An evaluation of quality in context based sequential pattern mining

Jerzy Stefanowski[1], **Radosław Ziembiński**[1]

Abstract: Context based sequential patterns include two additional sets of context attributes – one describing a complete sequence and the other describing each element of the sequence. The aim of this paper is to experimentally compare two approaches to mine such patterns from numerical data: a new proposed context based algorithm and a traditional one combined with discretization. The results of experiments show that a new algorithm has led to better re-discovery of reference patterns hidden in the artificially generated sequence databases. The other aim is to present a new measure for comparing two sets of context patterns.

Keywords: data mining, sequential patterns, context patterns

1. Introduction

The sequential pattern mining is an essential task of the data mining. Its definition has been introduced by Agrawal, Srikant (1995) and generalized in many ways - for a review see e.g. Morzy (2004). It could be shortly presented as: for given a *sequences database* find all sequential patterns with a user-specified *minimum support*. Each sequence is a list of elements (transactions) ordered by an associated identifier. The transaction contains a set of items. The support is counted as a number of sequences in the database including the frequent sub-sequence. An inclusion means that itemsets in the pattern are subsets of appropriate itemsets in the supporting sequence with preserving a pattern's elements order. This problem is illustrated by a shopping example in figure 1.

Several algorithms for mining sequential patterns were already developed, see e.g. reviews in Han, Pei (2001). Majority of proposed methods works with sets of nominal items only. This is also reflected by an inclusion operation while comparing sequences which is feasible for nominal values. These properties may limit modelling of some more complex real-life problems. In many cases sources of transactions can provide additional non-nominal information associated with either circumstances of transactions or the sequence itself (e.g. a description of the place, environment factors, duration, engaged tools or persons etc.). Handling such information can be done by introducing two different *sets of context attributes* attached to sequence and all sequence's elements. Attributes can be defined using any scale also numeric ones. Such context attributes allow to mine "richer" sequence patterns – see figure 1. The traditional approach (shortly TSPM) cannot handle directly context data, in particular numeric ones.

Shortcomings of this traditional problem have lead us to formulation its generalization, called *context based sequential pattern mining* (shortly CBSPM), see

[1] Institute of Computing Science, Poznan University of Technology, ul. Piotrowo 2, 60–965 Poznan e-mail: `{jstefanowski,rziembinski}@cs.put.poznan.pl`

(Stefanowski, Ziembiński 2005). The important property is ability to handle directly numeric context attributes which implies using special functions measuring similarity of their values in contexts of a pattern and a sequence instead of the sets inclusion operation used in TPSM. As CBSPM can not be solved by simple extensions of traditional mining algorithms, a new algorithm, called *ContextMappingImproved*, was developed in (Ziembiński 2007). Previous experiments showed that looking for context patterns may require more computational costs. Taking into account these costs, one can consider the other approach to handle context numeric data as a "transformation approach", where all numeric attributes are pre-discretized and transformed to artificially items suitable to be processed by a "traditional" algorithm like PrefixSpan. However, it leads us also to another question about a quality measure other than the time efficiency.

Therefore, we decide to consider the problem of rediscovery of patterns by compared algorithms, i.e. we want to introduce several "reference patterns" in the artificially generated sequence database and compare to them the sets of patterns discovered by both algorithms (ContextMappingImproved and pre-discretized PrefixSpan - with equal width and equal frequency local discretization). We are interested in verifying differences between results of both algorithms and checking whether the transformation approach could be an alternative to a specific algorithm for CBSPM. Carrying out such an experimental study is the main aim of this paper. Measuring patterns re-discovery degree requires a research work as the problem of calculating similarities of two sets of context sequential patterns has not been extensively studied. This is the second aim of this paper.

