Proc. ACM Symposium on Applied Computing (SAC 2004),
March 14-17, 2004, Nicosia, Cyprus, 525-529: http://portal.acm.org/citation.cfm?id=968011

Mining Dependence Rules by Finding Largest Itemset Support Quota

Alexandr Savinov
Fraunhofer Institute for Autonomous Intelligent Systems
Schloss Birlinghoven, Sankt-Augustin
D-53754 Germany
savinov@ais.fraunhofer.de

ABSTRACT

In the paper a new data mining algorithm for finding the most interesting dependence rules is described. Dependence rules are derived from the itemsets with support significantly different from its expected value and therefore considered interesting. Since such itemsets are distributed non-monotonically in the lattice of all itemsets the support monotonicity property cannot be used for their search. Instead we estimate upper/lower bounds for the support to find itemsets with large interval of possible support values called support quota. Since the support quota is known to be monotonically decreasing the search space can be effectively restricted. Strongly dependent itemsets are selected by computing their expected support using iterative proportional fitting algorithm and comparing it with the real itemset support.

Keywords

Data Mining, Dependence Rules, Support Bounding, Support Quota, Expected Support

1. INTRODUCTION

Rules are semantic constructions having the form IF X THEN Y where X restricts objects by means of some logical condition while Y is some (interesting) semantic statement about these objects. For example, a rule might say that “IF SEX is Male AND AGE is Young AND ACTIVITY is Low THEN PAYMENT_METHOD is Cash (55%), DebitCard (15%), CreditCard (10%) or OtherMethods (20%)”. This rule predicts probability of payment method for the selected subgroup of customers. There exists a huge number of syntactically valid antecedents and for each of them we can generate the corresponding consequent from the dataset as probabilities of the target values under the specified constraints. So the crucial question is whether a rule is interesting or not (see, e.g., [1]). In particular, what is interesting in the above rule? Is it interesting that payment in cash is relatively high (high confidence [2]) or payment with credit card is lower (hole in data [3,4,5])? There exists no one ultimate answer for this question and different rule induction methods provide their own rule interestingness measures and their interpretations.

In association rule mining interestingness is based on two parameters: support and confidence. Support characterizes rule generality and is equal to the percentage of objects covered by the corresponding itemset. Confidence characterizes the rule surpriseness and is equal to the percentage of objects satisfying consequent among those satisfying the antecedent. Here very high values of confidence are assumed to be surprising or unexpected. The main problem of such a support-confidence framework is that rule confidence does not reflect correctly what is meant by surpriseness or unusualness because this parameter relates to the only rule and does not take into account other available rules. For example, a rule might say that the probability to buy beer is 99% if chips have been bought and it is regarded as an extremely surprising fact and hence very interesting association rule. However it may well happen that the default (unconditioned) probability of buying beer is 95% so taking this into account the above rule produces almost no surprise and thus is regarded as non-interesting. Thus confidence can be considered as a factor of surprise only in rare situations where the itemset does not inherit information from its subsets while in general case we need to provide more comprehensive framework, which could take into account more complex *dependencies*.

The corresponding critique of the conventional association rule mining framework has been presented in [6,7] where the notion of dependence rule has been proposed to overcome its shortcomings. In particular, in order to reveal highly dependent itemsets characterized by their *own* high interestingness rather than inheriting it from its subsets the chi-square test for independence has been applied. This parameter has been proven to be monotonically decreasing with the itemset size. This allows for organizing an efficient search for highly dependent itemsets and hence dependence rules derived from them.

A continuation of this line is presented in [8] in the form of the theory of dependence values where a new model to *evaluate* dependencies is proposed. The idea consists in computing for each itemset its expected support and then comparing it with the real one. Large difference means high surprise and hence interestingness. The crucial moment in comparing expectations with reality is how expected value is calculated. In this model it is calculated by using maximum independence estimate with the help of the iterative proportional fitting algorithm.

