Significant Frequent Item Sets via Pattern Spectrum Filtering

Christian Borgelt and David Picado-Muiño

European Centre for Soft Computing
Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Spain
christian@borgelt.net,david.picado@softcomputing.es

Abstract. Frequent item set mining often suffers from the grave problem that the number of frequent item sets can be huge, even if they are restricted to closed or maximal item sets: in some cases the size of the output can even exceed the size of the transaction database to analyze. In order to overcome this problem, several approaches have been suggested that try to reduce the output by statistical assessments so that only significant frequent item sets (or association rules derived from them) are reported. In this paper we propose a new method along these lines, which combines data randomization with so-called pattern spectrum filtering, as it has been developed for neural spike train analysis. The former serves the purpose to implicitly represent the null hypothesis of independent items, while the latter helps to cope with the multiple testing problem resulting from a statistical evaluation of found patterns.

1 Introduction

Frequent item set mining (see, e.g., [11, 6] for an overview) has been an area of intense research in data mining since the mid 1990s. Up to the early 2000s the main focus was on developing algorithms that can find all frequent, all closed or all maximal item sets as fast as possible. The substantial efforts devoted to this task led to a variety of very sophisticated algorithms, the best-known of which are Apriori [2], Eclat [28, 29], FP-Growth [14, 12, 13], and LCM [21–23]. Since the efficiency problem can be considered solved with these algorithms, the focus has shifted since then to the grave problem that the number of found frequent item sets can be huge, even if they are restricted to closed or maximal item sets: in some cases the size of the output can even exceed the size of the transaction database to analyze. As a consequence, relevant frequent item sets (or association rules derived from them) can drown in a sea of irrelevant patterns.

In order to overcome this problem, several approaches have been suggested, which fall mainly into two categories: in the first place, it is tried to reduce the output by statistical assessments so that only significant patterns are reported. Such approaches include mining only part of the data and statistically validating the results on a hold-out subset [25] or executing statistical tests directly in the search [26], corrected by Bonferroni [5, 1], Bonferroni-Holm [15], Benjamini-Hochberg [3] or similar methods for multiple testing. A related approach in the

spirit of closed item sets are self-sufficient item sets [27]: item sets the support of which is within expectation (under independence assumptions) are removed. A second line in this category consists in randomization approaches (like [9]), which create surrogate data sets that implicitly encode the null hypothesis.

The second category is the selection of so-called *pattern sets*, for example, a (small) pattern set that covers the data well or exhibits little overlap between its member patterns (low redundancy). Such approaches include finding pattern sets with which the data can be compressed well [19, 24] or in which all patterns contribute to partitioning the data [7]. A general framework for this task, which has become known as *constraint based pattern mining*, has been suggested in [8]. Note that in this second category pattern sets are selected, with an emphasis on the interaction between the patterns, while the approaches in the first category rather try to find patterns that are significant individually.

In this paper we propose an approach that falls into the first category and is closest in spirit to [9], mainly because we also use swap randomization to generate surrogate data sets. However, we consider other randomization methods as well, in particular if the transactional data is derived from a table, that is, if the individual items are actually attribute-value pairs. Our method also goes beyond [9] by considering the significance of individual patterns, while [9] only considered the total number of patterns. Finally, we discuss pattern spectrum filtering as a simple, yet effective way to cope with the multiple testing problem.

The remainder of this paper is organized as follows: in Section 2 we briefly review frequent item set mining to introduce notation as well as core concepts. In Section 3 we discuss randomization or surrogate data generation methods, with which the null hypothesis of independent items is represented implicitly. Section 4 introduces the notion of a pattern spectrum (adapted from [18]) as a way to handle the multiple testing problem that results from the combinatorial explosion of potential patterns. In Section 5 we report about experiments that we carried out with several publicly available data sets that are commonly used for benchmarks. Finally, in Section 6, we draw conclusions from our discussion.

