Local Projection in Jumping Emerging Patterns Discovery in Transaction Databases*

Pawel Terlecki and Krzysztof Walczak

Institute of Computer Science, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
{P.Terlecki, K.Walczak}@ii.pw.edu.pl

Abstract. This paper considers a rough set approach for the problem of finding minimal jumping emerging patterns (JEPs) in classified transactional datasets. The discovery is transformed into a series of transaction-wise local reduct computations. In order to decrease average subproblem dimensionality, we introduce local projection of a database. The novel algorithm is compared to the table condensation method and JEP-Producer for sparse and dense, originally relational data. For a more complete picture, in our experiments, different implementations of basic structures are considered.

Keywords: jumping emerging pattern, transaction database, local reduct, rough set, local projection.

1 Introduction

Pattern mining is one of key tasks in contemporary knowledge discovery. Although recent years have brought a wide spectrum of pattern types, discovery algorithms still follow common strategies such as Apriori, operations on concise representations ([1]), pattern trees ([2]). Regardless of a particular method, processing may involve exponentially large item set collections, which makes overall feasibility very sensitive to input data. Therefore, in our opinion, it is crucial to study, how to approach datasets of certain characteristics.

Here, we look at the problem of finding jumping emerging patterns (JEPs) in classified transaction databases. A JEP refers to an itemset that is supported in one class and absent from others. This highly discriminative idea was introduced in [3], and, since then, it has been successfully applied to business and gene expression problems. Because all JEPs constitute a convex space, the task is often perceived as finding minimal patterns. In fact, these patterns have found valuable applications to classification and clustering ([3]).

Among known algorithms, JEP-Producer ([1]) is believed to be the most efficient solution for finding a complete space of JEPs. It operates on concise representation of convex collections and employs a border differentiation operation to obtain a result set. In our previous works ([4]), it has been demonstrated

* The research has been partially supported by grant No 3 T11C 002 29 received from Polish Ministry of Education and Science.

T. Washio et al. (Eds.): PAKDD 2008, LNAI 5012, pp. 723–730, 2008.

that reducts from the rough set theory ([5]) are closely related to JEPs. Algorithms based on these relations appeared superior in experiments for relational data ([6]). Moreover, even if data is originally given in a transactional form, a condensed decision table can be efficiently obtained by finding an approximate graph coloring in an item-conflict graph ([7]).

Following successful results for dense, originally relational data, we decided to examine opportunities for reduct-based methods against a popular class of sparse transaction databases. Note that, for large datasets, table condensation may likely deal with adverse item distribution in transactions, which results in low dimensionality reduction and inefficient discovery. The method of local projection that is put forward in this paper ascertains average dimensionality to depend only on average transaction length, not on item distribution in a database. The problem is decomposed into a series of per transaction local reduct computations in a locally projected decision table. For each subproblem only objects and attributes substantial for reduct induction are taken into account, which significantly improves overall efficiency. In addition, we propose several optimization to decrease a construction overhead of discernibility matrices.

Our experiments covered efficiency comparison between JEP-Producer, table condensation and local projection with different reduct computation methods. Approaches were tested against originally relational and sparse datasets. Since actual performance depends strongly on implementation, different structures to represent an attribute/item set were tested.

Section 2 provides fundamentals of emerging patterns and border representation. In Sect. 3, we present basic elements of the rough set theory. Local projection is introduced and proved correct in Sect. 4. In Sect. 5 the novel algorithm is described. It also discusses optimizations for discernibility matrix computation and impact of different implementations of main structures. Section 6 covers testing procedure and experimental results. The paper is concluded in Sect. 7.

2 Emerging Patterns

Let a transaction system be a pair $(\mathcal{D}, \mathcal{I})$, where $\mathcal{D}$ is a finite sequence of transactions $(T_1, .., T_n)$ (database) such as $T_i \subseteq \mathcal{I}$ for $i = 1, .., n$ and $\mathcal{I}$ is a nonempty set of items (itemspace). The support of an itemset $X \subseteq \mathcal{I}$ in a sequence $D = (T_i)_{i \in K} \subseteq \mathcal{D}$ is defined as $supp_D(X) = \frac{|\{i \in K : X \subseteq T_i\}|}{|K|}$, where $K \subseteq \{1, .., n\}$.

