Randomized Quantitative Association Rules

Bart Goethals[1,2], Emmanuel Müller[1,2], and Thomas Van Brussel[1]

[1] University of Antwerp, Belgium
[2] Monash University, Australia
[3] Hasso-Plattner-Institute, Germany

Abstract. Traditional association rules are commonly used to find relationships in binary or categorical data and have been extensively studied in the literature. Quantitative association rules are a natural extension, which allow for detection of interpretable relationships in numeric data. However, quantitative association rule mining suffers from tremendously large result sets and inefficient computation just like traditional techniques. Here, we propose a randomized technique for quantitative association rules. We statistically select unexpected rules, which provide sufficient new information about the underlying data relationships. Furthermore, we exploit unexpectedness for the efficient steering of our randomized search and as termination criterion of our algorithm. Overall, our algorithm can quickly generate a small number of high quality rules that are easily interpretable by the user.

1 Introduction

Association rule mining has been a popular method for discovering interesting patterns in databases, ever since its introduction [1]. Originally, these methods were used to find patterns in boolean or categorical data, i.e. rules of the form Buy[Toothbrush] $\rightarrow$ Buy[Toothpaste] or Healthy[Yes] $\rightarrow$ Sports[Yes]. These relations are easily interpreted and usually require no further explanation. Over time, techniques were introduced that found these patterns more optimally, or that focused on more complex types of patterns [4, 8].

Since more and more data is being collected and large quantities of this data are of the numerical kind, the previous rules are no longer adequate to represent all patterns in the data. To this end, quantitative association rule mining was introduced [2, 17, 21]. In their purest form, quantitative association rules handle numeric data but can, in theory, easily be combined with regular association rules in order to handle all kinds of data. Two examples of such rules are the following: Age$[16, 90] \wedge$ US-citizen[Yes] $\rightarrow$ Drivers-License[Yes] and Age$[0, 21] \wedge$ US-citizen[Yes] $\rightarrow$ Alcohol-Allowed[No].

Unfortunately, current techniques detect these rules together with very many other rules in a tremendously large result set. This so-called *pattern explosion* phenomenon can obfuscate these rules [20]. It is well-explored in traditional association rule mining, but has raised rather little attention for numeric predicates in association rules. Another severe issue are the parameters of these techniques.

They have to be chosen very carefully, as they affect both result size and the runtime of current algorithms. Detection of a small set of relevant rules is a very hard task with the state-of-the-art algorithms. Either they compute few rules in short time and miss interesting rules, or they run out of time with huge result sets that obfuscate the interesting rules by many redundant rules [17, 7, 11, 3, 10, 5, 9, 16, 19].

In the present paper, we focus on discovering a compact set of quantitative association rules while avoiding the pitfall of trying to find all rules. We propose a randomized technique that is capable of iteratively discovering rules in a Monte Carlo approach. We statistically select unexpected rules, which provide sufficient new information about the underlying data relationships. Thus, we avoid the redundant rules and keep the result set compact. Furthermore, we exploit unexpectedness for the efficient steering of our randomized search and as termination criterion of our algorithm. Thus, we even avoid the generation of redundant rules within the Monte Carlo approach. New rules are generated in vastly unexplored regions of the numeric attribute space. Using this steered processing, our algorithm quickly terminates as soon as no further rules can be expected. Overall, our algorithm can quickly generate a small number of high quality rules that are easily interpretable by the user.

2 Formal Problem Statement

For a database $\mathcal{DB}$ of numeric attributes $\mathcal{A}$, we consider each object $o \in \mathcal{DB}$ to be represented by a vector $\boldsymbol{o} \in \mathbb{R}^m$. We denote $o(A_i) \in dom(A_i)$ as the value of object o within the attribute domain of A_i. We use the notation $n = |\mathcal{DB}|$ for the size of the database and $m = |\mathcal{A}|$ as the dimensionality of the database. Let $\mathcal{A} = \{A_1, \ldots, A_m\}$ be the set of all attributes in the database. A *quantitative predicate* is then defined as follows.

