Mining Rank-Correlated Sets of Numerical Attributes

Toon Calders
Department of Mathematics and Computer Science
University of Antwerp
toon.calders@ua.ac.be

Bart Goethals
Department of Mathematics and Computer Science
University of Antwerp
bart.goethals@ua.ac.be

Szymon Jaroszewicz
Szczecin University of Technology
National Institute of Telecommunications
Szachowa 1, 04-894, Warsaw, Poland
sjaroszewicz@wi.ps.pl

ABSTRACT

We study the mining of interesting patterns in the presence of numerical attributes. Instead of the usual discretization methods, we propose the use of rank based measures to score the similarity of sets of numerical attributes. New support measures for numerical data are introduced, based on extensions of *Kendall's tau*, and *Spearman's Footrule* and *rho*. We show how these support measures are related. Furthermore, we introduce a novel type of pattern combining numerical and categorical attributes. We give efficient algorithms to find all frequent patterns for the proposed support measures, and evaluate their performance on real-life datasets.

Categories and Subject Descriptors: H.2.4 [Database Management]:Systems I.2.6[Artificial Intelligence]:Learning Knowledge Acquisition

General Terms: Algorithms, Experimentation, Theory.

Keywords: Data mining, Numerical, Rank Correlation.

1. INTRODUCTION

The motivation for the research reported upon in this paper is an application where we want to mine frequently occurring patterns in a meteorological dataset containing measurements from various weather stations in Belgium over the past few years. Each record contains a set of measurements (such as temperature or pressure) taken in a given station at a given time point, together with extra information about the stations (such as location or altitude).

The classical association rule framework [1], however, is not adequate to deal with numerical data directly. Most previous approaches to association rule mining for numerical attributes were based on discretization, see for example [16]. Discretization, however, has serious disadvantages. First of all it *always* incurs an information loss, since values falling in the same bucket become indistinguishable and small differences in attribute value become unnoticeable. On the other hand, very small changes in values close to a discretization border may cause unjustifiably large changes in the set of active rules. Second, if there are too many discretization intervals, discovered rules are replicated in each interval making the overall trends hard to spot. It is also possible that rules will fail to meet the minimum support criterion when they are split among many narrow intervals. In [16] a method for merging narrow intervals into wider ones was combined with a special scheme to prune spurious rules. The method, however, cannot entirely solve the problems related to discretization, since it is impossible to decide with certainty which rules were true associations and which were just artifacts of discretization. Also, information loss and instability at interval borders is inherent to discretization and cannot be eliminated entirely.

To tackle the problem of mining the meteorological dataset without relying on discretization methods, we propose a new technique based on well established statistical studies of rank correlation measures [12, 13]. More specifically, we propose to compare attributes by the rank their values impose on the records in the database. For example, for a given set of attributes, this can be done by counting the number of pairs of records such that all attributes rank the first tuple higher than the second tuple. When this number is high, it gives a clear indication that the attributes in the set behave similarly, and hence, reveals an interesting pattern. As it turns out, this number is related to the well known Kendall's τ [12] rank correlation measure, which will be thoroughly explained in the next section.

Some examples of the types of rules we are able to discover are the following: given two records t_1 and t_2,

> *If the altitude of the sun in t_1 is higher than in t_2, then temperature is likely to be higher as well.*
>
> *If t_1 comes from a weather station in Antwerp, and t_2 from Brussels, and wind speed in t_1 is higher than in t_2, then it is likely that cloudiness is higher as well.*

The main contributions of our paper are as follows:

1. We propose three new support measures for sets of numerical attributes; supp_τ, supp_ρ, and supp_F, which are based on well-known statistical rank correlation measures, *i.e.*, respectively *Kendall's* τ, *Spearman's* ρ and *Spearman's Footrule* F [12, 13].

2. We show how to combine the mining of sets of numerical attributes with ordinal and categorical attributes and how to extend it to association rules.

3. We propose and discuss algorithms to mine sets of numerical attributes with these support measures, based on spatial indexing techniques.

4. We present an empirical evaluation of our proposed solutions on real-life datasets to show the feasibility and effectiveness.

2. RANK-BASED SUPPORT MEASURES

For now, we assume that all values of numerical attributes are distinct, which is a reasonable assumption for real-valued attributes. Situations with tied values are taken care of in Section 3.

Let D be a database. Elements of D are called transactions, or records. Let H be the header of D, *i.e.*, the set of its attributes.

An attribute is called *numerical* if its domain is the set of real numbers. A *categorical* attribute has a finite **unordered** domain, and an *ordinal* attribute has a finite **ordered** domain. An example of an ordinal attribute's domain is $\{low, medium, high, veryhigh\}$.

The rank of a value $t.A$ of a numerical attribute A, is the index of t in D when D is ordered ascending w.r.t. A. That is, the smallest value $t.A$ of A gets rank 1, etc. We denote the rank of $t.A$ in A by $r_t.A$; *i.e.*, r_t denotes the tuple t where all values have been replaced by their rank. In Table 1, an example weather database has been given. The number between brackets in the entry for $t.A$ is $(r_t.A)$.

Rank methods use the ranks of attribute values, not the values themselves. They assess and test correlations between attributes with ordered domains.

2.1 Rank Correlation Measures

Three well-known rank correlation measures are *Kendall's* τ, *Spearman's* ρ, and *Spearman's Footrule* F [12, 13]. Let A and B be two numerical attributes.

The measures can now be defined as follows:

$$\begin{aligned}\rho(A,B) &= 1-\frac{6\sum_{t\in D}(r_t.A-r_t.B)^2}{|D|^3-|D|}\\ F(A,B) &= 1-\frac{4\sum_{t\in D}|r_t.A-r_t.B|}{|D|^2}\\ \tau(A,B) &= \frac{4|\{(s,t)\in D^2 \mid s.A<t.A\wedge s.B<t.B\}|}{|D|(|D|-1)}-1\end{aligned}$$

The observation behind ρ and F is that when two attributes are highly positively correlated, the ranks of their values will be close within most tuples. Hence, the sums $\sum_{t\in D}(r(t.A)-r(t.B))^2$ and $\sum_{t\in D}|r(t.A)-r(t.B)|$ will be small if A and B are correlated. On the other hand, if A and B are uncorrelated, the expected absolute difference between the ranks of A and B is $(|D|-1)/2$, thus resulting in very high values for these sums. The formulas are chosen in such a way that they are 1 for total positive correlation (*e.g.*, $A=B$), and -1 for total negative correlation (*e.g.*, A and B are in reverse order). See [12] for details.

For $\tau(A,B)$, it suffices to notice that

$$\frac{|\{(s,t)\in D^2 \mid s.A<t.A\wedge s.B<t.B\}|}{|D|(|D|-1)}$$

is actually the probability that for a randomly chosen pair of tuples (s,t), $s.A<t.A$ and $s.B<t.B$. Again, the formula is such that it is 1 for total positive correlation, and -1 for total negative correlation. Notice that this connection of τ with this probability, makes its interpretation very natural. The other two measures do not possess such clear interpretations.