Let a decision transaction system be a tuple $(\mathcal{D}, \mathcal{I}, \mathcal{I}_d)$, where $(\mathcal{D}, \mathcal{I} \cup \mathcal{I}_d)$ is a transaction system and $\forall_{T \in \mathcal{D}} |T \cap \mathcal{I}_d| = 1$. Elements of $\mathcal{I}$ and $\mathcal{I}_d$ are called condition and decision items, respectively. Support in a decision transaction system $(\mathcal{D}, \mathcal{I}, \mathcal{I}_d)$ is understood as support in the transaction system $(\mathcal{D}, \mathcal{I} \cup \mathcal{I}_d)$.

For each decision item $c \in \mathcal{I}_d$, we define a decision class sequence $C_c = (T_i)_{i \in K}$, where $K = \{k \in \{1, .., n\} : c \in T_k\}$. For convenience, the notations C_c and $C_{\{c\}}$ are used interchangeably. Note that each of the transactions from $\mathcal{D}$ belongs to exactly one class sequence. In addition, for a database $D = (T_i)_{i \in K \subseteq \{1,..,n\}} \subseteq \mathcal{D}$, we define a complementary database $D' = (T_i)_{i \in \{1,..,n\} - K}$.

Given two databases $D_1, D_2 \subseteq \mathcal{D}$, in particular decision classes, we define a jumping emerging pattern (JEP) from D_1 to D_2 as an itemset $X \subseteq \mathcal{I}$ such as $supp_{D_1}(X) = 0$ and $supp_{D_2}(X) > 0$. A set of all JEPs from D_1 to D_2 is called a JEP space and denoted by $JEP(D_1, D_2)$.

JEP spaces can be described concisely by borders ([1]). For $c \in I_d$, we use a border $< \mathcal{L}_c, \mathcal{R}_c >$ to uniquely represent a JEP space $JEP(C'_c, C_c)$. Members of the left bounds are minimal JEPs, whereas member of the right bounds are maximal JEPs, i.e. distinguishable transactions.

The problem of JEP discovery can be defined as computing $\{< \mathcal{L}_c, \mathcal{R}_c >\}_{c \in I_d}$ for a decision transaction system $(\mathcal{D}, \mathcal{I}, \mathcal{I}_d)$. Since finding of right bounds is trivial ([1]) and not interesting from a practical point of view, we focus on the collection of left bounds $\{\mathcal{L}_c\}_{c \in I_d}$.

3 Rough Sets

Let a decision table be a triple $(\mathcal{U}, \mathcal{C}, d)$, where $\mathcal{U}$ (universum) is a non-empty, finite set of objects, $\mathcal{C}$ is a non-empty finite set of condition attributes and d is a decision attribute. A set of all attributes is denoted by $\mathcal{A} = \mathcal{C} \cup \{d\}$. The domain of an attribute $a \in \mathcal{A}$ is denoted by V_a and its value for an object $u \in \mathcal{U}$ is denoted by a(u). In particular, $V_d = \{c_1, .., c_{|V_d|}\}$ and the decision attribute induces a partition of $\mathcal{U}$ into decision classes $\{U_c\}_{c \in V_d}$. Hereinafter, we use the term *attribute* to denote a condition attribute.

Consider $B \subseteq \mathcal{A}$. An indiscernibility relation $IND(B)$ is defined as $IND(B) = \{(u, v) \in \mathcal{U} \times \mathcal{U} : \forall_{a \in B}\ a(u) = a(v)\}$. Since $IND(B)$ is an equivalence relation it induces a partition of $\mathcal{U}$ denoted by $\mathcal{U}/IND(B)$. Let $B(u)$ be a block of the partition containing $u \in \mathcal{U}$. A B-lower approximation of a set $X \subseteq \mathcal{U}$ is defined as follows: $B_*(X) = \{u \in \mathcal{U} \mid B(u) \subseteq X\}$ and a B-positive region with respect to a decision attribute d is defined as $POS(B, d) = \bigcup_{X \in \mathcal{U}/IND(\{d\})} B_*(X)$.