Definition 1. *Given one attribute $A_i \in \mathcal{A}$ and a pair of lower and upper bounds $[l, u] \in \mathrm{dom}^2(A_i)$, with $l \leq u$. We define a Quantitative Predicate as: $A_i[l, u]$*

Please note, that this definition considers numeric attributes only. Although we are not considering other attribute types in this paper our algorithm is not restricted to numeric attributes. It can be easily extended to allow for binary and categorical attribute types as well.

For our basic notions, we consider an object $o \in \mathcal{DB}$ to be covered by a quantitative predicate $A_i[l, u]$ *iff* $l \leq o(A_i) \leq u$. That is, the value of object o for attribute A_i is contained within the respective interval. Based on Definition 1, a *set of predicates* is then defined as a conjunction of predicates. Consequently, an object is covered by a set of predicates *iff* that object is covered by each of the predicates in that set. For such a predicate set $\mathcal{X}$, we denote its attributes by $attr(\mathcal{X}) \subseteq \mathcal{A}$. Furthermore, we denote all objects that are covered by predicate set $\mathcal{X}$ as $I_{\mathcal{DB}}(\mathcal{X})$, with $I_{\mathcal{DB}}(\cdot)$ the function that returns all objects $o \in \mathcal{DB}$ that are covered by all predicates given in $\mathcal{X}$. That is, $I_{\mathcal{DB}}(\mathcal{X}) = \{o \in \mathcal{DB} \mid o \text{ is covered by } \mathcal{X}\}$. We write $I_{\mathcal{DB}}(\cdot)$ as $I(\cdot)$ whenever the database we are working on is clear from context.

We define the *support* and frequency of $\mathcal{X}$ as $\text{supp}_{\mathcal{DB}}(\mathcal{X}) = |I_{\mathcal{DB}}(\mathcal{X})|$, and $\text{freq}_{\mathcal{DB}}(\mathcal{X}) = |I_{\mathcal{DB}}(\mathcal{X})|/|\mathcal{DB}|$. Again, we omit $\mathcal{DB}$ whenever it is clear from context.

Definition 2. *A Quantitative Association Rule (QAR) R is defined as $R : \mathcal{P} \rightarrow \mathcal{Q}$ with $\mathcal{P}, \mathcal{Q} \neq \emptyset$, and $\mathcal{P} \cap \mathcal{Q} = \emptyset$.*

Following Definition 2, we refer to $\mathcal{P}$ as the *left hand side (LHS)* of the rule and $\mathcal{Q}$ the *right hand side (RHS)* of rule R. The support of a rule R is defined as $\text{supp}(R) = \text{supp}(\mathcal{P} \cup \mathcal{Q})$. That is, the number of objects that satisfy both the predicates in the left and right hand side of the rule. These objects are also said to be covered by rule R. The confidence of rule R is defined as $\text{conf}(R) = \text{supp}(R)/\text{sup}(\mathcal{P})$. That is, the fraction of objects that are covered by rule R given that they are covered by predicate set P. Please note, that we decided to use these two traditional measures while there are many other interestingness measures possible [18]. Given any of these measures, one can detect all interesting rules $\mathcal{ALL}$. However, this set is known to be highly redundant.

Given the set of all interesting rules $\mathcal{ALL}$, we select only those rules that are unexpected and provide sufficient new information about the underlying data relationships. We define our result set as subset $\mathcal{RS} \subseteq \mathcal{ALL}$:

Definition 3. *A set of quantitative association rule $\mathcal{RS}$ is* non-redundant *if*

$$\forall R_i \in \mathcal{RS} : \text{unexpected}(R_i, \mathcal{RS} \setminus R_i)$$

Each rule R_i has to be unexpected w.r.t. all other given rules $\mathcal{RS} \setminus R_i$. In our case we define the function *unexpected*() by a significance test (cf. Section 3.4). However, in general it could be any function that checks the quality of a rule R_i w.r.t. all other given rules. With this check we try to prevent users from a tremendously large result set and exclude redundant rules as in the following example.

Example 1.

$$R_1 : \text{Age}[30, 42] \wedge \text{Income}[27000, 32000] \rightarrow \text{Children}[2, 3].$$
$$R_2 : \text{Age}[30, 40] \wedge \text{Income}[27000, 32000] \rightarrow \text{Children}[2, 3].$$
$$R_3 : \text{Age}[32, 39] \wedge \text{Income}[27000, 32000] \rightarrow \text{Children}[2, 3].$$
$$R_4 : \text{Age}[31, 40] \wedge \text{Income}[27000, 32000] \rightarrow \text{Children}[2, 3].$$

all of which fulfill support = 10% and confidence = 70%.