2 Mining Frequent Item Sets

Formally, frequent item set mining is the following task: we are given a set $B = \{i_1, \ldots, i_n\}$ of *items*, called the *item base*, and a database $T = (t_1, \ldots, t_m)$ of *transactions*. An item may, for example, represent a product offered by a shop. In this case the item base represents the set of all products offered by, for example, a supermarket or an online shop. The term *item set* refers to any subset of the item base B. Each transaction is an item set and may represent, in the supermarket setting, a set of products that has been bought by a customer. Since several customers may have bought the exact same set of products, the total of all transactions must be represented as a vector (as above) or as a multiset (or bag). Alternatively, each transaction may be enhanced by a *transaction identifier (tid)*. Note that the item base B is usually not given explicitly, but only implicitly as the union of all transactions, that is, $B = \cup_{k \in \{1,\ldots,m\}} t_k$.

The *cover* $K_T(I) = \{k \in \{1, \ldots, m\} \mid I \subseteq t_k\}$ of an item set $I \subseteq B$ indicates the transactions it is contained in. The *support* $s_T(I)$ of I is the number of these transactions and hence $s_T(I) = |K_T(I)|$. Given a user-specified *minimum support* $s_{\min} \in \mathbb{N}$, an item set I is called *frequent* (in T) iff $s_T(I) \geq s_{\min}$. The goal of frequent item set mining is to find all item sets $I \subseteq B$ that are frequent in the database T and thus, in the supermarket setting, to identify all sets of products that are frequently bought together. Note that frequent item set mining may be defined equivalently based on the *(relative) frequency* $\sigma_T(I) = s_T(I)/m$ of an item set I and a corresponding lower bound $\sigma_{\min} \in (0, 1]$.

A typical problem in frequent item set mining is that the number of patterns is often huge and thus the output can easily exceed the size of the transaction database to mine. In order to mitigate this problem, several restrictions of the set of frequent item sets have been suggested. The two most common are *closed* and *maximal* item sets: a frequent item set $I \in \mathcal{F}_T(s_{\min})$ is called

- a *maximal (frequent) item set* iff $\forall J \supset I : s_T(J) < s_{\min}$;
- a *closed (frequent) item set* iff $\forall J \supset I : s_T(J) < s_T(I)$.

In this paper we mainly consider closed item sets, because they not only preserve knowledge of what item sets are frequent, but also allow us to compute the support of non-closed frequent item sets with a simple formula (see, e.g., [6]).

Frequent item set mining usually follows a simple *divide-and-conquer* scheme that can also be seen as a *depth-first search* (essentially only Apriori uses a *breadth-first search*): for a chosen item i, the problem to find all frequent item sets is split into two subproblems: (1) find all frequent item sets containing i and (2) find all frequent item sets *not* containing i. Each subproblem is then further split based on another item j: find all frequent item sets containing (1.1) both i and j, (1.2) i, but not j, (2.1) j, but not i, (2.2) neither i nor j etc.

All subproblems occurring in this recursion can be defined by a *conditional transaction database* and a *prefix*. The prefix is a set of items that has to be added to all frequent item sets that are discovered in the conditional transaction database. Formally, all subproblems are pairs $S = (C, P)$, where C is a conditional database and $P \subseteq B$ is a prefix. The initial problem, with which the recursion is started, is $S = (T, \emptyset)$, where T is the given transaction database.

A subproblem $S_0 = (C_0, P_0)$ is processed as follows: choose an item $i \in B_0$, where B_0 is the set of items occurring in C_0. This choice is, in principle, arbitrary, but often follows some predefined order of the items. If $s_{C_0}(\{i\}) \geq s_{\min}$, then report the item set $P_0 \cup \{i\}$ as frequent with the support $s_{C_0}(\{i\})$, and form the subproblem $S_1 = (C_1, P_1)$ with $P_1 = P_0 \cup \{i\}$. The conditional database C_1 comprises all transactions in C_0 that contain the item i, but with the item i removed. This also implies that transactions that contain no other item than i are entirely removed: no empty transactions are ever kept. If C_1 is not empty, process S_1 recursively. In any case (that is, regardless of whether $s_{C_0}(\{i\}) \geq s_{\min}$ or not), form the subproblem $S_2 = (C_2, P_2)$, where $P_2 = P_0$. The conditional database C_2 comprises *all* transactions in C_0 (including those that do not contain the item i), but again with the item i (and resulting empty transactions) removed. If the database C_2 is not empty, process S_2 recursively.

Concrete algorithms following this scheme differ mainly in how they represent the conditional transaction databases and how they derive a conditional transaction database for a split item from a given database. Details about such algorithms (like Eclat, FP-Growth, or LCM) can be found, for example, in [11, 6].