A local reduct for an object $u \in \mathcal{U}$ is a minimal attribute set $B \subseteq \mathcal{C}$ such that $\forall_{c \in V_d}(\mathcal{C}(u) \cap U_c = \emptyset \implies B(u) \cap U_c = \emptyset)$. It means that the object u can be differentiated by means of B from all objects from other classes as well as using $\mathcal{C}$. The set of all local reducts for an object u is denoted by $REDLOC(u, d)$.

4 Local Projection

In order to apply the rough set framework to transactional data, transformation to a respective relational form is required. We consider two representations: a binary decision table, which already found an application to negative pattern discovery ([4]), and a locally projected form - introduced in this paper for efficient finding of positive patterns.

Hereinafter, we assume that our input data is represented by a decision transaction system $\mathcal{DTS} = (\mathcal{D}, \mathcal{I}, \mathcal{I}_d)$, where $\mathcal{D} = (T_1, .., T_n)$, $\mathcal{I} = \{I_1, .., I_m\}$, $\mathcal{I}_d = \{c_1, .., c_p\}$, $K = \{1, .., n\}$.

A binary decision table for a decision transaction system $\mathcal{D}TS$ is a decision table $BDT_{\mathcal{D}TS} = (\mathcal{U}, \mathcal{C}, d)$ such that $\mathcal{U} = \{u_1, .., u_n\}$, $\mathcal{C} = \{a_1, .., a_m\}$, $V_d = \{c_1, .., c_p\}$; $a_j(u_i) = \begin{cases} 0, I_j \notin T_i \\ 1, I_j \in T_i \end{cases}, \forall_{i \in 1..n, j \in 1..m}$; $d(u_i) = T_i \cap \mathcal{I}_d, \forall_{i \in 1..n}$.

Local reducts in a binary decision table correspond to jumping emerging patterns with negation (JEPNs, [4]). JEPNs constitute a convex space that contains JEPs for the same transaction system. Note that, although $\{e, g\}$ and $\{d, f\}$ are both local reducts for u_1, the pattern eg is a minimal JEP, whereas df is not, since it is not supported by the respective transaction.

Solving a problem of a double dimensionality and filtering positive patterns is most often expensive, thus, the idea of a table condensation was proposed ([7]). Before local reduct computation, binary attributes are aggregated into multivalued attributes by means of an approximate graph coloring. This approach is efficient for originally relational datasets, however, remains sensitive to a distribution of items in transactions.

The table condensation leads to an alternative representation of a decision transaction system. However, one may get much higher complexity reduction if transformation is performed independently for every transaction. The following structure demonstrates how we may limit our interest only to items that are indispensable to compute complete discernibility information for a transaction.

A locally projected decision table for: DTS, a decision transaction system, and $T_i \in \mathcal{D}$, where $i = 1, .., |\mathcal{D}|$, a transaction, is a binary decision table $LPDT_{\mathcal{D}TS,T_i} = BDT_{DTS_i,T_i}$, where $DTS_i = (\mathcal{D}_i, \mathcal{T}_i, \mathcal{I}_d)$ and $\mathcal{D}_i = (T_k \cap T_i)_{k \in K}$.

Hardness of an input decision system DTS can be characterized by average (maximal) dimensionality of subproblems, i.e. a locally projected decision table for distinguishable transactions, namely $avgDim(DTS) = \{|T| : T \in \mathcal{R}_c \wedge c \in I_d\} / \sum_{c \in I_d} |\mathcal{R}_c|$ and $maxDim(DTS) = max_{T \in \mathcal{R}_c \wedge c \in I_d} |T|$. Note that, when all transactions are distinguishable, these parameters refer to an average (maximal) transaction length DTS.