This approach however does not provide a search method, which could efficiently find itemsets characterized by high dependence so it can be used as an additional evaluation technique (like confidence) applied after the frequent itemset search. The main problem here is that highly dependent itemsets are distributed non-monotonically so essentially they can be found anywhere in the lattice. In other words, if we know dependence value for an itemset then it says absolutely nothing about the possibility to have dependencies in its supersets. In this paper we present one solution to this problem by describing an algorithm, called Quota, which is able to directly search for dependent itemsets even if no other criteria are provided. Thus instead of generating a huge number of all frequent itemsets and then evaluating their dependence value this algorithm effectively predicts directions where dependent itemsets can reside and cuts off all branches, which are guaranteed to have no dependent itemsets.

The idea of the Quota algorithm is that each itemset is characterized by real support determined from data and expected support derived from its subsets. These values are restricted by lower and upper bounds, which are also computed from all the subsets. The size of this interval of possible support values allocated for each itemset is said to be *support quota*. The dependence value is defined as difference between real and expected support and hence its absolute value is always less than or equal to support quota. In other words, if support quota is too small then the itemset cannot have high dependence value. For example, if an itemset is known to take its support between 2% and 5% then its dependence must be less than the support quota 3%. Support quota can be derived from known supports of subsets using different support bounding techniques, e.g., described in [9]. For the algorithm it is important that support quota is known to monotonically decrease with the rank, e.g., for a 4-itemset it is less than for any its parent 3-itemset. Thus if an itemset support quota has been found too small then all its supersets are known in advance to have it even smaller and hence no dependent itemsets can be found in this branch. Thus the kernel of the algorithm finds itemsets with high support quota because only such itemsets can be dependent. Informally, support quota for dependence value is like support for confidence in the conventional association rule mining. We search for high support quota itemsets just like we do it for frequent itemsets and then select among them those with large difference between real and expected supports.

Normally only a small portion of high support quota itemsets are really dependent. To find them we compare their real support with the expected one. In addition other secondary measures of interestingness can be applied. Finally the dependence rules are built by inverting some items. The generated dependence rules are guaranteed to be highly informative in the sense that their information is not contained in and cannot be derived from other rules. Such rules may have any confidence for the target items starting from 0 and ending with 100% — the main thing is that this probability is significantly different from what is expected. For example, a rule might say that the probability to buy beer under some conditions is 50% and it is extremely surprising since from all other rules 95% is expected.

The expected value can be calculated from very different principles but the most fundamental one supposes that complete independence of items means maximum entropy of the corresponding distribution. Unfortunately currently there is no closed formula for calculating such an expected probability and we use iterative proportional fitting algorithm [10], which can find this estimate at any desired level of accuracy. It is worth noticing that similar ideas are used by log-linear methods in statistics where the degree of dependence associated with a set of variables is called an interaction term. Yet in statistics the main task is how to calculate such interaction terms for a set of (normally 2 or 3) variables while in data mining the main problem is how to efficiently organise the search.

2. DEPENDENT ITEMSETS

Dependence rules are derived from itemsets characterized by strong dependence value by choosing some target item so the task of dependence rule mining can be reduced to finding such dependent itemsets. Informally dependence value associated with each itemset represents new information belonging to this and only this combination of items. Having such coefficients for all itemsets means complete representation of the dataset semantics, i.e., we can precisely reproduce the underlying probability distribution or each itemset support. If we have these coefficients only for a subset of all itemsets then some information is lost. Thus in order to select only informative itemsets we need to define their degree of dependence in such a way that it reflects exclusively own information of an itemset that cannot be derived (expected) from its subsets. The goal of data mining in this context can be defined as *simplifying* representation of data semantics by finding a *small* number of patterns representing *most* of information in the dataset. For example, we might find 1% of itemsets explaining 99% of dependencies in a dataset. The use of dependence values (rather than other measures of itnerestingness) allows us to avoid itemsets with semantics (support) derived from its subsets. Thus the rules produced from such dependent itemsets are highly informative because their prediction cannot be derived and hence is highly unexpected.