2.2 Support For Numerical Attributes

The support measures we propose are the following:

$$\begin{aligned}\text{supp}_\rho(I) &= 1-\frac{\sum_{t\in D}(\max_{A\in I} r_t.A-\min_{A\in I} r_t.A)^2}{|D|(|D|-1)^2}\\ \text{supp}_F(I) &= 1-\frac{\sum_{t\in D}(\max_{A\in I} r_t.A-\min_{A\in I} r_t.A)}{|D|(|D|-1)} \qquad (1)\\ \text{supp}_\tau(I) &= \frac{|\{(s,t)\in D^2 \mid \forall A\in I: s.A<t.A\}|}{\binom{|D|}{2}}\end{aligned}$$

Similarly as for the correlation measures ρ and F, the intuition behind the measures supp_ρ and supp_F is that attributes can be said to be similar if they result in similar ranks for the tuples. Hence, when the attributes are similar, $(\max_{A\in I} t.A-\min_{A\in I} t.A)$ tends to be small. Both supp_ρ and supp_F are based on aggregating this difference over all tuples. For $\text{supp}_\tau(I)$, we take all pairs of tuples (s,t), and count the number of pairs such that s comes before t for all attributes of I. This number is then divided by the maximal possible number of such pairs. Notice incidentally that this maximum is $\binom{|D|}{2}$, and not $|D|(|D|-1)$, because it is not possible that s comes before t, and t before s at the same time.

PROPERTY 1. *For all sets of numerical attributes $I\subseteq J$, and for supp_m any of the measures supp_ρ, supp_F, and supp_τ, the following holds:*

$$0\le \text{supp}_m(J)\le \text{supp}_m(I)\le 1\ .$$

EXAMPLE 1. *Consider the example weather data given in Table 1. This relation has 4 numerical attributes: W, T, P, and H. In order to compute the supp_ρ and supp_F, we first need to compute the ranks. This can easily be done by sorting the table once for every numerical attribute. In Table 1, the ranks have been given in brackets. In this example,*

$$\begin{aligned}\text{supp}_\rho(\{T,P,H\}) &= 1-\frac{1^2+2^2+2^2+1^2+1^2}{5\cdot 16} = \frac{69}{80}\\ \text{supp}_F(\{T,P,H\}) &= 1-\frac{1+2+2+1+1}{5\cdot 4} = \frac{13}{20}\end{aligned}$$

	W		T		P		H	
t_1	3.15	(4)	20.1	(4)	1 030.3	(5)	0.75	(4)
t_2	2.12	(2)	20.5	(5)	1 025.7	(4)	0.65	(3)
t_3	5.19	(5)	13.7	(3)	1 015.6	(3)	0.80	(5)
t_4	1.05	(1)	12.8	(2)	1 012.3	(2)	0.25	(1)
t_5	2.13	(3)	5.3	(1)	1 005.7	(1)	0.30	(2)

Table 1: Example weather data; W stands for Wind speed, T for Temperature, P for Pressure, and H for Humidity. The numbers between brackets indicate the rank of the value within the attribute.

In order to compute the support $\text{supp}_\tau(\{T,P,H\})$*, we need to count the number of pairs of tuples such that the first tuple is smaller than the second tuple in* T*,* P*, and* H*. E.g., the pair* (t_5, t_1) *is such a pair, and* (t_2, t_3) *is not. As there are* 6 *pairs of tuples that are in order (*3 *with* t_4 *and* 3 *with* t_5 *as first element),* $\text{supp}_\tau(\{T,P,H\})$ *is* 6/10*.*

Extensions of the rank correlation measures. Notice that Property 1 states that the measures are scaled in such a way that they fall into the interval $[0,1]$. For $\text{supp}_\rho I$ and supp_F, however, 0 can only be obtained in the special case that $|I| \geq |D|$. We can, though, re-scale by making the multiplicative constants depending on $|I|$. As a by-product, we provide the following extensions of ρ and F to multiple attributes, where the range is $[-1,1]$, and these bounds are attainable, independently of $k = |I|$. The construction is based on the observation that in worst case, with k attributes, there are at most k tuples with the maximal difference $|D|-1$, k with difference $|D|-3$, etc. Subsequently, the sum over the tuples in the formulas is divided by the support in this worst case.

$$\begin{aligned} \rho'(I) &= 1 - \frac{2\sum_{t\in D}(\max_{A\in I} r_t.A - \min_{A\in I} r_t.A)^2}{\frac{3k^2-6k+4}{3k^2}|D|^3 - \frac{1}{3}|D|} \\ F'(I) &= 1 - \frac{2\sum_{t\in D}(\max_{A\in I} r_t.A - \min_{A\in I} r_t.A)}{\frac{k-1}{k}|D|^2} \end{aligned} \tag{2}$$

Note that for $k = 2$, these measures indeed correspond to the rank correlation measures ρ and F. There are, however, a couple of reasons why we prefer to use our support measures given in (1). First and most important, ρ' and F' do not have the monotonicity property. Second, for our purposes, the formulas ρ' and F' are needlessly complicated, while they are linearly related to our support measures anyway.

Discussion. It is worth remarking that supp_τ inherits the nice statistical interpretation of τ. Indeed, $\text{supp}_\tau(I)$ is twice the probability that in a randomly selected pair of tuples (s,t), s is smaller than t in all attributes of I. This connection makes that supp_τ, in contrast to supp_ρ and supp_F, is easy to interpret, and that we can extend it easily to other interestingness measures, such as *e.g.*, confidence of association rules between sets of numerical attributes. This extension is discussed later on in this section. On the negative side, however, it should be noted that supp_τ is much harder to compute than the other two support measures, as τ is defined over *pairs* of tuples while the other two over the ranks of *single* tuples. The many desirable properties of supp_τ, however, are our main motivation for dealing with the challenging computational difficulties of supp_τ, instead of just taking one that is easier to compute.

2.3 Properties of the Measures.

The following theorem gives the average supp_τ in the case of statistically independent attributes. This expected support under independence can be used as a reference point to assess the support of a set of numerical attributes.

Theorem 1. *Let* $A_1, A_2, \ldots, A_k$ *be* k *statistically independent numerical attributes. The expected value of the support* $\text{supp}_\tau(A_1 A_2 \ldots A_k)$ *is* $\frac{1}{2^{k-1}}$*.*

Proof. Let us first prove a helpful fact. Let X be a numerical random variable and let x_1 and x_2 be its two instantiations. It is easy to see that $Pr\{x_1 < x_2\} = \frac{1}{2}$. Indeed, let $F(x)$ be the cumulative probability distribution function of X and let $f(x)$ be its probability density function. Then Let us pick any pair of distinct records (s,t) from D. Since all attributes are independent, it follows that $Pr\{s[A] < t[A] \text{ forall } A \in I\} = \frac{1}{2^k}$. Hence, the expected number of pairs that are in order is $\frac{|D|(|D|-1)}{2^k}$. Therefore, the expected $\text{supp}_\tau(I) = \frac{1}{2^{k-1}}$ (Q.E.D.)