For the sake of convenience, we use the notation: $itemPatt_{DTS,T_i}(u, B) = \{I_k \in T_i : a_k \in B \wedge a_k(u) = 1 \wedge k \in M'\}$, where $u \in \mathcal{U}$, $B \subseteq \mathcal{C}_i = \{a_k\}_{k \in M'}$, $LPDT_{\mathcal{D}TS,T_i} = (\mathcal{U}, \mathcal{C}_i, d)$ is a locally projected decision table and a transaction $T_i = \{I_k\}_{k \in M'}$, $M' \subseteq \{1, .., m\}$. Note that $|itemPatt_{DTS,T_i}(u_i, B)| = |B|$. Whenever a decision transaction system is known from the context, the respective subscript is omitted.

The following theorem states that the complete JEP space for the DTS and a given class can be obtained by finding a locally projected tables for each distinguishable transaction and generating patterns for the respective objects and any attribute set in the respective table.

Theorem 1. $\forall_{c \in \mathcal{I}_d} \{itemPatt_{DTS,T_i}(u_i, R) : i \in K \wedge LPDT_{DTS,T_i} = (\mathcal{U}, \mathcal{C}_i, d) \wedge u_i \in POS(\mathcal{C}_i, d) \cap U_c \wedge R \subseteq \mathcal{C}_i\} = JEP(C'_c, C_c)$.

The respective left bound of a JEP space can be found by applying local reducts for a given object instead of any attribute sets.

Theorem 2. $\forall_{c \in \mathcal{I}_d}\{itemPatt_{DTS,T_i}(u_i, R) : i \in K \wedge LPDT_{DTS,T_i} = (\mathcal{U}, \mathcal{C}_i, d) \wedge u_i \in POS(\mathcal{C}_i, d) \cap U_c \wedge R \in REDLOC(u_i, d)\} = \mathcal{L}_c$.

The proofs are omitted here due to space limitations.

5 JEP Computation

Minimal jumping emerging patterns in $\mathcal{DTS} = (\mathcal{D}, \mathcal{I}, \mathcal{I}_d)$ can be computed by local reduct computation in locally condensed tables for all transactions. The actual procedure is straightforward and fully based on Theorem 2.

1: $\mathcal{L}_c = \emptyset$ for each $c \in I_d$
2: **for** $(k = 1; 1 <= |\mathcal{D}|; k++)$ **do**
3: Construct a locally projected decision table $LPDT_{DTS,T_k}$
4: Compute $REDLOC(u_k, d)$ in $LPDT_{DTS,T_k}$
5: $\mathcal{L}_c = \mathcal{L}_c \cup \{itemPatt_{DTS,T_k}(u_k, R) : R \in REDLOC(u_k, d)\}$, $c = T_k \cup \mathcal{I}_d$
6: **end for**

Identification of minimal patterns by means of local reduct induction is the most complex part of our approach. It is normally addressed with methods used for global reducts ([5]). Unfortunately, all known exact solutions are pessimistically exponential.

Here, we look at two algorithms that employ a discernibility matrix. The first one reduces the problem to finding prime implicants of a monotonous boolean functions ([5], RedPrime). It loops over elements of a matrix and extends a collection of reducts for rows seen so far, so that they are sufficient to discern the current row as well. The second algorithm traverses a lattice of all subsets of an attribute space using the apriori scheme ([8], RedApriori). Successive collections of candidates are pruned basing on a degree of attribute set dependence, which is calculated by means of a discernibility matrix. Also, in order to optimize this stage, one may eliminate transactions that are not maximal JEPs and group transactions by their classes.

6 Experimental Results

Experiments focused on efficiency of the new algorithm, table condensation and JEP-Producer for synthetically generated sparse data and dense data obtained from relational tables. Each result was averaged over several executions.

