, and A. Zhou. Dynamically maintaining frequent items over a data stream. In *Proc. of CIKM'03*. ACM Press, 2003.
8. M. J. Kearns and Y. Mansour. A Fast, Bottom-up Decision Tree Pruning algorithm with Near-Optimal generalization. In *Proc. of ICML'98*, 1998.
9. H.-F. Li, S.Y. Lee, and M.-K. Shan. An efficient algorithm for mining frequent itemsets over the entire history of data streams. In *Proc. of the 1st Intl. Workshop on Knowledge Discovery in Data Streams*, 2004.
10. G. Manku and R. Motwani. Approximate frequency counts over data streams. In *Proc. of VLDB'02*, 2002.
11. F. Masseglia, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In *Proc. of PKDD'98*, 1998.
12. R. Nock and F. Nielsen. Statistical Region Merging. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2004. to appear.
13. Tobias Scheffer and Stefan Wrobel. Finding the most interesting patterns in a database quickly by using sequential sampling. *Proc. of JMLR'02*, 3, 2002.
14. R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In *Proc. of EDBT'96*, 1996.
15. R. Agrawal R. Srikant. Mining sequential patterns. In *Proc. of ICDE'95 Conference*, 1995.
16. W.-G. Teng, M.-S. Chen, and P.S. Yu. A regression-based temporal patterns mining schema for data streams. In *Proc. of VLDB'03*, 2003.
17. Hannu Toivonen. Sampling large databases for association rules. In *Proc. of VLDB'96*, 1996.
18. V. Vapnik. *Statistical Learning Theory.* John Wiley, 1998.
19. D. Wettschereck and D. Aha. Mining concept-drifting data streams with ensemble classifiers. In *Proc. of KDD'03*, 2003.
20. M.J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning Journal*, 42, 2001.
21. Q. Zheng, K. Ksu, S. Ma, and W. Lv. The algorithms of updating sequential patterns. In *Proc. of ICDM'02*, 2002.