3 Surrogate Data Generation

The general idea of data randomization or surrogate data generation is to represent the null hypothesis (usually an independence hypothesis; here: independence of the items) not explicitly by a data model, but implicitly by data sets that are generated in such a way that their occurrence probability is (approximately) equal to their occurrence probability under the null hypothesis. Such an approach has the advantage that it needs no explicit data model, which in many cases may be difficult to specify, but can start from the given data. This data is modified in random ways to obtain data that are at least analogous to those that could be sampled under conditions in which the null hypothesis holds.

A randomization or surrogate data approach also makes it usually easier to preserve certain frame conditions and properties of the data to analyze that one may want to keep, in order not to taint the test result by having destroyed features that the data possess, but in which one is not directly interested. In the case of transactional data, such features are the number of items, the number of transactions, the size of the transactions and the (relative) frequency of the items. That is, for standard transactional data, we want a randomization method that only changes the composition of the given transactions, but keeps their sizes and the overall occurrence frequencies of the individual items.

A very simple method satisfying these constraints is *swap randomization* [9], which is best explained with the help of how it modifies a binary matrix representation of a transaction database. In such a representation each column refers to an item, each row to a transaction, and a matrix element is 1 iff the item corresponding to the element's column is contained in the transaction corresponding to the element's row. Otherwise the element is 0. Swap randomization consists in executing a large number of *swaps* like the one depicted in Figure 1. Each swap affects two items and two transactions. Each of the transactions contains one item, but not the other; the swap exchanges the items between the transactions.

In a set representation, as we used it in Section 2, a swap can be described as follows: let t_j and t_k be two transactions with $t_j - t_k \neq \emptyset$ and $t_k - t_j \neq \emptyset$, that is, each transaction contains at least one item not contained in the other. Then we choose $i_j \in t_j - t_k$ and $i_k \in t_k - t_j$ and replace t_j and t_k with $t'_j = (t_j - \{i_j\}) \cup \{i_k\}$ and $t'_k = (t_k - \{i_k\}) \cup \{i_j\}$ thus exchanging the items between the transactions. Such a swap has the clear advantage that it obviously maintains the sizes of the transactions as well as the (exact) occurrence frequencies of the items.

If a sufficiently large number of swaps is carried out (in [9] it is recommended to use a number in the order of the 1s in a binary matrix representation of the data), the resulting transaction database can be seen as being sampled from the null hypothesis of independent items, because all (systematic, non-random) co-

Fig. 1. A single swap of swap randomization in a matrix representation.

occurrences of items have been sufficiently destroyed. Note that it is advisable to apply swap randomization to already generated surrogates to further randomize the data, rather than to start always from the original data. In this way the number of swaps may also be reduced for later surrogates. In our implementation we execute as many swaps as there are 1s in a binary matrix representation only for the first surrogate, but only half that number for every later surrogate. This provides a good trade-off between speed and independence of the data sets.

An obvious alternative consists in retrieving the (overall) item probability distribution and randomly sampling from it to fill the given transactions with new items (taking care, of course, that no item is sampled more than once for the same transaction). This methods looks simpler (because one need not find transactions first that satisfy the conditions stated above), but has the drawback that it preserves the item frequencies only in expectation. However, this can be corrected (to some degree) by checking the item distribution in a generated surrogate and then adapting the transactions as follows: if there is a transaction (selected randomly) in which an item i occurs that is over-represented relative to the original data, while it lacks an item j that is under-represented, item i is replaced by item j. This procedure is repeated until the item distribution meets, or is at least sufficiently close to the distribution in the original data. In our experiments we found that it was always possible, with fairly little effort in this direction, to meet the actual item frequency distribution exactly.

While these methods work well for actual transactional data, we also have to take care of the fact that many data sets that might be submitted to frequent item set mining (including many common benchmark data sets) are actually derived from tabular data. That is, the items are actually attribute-value pairs, and thus the transactions are sets $t_k = \{A_1 = a_{1k}, \ldots, A_n = a_{nk}\}$, where the A_j, $j = 1, \ldots, n$, are attributes and a_{jk} is the value that attribute A_j has in the k-th transaction, $k = 1, \ldots, m$. For such data the methods described above are not applicable, because we have to ensure that each transaction contains exactly one item for each attribute, which is not guaranteed with the above methods.

