Is Pushing Constraints Deeply into the Mining Algorithms Really What We Want?

An Alternative Approach for Association Rule Mining

Jochen Hipp
DaimlerChrysler AG,
Research & Technology, Ulm, Germany
jochen.hipp@daimlerchrysler.com

Ulrich Güntzer
Wilhelm Schickard-Institute,
University of Tübingen, Germany
guentzer@informatik.uni-tuebingen.de

ABSTRACT

The common approach to exploit mining constraints is to push them deeply into the mining algorithms. In our paper we argue that this approach is based on an understanding of KDD that is no longer up-to-date. In fact, today KDD is seen as a human centered, highly interactive and iterative process. Blindly enforcing constraints already during the mining runs neglects the process character of KDD and therefore is no longer state of the art. Constraints can make a single algorithm run faster but in fact we are still far from response times that would allow true interactivity in KDD. In addition we pay the price of repeated mining runs and moreover risk reducing data mining to some kind of hypothesis testing. Taking all the above into consideration we propose to do exactly the contrary of constrained mining: We accept an initial (nearly) unconstrained and costly mining run. But instead of a sequence of subsequent and still expensive constrained mining runs we answer all further mining queries from this initial result set. Whereas this is straight forward for constraints that can be implemented as filters on the result set, things get more complicated when we restrict the underlying mining data. Actually in practice such constraints are very important, e.g. the generation of rules for certain days of the week, for families, singles, male or female customers etc. We show how to postpone such row-restriction constraints on the transactions from rule generation to rule retrieval from the initial result set.

Keywords

Association Rules, KDD Process, Constrained Mining

1. INTRODUCTION

Since its introduction in 1993 [1] association rule mining has become one of the fundamental analysis methods in knowledge discovery in databases (KDD). In the following section we briefly introduce the basic idea behind this popular mining approach. Moreover we point out the implications of understanding KDD as a nontrivial and interactive process in the context of association rule mining. This finally leads to a better understanding of constraints in association rule mining.

1.1 Association Rules

Association rules model dependencies between items in transactional data. Let $\mathcal{I} = \{x_1, \dots, x_n\}$ be a set of distinct literals, called items. A set $X \subseteq \mathcal{I}$ with $k = |X|$ is called a k-itemset or simply an itemset. Let a database $\mathcal{D}$ be a multi-set of subsets of $\mathcal{I}$. Each $T \in \mathcal{D}$ is called a transaction.

A transaction $T \in \mathcal{D}$ supports an itemset $X \subseteq \mathcal{I}$ if $X \subseteq T$. Let $X, Y \subseteq \mathcal{I}$ be nonempty itemsets with $X \cap Y = \emptyset$. Then an association rule is an expression

$$X \to Y,$$

with rule body X, rule head Y, and rule confidence

$$\mathsf{conf}(X \to Y) = \frac{|\{T \in \mathcal{D} \mid X \cup Y \subseteq T\}|}{|\{T \in \mathcal{D} \mid X \subseteq T\}|}.$$

The confidence can be understood as the conditional probability $P(Y|X)$. The fraction of transactions T supporting an itemset X with respect to database $\mathcal{D}$ is called the support of X,

$$\mathsf{supp}(X) = \frac{|\{T \in \mathcal{D} \mid X \subseteq T\}|}{|\mathcal{D}|}.$$

An itemset reaching a predefined threshold for support is called a frequent itemset. The support of a rule $X \to Y$ is defined as

$$\mathsf{supp}(X \to Y) = \mathsf{supp}(X \cup Y).$$

In practice the support-confidence framework, as described above, shows severe limitations. The reason is that association rules are based on correlations and even for large confidence values do not necessarily imply causation. As a consequence supplementary rule quality measures have been developed over the years, e.g. lift (interest) [6; 15]:

$$\mathsf{lift}(X \to Y) = \frac{\mathsf{conf}(X \to Y)}{\mathsf{supp}(Y)}$$

The measure lift expresses the deviation of the rule confidence from the a priori probability of Y, $\mathsf{supp}(Y)$. That is, in how far does the rule body X "lift" the probability for the rule head Y to occur in the same transaction. Alternatively conviction [6] expresses in how far X and $\neg Y$ are stochastically independent:

$$\mathsf{conv}(X \to Y) = \frac{\mathsf{supp}(X) \cdot \mathsf{supp}(\neg Y)}{\mathsf{supp}(X, \neg Y)}$$

High values for $\mathsf{conv}(X \rightarrow Y)$, up to ∞ for $\mathsf{supp}(X, \neg Y) = 0$, express the conviction that $X \rightarrow Y$ represents a causation.
In order to restrict the otherwise enormous result set today's mining algorithms expect the analyst to give minimal thresholds on rule quality measures, e.g. [2; 11; 13].
Association rules origin from basket analysis but easily are transfered to a broad variety of applications from many different domains.