The main problem here is how to measure itemset's own information, i.e., how to separate information that can be derived from subsets and select only what is absolutely unique in it. For two variables the answer is well known and consists in calculating their correlation (normalized covariance), which effectively removes influence of individual items from the pair. Obviously correlation is much better solution than association rules for measuring *dependence* between buying chips and beer because now it does not depend on probabilities of individual items. Unfortunately there is no such a simple measure for an arbitrary set of items characterizing their own (mutual) correlation. In statistics there exist a number of methods that are considered good for various particular cases. One such method is chi-squared test for independence used in [6,7] to generate dependence rules. This measure of dependence is upward close, i.e., if A and B are dependent then ABC is also dependent. This is not quite natural if we define dependence as itemset's own characteristic. In other words, if A and B are dependent (or independent) then it should say nothing about ABC or A, which may have their own dependence value.

Instead of directly computing degree of dependence like correlation we can use an approach consisting in comparing its expected and real support where expected support is the value derived from all subsets of this itemset. Here the center of the problem is shifted to computing expected value. The most justified approach consists in calculating it from the maximum entropy principle. The idea here is that expected support must maximize entropy of the underlying distribution if supports of all subsets are fixed. In other words, expected support is the only unknown parameter, which can be varied and each its value allows us to reconstruct the probability distribution. We choose among them the distribution with maximum entropy and declare the corresponding support as an expected value because it does not add any new information to the distribution and hence the items are completely (mutually) independent. Yet if the real value differs from what is expected then the itemset is said to have some dependence among its items (but not in their subsets).

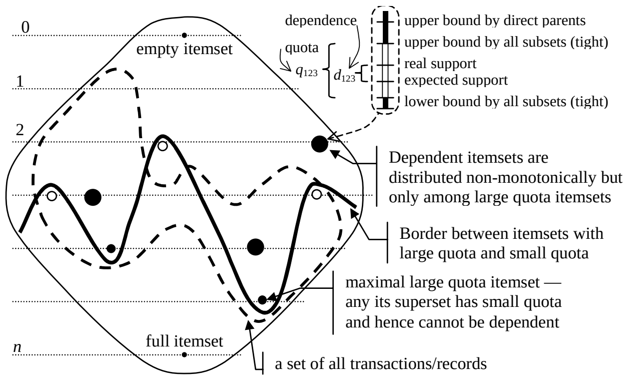

0
1
2
n
empty itemset
full itemset
dependence
quota
q_{123}
d_{123}
upper bound by direct parents
upper bound by all subsets (tight)
real support
expected support
lower bound by all subsets (tight)
Dependent itemsets are distributed non-monotonically but only among large quota itemsets
Border between itemsets with large quota and small quota
maximal large quota itemset — any its superset has small quota and hence cannot be dependent
a set of all transactions/records

Figure 1. Lattice of itemsets.

This idea has been applied to evaluate itemset dependence in [8], It has been also described how this measure of dependence can be used to prune redundant association rules [11]. However currently there is no method to directly search for such dependent itemsets so that this method turns into a filtering or postprocessing technique. The main problem is that dependent itemsets are distributed non-monotonically in the lattice (Fig. 1). In particular, it is quite possible that there are no dependent itemsets up to the level 5 and only itemsets consisting of 6 items possess some information. In the next sections we describe an algorithm, which makes it possible to efficiently search for such nuggets of dependence in huge lattice of itemsets based on predicting an interval of possible support values called support quota.

3. FINDING HIGH SUPPORT QUOTA ITEMSETS

Data semantics is represented as a probability distribution $p(x_1, x_2, \ldots, x_n)$ over the hypercube where each vertex $\langle x_1, x_2, \ldots, x_n \rangle$ corresponds to one database transaction, i.e., one combination of items or one record. Here variables $x_1, x_2, \ldots, x_n$ take values 0 or 1, where 1 means that the corresponding item is present and 0 means its absence. The number of variables taking value 1 is the transaction size, rank or level. For example, the vertex with all zeros corresponds to the empty transaction with no items and $p(0,0,\ldots,0)$ is its probability. $p(1,0,\ldots,0)$ is probability of the transaction consisting of item x_1 and so on. Notice that $p(x_1, x_2, \ldots, x_n)$ is probability of this and only this transaction rather than the itemset support. The probability distribution $p(x_1, x_2, \ldots, x_n)$ is supposed to encode the complete data semantics and theoretically once we have it, further we can calculate any its parameters. The problem however is that in practice the hyper cube is extremely large and it is not possible to represent the semantics explicitly in each point. This is why we use itemset support defined as a coefficient $p_{12\ldots k}$ calculated for each combination of variables as follows:

$$p_{12\ldots k} = \sum x_1 x_2 \ldots x_k \, p(x_1, x_2, \ldots, x_n)\,, \quad x_i \in \{0,1\} \qquad (1)$$

In particular, support of the empty itemset is the sum of probabilities over the whole hypercube and is equal 1 while support of the full itemset is equal to its own probability.

Below we show how lower $\breve{p}_{12\ldots n}$ and upper $\hat{p}_{12\ldots n}$ bounds for the itemset support can be found based on available information about supports of all its subsets. Why itemset support has to be bounded and why it cannot take any value? Assume that we know supports of all subsets of an itemset. If now we learn the support of the itemset itself then we can completely reconstruct the corresponding probability distribution. Depending on the itemset support chosen we will obtain different probability distributions. The main idea underlying the support bounding mechanism is that since the values of the probability distribution are bounded by 0 and 1 the itemset support we vary as a parameter is also bounded by some values. In other words, if the itemset support is too low or too high then the uniquely reconstructed probability distribution may well turn out to be less than 0 or greater than 1 in some points. Formally it is expressed as follows:

$$0 \le p(x_1, x_2, \ldots, x_n) \le 1 \quad \Rightarrow \quad \breve{p}_{12\ldots n} \le p_{12\ldots n} \le \hat{p}_{12\ldots n}\,. \qquad (2)$$

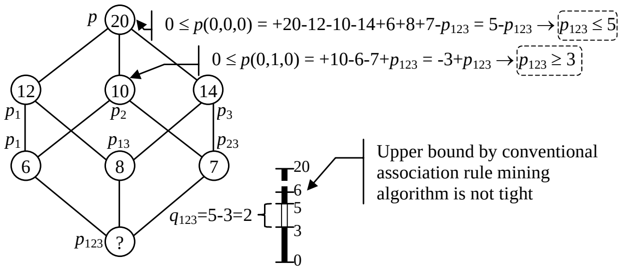
Taking this into account the next task is to reconstruct 2^n values $p(x_1, x_2, \ldots, x_n)$ of probability distribution from its known support values. In the trivial case $p(1,1,\ldots,1) = p_{12\ldots n}$ and it follows from (2) that $0 \le p_{12\ldots n} \le 1$, i.e., support is always within the interval [0,1]. If only one variable takes value 0 then the reconstruction is also simple and we derive that $p(0,1,\ldots,1) = p_{2\ldots n} - p_{12\ldots n}$ from which n additional bounds can be derived: $p_{2\ldots n} - 1 \le p_{12\ldots n} \le p_{2\ldots n}$. It means that support of an itemset is less than or equal to that of any its direct subset. Obviously, it is precisely the main optimisation used in all association rule mining (support counting) algorithms [2].

In general case the idea of reconstructing the probability distribution given its supports consists in finding cross-sum between two points in the lattice:

$$p(0,0,\ldots,0) = \left(\begin{array}{ccc} & p & \\ -\,p_1 & \cdots\cdots\cdots & -\,p_n \\ \vdots & \vdots & \vdots \\ \mp\, p_{2\ldots n} & \cdots\cdots\cdots & \mp\, p_{1\ldots n-1} \\ & \pm\, p_{12\ldots n} & \end{array} \right) \qquad (3)$$

Thus for each candidate itemset for which we are estimating support bounds we need to calculate the cross-sum (3) between each its subset and the candidate itemset $p_{12\ldots n}$. Each such cross-sum is used to calculate new bounds according to (2).