To randomize such data we use a *column shuffling scheme*. That is, we generate r permutations π_j, $j = 1, \ldots, r$, of the numbers $\{1, \ldots, m\}$ (one permutation for each attribute), where m is the number of transactions. Then we replace each transaction t_k, $k = 1, \ldots, m$, with $t'_k = \{A_1 = a_{1\pi_1(k)}, \ldots, A_n = a_{n\pi_n(k)}\}$. This guarantees that each transaction contains one item for each attribute. It only shuffles the attribute values, respecting the domains of the attributes.

Other surrogate data generation methods, which are designed for data over an underlying continuous domain (like a time domain), from which the transactions are derived by (time) binning, are discussed in [17]. Unfortunately, they cannot be transferred directly to the transactional setting, because most of them require the possibility to dither/displace items on a continuous (time) scale.

4 Pattern Spectrum Filtering and Pattern Set Reduction

Trying to single out significant patterns proves to be less simple than it may appear at first sight, since one has to cope with the following two problems: in the first place, one has to find a proper statistic that captures how (un)likely it is to observe a certain pattern under the null hypothesis that items occur independently. Secondly, the huge number of potential patterns causes a severe multiple testing problem, which is not easy to overcome with standard methods. In [18] we provided a fairly extensive discussion in the framework of spike train analysis (trying to find patterns of synchronous activity) and concluded that an approach different to evaluating specific patterns with statistics is needed.

As a solution, *pattern spectrum filtering* was proposed in [18, 20] based on the following insight: even if it is highly unlikely that a *specific group* of z items co-occurs s times, it may still be likely that *some group* of z items co-occurs s times, even if items occur independently. The reason is simply that there are so many possible groups of z items (unless the item base B as well as z are tiny) that even though each group has only a tiny probability of co-occurring s times, it may be almost certain that *one of them* co-occurs s times. As a consequence, since there is no *a-priori* reason to prefer certain sets of z items over others (even though a refined analysis, on which we are working, may take individual item frequencies into account), we should not declare a pattern significant if the occurrence of a counterpart (same size z and same or higher support s) can be explained as a chance event under the null hypothesis of independent items.

Hence we pool patterns with the same *pattern signature* $\langle z, c \rangle$, and collect for each signature the (average) number of patterns that we observe in a sufficiently large number of surrogate data sets. This yields what is called a *pattern spectrum* in [18, 20]. Pattern spectrum filtering keeps only such patterns found in the original data for which no counterpart with the same signature (or a signature with the same z, but larger s) was observed in surrogate data, as such a counterpart would show that the pattern can be explained as a chance event.

While in [18, 20] a pattern spectrum is represented as a bar chart with one bar per signature, this is not feasible for the data sets we consider in this paper, due to the usually much larger support values. Rather we depict a pattern spectrum as a bar chart with one bar per pattern size z, the height of which represents the largest support $s_{\max}(z)$ that we observed for patterns of this size in surrogate data sets. An example of such a pattern spectrum is shown in the top part of Figure 2 (mind the logarithmic scale). Note that this reduced representation, although less rich in information, still contains all that is relevant, namely the support border, below which we discard patterns found in the original data.

Table 1. Data sets for which we present results in this paper together with their sizes, the minimum support used for mining and the number of found patterns.

			closed patterns		
data set	trans.	$s_{\min}$	unfiltered	filtered	reduced
census	48842	40	850932	779	17
breast	350	20	965	323	1
webview1	59602	60	3974	259	42
retail	88162	45	19242	3	1

Table 2. Top-ranked closed frequent item sets in the census data.

z	s	q	items
12	382	1.425	country=United-States edu_num=10 education=Some-college salary<=50K loss=none gain=none hours=half-time marital=Never-married relationship=Own-child age=young sex=Female workclass=Private
12	362	1.351	country=United-States edu_num=10 education=Some-college salary<=50K loss=none gain=none hours=full-time marital=Never-married relationship=Own-child age=young sex=Male workclass=Private
11	882	1.256	country=United-States edu_num=13 education=Bachelors salary>50K loss=none age=middle-aged marital=Married-civ-spouse relationship=Husband sex=Male race=White workclass=Private

Note also that pattern spectrum filtering still suffers from a certain amount of *multiple testing*: every pair $\langle z, c\rangle$ that is found in the original data gives rise to one test. However, the pairs $\langle z, c\rangle$ are *much fewer* than the number of specific item sets. As a consequence, simple approaches like *Bonferroni correction* [5, 1] become feasible, with which the number of needed surrogate data sets can be computed [18]: given a desired overall significance level α and the number k of pattern signatures to test, at least k/α surrogate data sets have to be analyzed.