1.2 The KDD Process

One of the most important and influential insights in KDD research is the understanding of KDD as a process, as e.g. Fayyad et al. formulate in [8]:

> KDD is the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.

There are several different more or less concrete process models, e.g. [5; 8; 20], but the key message is always the same: data mining, that is applying a sophisticated mining algorithm to a dataset, is just one out of several steps in a KDD project. Moreover typically the KDD process steps are not passed in any prescribed order. The analyst decides "on the fly" wether to proceed to the next step, to redo the current step or even to return to one of the previous tasks. In Figure 1 the most important dependencies between the common mining steps are indicated by arrows. The cycle around

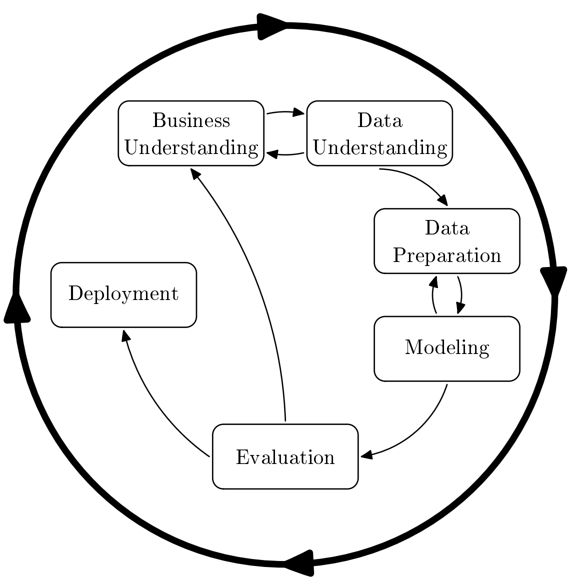

Figure 1: The steps (phases) of the CRISP-Data Mining process.

the process indicates its overall cyclic character. Obviously such a human centered and interactive process highly depends on the analyst's skills and creativity. So adequately supporting the analyst is one of the keys to successful application of KDD techniques.
Unfortunately the response times of today's highly optimized association mining algorithms still range from minutes to hours, depending on the dataset and the employed minimal thresholds for the rule quality measures, e.g. [13]. A rerun of the mining algorithm is often required for investigating even speculative ideas. Yet if every simple and speculative idea implies to be idle for a few minutes, then analysts will – at least in the long run – brake themselves in advance instead of trying out diligently whatever pops into their minds.
Generally speaking, we cannot expect to reach a smooth integration of the mining algorithms into the interactive KDD process. In other words, in practice often creativity and inspiration of the analysts are smothered by the annoying inefficiencies of the underlying algorithms.

1.3 Constraints in Association Rule Mining

Constraints aim at reducing the mining result. In its simplest form constraints are filtering thresholds on the rule quality measures support and confidence. More elaborated constraints formulate conditions on the presence or absence of certain items in head and/or body. In general constraints help preventing the analysts from drowning in large result sets. By specifying constraints on the result set an analyst can deliberately focus his search on what he is really interested in. In the context of result sets easily reaching thousands or ten thousands of association rules, constraints are a powerful and hardly replaceable means in the analyst's hands.
Furthermore it became clear that pushing constraints deeply into the actual mining run is a way to tackle the performance problem of association rule mining algorithms. If the analyst is able to specify what he is interested in then run times can be drastically reduced. At first glance exploiting constraints during the mining run seems to be a promising approach in order to get closer to the ideal of a truly interactive KDD process. As a consequence a variety of algorithms have been developed by research that employ constraints for performance improvement, e.g. [18; 19]. Although obviously there is a significant effect on performance when pushing constraints deeply into the mining procedure, we propose to do exactly the contrary [14], for reasons we will see shortly.