2. Basic concepts of mining context sequential patterns

We only briefly describe CBSPM problem – for details see (Ziembiński 2007). Two kinds of context attribute sets are introduced to the structure of sequences. The first set, called a *sequence context* $D = \langle D_1, ...D_v \rangle$ is a set of attributes describing complete sequence. In general it describes main properties of the data source (e.g. a user profile). The second set of different attributes is called an *element context* $C = \langle C_1, ..., C_w \rangle$. In contradiction to the former it is attached to each element in the sequence. Structure of contexts is homogeneous in the scope of described class of object (e.g. sequence, element). Attributes of both contexts can be defined on either nominal, ordinal or numerical scales. An example of the context database sequence and context pattern is presented in figure 1.

The similarity between a sequence and a pattern is calculated in two phases. In the first phase values of appropriate context attributes are compared to each other to decide whether they are similar enough. It is done by means of similarity functions $\sigma^k(c_1^k, c_2^k)\ c_1^k, c_2^k \in C_{k \in 1..w}$ $(\sigma^k(d_1^k, d_2^k)\ d_1^k, d_2^k \in D_{k \in 1..v})$. The value of this function is normalised to values from 1.0 to 0.0 where 1.0 means identity of attribute values and 0.0 means total dissimilarity of attribute values. In the second phase for each context attribute in two sets occurring in compared pattern and sequence their values of similarities are aggregated to a single value with operators:

$$\Theta^C(CI_1, CI_2) = \Theta^C(\sigma_1^C(c_1^1, c_2^1), \sigma_2^C(c_1^2, c_2^2), ..., \sigma_w^C(c_1^w, c_2^w)), CI_1, CI_2 \in C$$

$$\Theta^D(DI_1, DI_2) = \Theta^D(\sigma_1^D(d_1^1, d_2^1), \sigma_2^D(d_1^2, d_2^2), ..., \sigma_v^D(d_1^v, d_2^v)), DI_1, DI_2 \in D$$

apriopriately for elements and for sequence context. This aggregation of can be performed using various operators like a minimum, a maximum, a weighted sum, other aggregation extra functions like OWA. It is required that the aggregation result must be normalised within <0.0;1.0> range like an attribute's similarity values.

Then, counting support of a pattern by a sequence requires changing this continuous value into a binary one - support or not. So, if an aggregated similarity value Θ^C (or Θ^D) is equal or greater than a user's defined threshold of minimal similarity level θ^C (or θ^D) the context attribute sets are similar otherwise they are considered as dissimilar.

A element of a pattern is similar to a sequence element only if (1) their contexts are similar (according to the thresholds θ^D and θ^C.) and (2) an itemset in the element of a pattern is included in an itemset of the respective element in the a sequence. The aim of the CPSPM problem is to find all patterns supported by at least *min_ support* number of sequences in a context sequence database. Thresholds θ^D , θ^C and the *min_ support* are parameters to be tuned by an analyst.

Algorithms specific for CBSPM have been introduced in (Ziembiński 2007). A limited space of this paper allows to present only main concepts of the ContextMappingImproved algorithm. The problem definition states that element is frequent if the number of sequences containing similar elements is greater than required minimal support threshold. The algorithm for each context in the database builds a list of contexts similar to it. If the list has more members from distinct sequences than threshold *min_ support*, then the context is considered as frequent. Frequent context is transformed into two connected artificial items, called A and B respectively. Item A replaces frequent context and the item B is added to all itemset of elements from the list. Item B supports item A but A does not support B. A sequence context is mapped to additional element added to sequence in similar way as element contexts. This mapping somehow transforms similar context sequences into a traditional sequences database. Then, context patterns could be mined by specifically adopted PrefixSpan algorithm ensuring that all patterns must contain additional element representing sequence's context and a sequence of elements. Each pattern's element must contain an artificial item Reverse mapping of the patterns and replacement of artificial items with corresponding context gives a final set of context patterns. The experiments from Ziembiński (2007) showed that this algorithm is slower than original non-transformed PrefixSpan, however additional processing costs could be constrained by tuning minimal similarity thresholds. As the mapping can be memory consuming and a heuristic algorithm was proposed. The heuristic algorithm preserves limited number of the most similar mappings and avoid the itemset size "explosion".