These four rules form our hypothetical set of all interesting rules $\mathcal{ALL}$. However, we observe that all rules have very similar coverage of objects. In theory, one could now detect all of these rules and perform an optimal selection of a subset $\mathcal{RS}$. In our case, we do not aim for such an optimal solution due to the tremendous size of $\mathcal{ALL}$ in practice and the computational effort to detect this set. In contrast to this optimal solution, we aim at an efficient greedy heuristic that selects iteratively the best possible rule at a time.

In the following we focus on the main challenges for such a greedy selection of $\mathcal{RS}$: i.e. the Efficient (randomized) generation of R_i candidates, the selection criterion *unexpected*$(R_i, \mathcal{RS} \setminus R_i)$, and the termination criterion for the iterative selection of new candidates.

3 Algorithm

We first describe the general outline of the algorithm before zooming in on the more important aspects of it.

(i) During each step of the algorithm, a weight distribution over all transactions is kept. At first, this distribution is uniform but can, and will, change after each iteration. The algorithm creates rules by starting from random seed points. Complete randomness has the disadvantage of potentially recomputing overlapping rules that provide little to no extra information or that regions are explored where no useful information can be found. However, selecting seeds in a Monte Carlo manner has the advantage of being faster than deterministically computing all rules.

(ii) After such a point has been chosen, the algorithm continues by performing a transformation of the original database into a transactional database [22]. This transformation is done using a parameter w, which denotes a window width. For each point and for each dimension, we check whether a different point is within distance w of the seed point in that dimension. If it is, we update the transaction for that point in our new database to list that dimension. In the end, each point can be resolved to a list of dimensions, an *itemset* in which it is close to the seed point.

(iii) Using this transactional database, we mine it for interesting frequent itemsets, as these tell us something about which dimensions are relevant. Since we want to avoid pattern explosion, we only need a small subset of frequent itemsets. This can be done using techniques called random (maximal) itemset mining. These techniques compute only a small number of good patterns. Each of the itemsets generated this way can be related back to a hyper-rectangle in the original data.

(iv) In the final step, we evaluate the significance and quality of the hyper-rectangles generated in the previous step. Conceptually, this is done by comparing the amount of points within the hyper-rectangle to the amount of points we would expect to be within it following a certain distribution — usually the uniform distribution. A significant region would contain significantly more points than expected.

(v) Update the weights of the points using a mixture model. That is, points that are not yet included in a rule or are less frequently included in a rule than others will gain a higher weight than points that are frequently visited. This allows the algorithm to go for the patterns that would otherwise not be discovered.

The algorithm terminates when either no more rules can be discovered or enough rules have been discovered.

The algorithm is summarized in Algorithm 1. For further clarification of the steps, we refer to their corresponding subsections.

Algorithm 1 Rule mining(α, support, window width w, *max rules*)

```
Initialize array of weights W to 1/|DB|
while # rules found ≤ max rules do
    Select random point p ∈ DB according to W
    TDB = {}
    for point o ∈ DB, o ≠ p do
        t = {}
        for attribute A_i ∈ A do
            if |o(A_i) − p(A_i)| ≤ w then
                t = t ∪ A_i
        TDB = TDB ∪ t
    Find random maximal itemset(s) in TDB
    if points in hyper-rectangle R defined by transaction ≥ θ then
        Add R to mixture model
        Recompute weights
Optional: Generate rules based on hyper-rectangles in mixture model
```

3.1 Weighting and Selecting Seed Points

As mentioned above, the main focus of our algorithm is not on finding all quantitative association rules in the data. We are, however, still interested in a subset of them. To this end, we propose the use of a method similar to the one introduced by the DOC algorithm [15]. That is, starting from a random seed point, determine an interesting region around it and use that region to generate rules.

While this approach shows some promising results, it has one major downside. That is, each time the algorithm finishes handling one point, the next point is selected without taking the knowledge accumulated from handling the previous points into account. Selecting the wrong seeds will then lead to recomputing (partially) overlapping regions that provide little to no extra information. We therefore focus on a weighting approach for the transactions.