2. POSTPONING CONSTRAINTS FROM MINING TO EVALUATION

In this section we explain why we generally favor postponing constraints instead of pushing them into the mining procedure. After this motivation we come to the main contribution of our paper: we show how to do this for the practically very important case of constraining the rows of the mining data, a case where postponing constraints implies much more than simply filtering an existing result set.

2.1 Why Postponing Constraints?

KDD is more than examining some concrete hypothesis. Especially association rules as an unsupervised mining method aim at finding novel knowledge from the data. Pushing constraints on the result already from the very beginning means narrowing the result set and therefore is somehow contrary to this goal. So on the one hand constraints can help cutting an otherwise overwhelming search space to manageable pieces. But on the other hand we should always have in mind that constraints also involve the danger of reducing data mining to pure hypothesis investigation.
With this background it is clear that especially during the initial iterations of a KDD process constraints should always

be used with caution. In the beginning of a KDD project typically there will be a general search phase before the analyst begins to focus on specifics. So we mainly benefit from the performance improvements of constraints in the second half but cannot count on them during the initial orientation phases. In addition even when pushing rather strict constraints into the mining process we only get performance improvements but especially on really large databases we are still far from gaining true interactivity.

So why not accepting one initial mining run without applying any constraints in order to mine "all potentially interesting rules"? [14] Of course this initial run will take its time but as this is expected it should not be a severe problem, e.g. we might run it over night. Presuming that the generated rule set is broad enough, running the mining algorithm is a costly but unique event. The result is a rule set that will serve for both, the initial orientation phases and the phases where the analyst focuses on specifics.

In the latter phases instead of pushing them into the mining we simply apply the constraints to the result set. That is to say, we postpone the constraints from mining to rule set evaluation. Applying the constraints then typically means filtering the generated rules.

The benefit of the initial costly mining run is true interactivity throughout the whole subsequent KDD process. In [14] we suggest storing the result set in an appropriate rule cache. In all following mining iterations this cache is efficiently accessed through an evolved mining language, e.g. [10; 14; 16; 17]. Especially for large data sets selecting ("filtering") appropriate rules from this cache is always unbeatably much faster than generating a constraint rule set from the mining data itself.

2.2 Postpone Row-Restriction Constraints

Row-restriction constraints restrict the mining data to certain subsets. In practice such constraints are among the most important constraints in association rule mining. Typical applications from basket analysis are the generation of rules for certain days of the week, e.g. rules for only those customer transactions collected on Saturdays. Or the restriction of the transactions is based on customer information, e.g. rules for families, singles, male or female customers etc. Examples from other domains like manufacturing are production year, model type etc.

Pushing such a row-restriction constraint into the mining run is trivial: we simply run the mining algorithm on the selected subset. Unfortunately postponing such a constraint to rule retrieval from an existing rule set is not straight forward. It no longer suffices to simply filter the rules but we need to adapt the values of the rule quality measures adequately. In the following we show how to do this for the most common rule quality measures.

2.2.1 *Basic Idea*

We presume that all information employed for restricting the mining data were already stored as items in the transactions when the initial mining run took place. Such items can be seen as pseudo items in the sense that they are not "contained" in a transaction but describe the transaction as a whole. For instance the family status of a customer or information on gender can be added as an item to each transaction. The same holds for day of the week etc.

Let the database $\mathcal{D}'$ be a subset of $\mathcal{D}$ with $\mathcal{D}'$ being restricted to only those transactions from $\mathcal{D}$ that contain a certain itemset R. Then the support of an itemset A in the database $\mathcal{D}'$ can be derived from the support values in $\mathcal{D}$ as follows:

$$\mathsf{supp}_{\mathcal{D}'}(A) = \mathsf{supp}_{\mathcal{D}}(A \cup R) \cdot \frac{|\mathcal{D}|}{|\mathcal{D}'|}$$

The transactions in $\mathcal{D}'$ that contain A are those transactions from $\mathcal{D}$ that contain A and R. In other words, the support of A in the constrained database $\mathcal{D}'$ is the support of A extended with R in the unconstrained database $\mathcal{D}$, multiplied by factor $|\mathcal{D}|/|\mathcal{D}|'$. So if we want to derive the support of A in $\mathcal{D}'$ we depend on knowing the support of $A \cup R$ in $\mathcal{D}$. This implies that $A \cup R$ must be frequent in $\mathcal{D}$ in order to derive $\mathsf{supp}_{\mathcal{D}'}(A)$ (in association rule mining only the support values of frequent itemsets are known).