Let us notice that TSPM is adopted to handle numerical values in a different way. During the discretization disjoint subspaces (discretization folds) of the context values domain are created. Each discretization fold is transformed to an artificial item replacing the coded value corresponding this fold. However, it also reduces information available to PrefixSpan algorithm and discovered patterns do

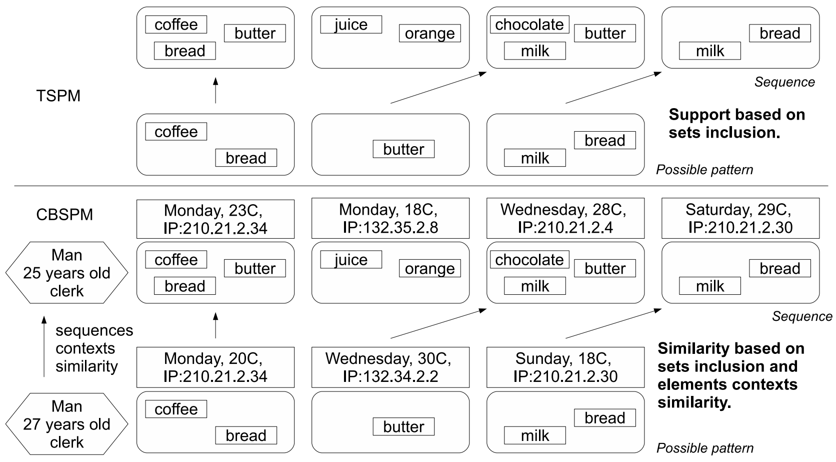
TSPM
coffee
bread
butter
juice
orange
chocolate
milk
butter
milk
bread
Sequence
Support based on sets inclusion.
coffee
bread
butter
milk
bread
Possible pattern
CBSPM
Monday, 23C, IP:210.21.2.34
Monday, 18C, IP:132.35.2.8
Wednesday, 28C, IP:210.21.2.4
Saturday, 29C, IP:210.21.2.30
Man 25 years old clerk
sequences contexts similarity
Monday, 20C, IP:210.21.2.34
Wednesday, 30C, IP:132.34.2.2
Sunday, 18C, IP:210.21.2.30
Man 27 years old clerk
Similarity based on sets inclusion and elements contexts similarity.
Possible pattern

Figure 1. Comparing a pattern and a sequence in TSPM (an upper part of the figure) and CBSPM (a lower part).

not contain original values of attributes as it occurs in the case of the CBSPM.

3. Measuring similarity between context patterns

A similarity of traditional sequences have been already studied mainly for a clustering task – see e.g. Ronkainen (1998), Morzy, Wojciechowski, Zakrzewicz (1999), Guralnik, Karypis (2001). However the task of comparing context sequences is more complex. So, a new measure must be designed. General assumptions for this measure are the following:

1. Measure values are normalised in the range from 1.0 to 0.0 where 1.0 means that both sets of patterns are exactly the same and the value 0.0 means that there are no similar information related to the context, itemsets and elements order between both pattern sets.

2. The measure should "punish" two undesired situations: when the set of discovered patterns is more numerous than the reference set (CBSPM may produce a lot of patterns) and when the algorithm discovers a smaller set of patterns than a reference one (e.g. as a result of inappropriate discretization).

3. It should reflect a case when a shorter discovered pattern could be compared to a different combination of elements from the longer reference pattern. It may happen in a situation, where a longer pattern from the reference set contains few repetitions of the shorter one from the discovered set.

4. A non-similar element in comparing patterns is treated as a kind of noise. Adequately occurrence of unused elements in reference patterns should be negatively reflected.

3.1. Comparing two context patterns

Each element E from pattern mp =$(D, S = \langle E_1, E_2, ..., E_k\rangle)$ contains an element context instance c and an itemset X . The evaluation of an itemsets similarity requires application of Jaccard's coefficient (Guralnik, Karypis (2001)). Assuming that element $E_1(C_1, X_1)$ is compared to the element form the second pattern $E_2(C_2, X_2)$ the Jaccard's coefficient is defined as:

$$\Theta^X(X_1, X_2) = \frac{|X_1 \cap X_2|}{|X_1 \cup X_2|}$$

If both elements have no common items the measure returns 0.0. Respectively if both itemsets are identical it gives 1.0. Similarity of context attributes is calculated using exactly the same way as it was done in context patterns mining. However, let us stress that in the thresholds for minimal similarities are not used here. Therefore a similarity between context attributes is calculating straightforward using attribute similarity function: $\sigma^C(c_1, c_2)$ for element contexts or $\sigma^D(d_1, d_2)$ for sequence contexts.