For the rank correlation measures ρ, τ, and F, there exist many inequalities relating them to each other. For an overview, see, *e.g.*, [12, 13, 6]. We extended some of these bounds to our support measures. Especially the relations where supp_ρ and supp_F that can be used to upper bound supp_τ are of particular interest to us, because, in contrast to supp_τ itself, such an upper bound is easy to compute, and might allow for pruning if it is below the support threshold.

Theorem 2. *For any set of numerical attributes* I *of any database,*

$$\text{supp}_\tau(I) \leq \text{supp}_F(I)$$
$$1 - |D|(1-\text{supp}_F(I))^2 \leq \text{supp}_\rho(I) \leq 1 - \tfrac{1-\text{supp}_F(I)}{|D|-1}$$

Proof. The first inequality is based on the following observation. Suppose that for a tuple t, A is the attribute in I with the lowest rank $r_t.A$, and B the one with the highest $r_t.B$. Then we know that in the database, there are exactly $r_t.A-1$ tuples smaller in A than t, and $r_t.B-1$ tuples smaller than t in B. Hence, there are at least $(r_t.B-1)-(r_t.A-1) = \max_{A\in I} r_t.A - \min_{B\in I} r_t.B$ tuples that are smaller in B, but not smaller in A than t. For these tuples s, neither (t,s), nor (s,t) will contribute to the support. If we sum this number over all tuples, we get a lower bound on the number of pairs (s,t) such that neither (s,t), nor (t,s) contributes to the support. Since every incomparable couple of tuples s,t is taken into account twice by this measure, we need to divide it by 2 to get a lower bound on the pairs of tuples not contributing to the support. Hence, an upper bound on the number of pairs (s,t) such that $s.A < t.A$ for all $A \in I$ is:

$$\binom{n}{2} - \frac{\sum_{t\in D} \max_{A\in I} r_t.A - \min_{B\in I} r_t.B}{2} \quad .$$

To get an upper bound on $\text{supp}_\tau(I)$, we still have to divide by $\binom{n}{2}$, resulting in $\text{supp}_F(I)$.

The other inequalities are based on the observation that

$$\begin{aligned} \sum_{t\in D}(\max_{A\in I} r_t.A - \min_{A\in I} r_t.A) &\leq \sum_{t\in D}(\max_{A\in I} r_t.A - \min_{A\in I} r_t.A)^2 \\ &\leq \left(\sum_{t\in D}(\max_{A\in I} r_t.A - \min_{A\in I} r_t.A)\right)^2 \end{aligned}$$

(Q.E.D.)

Example 2. *Consider again the example relation given in Table 1. For the set* $\{T,P,H\}$*, we illustrate the bound* $\text{supp}_\tau \leq \text{supp}_F$*. Consider, e.g., the tuple* t_3*. In this tuple, the rank of* T *is* 3*, and of* H *is* 5*. Therefore, there must be at least* 2 *tuples that are smaller than* H*, and not smaller than* T*.*

2.4 Categorical attributes

Our definitions of support are also applicable to ordinal attributes, since their domains are ordered. For ordinal attributes, however, we cannot assume that tied pairs do not occur, so in order to maintain the relationship to Kendall's τ tied pairs would also need to be counted, see [13] for details. In Section 3, we discuss what to do in case of ties.

Let us now look at how to integrate categorical attributes. Without loss of generality we will look only at binary attributes, also called *items*. When a given item is set to 1 we say that it is *present* in the transaction.

Categorical attributes will be used to define the context in which the support of numerical or ordinal attributes is computed. Let B be the set of binary attributes and O the set of numerical and ordinal attributes of the dataset D.

Definition 1. *Let $I \neq \emptyset, I \subseteq O, J_1, J_2 \subseteq B$. The support of I in context (J_1, J_2) is defined as*

$$\begin{aligned}\mathrm{supp}_\tau(I|(J_1,J_2)) = |\{(t_1,t_2) : t_1, t_2 \in D, \textit{ and } J_1 \subseteq t_1, \\ J_2 \subseteq t_2, \textit{ and } \forall A \in I : t_1.A < t_2.A\}| / \binom{|D|}{2},\end{aligned}$$

where $J_1 \subseteq t_1, J_2 \subseteq t_2$ denotes the fact that all items from J_1 are present in transaction t_1 and all items from J_2 are present in t_2.

Notice that this definition of support for numerical and categorical attributes captures exactly the patterns we wanted to find in the first place.

Example 3. *Suppose that we have attributes for the location where the weather data was gathered. Then,* e.g.*, the rule "In measurements of Antwerp and Brussels, the temperature and wind speed in Antwerp are higher than in Brussels in 70% of the cases" could be expressed as:*

$$\mathrm{supp}_\tau(\{T, W\}|(\mathit{Antwerp}, \mathit{Brussels})) = 70\%$$

Note that this rule does not *require that the measurements that are compared are of the same point in time. Rather it expresses that wind speeds and temperatures in Antwerp are in general higher than in Brussels.*

2.5 Association rules

Because of the interpretation of $\mathrm{supp}_\tau(I|(J_1,J_2))$ as the conditional probability that two pairs that satisfy respectively J_1 and J_2 are concordant on all attributes of I, it is straightforward to define the confidence of association rules in this context. Notice that for the other support measures, it is not at all straightforward how to score an association rule.

Definition 2. *Let I_1, I_2 be sets of numerical attributes, and let J_1 and J_2 be sets of binary attributes. Then, the confidence of the association rule $I_1 \to I_2|(J_1,J_2)$ is defined as*

$$\mathrm{conf}(I_1 \to I_2|(J_1,J_2)) = \frac{\mathrm{supp}(I_1 \cup I_2|(J_1,J_2))}{\mathrm{supp}(I_1|(J_1,J_2))}$$

The interpretation of an association rule $I_1 \to I_2|(J_1,J_2)$ with confidence c, is simply that for all pairs of tuples (t_1,t_2), with t_1 satisfying J_1, and t_2 satisfying J_2, if t_1 is smaller than t_2 on I_1, then there is a chance of c that t_1 is also smaller than t_2 on I_2.

Example 4. *The following rule expresses that in Antwerp, if the wind speed in one measurement is greater than in another measurement, then there is a probability of 65% that the cloudiness will be greater as well.*

$$\mathrm{conf}(\{W\} \to \{C\}|(\mathit{Antwerp}, \mathit{Antwerp})) = 65\%.$$

3. RANKS WITH TIES

In the beginning of the paper, we assumed that we are working with numerical data, in which no identical values occur. In reality, however, this assumption is not always true. Often, the data contain measurements of finite precision, *e.g.*, temperatures up to .1 degree. In those cases, many numerical values will be duplicated, and the ranks implied by the attributes are no longer total, because ties will occur. Also for ordinal data, many ties will occur. Consider, *e.g.*, an attribute *Cloudiness* that can take values $\{\mathit{low}, \mathit{medium}, \mathit{high}\}$. For this attribute, at least 1/3th of the pairs will be tied. In this section we describe how we can extend our support measures to work with rankings with ties.