At first glance this might look like a severe restriction. Fortunately it turns out to be only a minor problem in practical applications. It simply implies that the subset $\mathcal{D}'$ to which we restrict $\mathcal{D}$ must be a reasonable portion of $\mathcal{D}$. For typical row-restriction constraints, e.g. gender, family status or day of the week, normally this is always the case. For instance presuming approximately the same number of customer transactions each day, lowering the minimal support for $\mathcal{D}$ by factor $< 1/7$ is a practical guess for constraining the result set to different days of the week.

Another important point is that we store rules not frequent itemsets. To derive the support of an itemset A in $\mathcal{D}'$ there must be at least one corresponding rule being generated from $\mathcal{D}$ that reaches all minimal thresholds for the rule quality measures. Whereas we saw that frequency is only a minor problem we suggest to restrict the rule set only by moderate values for minimal confidence, e.g. 75% proved to be a good choice in our experiments. Furthermore during the initial mining run we recommend no minimal thresholds at all for the supplementary rule quality measures.

2.2.2 *Common Rule Quality Measures*

Based on the support it is straight forward to postpone row-restriction constraints from mining to rule retrieval from a cache. In the following we show how to do this for the most relevant rule quality measures (Again $\mathcal{D}'$ is the transaction set obtained when constraining $\mathcal{D}$ by itemset R):

Support

$$\begin{aligned}\mathsf{supp}_{\mathcal{D}'}(A \to B) &= \mathsf{supp}_{\mathcal{D}'}(A \cup B)\\ &= \mathsf{supp}_{\mathcal{D}}(A \cup B \cup R) \cdot \frac{|\mathcal{D}|}{|\mathcal{D}'|}\end{aligned}$$

Confidence

$$\begin{aligned}\mathsf{conf}_{\mathcal{D}'}(A \to B) &= \frac{\mathsf{supp}_{\mathcal{D}'}(A \cup B)}{\mathsf{supp}_{\mathcal{D}'}(A)}\\ &= \frac{\mathsf{supp}_{\mathcal{D}}(A \cup B \cup R)}{\mathsf{supp}_{\mathcal{D}}(A \cup R)}\end{aligned}$$

Lift

$$\begin{aligned}\mathsf{lift}_{\mathcal{D}'}(A \to B) &= \frac{\mathsf{supp}_{\mathcal{D}'}(A \cup B)}{\mathsf{supp}_{\mathcal{D}'}(A) \cdot \mathsf{supp}_{\mathcal{D}'}(B)}\\ &= \frac{\mathsf{supp}_{\mathcal{D}}(A \cup B \cup R)}{\mathsf{supp}_{\mathcal{D}}(A \cup R) \cdot \mathsf{supp}_{\mathcal{D}}(B \cup R)} \cdot \frac{|\mathcal{D}'|}{|\mathcal{D}|}\end{aligned}$$

Conviction

$$
\begin{aligned}
\mathsf{conv}_{\mathcal{D}'}(A \rightarrow B) &= \frac{\mathsf{supp}_{\mathcal{D}'}(A) \cdot \mathsf{supp}_{\mathcal{D}'}(\neg B)}{\mathsf{supp}_{\mathcal{D}'}(A, \neg B)} \\
&= \frac{\mathsf{supp}_{\mathcal{D}'}(A) \cdot (1 - \mathsf{supp}_{\mathcal{D}'}(B))}{\mathsf{supp}_{\mathcal{D}'}(A) - \mathsf{supp}_{\mathcal{D}'}(A \cup B)} \\
&= \frac{\mathsf{supp}_{\mathcal{D}}(A \cup R) \cdot (1 - \mathsf{supp}_{\mathcal{D}}(B \cup R) \cdot \frac{|\mathcal{D}'|}{|\mathcal{D}|})}{\mathsf{supp}_{\mathcal{D}}(A \cup R) - \mathsf{supp}_{\mathcal{D}}(A \cup B \cup R)}
\end{aligned}
$$

3. EVALUATION

In the following section we evaluate an implementation of our approach on a mining scenario from supermarket basket analysis.