The algorithm starts by initialising the weights of all transactions to a single uniform weight. Let $o_i \in \mathcal{DB}$ be the i-th record in the database, then $W(o_i)$ denotes the weight of that record. During the initialisation step, it holds that $\forall i, j, W(o_i) = W(o_j)$. The choice for a uniform distribution is obvious as it simply means that no information has been extracted from the data yet and all points are equally likely to produce results. In the following iterations, these weights will be recomputed based on how well the current rule set (rules discovered so far) describe each point. This approach is similar to existing boosting approaches such as AdaBoost [6]. The re-weighting is done according to a mixture model as used by the StatPC algorithm [13] and has the following advantages,

– allows us to find rules for areas that might otherwise be ignored by guiding the selection of seeds;
– enforces the detection of novel rules by steering the algorithm in each iteration to new (not yet discovered) patterns;
– ensures non-redundant rules due to the re-weighting after the detection of new rules, but it also allows for overlapping rules by overlapping regions;
– high weight points can, after the algorithm has terminated, be seen as outliers.

For a given rule set $RS = \{R_1 \dots R_k\}$ we use the respective predicate sets $\{\mathcal{X}_1 \dots \mathcal{X}_k\}$ as the basis of our mixture model. The mixture model represents the expected data distribution for this given predicate set. In the ideal case this set captures the hidden rules in our data, and thus, the mixture model represents the true data distribution. However, during our process we start with an empty set of rules and add rules incrementally to this set. During this incremental computation we would like to steer our algorithm towards the detection of novel and non-redundant rules.

Formally, we model the entire rule set as a mixture model of k components:

$$p_{RS}(x) = \sum_{i=1}^{k} w_i \cdot f_i(x, R_i)$$

Each component $f_i(x, R_i)$ represents the data distribution of one rule R_i and is weighted by w_i for the entire mixture.

Given such a mixture model we can test the significance of a candidate rule R' before adding it to the rules set. The support of such a rule has to be significantly deviating from the expected support given by the current model $p_{RS}(x)$. Each candidate rule can be simply checked by a statistical test with some given significance level α. We will go more into details for this statistical test in Section 3.4. However, the mixture model is even more powerful and can be used for our steered creation of such candidates. We use it as a re-weighting scheme for the seed points of our Monte Carlo algorithm. Unexpected points lead us to both novel and non-redundant rules and to outliers that are highly unexpected after termination of the algorithm.

For both the significance test (cf. Section 3.4) and the re-weighting we need this mixture model. However, it is hard to learn this mixture model, especially for complex definitions of $f_i(x, R_i)$ components. As such information about the internal distribution of rules is not given and due to practical computational reasons we stick to a uniform random distribution of objects for each component. We model each predicate set $\mathcal{X}_i$ as a uniform distribution:

$$f_i(x, R_i) = \begin{cases} freq_{DB}(\mathcal{X}_i) & \text{, if } x \text{ satisfies predicate } \mathcal{X}_i \\ 0 & \text{, otherwise.} \end{cases}$$

For the coverage area of predicate $\mathcal{X}_i$ we model a uniform distribution. Outside of this area the probability of seeing an object drops to zero. Overall, the

sum of all components f_i represents the expected data distribution w.r.t. the given set of rules RS.

The new weights are then defined as the inverse of the computed distribution.

3.2 Transformation to Transactional Database using a Window

To find interesting regions (*hyper-rectangles*), we will use existing itemset mining approaches. This does, however, require us to transform the original data into transactional data. We opt to view the dimensions as items as this has several advantages, most notably the advantage that is easier to determine whether adding a dimension to the mix adds any value. Using a window width w, we determine which points are within distance w of our seed point and take note of for which dimension this occurs. This list of dimensions forms our itemset. After removing all empty itemsets, we can then mine the resulting data to find dimensions that occur together often. Using a fixed window width has the following advantages,

- it is easy to compute statistical tests, i.e., we can easily compare to how many points we expect there to be in an interval;
- it is easy to test whether adding a dimension adds any value compared to how many points we drop;
- there are less parameter issues due to statistical test and notion of unexpectedness compared to uniformly distributed data;
- it is novel notion of confidence in QARM by exploiting directly the expectation of intervals (1D) and hyper-rectangles (mD).