The testing environment and algorithms were coded in Java 5. Since the rough set methods and JEP-Producer differ significantly, it is not possible to come up with one single dominant operation for time complexity representation. Therefore, in order to provide reliable time measurements, we based their implementations on mostly the same structures. In particular, all the studied approaches process large collections of attribute/item sets. To obtain results possibly independent from what a data structure was used to represent such a set, three

implementations were tested. The first two are characteristic vectors of an attributes/item space, one based on a regular byte array (Array) and the other one - on java.util.BitSet (BitSet). The third structure is a balanced binary tree implemented by means of java.util.TreeSet (TreeSet). Array is generous in memory allocation, but assures the most efficient access. Bit and dynamic structures are slower, however, they may take precedence for large attribute/item spaces when a high number of sets is created.

Table 1. Synthetic dataset summary with problem hardness characteristics

No	Trans	Items	Classes	MaxTrans	JEPs	avgDim	maxDim
1	2000	20	2	326	539	5,09	9,00
2	2000	40	3	1075	5967	6,33	14,00
3	2000	60	3	1551	19140	6,79	16,00
4	2000	80	2	1858	71250	9,37	19,00
5	5000	50	2	2918	20088	6.25	15.00
6	10000	50	3	4119	18673	5.57	12.00
7	15000	50	2	7920	94252	7.57	18.00
8	20000	50	2	10300	126162	7.63	18.00

Sparse Data. In this test local projection and JEP-Producer are compared for sparse datasets. Since itemspaces are commonly much larger than average transaction size, this kind of data is substantial for practical tasks. Unfortunately, it is hard to find publicly available classified sparse datasets, thus, the test was performed against synthetic data. Transaction databases were produced by means of the IBM generator ([9]) and, then, the CLUTO package was used to classify transactions (Tab. 1). The density of each database was set up at 5-15% of a respective item space. We studied behavior of the algorithms when a size of a database or an item space increases. In order to describe the actual hardness of each problem, additional measures were provided, in particular, a total number of JEPs over all classes, number of maximal transactions and average (maximal) dimensionality.

The local projection algorithm was tested with two different reduct computation methods: RedPrime ([5]) and RedApriori ([8]). JEP-Producer was implemented according to the scheme and optimizations described in [1]. To optimize all computations a database is always reduced to contain only maximal transactions. Measurements for all the algorithms were taken for the aforementioned implementations of an attribute/item set.

Table 2 shows that the rough set approach outperforms JEP-Producer. In particular, for RedApriori and Array, there is a difference of 1-2 orders of magnitude. In general, all the methods perform well for Array. Reduct computations are performed for locally projected tables with small attribute spaces, thus, slower structures significantly affect the overall performance. For example, for TreeSet, efficiency of RedPrime and JEP-Producer remain very close. On the other hand, JEP-Producer is sensitive to the size of a whole item space, therefore, BitSet led to slightly better results in almost all the cases.

Table 2. Execution time comparison for sparse datasets between local projection and JEP-Producer for different implementations of an attribute/item set

No	RedPrime			RedApriori			JEP-Producer		
	Array	BitSet	TreeSet	Array	BitSet	TreeSet	Array	BitSet	TreeSet
1	297	484	797	138	218	470	594	640	1296
2	4906	8938	19063	1796	3196	10262	14375	13469	28547
3	20453	34860	78406	4553	8120	29351	53281	47250	93562
4	202328	323296	810360	48469	77129	340541	217796	164594	346265
5	26532	45219	95203	6573	10669	36239	118453	97735	200094
6	28750	55906	104812	4071	7608	21565	215250	190906	390937
7	671390	1123562	2739329	162874	263418	1033108	1311203	1169141	2623266
8	877655	1468744	3582734	243205	393338	1490587	2153109	1982421	4316875

Originally Relational Data. Earlier tests demonstrated that, for dense, originally relational datasets, condensation successfully reduces dimensionality and performs better that JEP-Producer ([6]). Here, it is contrasted with local projection. Due to space limitations, results for RedPrime and Array-based attribute set implementation are presented. Transactional databases for this test were generated from relational tables from UCI Repository. Average time and dimensionality is given for each of the methods.