3.1 Algorithm and Dataset

For our evaluation we employed a C++-implementation of the popular Apriori-algorithm [2]. The experiments were run on a Pentium III Linux machine clocked at 500 Mhz.
The data is from a supermarket and consists of 77,588 customer transactions that were collected over a period of one month. The transactions contain items out of an assortment of 19,535 different articles plus the day of the week as pseudo item. The data resided in IBM's DB2 relational database system. Data access was implemented as described in [12].

3.2 Scenario

For our evaluation we chose a typical scenario. An analyst investigates the above customer data in a KDD process. He starts with a general exploration phase and his early "blurred" insights serve as starting points for further investigation. In our supermarket scenario the day of the week stored with each transaction obviously is worth a deliberate analysis. The question is, in how far the behavior of customers on the different days of the week differs. For instance people make supply purchases before the weekend but typically not in the beginning of the week. In addition different people – families, singles, business people, housewives etc – may favor different days of the week for their purchases.
Normally an analyst will start with mining the whole dataset. Then, on demand, he will select subsets of the data for each day of the week and rerun the mining algorithm on these datasets. Depending on the size of the data each of these reruns will interrupt his work for several minutes upto hours.
By postponing the row-restriction constraints as described in this paper the analyst gets around the repeated mining runs. After an initial mining run (at lowered minimal thresholds for the rule quality measures) restricting the rows of the datasets may take less than a second.

3.3 Experiments and Results

In our experiments we made two mining runs over the complete database. One at minimum support of 0.1%, the other at 0.7% (minimum confidence was always set to 75%). To the rule set generated at 0.1% we apply our postponed row-restriction constraints. That is to say, without rerunning the mining algorithm we get those rules with adjusted quality measures that would have been generated if we had restricted the datasets in advance. We do this for each day of the week (except Sunday because our shop was closed on Sundays).
In order to compare the results we selected the appropriate data subsets for each day of the week and made separate mining runs at minimal support of $0.7\% = (6+1) \times 0.1\%$ on them. We divided the minimal support by seven (six days of the week plus one as safety interval) to compensate the chance of missing rules when postponing constraints compared to true mining on the restricted subsets, c.f. Section 2. Finally for each day of the week the rule set generated by postponing constraints was equal to or even slightly larger than the rule set generated by a separate mining run, at minimum support 0.7% of course. So mining with postponed row-restriction constraints did not miss any rules. The number of rules generated for each day differed between approximately 50 and 90 rules.
The benefit of our approach becomes clear when looking at the performance result in Figure 2. Of course postponed

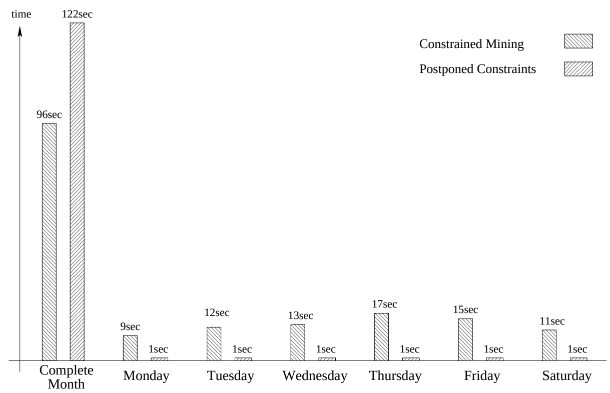

Figure 2: Performance results on supermarket transactions (including the time for database access).

constraints require an initial mining run where none of the constraints is applied (in our experiment the run at the left that takes 122sec on the complete month). Whereas rerunning the algorithm for each of the days always implies an interruption of the analyst, in our experiments the time for applying row constraints to an existing result set is about a second and can be neglected for each of the restrictions[1]. This performance gap becomes even larger when mining larger datasets. In brief, mining the data of a complete year will take about 12 times longer than this single month. But we experienced that presuming the number of items remains the same, the number of rules also stays nearly constant. Consequently pushing row-restriction constraints into the mining run – selecting subsets of the mining data – scales linearly with the number of rows. In contrast typically the time for restricting a rule set as described in this paper can be neglected and more important is practically independent of the actual size of the database.

4. DISCUSSION

In this paper we argue in favor of an approach that to some extend stands in contradiction to today's common opinion. Therefore we think it is worth to spent some more time on discussing the pros and cons of our approach in order to face the most important counter-arguments.

[1]In contrast to [14] we store the rules directly in main memory instead of a relational database system.