For our support measure supp_τ, we can just keep our original definition; that is, if two tuples have a tie on an attribute of I, they won't contribute to $\mathrm{supp}_\tau(I)$. For the other measures, we need to decide what rank we want to assign to a value that is repeated; *i.e.*, suppose a value $t.A$ is repeated m times in attribute A, and there are p tuples with an A-value smaller than $t.A$. Then, in any order consistent with the ranking with ties, $t.A$ will have a rank in the range $p+1, \ldots, p+m$. There are now several choices for the rank of $t.A$; we could take the average of the ranks $ar_t.A = (p+1+m)/2$, the minimal rank $\underline{r}.A = p+1$, or the maximal rank $\overline{r}.A = p+m$. The choice we make will have a large influence on the supports measured by supp_ρ and supp_F. In our experimental evaluations, we have chosen for assigning the average rank to tied values, because, intuitively, this is the most natural choice, as it keeps, *e.g.*, the property that the sum of all ranks is $\sum_{i=1}^{|D|} i$. For the minimal and maximal rank, this sum respectively decreases and increases, hence introducing a bias. Thus, to summarize, for rankings with ties, the definition of supp_τ does not change, and $\mathrm{supp}_\rho(I)$ and $\mathrm{supp}_F(I)$ become:

$$\begin{aligned}\mathrm{supp}_\rho(I) &= 1 - \frac{\sum_{t\in D}(\max_{A\in I} ar_t.A - \min_{A\in I} ar_t.A)^2}{|D|(|D|-1)^2} \\ \mathrm{supp}_F(I) &= 1 - \frac{\sum_{t\in D}(\max_{A\in I} ar_t.A - \min_{A\in I} ar_t.A)}{|D|(|D|-1)}\end{aligned}$$

Notice that these measures reduce to the original definitions of supp_ρ and supp_F when there are no ties.

With these definitions, we can still upper bound $\mathrm{supp}_\tau(I)$ with supp_F. Indeed; similarly as in the proof of the bound without ties, let A and B be two attributes of a set I. Consider a tuple t in the relation D, then, on the one hand, there are *at least* $\lceil ar_t.A \rceil$ tuples s such that $s.A \leq t.A$. On the other hand, there are *at most* $\lfloor ar_t.B \rfloor$ tuples s such that $s.B < t.B$. Therefore, there are at least $\lceil ar_t.A \rceil - \lfloor ar_t.B \rfloor$ tuples s, such that $s.A \leq t.A$ and not $s.B \geq t.B$, and that thus cannot contribute to supp_τ. The rest of the proof of the bound is now the same as for the case without ties.

For the sake of providing an upper bound to supp_τ, the result can still be improved:

THEOREM 3. *Let I be a set of numerical attributes. Let, for any tuple t of the database,*

$$\text{maxdiff}(t) := \max_{A \in I} \overline{r}_t.A - \min_{A \in I} \underline{r}_t.A \ .$$

Then,

$$\text{supp}_\tau(I) \ \leq \ 1 - \frac{\sum_{t \in D} \text{maxdiff}_I(t)}{|D|(|D|-1)}$$

Proof. Consider a tuple t, with $\text{maxdiff}(t) = \overline{r}_t.A - \underline{r}_t.B$. There are exactly $\overline{r}_t.A$ tuples s such that $s.A \leq t.A$, and there are exactly $\underline{r}_t.B$ tuples s such that $s.B < t.B$. Therefore, there are at least $\overline{r}_t.A - \underline{r}_t.B$ tuples s, such that $s.A \leq t.A$ and not $s.B \geq t.B$, and thus cannot contribute to supp_τ. The remainder of the proof is the same as for Theorem 2. (Q.E.D.)

4. ALGORITHMS

As only ranks are compared instead of the actual values in the database, we first transform the database by replacing every attribute value by its rank w.r.t. this attribute, which can be done in $\mathcal{O}(|H||D|\log(|D|))$ time.

We now present several techniques to efficiently generate all sets of attributes satisfying the minimum support threshold, where support is defined as supp_ρ, supp_F, or supp_τ. Essentially, the problem comes down to frequent itemset mining, with the important difference that we can no longer use most of the support counting optimizations used by many algorithms. Indeed, we do not have to count the number of transactions in which an itemset is present, but instead, we have to compare the ranks of all items of an itemset in each transaction. Therefore, we can never shrink the database by removing attributes or transactions, as is done in e.g. FP-growth or Eclat [9].

In the case of supp_ρ, or supp_F, the solution is straightforward as it can be solved by the standard level-wise search in which candidate itemsets are generated only when all of its subsets are known to be frequent. Counting the supports requires only a single scan through the data in every iteration, as is done in Apriori [1].

In the case that supp_τ is used as support measure, the mining task becomes inherently more difficult since the number of pairs of transactions is quadratic in the size of the database. In the next sections, we describe several methods which we investigated to tackle this problem.

4.1 Explicit creation of pairs of transactions

A brute force solution to the presented mining problem, is to explicitly combine all pairs of transactions from the original database D, and create a Boolean database D', such that

$$D' = \{t_{ij} \mid t_{ij} = \{A \mid t_i.A < t_j.A, \text{ with } t_i, t_j \in D\}\}$$

The problem is then reduced to standard frequent itemset mining on the new database D'.

The main disadvantage of this approach is that the size of the new database D' (and the time needed to create it) is quadratic in the size of the original database, so it is only applicable to small problems. If, however, the approach can be applied, it is most of the time very efficient in combination with depth-first frequent itemset mining algorithms such as Eclat, because all pairs that do not support an itemset will be removed from its branch, and hence, will not be considered anymore in the support computation of its supersets in that branch. Such fine level of pruning will not be possible in the other approaches presented hereafter.

4.2 Spatial indices

When the database is too large and the latter brute force method is not applicable, we need an efficient mechanism to count supp_τ for all candidate itemsets. For a given attribute set I, this comes down to the following nested for loop.

```
for all $t_i \in D$ do
  for all $t_j \in D$ do
    if $t_i.A < t_j.A$ for all $A \in I$ then
      count++
    end if
  end for
end for
```

Obviously, this double loop is also infeasible as it performs a quadratic operation for each candidate set. A better approach is to replace the inner for-loop with a more directed search for all transactions that have smaller values on all attributes in I. This is possible by using so called *spatial indices*. These are data structures especially designed to allow for searching in multidimensional data and to allow searches in several dimensions simultaneously.