For example, in Fig. 2 we have one candidate itemset of rank 3 with unknown support p_{123}. We need to find its support quota in order to predict (without explicit frequency counting through the dataset) if it can be dependent or not. First, we reconstruct 3 dimensional probability distribution by finding cross-sum of supports between each of 7 subsets and the candidate itemset. For example, for itemset x_2 the cross sum is $p(0,1,0) = p_2 - p_{12} - p_{23} + p_{123} = +10 - 6 - 7 + p_{123}$. Assuming that this value must be between 0 and $p = 20$ we derive using (2) two bounds: $3 \le p_{123} \le 23$. From this constraint we see that the candidate itemset is guaranteed to have support greater than 3. To get the most precise tight bounds we need to repeat this procedure for all subsets. In particular, for the empty set we obtain $p(0,0,0) = p - p_1 - p_2 - p_3 + p_{12} + p_{13} + p_{23} - p_{123} =$ $5 - p_{123}$ and derive the upper bound: $-15 \le p_{123} \le 5$. Finally, we get that the interval of possible support values is [3,5] with the support quota $q_{123} = \hat{p}_{123} - \breve{p}_{123} = 2$.



p
20
0 ≤ p(0,0,0) = +20-12-10-14+6+8+7-p123 = 5-p123 → p123 ≤ 5
0 ≤ p(0,1,0) = +10-6-7+p123 = -3+p123 → p123 ≥ 3
12
10
14
p1
p2
p3
p1
p13
p23
6
8
7
p123
?
q123=5-3=2
20
6
5
3
0
Upper bound by conventional association rule mining algorithm is not tight

Figure 2. Support bounding.

The support quota (2 in our example) allows us to determine in advance if this itemset can be interesting or not because support quota restricts dependence value $d_{12\ldots n}$:

$$|d_{12\ldots n}| = |p_{12\ldots n} - \tilde{p}_{12\ldots n}| \le (\hat{p}_{12\ldots n} - \breve{p}_{12\ldots n}) = q_{12\ldots n} . \quad (4)$$

Here we assume that expected support is always within the interval of possible values what is quite natural but must be guaranteed by the corresponding algorithm. For example, for linear models this condition is not satisfied while for log-linear models it is so.

For our algorithm the most important property of support quota is that it may only decrease with the itemset size. A formal proof of this fact can be found in [9]. Thus if we found support quota for a candidate itemset we can predict not only its own dependence value but also that for all its supersets. For example in Fig. 2 it means that no one superset of the candidate 3-itemset can have quota and dependence greater than 2. Notice here that absolute values of support do not play any role and it is the difference between maximum and minimum possible support that determines if we continue the search in this direction or stop. Intuitively it is clear that support quota drops 2 times faster with the rank, i.e., quota of a direct superset is at least 2 times less than that for its parent. However with no proof we regard it as a hypothesis.

4. MINING DEPENDENCE RULES

The computational kernel of the Quota algorithm implemented in C++ in Windows is aimed at finding itemsets with high support quota and cutting off all itemsets with the support quota less than the specified threshold specified in % as a minimum (absolute) dependence (minDep parameter). For example, minimum dependence 1% means that the quota of each itemset must be at least 1% of the dataset size. From computational point of view the primary distinction of the algorithm is that we use support quota as the main restricting parameter instead of an absolute value of support. Yet the minimum support threshold can be specified as a secondary parameter, e.g., to guarantee statistical significance. Notice however that minimum quota effectively restricts itemset support so that minimum support has to be specified only if it is greater than quota.

The algorithm carries out the conventional level-wise search and its main loop consists of three methods: generate new set of candidates by estimating which of them might be interesting, count frequencies of all existing candidate itemsets from the dataset, prune candidates on the basis of their real parameters. Notice that the quota mining kernel does not need to know expected support of generated itemsets. So essentially we do not need to know if generated itemsets are interesting or not (just like in association rule mining we do not need to know which itemsets produce high confidence rules). However, it is convenient to find expected support and thus dependent itemsets during the search, e.g., to restrict the maximum number of interesting itemsets. Frequency counting and pruning have been implemented in the conventional way. For finding real frequencies we increment each itemset covered by one transaction while pruning is reduced to removing low support itemsets.