Formally, for a centre point p and for every object o in the original database $\mathcal{DB}$, we compute the set of dimensions $X_{o,p}$ for which o is within distance w of p, i.e., $X_{o,p} = \{A_i \mid |o(A_i) - p(A_i)| \leq w\}$. The new transactional database $\mathcal{TDB}$ is then defined as $\mathcal{TDB} = \{X_{o,p} \mid o \in \mathcal{DB} \wedge X_{o,p} \neq \emptyset\}$

3.3 Random Maximal Itemset Mining

Once the data has been transformed into transactional data, we still have to determine which dimensions work well together. To avoid pattern explosion – and increasing the number of combinations of dimensions we have to consider – we have decided to use a random maximal itemset mining approach. This has the advantage of only reporting a few good itemsets from which others can be derived. Using only those that are *unique enough*, leaves with only a few hyper-rectangles we will have to further explore. Out of the possible options, we select the best one.

Thus, in each run we filter the best rule out of a set of (many) randomly generated candidates. However, it still leaves the question of how we measure the quality of these rules.

To further explain how this step works, we refer to the original paper by Moens et al. [12].

3.4 Evaluation of Discovered Hyper-Rectangles

We will evaluate a hyper-rectangle on its significance. That is, we determine how many points would be in the hyper-rectangle if the data was uniformly distributed. This can be done using a binomial distribution.

Let H be a hyper-rectangle in a set of dimensions X. We determine the probability that H contains $T(H)$ objects under the null hypothesis that the data is uniformly distributed in X [13]. The distribution of the $T(\cdot)$ is the Binomial distribution with parameters n and $vol(H)$, $T(H) \sim Binomial(n, vol(H))$. The significance level α is then used to decide whether we reject or accept the null hypothesis. For our one-sided test, the critical value θ_α is computed based on $\alpha = Probability(T(H) \geq \theta_\alpha)$.

3.5 Termination Conditions

The algorithm has 2 possible termination conditions. The first such condition is one that is inherent to random mining algorithms: stop when enough results have been produced. The second condition is reached when the algorithm can no longer find any significant rules within a certain amount of iterations.

3.6 Parameters

The algorithm has 4 parameters, one of which is optional.

- Minimum Support: the minimum number of objects that have to be contained within a hyper-rectangle;
- α: the minimum significance level for hyper-rectangles; the probability of observing too few points compared to a uniform distribution is less than α;
- Window width w (relative or absolute): the width used to determine the hyper-rectangles and start the mining of interesting hyper-rectangles;
- Maximum number of rules (optional): the algorithm stops after this many rules have been found.

3.7 Complexity

The average run-time complexity of our algorithm is $O(nm)$, with n the number of data points and m the number of dimensions. This can be demonstrated by looking at Algorithm 1. In the first step, we select a random point. This can be done in constant time. After this, we compute, for each other point, for which dimension it is within distance w of our center point. This requires two loops, one over all points, and one over all dimensions. This therefore has a complexity of $O(nm)$. After this step, we perform our random maximal itemset mining. This has an average run-time complexity of $O(n)$. Lastly, we add our recently discovered rectangle to our mixture model and recompute the weights. This step has an average complexity of $O(n)$. Not all points have to be visited for re-weighting, but on average, we are still dependent on n. Combining of all these, we find that the largest contributing factor to our algorithm's complexity of $O(nm)$.

4 Experiments

The experiments were run on a laptop with a 2.3 Ghz Intel Core i7 and 8 GB DDR3 running OSX 10.9.2. All experiments were run using Python 2.7.x and the NumPy library.

4.1 Datasets

For our experiments, we used three datasets of different sizes, namely the well known Iris dataset (150 tuples, 4 dimensions) and two synthetic datasets, S1500 (1595 tuples, 20 dimensions), and D50 (1596 tuples, 50 dimensions), both of which have also been used for subspace clustering algorithms [14]. We strip all datasets of their non-numeric attributes to focus on the numeric aspect of the algorithm.