In a more difficult case of TSPM the discretization fold is compared to a value of context attributes in the reference set. For a discretization fold an average similarity value is build using upper $\overline{X_2^{D,C}}$ and lower $\underline{X_2^{D,C}}$ boundary of discretization fold. The aggregated similarity value is calculated using the same functions as in the corresponding CBSPM:

$$\Theta^C(C_1, C_2) = \Theta^C(\sigma_1^{CX}(c_1^1, X_2^C), \sigma_2^{CX}(c_1^2, X_2^C), ..., \sigma_w^{CX}(c_1^w, X_2^C))$$

$$\Theta^D(D_1, D_2) = \Theta^D(\sigma_1^{DX}(d_1^1, X_2^D), \sigma_2^{DX}(d_1^2, X_2^D), ..., \sigma_v^{DX}(d_1^v, X_2^D))$$

The TSPM uses the same context similarity aggregation function in evaluation experiments as CBSPM to achieve comparable results. In experiments presented later in this papera weighted sum function (weights were equal) has been applied to aggregate similarities of context's attributes. Total similarity of compared elements (considering both aggregated similarities of contexts and itemset similarity) is calculated according to the following formulas:

$$\Psi^C(E_1(C_1, X_1), E_2(C_2, X_2)) = \Theta^C(C_1, C_2) \cdot \Theta^X(X_1, X_2)$$

$$\Psi^D(mp_1(D_1, S_1), mp_2(D_2, S_2)) = \Theta^D(D_1, D_2)$$

where $E_1 \in mp_1$ and $E_2 \in mp_2$ in both CBSPM and TSPM approaches. While comparing two patterns mp_1 and mp_2 we often encounter a situation where compared patterns contain only a limited number of common elements. Such common elements create a core mr_{mp_1,mp_2} identified as a common maximal sub-sequence of both patterns. Then, the core sub-sequence must contains elements fulfilling the condition $\Psi^C(E_1(C_1, X_1), E_2(C_2, X_2)) > 0$. The core coefficient is defined as:

$$\Lambda(mp_1, mp_2) = \frac{|mr_{mp_1,mp_2}|}{max(|mp_1|, |mp_2|)}$$

to take into account covered common elements in the reference pattern mp_1 and other "noisy" elements in the discovered pattern mp_2 . The similarity of two patterns mp_1, mp_2 with the respect to the core mr_{mp_1,mp_2}, sequence's contexts and elements similarities is calculated according to a following equation:

$$mrsim(mp_1, mp_2, mr_{mp_1,mp_2}) = \frac{\Lambda(mp_1, mp_2) \cdot \Psi^D \cdot (\Psi_1^C + \Psi_2^C + ... + \Psi_r^C)}{r}$$

where $r = |mr_{mp_1,mp_2}|$. While comparing two patterns which are not the same size it is possible to met a case where a few different combinations of similar elements in multiple cores can be found. Let MR denotes a set of such cores. So, the total similarity of two patterns mp_1 , mp_2 is calculated by averaging similarities counted for all possible cores in both patterns:

$$mpsim(mp_1, mp_2) = \frac{\sum\limits_{mr_{mp_1,mp_2} \in MR_{mp_1,mp_2}} mrsim(mp_1, mp_2, mr_{mp_1,mp_2})}{|MR_{mp_1,mp_2}|}$$