For the itemsets search space traversal, we use a depth-first traversal (such as in e.g. FP-growth and Eclat) as this allows us to remove transactions from consideration in the current branch when they are no longer part of the support of the current itemset. Note, however, that since we consider pairs of transactions, we can only remove a transaction if it does no longer occur in any supporting pair of the current itemset. Of course, we will not duplicate the database for each branch, but only store the list of transaction identifiers of the transactions that are still under consideration. As a first optimization, we will actually store two separate lists of transaction identifiers, one for the outer for loop, and a second one on which the spatial index is created. This also allows us to prune some transactions earlier. Indeed, when a transaction in the outer loop list is no longer involved in any pair, then it can be removed from that list, although it might still be present and necessary in the index, and vice versa.

Another optimization, as we recall from Theorem 2, is the usage of the upper bound presented by supp_F. As supp_F can be computed a lot more efficiently compared to supp_τ, we will first compute it for every candidate itemset. In case it gives a value under the minimum support threshold, we can already prune the itemset from the search space without computing the more expensive supp_τ.

Typically, depth-first frequent itemset mining algorithms do not exploit the monotonicity of support optimally. That is, when a candidate itemset is generated, they do not check whether all of its subsets are known to be frequent, and hence, they generate more candidate itemsets than in a breadth-first approach (see e.g. [9] for more information). Nevertheless, using simple heuristics and fast support counting techniques, it turns out not to have a great effect for most cases [4]. In our setting, however, the support computation becomes a lot harder and it is important to reduce the number of candidate itemsets to be counted as much as possible. Fortunately, by generating all candidate itemsets in a reverse depth-first manner, it is guaranteed that all subsets of a given candidate itemset are generated before the candidate itself [5]. For example, the subsets of $\{a, b, c, d\}$ would be generated in the following order: $\{d\}$, $\{c\}$, $\{c, d\}$, $\{b\}$, $\{b, d\}$, $\{b, c\}$, $\{b, c, d\}$, $\{a\}$, $\{a, d\}$, $\{a, c\}$, $\{a, c, d\}$, $\{a, b\}$, $\{a, b, d\}$, $\{a, b, c\}$, $\{a, b, c, d\}$. Thus, when all candidate itemsets are generated in this manner, it is still possible to check for each candidate whether all of its subsets are frequent, and if not, we can already remove the candidate as it must also be infrequent, due to the monotonicity of supp_τ (see Property 1). To index all transactions for the inner loop, we considered two well known spatial indices, called kd-tree and range tree respectively [3]. Next, we discuss how they work, their advantages and disadvantages. Notice that any other spatial data structures could be used as well. We report, however, only on kd-trees and range trees because they were designed primarily for main memory, in contrast to, e.g., R-trees [**?**], which are more appropriate when secondary storage is being used.

4.2.1 Kd-tree

A kd-tree [3] is similar to a binary search tree, except that, at every level of the tree, the ordering is based on a different attribute. In our case, the root of the tree orders on the first attribute of the itemset, its children on the second attribute and so forth.

The advantages of a kd-tree are its simplicity and low memory consumption. As a matter of fact, instead of implementing a tree structure, we simply implemented a binary search on the database itself with appropriately sorted records. First, the whole dataset is partitioned around the median of the first attribute, then each of the halves is partitioned around the median of the second attribute within that half, and so on for each attribute. Then, when all attributes have been used, but some parts still contain multiple tuples, then the ordering starts over with the first attribute.

4.2.2 Range-tree

A 1-dimensional range tree is simply a binary search tree. For multiple dimensions, a binary tree is created for the first dimension. Then, every node in the tree stores all transactions that fall into this node in a range tree, which is recursively constructed for the remaining dimensions. We refer the interested reader to [3] for more detailed information.

The range tree has the important advantage that it is very easy to create the tree for an itemset of size k, given the range tree of its prefix of size $k - 1$. Indeed, when a new candidate itemset of size k is created in the depth-first traversal, we already created the range tree for its $k - 1$ prefix, and we only need to add the a binary tree for the new dimension to all nodes that have multiple transactions in the nodes of the range trees for dimension $k - 1$.

4.2.3 Comparison

Obviously, both kd-tree and range-tree have their advantages and disadvantages. For the full technical details on the comparison, we refer to [3]. On the one hand, kd-trees use far less memory than range-trees. On the other hand, however, finding the number of tuples that are greater than a given tuple on all attributes of the itemset under consideration is far more efficient in a range-tree than in a kd-tree. More concretely, a kd-tree for an itemset of size k, uses $\mathcal{O}(nk)$ space, where n is the number of tuples. For the range-tree, the size is as large as $\mathcal{O}(n \log^k(n))$. For the counting of the number of tuples larger than a given tuple t, the kd-tree uses time proportional to $\mathcal{O}(n^{1-1/k})$, thus resulting in a cost of $\mathcal{O}(n^{2-1/k})$ for the counting operation. For large k, this cost becomes quadratic. For the range trees, however, the cost for one tuple is $\mathcal{O}(\log(n)^k)$, resulting in a total cost of $\mathcal{O}(n \log(n)^k)$ for the counting operation. Clearly, there will be a trade-off between memory and time in the choice between the kd-tree and the range-tree. This trade-off will also become clear in the experiments.

5. EXPERIMENTAL EVALUATION

5.1 An illustrative example

Let us first look at the `letter` dataset from the UCI Machine Learning Repository [10]. The dataset consists of selected features of 20000 handwritten letter images. All values (except for the class) are integer in the range from 0 to 15. The table below briefly describes those of the attributes which are used later

`letter`	character represented by handwritten letter
`box-x`	x position of center of bounding box (left)
`box-x`	x position of center of bounding box (left)
`box-y`	y position of center of bounding box (bottom)
`box-w`	width of the bounding box
`box-h`	height of the bounding box
`onpix`	number of 'on' pixels
`y-bar`	mean y of 'on' pixels in box
`y2bar`	mean y variance
`x2ybr`	mean of $x * x * y$
`xy2br`	mean of $x * y * y$
`x-ege`	mean edge count left to right
`xegvy`	correlation of `x-ege` with y

See [8], where the dataset was introduced, for details.

Table 2 shows the 5 most frequent itemsets in the `letter` dataset according to the three support measures proposed. The values for supp_F and supp_ρ have been scaled using Equations (2) to obtain more understandable values of support, even though they were mined with the definitions in (1) because of the monotonicity property.