4.1 Does It Really Make Sense to Drastically Lower Minimum Support?

Typically the support of a rule is understood as one of the rule quality measures. So at a certain level further lowering the minimum support threshold should not make sense anymore because the quality (significance) of such low support rules is not acceptable. In some domains such rarely applicable rules may not be actionable. For example it probably does not make sense to decide upon special advertisements based on items bought only by a very small fraction of supermarket customers. In contrast in the medical domain the death or severe illness of a patient may also be quite rare but obviously is nevertheless of interest, c.f. [7].

In general we think support as a rule quality measure is commonly overestimated. In brief, we see support as an unavoidable means to make the complexity of the association mining problem manageable for current algorithms. Even rules at very low support may be interesting. For example item a and item b may occur rather infrequently in the data. So the support of rule $a \rightarrow b$ is also quite low. Nevertheless if such a rule with reasonable confidence exists this rule will be of interest because the implied co-occurrence in the same transactions of two such infrequent items is probably not by accident. Further examples can be found in [7].

4.2 Is The Initial Mining Run Feasible?

Obviously the initial mining is the critical part of our argumentation. On the one hand the benefit of the single unconstrained mining run cannot be stressed enough: we gain interactivity for the following phases of the KDD process, as we think an obligatory prerequisite for efficiently supporting the 'human in the loop'. On the other hand we must admit that the price to pay for the unconstrained mining seems to be quite high, easily reaching hours or even days of computation. The question arising is whether our approach is still appealing in practice or not. From our point of view it undoubtedly is. Running the mining algorithm e.g. over night is nearly always an option. During rule generation the machine does not rely on any human interaction. So letting the machine stupidly counting frequencies in the data actually does not bind any human resources. That is to say, the price to pay is not as high as it seems at first sight. Moreover when considering the 'real problems' occurring during a mining project such a break quickly looses its importance. This opinion is also supported by other researchers. For example DuMouchel and Pregibon in [7] say they deliberately did not focus on algorithmic issues in their research and consider the computing time negligible compared to the analysis time. In the same direction Goethals and Van den Bussche argument in [9] when introducing their interactive mining approach.

4.3 Can Constraints Always Be Avoided?

Of course in our scenario we presume that the mining algorithm still terminates in reasonable time, e.g. overnight. But runtime of the algorithms is exponential in minimum support and moreover highly depends on the characteristics of the underlying data, e.g. [13]. Whereas we did not experience severe problems in classical association mining domains like market basket analysis – even at quite low thresholds for minimum support – things are different for so called dense databases, c.f [3; 4]. Here the number of frequent itemsets may be infeasible high already at comparably high levels of minimum support. So situations may occur where even quite strict constraints become essential.

Such a strict constraint is the exact specification of the rule head before running the algorithm as proposed in [3]. At first glance this restriction looks quite drastic but based on it Bayardo and Agrawal are able to broaden the class of rules identified by their algorithm [3]: the precise interestingness measures can be adjusted during post-processing instead of a priori. So to some extend they anticipate our postponed constraints approach.

The general idea for such dense or otherwise problematic databases is to push as few constraints as possible into the mining. We want the algorithm to return in reasonable time but we do not want to carelessly restrict the initial rule set that is the basis of our further explorations.

A problem in this context is that currently there is no possibility to estimate the runtime of the algorithms in advance. So how to decide upon the constraints to enable reasonable run times? At the moment we do not have an answer to this question. What we actually suggest is quite pragmatic but helpful in practice: we start several mining runs on different machines with different constraints and thresholds on the rule quality measures. The idea is that at least some of these mining runs return in reasonable time. Then out of the terminated runs the one with the weakest constraints serves as basis for our explorations.

5. SUMMARY

In this paper we introduced the reader to association rule mining with postponed row-restriction constraints. We motivated our approach by discussing the basics of association rule mining and constraints in the context of an interactive KDD process. The common approach to employ constraints is to push them deeply into the mining run. This can make algorithm runs faster but in fact we are still far from a truly interactive KDD process.

The idea of postponing constraints is to do exactly the contrary. A single and possibly expensive mining run is accepted but all subsequent mining questions are supposed to be satisfied from the initial result set. Whereas this is straight forward as long as postponing constraints means simply filtering the initial rule set, things get problematic as soon as constraints operate as select statements on the underlying mining data.