The original part of the Quota algorithm is the candidate generation procedure. For each existing itemset it tries to build all its supersets by adding one item (with the number greater than any existing in the itemset). For each new syntactically correct candidate this procedure calculates its lower/upper support bounds. Depending on these bounds the algorithm decides if the itemset has to be built and included into the lattice. In our case only itemsets with high enough support quota and upper bound are built. Notice that even for mining frequent itemsets such an approach is more efficient because we find tight upper bound, which allows the algorithm to prune some branches earlier. For example, 3-itemset in Fig. 2 will be rejected by our algorithm and accepted by Apriori in the case of minimum support 5.5.

Candidate itemset lower/upper bounds estimation is the most difficult procedure since it requires passing through all the subsets and for each of them calculating cross-sum between this and the candidate itemset. Yet this is compensated by early detection of dead branches and the overhead is extremely low for large dataset.

To select interesting itemsets the algorithm needs to calculate each itemset expected support. Expected frequency is calculated from maximum entropy principle, i.e., the distribution $p(x_1, x_2, \ldots, x_n)$ reconstructed from all subset supports and unknown $\tilde{p}_{12\ldots n}$ must have maximum entropy. (One criterion is that its cross-product is 1.) Since there is no closed formula for computing such an expected value for ranks 3 and higher we apply iterative proportional fitting algorithm [10], which sequentially approximates the distribution in all its points decreasing the error on each step until the desired precision is reached. There is a lot of versions of this algorithm with various optimisations, however, we implemented our own simple variant. After each pass over all itemsets in the sublattice it checks the stop condition. In our algorithm we stop the process if the difference between new (proportionally adjusted) and previous support is lower than some threshold. If the obtained expected value is outside the interval of possible supports then we continue iterations with higher precision and if after that it is still outside the interval then we simply set it to the closest interval limit.

Once expected supports have been calculated we can select itemsets with dependence value higher than the minimum dependence threshold. This set of interesting itemsets can be used to generate various descriptive or predictive models including dependence rules. To further restrict the result we can apply additional parameters like maximal rank, coverage (antecedent support), target variable, confidence, lift (the relation of the target conditional probability to the default probability), leverage (covariance between antecedent and consequent).

Table 1. Parameters of datasets used in the experiments.

Dataset	*Attributes*	*Nominal*	*Binary*	*Items*	*Records*
A	497	0	497	497	59602
B	37	18	19	164	216688
C	15	7	8	58	31748
D	9	9	0	32	12960

Table 2. Mean support quota in % for each level (columns) and dataset (rows). Minimum dependence is 0.1%.

Dataset	*2*	*3*	*4*	*5*	*6*	*7*
A	0.35	0.14	0.12	0		
B	5.67	0.72	0.27	?		
C	6.60	1.49	0.49	0.24	0.16	0
D	24.49	6.76	1.97	0.59	0.18	0.12

Table 3. The number of dependent itemsets for each level (columns) and dataset (rows). Minimum dependence is 0.1%.

Dataset	*2*	*3*	*4*	*5*	*6*	*7*
A	1076	368	42	0		
B	5934	8021	105410	?		
C	482	527	611	168	0	
D	122	258	279	14059	11972	0

Table 4. The total number of dependent itemsets for different minimum dependence threshold (columns) and dataset (rows).

Dataset	*0.1%*	*0.2%*	*0.4%*	*0.6%*	*0.8%*	*1%*
A	1983	735	553	520	511	503
B		18084	3491	2122	1616	1232
C	1846	795	421	298	241	210
D	26720	2677	211	163	139	111

Table 5. The number of generated candidate itemsets for each level (columns) and dataset (rows) with the specified minimum dependence threshold. First line in each row — with support bounding, second line — no optimisations (Apriori).