3.2. Comparing two sets of context patterns

Let us consider two sets of reference patterns MP_r and the discovered ones MP_d. The values off similarity between pairs o patterns from sets MP_r, MP_d are stored in a similarity matrix. The *reconstruction measure* evaluates a degree of re-discovery hidden patterns and is defined as:

$$RMSim(MP_r, MP_d) = \frac{\sum_{i=1,...,k} Lsim(i)}{|MP_r|}$$

where $k = |MP_r|$ is the number of patterns in the reference set,

$$Lsim(i) = \sum_{mp_f \in L(mp_i)} mpsim(mp_f, mp_i)/|L(mp_i)|$$

and $L(mp_i)$ is a list of the most similar discovered patterns mp_f to the given i-th reference pattern mp_i. Technically this list is constructed in the following way: for each discovered pattern we find the most similar reference pattern and put it to its list. If the similarity value of the next similar reference patterns is very close (with respect to a threshold), the discovered pattern is also added to their lists. If any list is empty then $Lsim(i)$ is equal 0.0. We can say that the reconstruction measure reflects the coverage of reference patterns by discovered patterns.

A next measure defined here is an *average similarity measure*. Its purpose is to reflect an overall content of information related to the reference patterns found in discovered patterns regardless of any noise they can contain. By referencing to the previous measure this one is calculated simply by adding all similarity values between a reference pattern and mined patterns to the reference pattern list. Lists aggregation procedure is exactly the same. This similarity measure compares all discovered patterns with each reference pattern considering even completely dissimilar pairs. Therefore values of the measure are lower than values of reconstruction measure because later is focusing mainly on a small fraction of discovered patterns most similar to the reference pattern.