As a further filtering step, *pattern set reduction* was proposed in [20] to take care of the fact that an actual pattern induces other, spurious patterns that are subsets, supersets or overlap patterns. These patterns are reduced with the help of a preference relation between patterns and the principle that only patterns are kept to which no other pattern is preferred. Here we adopt the following preference relation: let $X, Y \subseteq B$ be two patterns with $Y \subseteq X$ and let $z_X = |X|$ and $z_Y = |Y|$ be their sizes and s_X and s_y their support values. Finally, let $s_{\max}(z)$ be the largest support of a pattern of size z observed in surrogate data. Then the excess support of Y (relative to X) can be explained as a chance event if $\phi_1 = (s_Y - s_X + 1 \leq s_{\max}(z_Y))$ holds and the excess items in X (relative to Y) can be explained as a chance event if $\phi_2 = (s_X \leq s_{\max}(z_X - z_Y + 2))$ holds. Finally, we use $\phi_3 = ((z_X - 1)s_X \geq (z_Y - 1)s_Y))$ as a heuristic tie-breaker if both ϕ_1 and ϕ_2 hold. As a consequence, the set X is preferred to the set Y iff

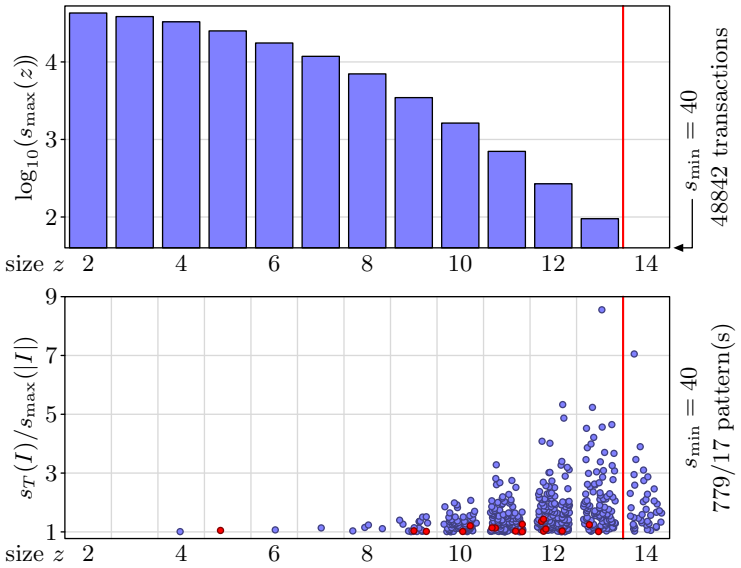

Fig. 2. Pattern spectrum (top) and filtered patterns (bottom) of the `census` data. Note the logarithmic scale in the top diagram. The red line marks the end of the pattern spectrum: no larger patterns were observed in surrogate data sets. The horizontal position of the dots representing the patterns in each size bin of the bottom diagram is random (to reduce the dot overlap). Reduced patterns are marked in red.

$\phi_1 \wedge (\neg\phi_2 \vee \phi_3)$ and the set Y is preferred to the set X iff $\phi_2 \wedge (\neg\phi_1 \vee \neg\phi_3)$. Otherwise X and Y are not comparable. More details, especially the reasoning underlying the conditions ϕ_1 and ϕ_2, can be found in [20].

5 Experiments

We implemented the described surrogate data generation methods as well as pattern spectrum filtering in C and made the essential functions available as a Python extension library, which simplifies setting up scripts for the experiments. Pattern set reduction was then implemented on top of this library in Python.

As data sets we chose common benchmark data sets, like the `census`, `chess`, `mushroom`, and `breast` data sets from the UCI machine learning repository [4], the `BMS-Webview-1` data set (or `webview1` for short) from the KDD cup 2000 [16], as well as the `retail`, `accidents` and `kosarak` data sets from the FIMI repository [10]. However, due to reasons of space we can only present some of the results, for which we selected `census`, `breast`, `webview1` and `retail`. The first two of these data sets are actually tabular data, and therefore we applied the column shuffling scheme described above, while the last two are genuinely transactional data, which we processed with swap randomization.