Dataset	*2*	*3*	*4*	*5*	*6*	*7*	*8*
A 0.05%	69751	158529	8035	5	0		
no optim	69751	159211	688836	?			
B 1%	11325	69304	42493	5036	7	0	
no optim	11325	75939	152573	249988	278701	?	
C 0.1%	861	3067	4346	1392	46	0	
no optim	861	3504	7415	6663	3117	663	43
D 0.05%	465	3091	13854	38869	61833	1154	0
no optim	465	3135	14354	41699	72603	5323	0

We applied the Quota algorithm to 4 real world data sets with characteristics specified in Tables 1-4. To check how efficient this algorithm is in comparison to the case where support bounding is not used we count the number of generated candidate itemsets on each level because it is the primary factor of performance. The results are shown in Table 5 for some fixed minimum dependence thresholds. We see that for different datasets we get different performance gains. For datasets A and B the gain is dramatic because the conventional generate-and-test approach simply results in exponential explosion (question mark in tables), e.g., 688836 candidate 4-itemsets against 8035 by our algorithm for dataset A. For datasets C and D the gain is moderate. This is because the efficiency of the described approach depends on how many strong dependencies exists in a dataset. In particular, if a dataset has no dependencies then this algorithm will provide no advantage at all because all support bounds will always be equal to their widest (default) values. On the other hand existing dependencies narrow down these bounds for all their supersets and the stronger the dependencies the smaller support quota allocated for the supersets. Thus the only thing this algorithm does is that it can efficiently and precisely use information provided by itemset support to restrict support of its supersets. For example, if a dataset has dependencies only of rank 8 then this algorithm provides no advantage and generates as many candidates as the conventional Apriori-like algorithm and only when this level is reached its information can be used to predict where dependencies can reside on the next level.

Dependence rules are generated from highly dependent itemsets by calculating the target real and expected confidence, which are guaranteed to be significantly different as well as other parameters like coverage, lift and leverage.

5. CONCLUSION

In the paper a new original algorithm for generating dependence rules has been described. This algorithm effectively searches for only high support quota itemsets using the monotonicity of this parameter. After that it finds the real dependence value of each itemset by comparing its real support with the expected support computed by iterative proportional fitting procedure. It allows us to find dependent itemsets distributed non-monotonically in the lattice. This idea is useful not only for rule induction but can be applied to other areas and other tasks, which will be considered in future work.

6. REFERENCES

[1] A.A. Freitas, On rule interestingness measures, Knowlege Based Systems 12, 309-315, 1999.

[2] R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. Proc. of the ACM SIGMOD Conference on Management of Data, Washington, D.C., May 1993, 207–216.

[3] B. Liu, L.-P. Ku and W. Hsu, Discovering Interesting Holes in Data, Proceedings of Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97), pp. 930-935, August 23-29, 1997, Nagoya, Japan.

[4] A. Savinov, Mining Possibilistic Set-Valued Rules by Generating Prime Disjunctions, Proc. 3rd European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'99), LNCS No. 1704, pp. 536-541.

[5] A. Savinov, Mining Interesting Possibilistic Set-Valued Rules, in: Fuzzy If-Then Rules in Computational Intelligence: Theory and Applications (Eds.: Da Ruan and Etienne E. Kerre), Kluwer, 2000, 107-133.

[6] S. Brin, R. Motwani, and C. Silverstein, Beyond market basket: Generalizing association rules to correlations, SIGMOD'97, pp. 265-276.

[7] C. Silverstein, S. Brin, and R. Motwani, Beyond Market Baskets: Generalizing Association Rules to Dependence Rules, Data Mining and Knowledge Discovery 2(1), 39-68.

[8] R. Meo, Theory of dependence values, ACM Transactions on Database Systems, 25(3), 2000, 380-406.

[9] T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. Proc. 6th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'02), LNCS No. 2431, pp. 74-85.

[10] Darroch and D. Ratchli, Generalized Iterative Scaling for Log-Linear Models, The Annals of Mathematical Statistics, Vol. 43, No. 5, pp. 1470-1480, 1972.

[11] S. Jaroszewicz and D.A. Simovici. Pruning Redundant Association Rules Using Maximum Entropy Principle. Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference, PAKDD'02, 135-147.