The patterns that come out on top are quite intuitive. For example {`box-y,box-h`} means that for tall letters the center of their bounding box is higher up. This patterns has supp_τ of 0.734, meaning that if we pick two pairs of records at random, 73% of them agree on `box-y` and `box-h`. An analogous pattern can be seen for box's width and x position. Another intuitive patterns is {`box-w,onpix`}, wider letters obviously require more pixels. The pattern {`box-x,box-y`} is an interesting pattern which we cannot explain, it says that x and y coordinates of the center of the bounding box are positively correlated. The pattern {`box-y,box-w`} means that wide letters tend to have the center higher up. We believe that this is due to letters with lower 'tails' such as 'j','p','q','y' are narrow, while wide letters such as 'm' and 'w' do not have such 'tails'. supp_F and supp_ρ produced a three item pattern {`box-x,box-w,onpix`}. This pattern made it to the top of the list after support scaling (Equations 2), and is a result of non-monotonicity of those measures. It seems that it is a consequence of patterns {`box-x,box-w`} and {`box-w,onpix`} and does not contain any extra information.

supp_τ		supp_F		supp_ρ	
{box-y,box-h}	0.734	{box-y,box-h}	0.786	{box-x,box-w}	0.939
{box-x,box-w}	0.706	{box-x,box-w}	0.785	{box-y,box-h}	0.914
{box-w,onpix}	0.667	{box-w,onpix}	0.728	{box-w,onpix}	0.904
{box-x,box-y}	0.663	{box-x,box-y}	0.697	{box-x,box-y}	0.879
{box-y,box-w}	0.631	{box-x,box-w,onpix}	0.693	{box-x,box-w,onpix}	0.877

Table 2: Five most frequent itemsets for each support measure in the letter dataset.

itemset (numerical)	letters	support
{xy2br}	P/U	629 957 (96.5%)
{x2ybr}	A/U	627 340 (97.8%)
{y2bar}	A/D	626 245 (98.6%)
{x-ege}	T/M	626 216 (99.3%)
{x2ybr}	D/Y	625 314 (98.8%)
{y-bar, x2ybr}	A/T	613 349 (97.7%)
{y-bar, y2bar}	A/T	610 605 (97.2%)
{y2bar, x2ybr}	A/U	608 084 (94.7%)
{y2bar, x2ybr}	A/T	607 727 (96.8%)
{x2ybr, xegvy}	A/Y	607 549 (98.0%)
{y-bar, y2bar, x2ybr}	A/T	604 593 (96.3%)
{y-bar, x2ybr, xegvy}	A/Y	593 157 (95.6%)
{y-bar, x2ybr, xegvy}	A/T	583 680 (92.9%)
{y-bar, y2bar, xegvy}	A/T	581 609 (92.6%)
{y2bar, x2ybr, xegvy}	A/T	581 338 (92.6%)

Table 3: Most frequent (supp_τ) patterns in letter data including the categorical 'letter' attribute

Table 3 shows the most frequent 1, 2 and 3 attribute patterns in the letter data when the categorical 'letter' attribute is considered. The 'letters' column shows which pair of letters is involved, and the 'support' column shows the actual value of support as well, as this value divided by the total number of pairs of records with for the given pair of records. For example, the first row denotes that for 96.5% of pairs of a letter P and a letter U, the mean of $x * y * y$ of the on-pixels is smaller for the P than for the U.

The top patterns involve arithmetic expressions of x and y coordinates of points which are not easily interpretable. It can be seen that the creators of the database have chosen the set of features quite well. Pattern {T/M,x-ege}, is quite interesting. It shows the average number of edges encountered, when traversing the letter image left-to-right. For 'T' this number is almost always small, while for 'M' there are often 4 edges to cross, so this number is high. The pattern tells us that in 99.3% of cases this number is parameter is indeed higher for letters 'M' than for letters 'T'.

5.2 Rules found in the weather data

In Table 4, some rules that were found in the meteorological dataset by the proposed method are presented. The input data was 5 years of hourly measured weather data taken from one weather station located at Brussels airport. In the table, the rule is reported, its confidence, its support, and also the support of the consequent of the rule. The meaning of the attributes is respectively the altitude of the sun (solar_alt), the amount of precipitation (precip), the amount of cloudiness (cloud), and the wind speed (w_speed).

Rule $X \to Y$	Supp(XY)	Conf	Supp(Y)
solar_alt→temp	33%	66%	50%
precip→cloud	21%	64%	43%
cloud→precip	21%	48%	32%
w_speed→precip	19%	44%	32%
cloud, w_speed→precip	13%	57%	32%

Table 4: Association Rules found in the weather data of one station in Brussels

The first rule simply claims that in 66% of the cases, the temperature is higher if the altitude of the sun is higher. The second rule indicates that if it is raining harder, then there is probability of 64% that the cloudiness is bigger as well. Intuitively, one would expect these confidences would be higher. This is not the case, however, as measurements of different days are compared (e.g., a measurement in winter vs a measurement in summer). Thus, for example, if the first measurement is in winter, and the second is in summer, the temperature of the former is likely to be lower than the latter, no matter what the solar altitudes in both measurements are.

The last three rules indicate that higher cloudiness or higher wind speed positively influence the amount of rain, and apparently, with both higher cloudiness and wind speed this effect becomes even stronger. Indeed; for two randomly selected tuples, the probability of having a higher precipitation is only 32%. When, however, we know that the cloudiness in the first tuple is higher, or the wind speed in the first tuple is higher, or both, the probability of the first tuple having a higher precipitation increases to respectively 48%, 44%, and 57%.

5.3 Performance evaluation of supp_τ

We evaluated the performance of the algorithms on a real-life meteorological dataset and on a number of benchmark datasets. All algorithms were implemented in the C programming language and tested on a 2.2GHz Opteron system with 2GB of RAM.

The meteorological data describes readings from several weather stations in Belgium at hourly intervals. The attributes include temperature, pressure, humidity, etc. There are about 5 million records in the database, and 57 attributes. There are many missing values, but about 20 attributes are present in most records.

We concentrate on performance of algorithms for supp_τ as it is much more computationally intensive than supp_F and supp_ρ. Some results for supp_F and supp_ρ are shown later in the section.

To evaluate performance of the algorithm for various database sizes we drew random samples of increasing sizes from

20%	10%	5%	1%
179 – 191	770 – 861	2 664 – 2 955	31 010 – 34 812

Table 5: Number of frequent itemsets for different values of minimum support for the meteorological data.

dataset	no. of records	no. of attributes
letters	20 000	16
elevators	16 599	19
2dplanes	40 768	11
puma8NH	8 192	9
bank32nh	8 192	33

Table 6: Characteristics of datasets used for experiments

the meteorological data and ran the algorithm on each sample for various minimum support thresholds.

Figure 1 shows the results for three supp_τ computation methods. It can be seen that range tree and kd-tree based algorithms scale to tens or hundreds of thousands of records depending on minimum support. In general we would not wait if the program ran for more than 3 hours. This was often the case with range trees, especially due to heavy swap memory usage. kd-trees incur almost no memory overhead so they always fit in main memory. The algorithm based on explicit pairs ran out of memory for 1, 000 records already.

Note that the number of records of the original table does not give full justice to the problem size, as for example for 500 000 records in the original database, there 124 999 750 000 unordered pairs of records to look at! This shows high usefulness of spatial indices applied.

It can be seen that range trees are the most efficient method, except for small minimum supports. It breaks down when itemsets become too long, due to large memory consumption. For 1% minimum support kd-trees achieve much better performance, since they incur almost no memory overhead. The third method: mining all explicitly precomputed pairs of records, cannot handle data of 10 000 records at all. This happens since 49 995 000 new records which are generated do not fit in memory. However, due to the types of pruning possible only in this method, it gives very good results for low minimum supports.