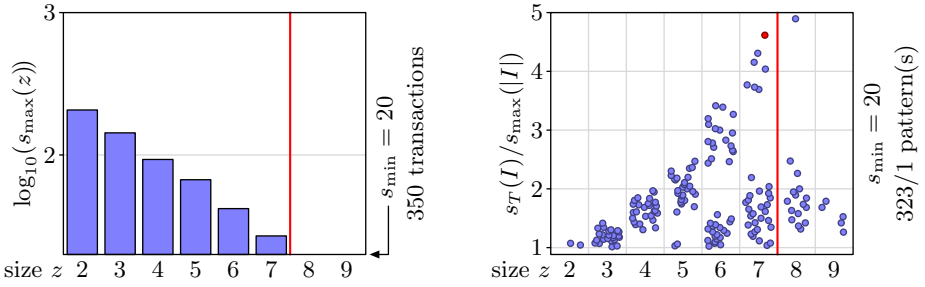

Fig. 3. Pattern spectrum (left) and filtered patterns (right) of the `breast` data.

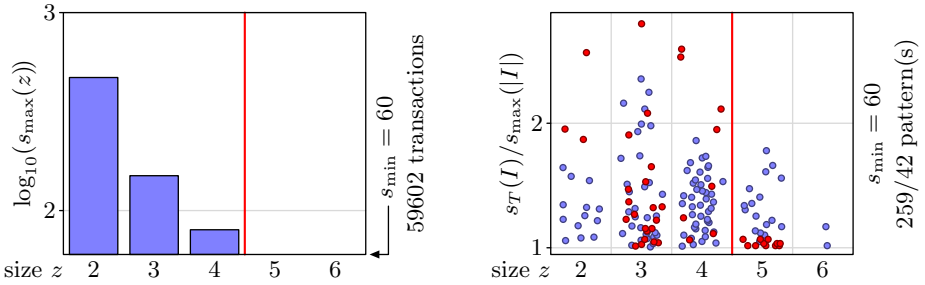

Fig. 4. Pattern spectrum (left) and filtered patterns (right) of the `webview1` data.

For all data sets we generated and analyzed 10,000 surrogate data sets and ranked the filtered item sets by how far they are from the support border of the pattern spectrum (using the ratio $s_T(I)/s_{\max}(|I|)$, where $s_T(I)$ is the support of I in the transactional database T). A summary of the number of transactions, minimum (absolute) support values, and discovered closed frequent patterns before and after pattern spectrum filtering is shown in Table 1.

On the `census` data (see Figure 2), our filtering methods reduce the huge number of 850932 closed frequent patterns that are found with minimum support $s_{\min} = 40$ to merely 17 statistically significant patterns. The top 3 patterns are shown in Table 2, which are nicely interpretable. The first two capture the children of a family that work directly after finishing college, the third pattern captures upper middle class husbands or family fathers. The differences of the first two patterns, which are highlighted in blue, are interesting to observe.

On the `webview1` data (see Figure 4) the 3974 closed frequent item sets that are found with minimum support $s_{\min} = 60$ are reduced to merely 42. The top ranked of these patterns are shown in Table 3. Due to the numerical encoding of the items, they are difficult to interpret without any data dictionary, though.

On the `retail` data (see Figure 5) the large number of 19242 closed frequent item sets found with minimum support $s_{\min} = 45$ is reduced to the single pattern $I = \{39, 41, 48\}$ with $s_I = 7366$ and $s_I/s_{\max}(3) = 1.12133$. Again an interpretation is difficult, due to the numeric encoding of the items.