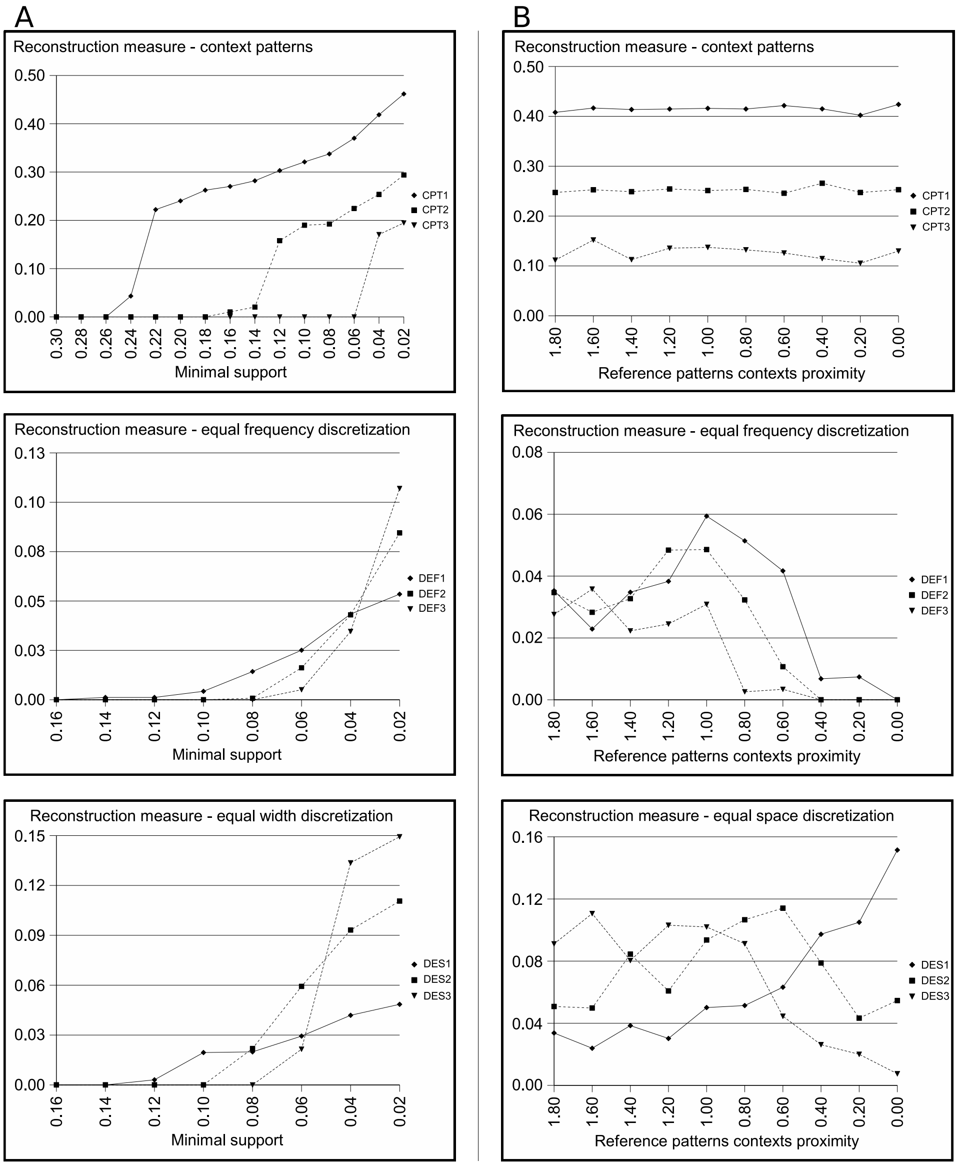

Figure 2. (A) The quality of patterns discovered by TSPM, CBSPM algorithms with respect to changes of minimal support threshold. (B) Changing the "proximity" between nodes for values of context attributes(TSPM, CBSPM).