4.2 Evaluation Metrics

We use the following evaluation metrics to evaluate our patterns:

- **Area Covered:** A higher coverage of the data results in better descriptiveness, while a smaller coverage results in more specialised patterns.
- **Overlap:** The higher this number, the more overlap we have and the more our patterns are clumped together.
- **Pattern Size:** The amount of data that is covered by our rules. If the average size of a pattern is small, we have a highly specialized rule (contrary to regular pattern mining), while larger patterns offer a broader view of the data.
- **Jaccard Similarity:** The Jaccard similarity coefficient gives us an intuition on the overlap between patterns.
- **Confidence:** as we are generating association rules, we are also interested in their quality.

The experiments are repeated several times after which the results are averaged.

4.3 Results

We have selected one real-world dataset and two datasets that best illustrate the results of our algorithm. Results on other datasets were comparable to the ones listed below.

A few observations can be made here. While the area explored increases as the number of rules increases, we can see that after a certain point the increase becomes less and less obvious. This is best demonstrated by looking at the results for the D50 dataset. Once a saturation point in terms of number of rules is achieved, we have explored most of what there is to explore in the data and any further rules will only result in marginal increases. On the other hand, once we have reached said saturation point, we can see a sharp increase in overlap.

$\mathcal{D}$	Samples	Total Size	Size	Pattern Size	Overlap	Jacard	Confidence	Runtime(s)
Iris	50	122.34	105.31	50.78	174.12	0.63	87.12	0.41
	20		42.16	48.65	115.17	0.485	83	0.26
	10		19.09	42.8	31.46	0.296	86	0.196
S1500	50	7.67e62	1.27e39	297.4	1.11e27	0.143	91	3.13
	20		1.23e33	298.1	5.81e23	0.131	87	2.51
	10		1.22e30	288.3	5.72e22	0.119	86	2.04
D50	50	6.54e152	1.24e36	409.74	1.818e18	0.166	80	3.2
	20		5.87e35	374.25	617e9	0.166	83	2.53
	10		6.57e26	366.4	223e7	0.165	82	2.09

Table 1. Experimental Results

Naturally, as less and less area is open for novel information, already explored areas are (partially) revisited.

Comparing the S1500 and D50 datasets, we can see that our algorithm performs better on datasets with higher dimensionality. Since we mine for maximal frequent itemsets when determining which dimensions can be combined, having a high dimensional dataset can be an advantage. That is, since the algorithm leans towards larger combinations of dimensions, we can achieve more spread out patterns in the data.

We also note that the size of the patterns, i.e., the number of items per pattern, does not change that much when considering more rules. This is the result of how we compute the windows and/or what the data looks like. Since we are using a fairly static approach in the transformation to a transactional database, we end up with patterns that, when looked at visually, all look the same. Combined with the distribution in the data, we end up with patterns of roughly the same size. This can however lead to our algorithm overlooking interesting smaller

As a final observation, we remark that the runtimes for the S1500 and D50 almost identical. Using a dataset of higher dimensionality does not greatly affect the runtime of the algorithm due to the sampling approach used.

We should however note that repeating results with this algorithm is not an easy task. Due to the fact that several steps in this algorithm rely on randomization, it is nearly impossible to achieve the same results and/or patterns on subsequent runs. While this is not problematic for exploring data, it can be problematic in terms of repeatability.

4.4 Efficiency

We compare the efficiency of our method to that of other exploratory methods. We only compare to equi-width discretization in combination with random maximal itemset sampling. We do not add any extra comparisons as we only chose discretization methods and the only overhead they cause is in the preprocessing step. All other remaining steps should be similar complexity-wise. We use synthetic datasets to demonstrate the runtime efficiency when considering

both the number of objects and the number of dimensions. As the amount of time spent per rule is nearly identical, we only generate 25 rules. The results