Table 3. Top-ranked closed frequent item sets in the `webview1` data.

z	s	q	items
3	417	2.780	10295 10307 10311
4	205	2.562	10295 10307 10311 10315
2	1204	2.561	33449 33469
4	200	2.500	10311 12487 12703 32213

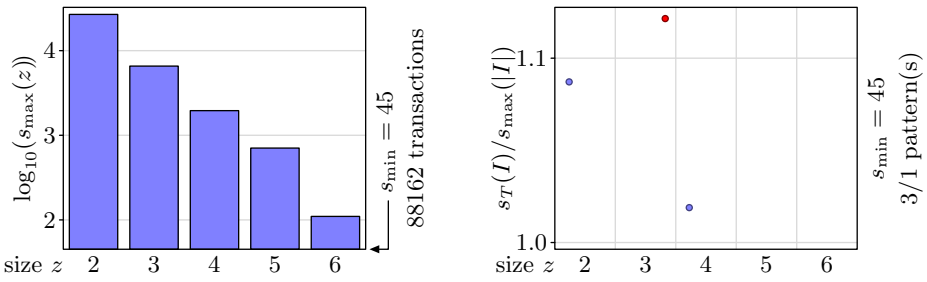

Fig. 5. Pattern spectrum (left) and filtered patterns (right) of the `retail` data.

6 Conclusions and Future Work

We demonstrated how data randomization or surrogate data generation together with pattern spectrum filtering and pattern set reduction can effectively reduce found (closed) frequent item sets to statistically significant ones. The reduction is often tremendous and leaves a user with a manageable number of patterns that is feasible to check manually. A shortcoming of our current method is, however, that it treats all item sets alike, regardless of the frequency of the individual items. We are currently working on an extension that allows for different support borders depending on the expected support of an item set as computed from the individual item frequencies under an independence assumption. Although this is likely to increase the number of filtered patterns, it may enable the method to detect significant item sets consisting of less frequent items.

Software and Source Code

Python and C implementations of the described surrogate data generation and frequent item set filtering procedures can be found at this URL:

`www.borgelt.net/pyfim.html`

References

1. H. Abdi. Bonferroni and Šidák Corrections for Multiple Comparisons. In: N.J. Salkind, ed. *Encyclopedia of Measurement and Statistics*, 103–107. Sage Publications, Thousand Oaks, CA, USA 2007

2. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. *Proc. 20th Int. Conf. on Very Large Databases (VLDB 1994, Santiago de Chile)*, 487–499. Morgan Kaufmann, San Mateo, CA, USA 1994
3. Y. Benjamini and Y. Hochberg. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society, Series B (Methodological)* 57(1):289–300. Blackwell, Oxford, United Kingdom 1995
4. C.L. Blake and C.J. Merz. *UCI Repository of Machine Learning Databases*. Dept. of Information and Computer Science, University of California at Irvine, CA, USA 1998 (`http://www.ics.uci.edu/~mlearn/MLRepository.html`)
5. C.E. Bonferroni. Il calcolo delle assicurazioni su gruppi di teste. *Studi in Onore del Professore Salvatore Ortu Carboni*, 13–60. Bardi, Rome, Italy 1935
6. C. Borgelt. Frequent Item Set Mining. *Wiley Interdisciplinary Reviews (WIREs): Data Mining and Knowledge Discovery* 2:437–456 (doi:10.1002/widm.1074). J. Wiley & Sons, Chichester, United Kingdom 2012
7. B. Bringmann and A. Zimmermann. The Chosen Few: On Identifying Valuable Patterns. *Proc. 7th IEEE Int. Conf. on Data Mining (ICDM 2007, Omaha, NE)*, 63–72. IEEE Press, Piscataway, NJ, USA 2007
8. L. De Raedt and A. Zimmermann: Constraint-Based Pattern Set Mining. *Proc. 7th IEEE Int. Conf. on Data Mining (ICDM 2007, Omaha, NE)*, 237–248. IEEE Press, Piscataway, NJ, USA 2007
9. A. Gionis, H. Mannila, T. Mielikäinen and P. Tsaparas. Assessing Data Mining Results via Swap Randomization. *ACM Transactions on Knowledge Discovery from Data* 1(3):article 14. ACM Press, New York, NY, USA 2007
10. B. Goethals. Frequent Itemset Mining Implementations Repository. University of Antwerp, Belgium 2003 (`http://fimi.ua.ac.be/`)
11. B. Goethals. Frequent Set Mining. *Data Mining and Knowledge Discovery Handbook*, 321–338. Springer-Verlag, Berlin/Heidelberg, Germany 2010
12. G. Grahne and J. Zhu. Efficiently Using Prefix-trees in Mining Frequent Itemsets. *Proc. Workshop Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL)*. CEUR Workshop Proceedings 90, Aachen, Germany 2003
13. G. Grahne and J. Zhu. Reducing the Main Memory Consumptions of FPmax* and FPclose. *Proc. Workshop Frequent Item Set Mining Implementations (FIMI 2004, Brighton, UK)*. CEUR Workshop Proceedings 126, Aachen, Germany 2004
14. J. Han, J. Pei, and Y. Yin. Mining Frequent Patterns without Candidate Generation. *Proc. 19th ACM Int. Conf. on Management of Data (SIGMOD 2000, Dallas, TX)*, 1–12. ACM Press, New York, NY, USA 2000
15. S. Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics* 6(2):65–70. J. Wiley & Sons, Chichester, United Kingdom 1979
16. R. Kohavi, C.E. Bradley, B. Frasca, L. Mason, and Z. Zheng. KDD-Cup 2000 Organizers' Report: Peeling the Onion. *SIGKDD Exploration* 2(2):86–93. ACM Press, New York, NY, USA 2000
17. S. Louis, C. Borgelt, and S. Grün. Generation and Selection of Surrogate Methods for Correlation Analysis. In: S. Grün and S. Rotter (eds.) *Analysis of Parallel Spike Trains*, 359–382. Springer-Verlag, Berlin, Germany 2010
18. D. Picado-Muiño, C. Borgelt, D. Berger, G.L. Gerstein, and S. Grün. Finding Neural Assemblies with Frequent Item Set Mining. *Frontiers in Neuroinformatics* 7:article 9 (doi:10.3389/fninf.2013.00009). Frontiers Media, Lausanne, Switzerland 2013