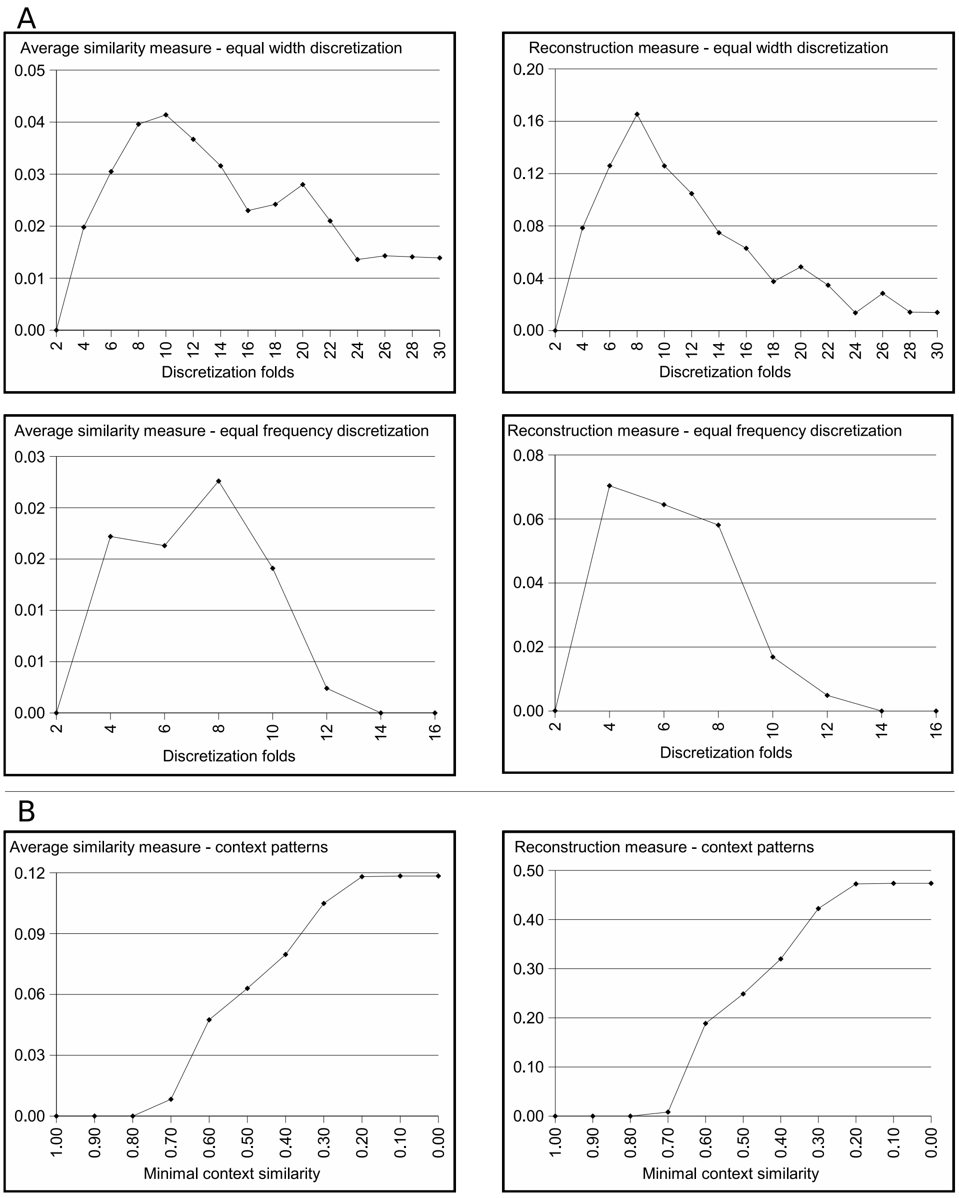

Figure 3. (A) The change of the quality of patterns discovered by TSPM algorithm in a relation to a number of discretization folds. (B) The change of the quality of patterns discovered by CBSPM algorithm with respect to the threshold of minimal context attribute similarity.

4. Experiments

The goal of the experiments is to compare the quality of patterns sets mined using CBSPM and TSPM with prediscretization. The quality reflects a rediscovery degree of reference patterns' set hidden in the sequences database. Compared algorithms are ContextMappingImproved and PrefixSpan with numerical attributes prediscretizations based on an equal contexts distribution in folds (equal frequency) and an equal folds width.

A data source for experiments is constructed from artificial databases containing a fixed number of hidden reference patterns. Patterns are characterized by defined length, itemsets structure, context size and support. For each reference pattern a required number of unique itemsets is randomly constructed from the set of available items. Then a required number of patterns is replicated to sequences in the database according to the assumed support threshold in such a way that they are super-sets of the reference pattern. If the length of sequence should be greater than the pattern's length, additional random itemsets are randomly introduced to sequences. Values of context attributes used in reference patterns are randomly generated using a kind of a hypergrid structure in the multidimensional real number space. Distances between nodes of this hypergrid can be changed to make contexts more distant or closer. Context values of reference patterns hidden in sequences database are distorted randomly within a defined sphere and rotated randomly in each dimension. It ensures more realistic distribution of context values.

For the context algorithm we will use a similarity function for a single attribute based on the following formula:

$$\sigma^{C,D}(c_1, c_2) = \begin{cases} 1.0 - |c_1 - c_2| & if \quad |c_1 - c_2| < 1.0 \\ 0.0 & if \quad |c_1 - c_2| \geq 1.0 \end{cases}$$

The ContextMappingImproved has been executed with context similarity thresholds θ^C and θ^D equal to 0.4 (CPT1 label on figures), 0.6 (CPT2) and 0.8 (CPT3). The PrefixSpan discretization splits each attribute domain on 3, 5, 7 folds. These "options" are denoted as DES1, DES2, DES3 for the equal width discretization and DEF1, DEF2, DEF3 for the equal frequency discretization on figures. The presented results were obtained for database containing 8 hidden reference patterns containing 4 elements with support equal to 0.25. Each database sequence has length of 12 elements and it contains sub-sequences that belongs to 2 hidden reference patterns and 4 additional randomly created elements. Random elements' itemsets do not overlap with itemsets used in reference pattern itemsets. A number of sequences in generated databases was 500. Sequence contexts and element contexts contain 2 attributes created around hypergrid nodes. The distance between nodes in three first experiments were greater than distortion spheres around nodes. In the fourth experiment the distance was reduced gradually to zero. Considering much higher numbers of sequences and the sequence's length could make experiment infeasible with low values of a minimal support threshold.