Such row-restriction constraints restrict the mining data to well defined subsets. Typical applications from basket analysis are the generation of rules for certain days of the week, e.g. generate rules for only those customer transactions collected on Saturdays. Or the restriction of the transactions based on customer information, e.g. rules for families, singles, male or female customers etc. Examples from other domains like manufacturing are production year, model type etc. In practice such constraints are among the most important constraints in association rule mining.

One of the topics of our paper was to show how to postpone such row-restriction constraints on the transactions from rule generation to rule retrieval from the initial result set. In fact, we showed that without actually rerunning the algorithm, we are able to efficiently construct those rules from the initial result set that would have been generated if the mining algorithm were run on only a subset of the transactions. In our experiments we demonstrated that the time needed for applying postponed row-restriction constraints to rule sets – at least in our scenario – is about a second and can be neglected. Furthermore the response time is even independent from the number of rows of the underlying database. The latter is an important advantage compared to rerunning constrained mining algorithms that still scale linearly with the number of rows.

Summing up, we introduced the reader to approaching true interactivity in KDD by postponing row-restriction constraints from the mining run to rule set evaluation. We discussed the pros and cons and finally admitted that pushing constraints into the algorithms is not always avoidable. But if constraints have to be used they bear the danger of reducing data mining to hypothesis testing. Consequently as few as possible constraints should be exploited already during the mining run.

So is pushing constraints deeply into the mining algorithms really what we want? No, for sure not, but unfortunately sometimes when its algorithmically impossible to completely postpone all constraints we cannot life without constraint-based mining.

6. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD '93)*, pages 207–216, Washington, USA, May 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large Databases (VLDB '94)*, Santiago, Chile, June 1994.

[3] R. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining (KDD '99)*, pages 145–154, San Diego, California, USA, August 1999.

[4] R. J. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the 15th International Conference on Data Engineering*, Sydney, Australia, March 1999.

[5] R. J. Brachman and T. Anand. The process of knowledge discovery in databases: A human centered approach. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, chapter 2, pages 37–57. AAAI/MIT Press, 1996.

[6] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (ACM SIGMOD '97)*, pages 265–276, 1997.

[7] W. DuMouchel and D. Pregibon. Empirical bayes screening for multi-item associations. In *Proceedings of The Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2001)*, pages 67–76, San Francisco, CA, USA, August 26-29 2001.

[8] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

[9] B. Goethals and J. V. den Bussche. A priori versus a posteriori filtering of association rules. In *Proceedings of the 1999 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD '99*, Philadelphia, USA, May 30 1999.

[10] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases. In *Proceedings of the 1996 SIGMOD'96 Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD '96)*, Montreal, Canada, June 1996.

[11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM-SIGMOD International Conference on Management of Data*, Dallas, Texas, USA, May 2000.

[12] J. Hipp, U. Güntzer, and U. Grimmer. Integrating association rule mining algorithms with relational database systems. In *Proceedings of the 3rd International Conference on Enterprise Information Systems (ICEIS 2001)*, pages 130–137, Setúbal, Portugal, July 7-10 2001.

[13] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[14] J. Hipp, C. Mangold, U. Güntzer, and G. Nakhaeizadeh. Efficient rule retrieval and postponed restrict operations for association rule mining. In *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)*, pages 52–65, Taipei, Taiwan, May 6-8 2002.

[15] IBM. *Intelligent Miner Handbook*, 1999.

[16] T. Imielinski, A. Virmani, and A. Abdulghani. DMajor - application programming interface for database mining. *Data Mining and Knowledge Discovery*, 3(4):347–372, December 1999.

[17] R. Meo, G. Psaila, and S. Ceri. A new sql-like operator for mining association rules. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB '96)*, Mumbai (Bombay), India, September 1996.

[18] R. Ng, L. S. Lakshmanan, J. Han, and T. Mah. Exploratory mining via constrained frequent set queries. In *Proceedings of the 1999 ACM-SIGMOD International Conference on Management of Data (SIGMOD '99)*, pages 556–558, Philadelphia, PA, USA, June 1999.

[19] R. Srikant, Q. Vu, and R. Agrawal. Mining association rules with item constraints. In *Proceedings of the 3rd International Conference on KDD and Data Mining (KDD '97)*, Newport Beach, California, August 1997.

[20] R. Wirth and J. Hipp. CRISP-DM: Towards a standard process modell for data mining. In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39, Manchester, UK, April 2000.