Overall we would recommend the use of range trees, unless minimum support is too low and extensive usage of swap memory becomes an issue. For low minimum supports and large datasets kd-trees are a better alternative. For low minimum supports and small datasets, the algorithm based on the explicit pairs becomes viable. A development of a hybrid approach, which would switch from one method to another at various stages of the mining process is a topic of future research.

We found the number of frequent itemsets to remain approximately constant with the increase in sample size as long as the support was constant percentage-wise. We summarize the results in Table 5 where for every value of minimum support, the range in which the numbers of frequent itemsets were contained for all considered sample sizes.

We now evaluate performance of the algorithm on various benchmark datasets. Table 6 summarizes their characteristics. The results are shown in Figure 2. Doubly logarithmic scales are used in both charts. We used several of Weka's regression benchmarks as well as the `letter` database described earlier (with the class attribute ignored). All attributes were continuous. The results are given for supp_τ and range tree based algorithm. It can be seen that the algorithm performs quite well even on large datasets. There are however some performance problems for datasets with large number of attributes.

minsupp[%]	time	#freq
supp_F		
50	2.736	945
40	10.350	4 229
30	1m54.026	21 573
20	6m41.215	63 387
10*	6m56.100	65 536
8*	6m54.100	65 536
supp_ρ		
$5 \cdot 10^{-6}$	0.865	1
$4 \cdot 10^{-6}$	0.862	1
$3 \cdot 10^{-6}$	0.864	1
$2 \cdot 10^{-6}$	0.863	1
$1 \cdot 10^{-6}$	0.955	17
$0.8 \cdot 10^{-6}$	1.457	215
$0.6 \cdot 10^{-6}$	10.465	4 332
$0.4 \cdot 10^{-6}$	5m35.282	49 165
$0.2 \cdot 10^{-6}$*	6m55.548	65 536
$0.1 \cdot 10^{-6}$*	6m56.231	65 536

*) all itemsets frequent

Table 7: Performance results for supp_F and supp_ρ for the `letter` data.

5.4 Performance of supp_F and supp_ρ

Table 7 shows performance results for supp_F and supp_ρ for the `letters` data. We used the monotone version of the measures (Definition 1).

It can be seen that mining patterns using those measures is much more efficient than for supp_τ. Notice that both measures are quite sensitive to minimum support level chosen. This is especially true for supp_ρ, which moves relatively fast from the extreme point of 1 frequent itemset (the empty set is always frequent) to all itemsets being frequent. Also note that for supp_ρ the values of minimum support required are very low, in the range of tenths of millionth of percent! It shows that the theoretical maximum value is rarely achieved.

6. RELATED WORK

There has been previous work by Han et. al [14] on mining itemsets which span more than one transaction. Their work however used discretization of numerical dimensions, and, they placed restrictions on which pairs of transactions were considered. More specifically only pairs of transactions which are close to each other in one or more numerical dimensions were considered. The semantics of their rules is thus different. While we find that approach useful, it does not solve all the problems we encountered. Algorithms used in [14] cannot be applied to our definition of support.

Another approach to mine numerical data, which does

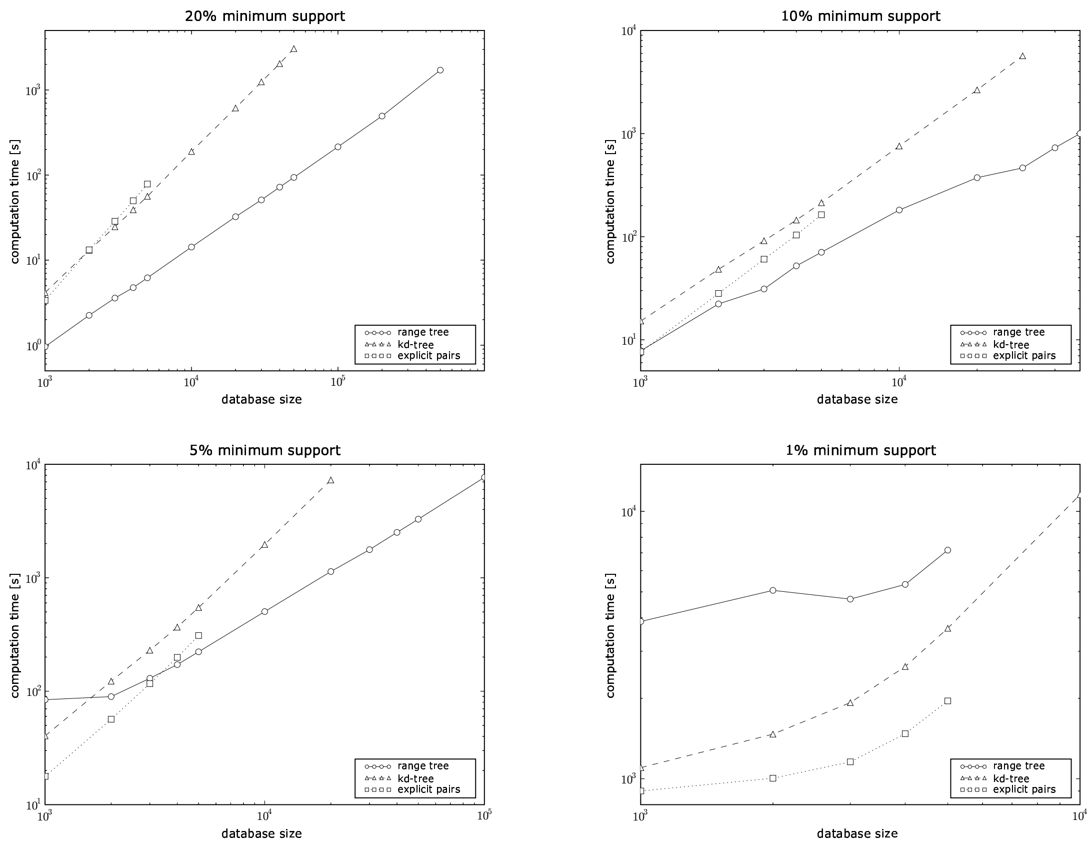


Figure 1: Performance of the supp_τ **mining algorithms for various database sizes and minimum supports for the meteorological data.**

not use discretization can be found in [2, 15]. However the case considered in those works involves discrete attributes in the antecedent and a single numerical attribute in the consequent.