19. A. Siebes, J. Vreeken, and M. van Leeuwen. Item Sets that Compress. *Proc. SIAM Int. Conf. on Data Mining (SDM 2006, Bethesda, MD)*, 393–404. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA 2006
20. E. Torre, D. Picado-Muiño, M. Denker, C. Borgelt, and S. Grün. Statistical Evaluation of Synchronous Spike Patterns Extracted by Frequent Item Set Mining. *Frontiers in Computational Neuroscience*, 7:article 132 (doi:10.3389/fninf.2013.00132). Frontiers Media, Lausanne, Switzerland 2013
21. T. Uno, T. Asai, Y. Uchida, and H. Arimura. LCM: An Efficient Algorithm for Enumerating Frequent Closed Item Sets. *Proc. Workshop on Frequent Item Set Mining Implementations (FIMI 2003, Melbourne, FL)*. CEUR Workshop Proceedings 90, TU Aachen, Germany 2003
22. T. Uno, M. Kiyomi and H. Arimura. LCM ver. 2: Efficient Mining Algorithms for Frequent/Closed/Maximal Itemsets. *Proc. Workshop Frequent Item Set Mining Implementations (FIMI 2004, Brighton, UK)*. CEUR Workshop Proceedings 126, Aachen, Germany 2004
23. T. Uno, M. Kiyomi, and H. Arimura. LCM ver. 3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset Mining. *Proc. 1st Open Source Data Mining on Frequent Pattern Mining Implementations (OSDM 2005, Chicago, IL)*, 77–86. ACM Press, New York, NY, USA 2005
24. J. Vreeken, M. van Leeuwen, and A. Siebes. Krimp: Mining Itemsets that Compress. *Data Mining and Knowledge Discovery* 23(1):169–214. Springer, Berlin, Germany 2011
25. G.I. Webb. Discovering Significant Patterns. *Machine Learning* 68(1):1–33. Springer, New York, NY, USA 2007
26. G.I. Webb. Layered Critical Values: A Powerful Direct-adjustment Approach to Discovering Significant Patterns. *Machine Learning* 71(2–3):307–323. Kluwer, Amsterdam, Netherlands 2008
27. G.I. Webb. Self-sufficient Itemsets: An Approach to Screening Potentially Interesting Associations between Items. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):Article 3. ACM Press, New York, NY, USA 2010
28. M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New Algorithms for Fast Discovery of Association Rules. *Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD 1997, Newport Beach, CA)*, 283–296. AAAI Press, Menlo Park, CA, USA 1997
29. M.J. Zaki and K. Gouda. Fast Vertical Mining Using Diffsets. *Proc. 9th ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD 2003, Washington, DC)*, 326–335. ACM Press, New York, NY, USA 2003