According to the results in Tab. 3, table condensation and local projection lead most often to the same subproblem dimensionality. Since databases are reduced in an analogical way, both methods achieve similar efficiency. Nevertheless, the former strongly relies on optimality of graph coloring solution. An overhead of generation and filtering of additional patterns is visible for *mushroom*.

Table 3. Execution time comparison for originally relational datasets between table condensation and local projection with RedPrime and Array-based implementation

Dataset	Trans	Items	Class	JEPs	Table Condensation		Local Projection	
					avgDim	Time	avgDim	Time
lymn	148	59	4	6794	18.00	13156	18.00	8359
house	435	48	2	6986	16.00	27218	16.00	21141
balance	625	20	3	303	4.00	671	4.00	406
tic-tac-toe	958	27	2	2858	9.00	9203	9.00	7578
car-mod	1728	21	4	246	6.00	4109	6.00	3985
mushroom	8124	117	2	3635	23.00	608546	22.00	440656
nursery	12960	27	5	638	8.00	477484	8.00	495375
krkopt	28056	43	18	21370	6.00	1754469	6.00	1728782

7 Conclusions

In this paper we have proposed a rough set approach to discovery of jumping emerging patterns (JEPs) in classified transaction databases. The problem is

decomposed into a series of local reduct computations performed for locally projected decision tables for each transaction.

Main benefit of our approach is that only the transactions and items necessary for each computation are considered, which results in potentially significant dimensionality reduction of subproblems. In this case, additional processing can be a significant factor. The way of discernibility matrices construction can be optimized by caching of partial per-attribute results in complementary form.

Experiments have proved that the method outperforms JEP-Producer, the most popular solution for the considered problem, for sparse, synthetically generated datasets. The high efficiency is a result of a dramatic decrease in average dimensionality. This fact was observed independently from a reduct computation method. Nevertheless, the algorithm based on attribute set dependence behaves much better than the classical one searching for prime implicants and is faster than JEP-Producer by 1-2 orders of magnitude. On the other hand, for dense, originally relational data the new approach achieves at least the same dimensionality gain as the previously proposed method of table condensation and gives similar overall efficiency.

The future research will extend our method to look for derivative types of patterns and confront its efficiency with existing tree-based strategies.

References

1. Dong, G., Li, J.: Mining border descriptions of emerging patterns from dataset pairs. Knowl. Inf. Syst. 8(2), 178–202 (2005)
2. Li, J., Liu, G., Wong, L.: Mining statistically important equivalence classes and delta-discriminative emerging patterns. In: KDD 2007, pp. 430–439 (2007)
3. Li, J., Dong, G., Ramamohanarao, K.: Making use of the most expressive jumping emerging patterns for classification. Kn. In. Sys. 3(2), 1–29 (2001)
4. Terlecki, P., Walczak, K.: Jumping emerging patterns with negation in transaction databases - classification and discovery. Information Sciences 177, 5675–5690 (2007)
5. Bazan, J., Nguyen, H.S., Nguyen, S.H., Synak, P., Wroblewski, J.: Rough set algorithms in classification problem. Rough set methods and applications: new develop. in knowl. disc. in inf. syst., 49–88 (2000)
6. Terlecki, P., Walczak, K.: Local reducts and jumping emerging patterns in relational databases. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 358–367. Springer, Heidelberg (2006)
7. Terlecki, P., Walczak, K.: Jumping emerging pattern induction by means of graph coloring and local reducts in transaction databases. In: An, A., Stefanowski, J., Ramanna, S., Butz, C.J., Pedrycz, W., Wang, G. (eds.) RSFDGrC 2007. LNCS (LNAI), vol. 4482, pp. 363–370. Springer, Heidelberg (2007)
8. Terlecki, P., Walczak, K.: Attribute set dependence in apriori-like reduct computation. In: Wang, G.-Y., Peters, J.F., Skowron, A., Yao, Y. (eds.) RSKT 2006. LNCS (LNAI), vol. 4062, pp. 268–276. Springer, Heidelberg (2006)
9. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB 1994, pp. 487–499 (1994)