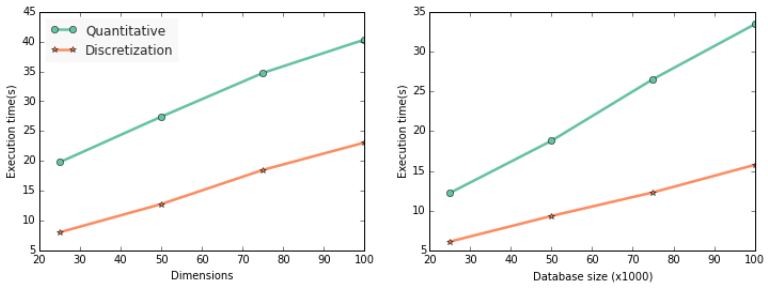

Fig. 1. Efficiency Results

of the experiment are shown in Figure 1. As we can clearly see, our method is slightly slower than the discretization approach but does not fall behind by a lot as the dimensionality of the dataset increases. The discretization approach loses a lot of time doing the actual preprocessing of the data. So having to guess the amount of bins or other, harder to estimate, parameters can lead to a large increase in runtime and require a larger knowledge of the data being processed. Our method, while slightly slower, is capable of generating a small amount of rules quickly without the hassle of having to preprocess the data.

While we allow for several parameters, only two have any real impact on the efficiency of the algorithm. That is, the window width and the number of rules to generate. Obviously, the algorithm takes longer as more rules have to be generated. The window width contributes in that it increases the size of the transactional database, also increasing the workload of further steps. While this effect is noticeable, it never has too great of an impact on the total runtime. The results show that each additional rule adds a near constant amount of time. We omit a graph due to space restraints.

5 Conclusion

We demonstrated a method that allows to rapidly find a limited set of quantitative association rules. This provides the user with a powerful tool to explore the data without having to worry about the massive runtimes or number of patterns that are typically associated with this task. We have shown that our method is efficient. We also demonstrated that our algorithm is capable of finding a nice spread of patterns over the data instead of generating a large amount of rules over the same regions.

References

1. R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD*, pages 207–216, 1993.
2. Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *ACM SIGKDD*, pages 261–270, 1999.
3. S. D. Bay. Multivariate discretization for set mining. *Knowl. Inf. Syst.*, 3(4):491–512, 2001.
4. R. Bayardo, Jr., B. Goethals, and M. Zaki, editors. *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI 2004)*, volume 126 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
5. S. Brin, R. Rastogi, and K. Shim. Mining optimized gain rules for numeric attributes. *IEEE Trans. Knowl. Data Eng.*, 15(2):324–338, 2003.
6. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.
7. T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining optimized association rules for numeric attributes. *J. Comput. Syst. Sci.*, 58(1):1–12, 1999.
8. B. Goethals, W. Le Page, and H. Mannila. Mining association rules of simple conjunctive queries. In *Siam Data Mining (SDM)*. SIAM, 2008.
9. J. W. Grzymala-Busse. Three strategies to rule induction from data with numerical attributes. *Transactions on Rough Sets*, 3135:54–62, 2005.
10. J. Mata, J.-L. Alvarez, and J.-C. Riquelme. An evolutionary algorithm to discover numeric association rules. In *ACM SAC*, pages 590–594, 2002.
11. R. J. Miller and Y. Yang. Association rules over interval data. In *ACM SIGMOD*, pages 452–461, 1997.
12. S. Moens and B. Goethals. Randomly sampling maximal itemsets. In *Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics*, IDEA '13, pages 79–86, New York, NY, USA, 2013. ACM.
13. G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *ACM SIGKDD*, pages 533–541, 2008.
14. E. Müller, S. Günnemann, I. Assent, and T. Seidl. Evaluating clustering in subspace projections of high dimensional data. *PVLDB*, 2(1):1270–1281, 2009.
15. C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *ACM SIGMOD*, pages 418–427, 2002.
16. A. Salleb-Aouissi, C. Vrain, C. Nortet, X. Kong, V. Rathod, and D. Cassard. Quantminer for mining quantitative association rules. *Journal of Machine Learning Research*, 14(1):3153–3157, 2013.
17. R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD*, pages 1–12, 1996.
18. P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *ACM SIGKDD*, pages 32–41, 2002.
19. N. Tatti. Itemsets for real-valued datasets. In *IEEE ICDM*, pages 717–726, 2013.
20. G. Webb and J. Vreeken. Efficient discovery of the most interesting associations. *Transactions on Knowledge Discovery from Data*, 8(3):15:1–15:31, 2014.
21. G. I. Webb. Discovering associations with numeric variables. In *ACM SIGKDD*, pages 383–388, 2001.
22. M. L. Yiu and N. Mamoulis. Iterative projected clustering by subspace mining. *IEEE Trans. Knowl. Data Eng.*, 17(2):176–189, 2005.