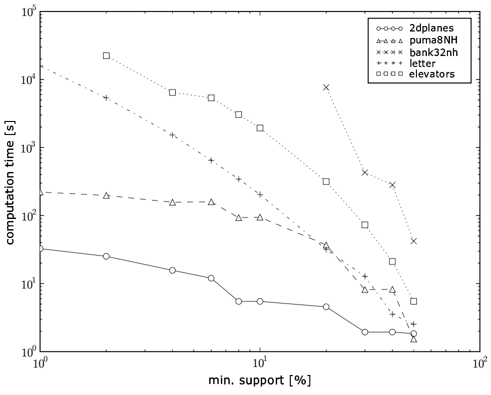
In [17] an interesting definition of support for continuous data is presented which does not require discretization. It is defined as the sum over all transactions of the squared minimum over all itemset's attributes in each transaction. Although a relationship between the support definition and the cosine similarity of pairs of attributes as well as a relationship with the notion of *h-confidence* is given, the interpretation of the definition is not clear. Also, the semantics of the definition given in [17] is clearly different from our approach. Also ranks and relations to statistical rank methods is unique to our paper. Note that after converting to ranks, F and ρ are expressible in the framework presented in [17] as σ_{range}, L_1 and σ_{range}, L_2 respectively. The original paper however does not discuss conversion to ranks and relation Spearman's ρ and Footrule. Also Dzeroski and Todorovski [7] propose an interesting approach to mining numerical data that does not rely on discretization. But, to the best of our knowledge, no previous work exists that uses quantities related to Kendall's τ as measures of support.

7. CONCLUSIONS

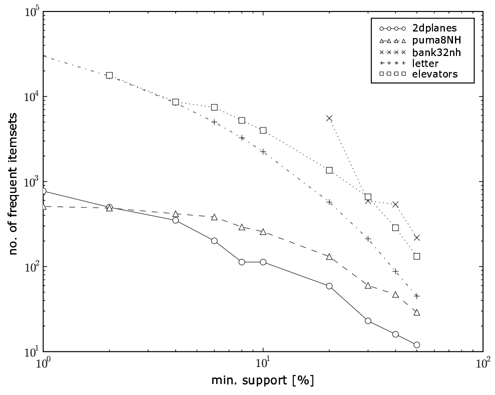
In this paper we presented three new support definitions for continuous attributes based on rank methods used in statistics. Those methods do not require discretization of numerical data. Relationships between the measures have been analyzed theoretically. Algorithms have been presented for mining frequent itemsets based on those definitions. Tests were performed on benchmark datasets as well as on a real-world meteorological data. The efficiency of the algorithm has been verified even on large databases.

An important direction for future work is improving even further the performance of the algorithms involving supp_τ. The first idea is to use a hybrid strategy which switches between various mining algorithms depending on the size of the dataset and the available memory. For example, an algorithm might decide to switch away from kd-tree or range tree and generate explicit pairs of records at a lower lever of the recursion tree, when many records have already been pruned and there are only few attributes involved.

Another optimization is possible for ordinal attributes with small domains, where support could be obtained from their contingency tables. For example, suppose we have two attributes x, y with domains $\{0, 1, 2\}$. Let n_{ij} denote the number of records with $x = i$ and $y = i$. Support of xy can now be computed as $n_{00}n_{11} + n_{00}n_{21} + n_{00}n_{12} + n_{00}n_{22} + n_{01}n_{12} + n_{01}n_{22} + n_{10}n_{21} + n_{10}n_{22} + n_{11}n_{22}$, which can be



a)



b)

Figure 2: Computation time (a) and number of frequent itemsets (b) for various datasets and levels of minimum support (supp_τ and range tree implementation used)

done with just a simple scan of the data. It is not clear how much performance improvement this approach would yield for larger itemsets with larger domains.

An ultimate approach, as far as performance goes would be sampling. Since based on relatively small samples, statistical inferences can be made even about infinite populations, it would remove the database size limitation completely. Sampling itself is relatively easy to implement; we would pick two records at random, and check which attributes of the first record are less than attributes of the second, thus obtaining a single sample from the database currently built in the explicit_pairs algorithm. A more difficult task is guaranteeing accuracy of the solution obtained. Methods presented in [11] can be used.

Currently we only consider support definitions where all items in one record have to be strictly less than all items in the other record. It would be interesting to consider cases where we require certain items in the first record to be less and others to be greater than (or equal) corresponding items in the other record. This way we could mine patterns where a decrease in value of one attribute causes an increase in the value of another.

Acknowledgements

The authors would like to thank Heikki Mannila for pointing out the relationship of our support measure to rank aggregation methods, and Geert Barentsen for his help with the experiments on the meteorological dataset.

Toon Calders is funded by the Fund for Scientific Research - Flanders (FWO-Vlaanderen) as a post-doctoral fellow. This work has been partially funded by the EU contract IQ FP6-516169.

8. REFERENCES

[1] R. Agrawal, T. Imielinski, and A.N. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD*, pages 207–216, 1993.

[2] Y. Aumann and Y. Lindell. A statistical theory for quantitative association rules. In *Proc. ACM SIGKDD*, pages 261–270, 1999.

[3] J. L. Bentley and J H. Friedman. Data structures for range searching. *ACM Comput. Surv.*, 11(4):397–409, 1979.

[4] F. Bodon and L. Schmidt-Thieme. The relation of closed itemset mining, complete pruning strategies and item ordering in apriori-based fim algorithms. In *Proc. PKDD*, pages 437 – 444, 2005.

[5] T. Calders and B. Goethals. Depth-first non-derivable itemset mining. In *Proc. SIAM DM*, 2005.

[6] P. Diaconis and R. Graham. Spearman's footrule as a measure of disarray. *J. of the Royal Statistical Society, Series B*, 39(2):262–268, 1977.

[7] S. Dzeroski and L. Todorovski. Discovering dynamics: From inductive logic programming to machine discovery. *Journal of Intelligent Information Systems*, 4:89–108, 1995.

[8] P. W. Frey and D. J. Slate. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2), 1991.

[9] B. Goethals. Frequent set mining. In *In The Data Mining and Knowledge Discovery Handbook*, chapter 17, pages 377–397. 2005.

[10] S. Hettich and S. D. Bay. The UCI KDD Archive [http://kdd.ics.uci.edu] Irvine, CA: University of California, Dept. of inf. and comp. science, 1999.

[11] S. Jaroszewicz and T. Scheffer. Fast discovery of unexpected patterns in data, relative to a bayesian network. In *Proc. ACM SIGKDD*, pages 118–127, 2005.

[12] M. Kendall. *Rank Correlation Methods*. Oxford University Press, 1990.

[13] A.M. Liebetrau. *Measures of Association*, volume 32 of *Quantitative Applications in the Social Sciences*. Sage Publications, Inc., 1983.

[14] H. Lu, L. Feng, and J. Han. Beyond intra-transaction association analysis: Mining multi-dimensional inter-transaction association rules. *ACM Transactions on Information Systems*, 18(4):423–454, 2000.

[15] G.I. Webb S. Huang. Discarding insignificant rules during impact rule discovery in large databases. In *Proc. SIAM DM*, pages 541–545, 2005.

[16] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM SIGMOD*, 1996.

[17] M. Steinbach, Tan P.-N., Xiong H., and V. Kumar. Generalizing the notion of support. In *Proc. ACM SIGKDD*, pages 689–694, 2004.