First results presented on figure 2A illustrate the influence of the minimal support thresholds on both quality measures for compared algorithms. The second experiment verifies a relation between a number of discretization folds and quality measures – see figure 3A (the minimal support during mining was reduced to 0.04 because of late finding of patterns in PrefixSpan with discretization noticed on figure 2A). The third experiment evaluates the change of minimal similarity thresholds for minimal element and sequence context's similarities in ContextMappingImproved, see figure 3B. The final experiment verifies the influence of the distance between nodes in the hypergrid structure (used for reference patterns' contexts generation) on the change of quality measures. The distance between nodes was reduced maintaining the same radius of the distortion sphere. In consequence contexts that belongs to different patterns elements begin to overlap in the attribute domain space creating a kind of noise – see figure 2B.

5. Conclusions

Let us comment results of experiments. In the first experiment values reconstruction measure results for specific CBSPM algorithm are at least 5 times higher than the transformed approach for TSPM (see figure 2A). Moreover, only the CBSPM algorithm could discover patterns with a minimal support threshold comparable to support of hidden patterns. On the other hand, we can say the prediscretization methods splited context attributes value space on small folds containing rather a small number of instances. Therefore TSPM algorithms could find patterns when density of instances in folds goes above the minimal support threshold - so for much smaller support threshold than CBSPM. It seems that CBSPM can work well because it recognises "dense neighborhoods" around reference sets of context attributes. The TSPM with equal width prediscretization detects dense clusters poorly. The worst quality was obtained by the TSPM with equal frequency. A greater number of prediscretization folds could improve working of TSPM algorithm. However, increasing the number of folds effects in a lower minimal support threshold required to find any patterns (see figure 2A where EFD3 finds patterns with a lower minimal support than EFD1.). However, the number of folds cannot be very high – see figure 3A – because a low minimal support threshold may result in mining a lot of patterns containing "noise". Further experiment for the CBSPM algorithm proved that minimal similarity thresholds should be tuned in the range 0.3-0.7, see figure 3B. Minimal similarity thresholds determine a size of a reference context "neighbourhood" where supporting instances of contexts can be found. The CBSPM algorithm mines more patterns if thresholds are getting lower (and the neighbourhood is growing). However such patterns can decrease quality. Finally, the fourth experiment proved that reducing distance between values of reference context attributes has much weaker impact on CBSPM algorithm performance than on TSPM algorithms – see figure 2B. All experiments with rediscovery of hidden patterns clearly showed that the ContextMappingImproved worked better than previously known algorithms using a transformation approach to pre-discretization. The computational costs were only slightly higher on some cases and even smaller on others comparing to pre-discretization with too many intervals.

References

Agrawal, R. and Srikant, R. (1994) Fast algorithms for mining association rules. *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, 487–499

Agrawal, R., Gehrke, J., Gunopulos, D. and Raghavan, P. (1998) Automatic subspace clustering of high dimensional data for data mining applications. 94–105

Guralnik, V. and Karypis, G. (2001) A Scalable Algorithm for Clustering Sequential Data. *ICDM*, 179–186.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. and Hsu, M.-C. (2001) PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. *Proceedings of the 17th International Conference on Data Engineering*, 215–224.

Morzy, T. (2004) Discovery associations; algorithms and datastructures. PAN Press, OWN Poznan.

Morzy, T., Wojciechowski, M. and Zakrzewicz, M. (1999) Pattern-Oriented Hierachical Clustering. *Advances in Databases and Information Systems*, 179–190.

Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q. and Dayal, U. (2001) Multi-dimensional sequential pattern mining. *Proceedings of the tenth international conference on Information and knowledge management*, 81–88.

Srikant, R. and Agrawal, R. (1996) Mining Sequential Patterns: Generalizations and Performance Improvements. *Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology*, **1057**, 3–17.

Srikant, R. and Agrawal, R. (1996) Mining Quantitative Association Rules in Large Relational Tables. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, 1–12.

Stefanowski, J. and Ziembiński, R. (2005) Mining Context Based Sequential Patterns. *Proceedings of the Third International Atlantic Web Intelligence Conference: Advances in Web Intelligence*, **3528**, 401–407.

Ziembiński, R. (2007) Algorithms for Context Based Sequential Pattern Mining. *Fundamenta Informaticae*, **76(4)**, 495–510.