Classification Rule Learning with APRIORI-C

Viktor Jovanoski and Nada Lavrač

J. Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
{Viktor.Jovanoski, Nada.Lavrac}@ijs.si

Abstract. Mining of association rules became one of the strongest fields of data mining. This paper presents a classification rule learning algorithm APRIORI-C, upgrading APRIORI to dealing with classification problems, decreasing its memory consumption and time complexity, further decreasing its time-complexity by feature subset selection, and improving the understandability of results by rule post-processing. This step also improved accuracy when dealing with unbalanced class distributions. The algorithm was applied to UCI domains as well as to the COIL challenge data.

1 Introduction

Mining of association rules is a field of data mining that has received a lot of attention in recent years. Its main advantage over most other machine learning techniques is a low number of database passes made when searching the hypothesis space. Its disadvantage is its time complexity.

One of the most famous association rule learning algorithms is APRIORI [3, 4]. The original APRIORI algorithm was extensively studied, adapted to other areas of machine learning and data mining, and successfully applied in many problem domains [19, 5, 15, 1, 14, 2].

An association rule has the following form:

$$X \Rightarrow Y, \quad where X, Y \subseteq I$$

where I is a set of all items, and X and Y are *itemsets*. Let $freq(X)$ denote the number of the transactions that are supersets of itemset X and let N be the number of all the transactions. Then the support of an itemset is defined by

$$Support(X) = \frac{freq(X)}{N}$$

Each rule is associated with its *confidence* and *support*. Confidence of a rule is an estimate of the conditional probability of Y given X: $p(Y|X)$. Support of a rule is an estimate of the probability of itemset $X \cup Y$: $p(XY)$. Confidence and support are computed as follows:

$$Confidence = \frac{freq(X \cup Y)}{freq(X)} \qquad Support = \frac{freq(X \cup Y)}{N}$$

Assuming that the reader has the basic knowledge of APRIORI and of the basic association rule learning techniques, this paper presents our algorithm APRIORI-C, based on the basic APRIORI learning algorithm, whose modifications enable it to be used for classification. The basic APRIORI-C algorithm is described in Section 2.

The idea of using association rules for classification is not new [13]. The main contribution of this paper is that we have succeeded to substantially decrease its memory consumption and time complexity (described in Section 2), to further decrease the algorithm's time-complexity by feature subset selection (described in Section 3.1), and improve the understandability of results by rule post-processing (described in Section 3.2).

The algorithm was applied to UCI domains where it achieved results comparable to algorithms that have been designed for classification purposes. Given a comparable accuracy, the advantage of using APRIORI-C rules is that, unlike in standard classification rule learning, each individual rule generated by APRIORI-C represents a reasonable "chunk" of generated knowledge about the problem domain, since each rule is guaranteed to have high support and confidence (above the user defined threshold), which is a valuable property for knowledge discovery. This property was successfully employed during the *COIL challenge* competition. The resulting rules were graded as very useful for practical situations, despite the average classifier accuracy.

2 Classification rule learning

Association rule learning can be adapted for classification purposes by implementing the following steps:

1. Discretize continuous attributes.
2. For each discrete attribute with N values create N items. Discrete value v_i means that i-th item of this attribute will have value '1' while others will have value '0'. A missing value means that all items have value '0'.
3. Run an association rule learning algorithm.
4. Collect rules whose right-hand side of the rule consists of a single item, representing a value of the target attribute.
5. Use this set of rules to classify the unclassified examples.

2.1 The basic APRIORI-C algorithms

The above modifications are too straightforward. To achieve a more reasonable variant of the association rule learning algorithm adapted for classification purposes, the following optimizations are suggested:

Rule generation Since we are interested only in rules having the target item at their right-hand side, such rules can be created during the search. This way we only need to save all the supported itemsets of sizes k and $k+1$.

This approach proved to be very effective in decreasing memory consumption up to a factor of 10. Notice, however, that this step does not improve the algorithm's time complexity.

Unneeded rules We suppress the generation of a rule if there already exists a rule that is its generalization and whose support and confidence are above the specified thresholds[1]. To prevent the generation of a rule we simply exclude the corresponding itemset from the set of supported *k+1*-itemsets. The savings in time and space complexity are considerable (a factor of 10 or more).

Unneeded items If there exists an item that cannot be found in any itemset together with a target item then no rule containing such an item will ever be created. Therefore we can safely prune the search by excluding all itemsets containing such an item from further consideration.

All the above optimizations assume that we want to find only rules which have a single item, called *target* item, at their right-hand side. Consequently, we have to run the algorithm for each target item, representing each individual class. The algorithm, named APRIORI-C, is shown in Figure 1.

```
for each target item T
    k = 1
    C_1 = set of all 1-itemsets
    check the support of all itemsets in C_1
    delete unsupported itemsets from C_1
    while C_k not empty do
        build all potentially supported k+1-itemsets
        put them into C_{k+1}
        check their actual support
        delete unsupported itemsets from C_{k+1}
        find all rules that have target item T at their right-hand side
        delete all itemsets that were used for constructing rules
        detect unneeded items and delete unneeded itemsets
        k = k + 1
    end while
endfor
```

Fig. 1. The APRIORI-C algorithm.

How do we classify an unclassified example with all the rules found by the algorithm? Firstly, sort rules according to some criteria. Next, go through the list of all rules until the first rule that covers the example is found. Classify the example according to the class at the right-hand side of the rule. If no rule covers this example, mark the example as unclassified.

[1] Note that this modification could not be made possible without the first modification of the algorithm as we wouldn't have had sufficient information to detect such situations.

2.2 Experimental evaluation of the basic APRIORI-C algorithm

Experimental domains APRIORI-C was evaluated on 17 datasets from the UCI ML databases repository at (*ftp://ftp.ics.uci.edu/pub/machine-learning-databases/*), listed in Table 1.

Dataset	Num. of examples	Num. of attributes	Num. of discrete attributes	Num. of continuous attributes	Num. of classes
australian	690	14	8	6	2
balance	625	5	1	4	3
breast-w	699	9	9	0	2
bridges-td	102	7	4	3	2
car	1728	6	6	0	4
diabetes	768	9	1	8	2
echocardiogram	131	6	1	5	2
german	1000	20	13	7	2
glass	214	10	1	9	6
hepatitis	155	19	13	6	2
hypothyroid	3163	25	18	7	2
image	2310	20	1	19	7
iris	150	5	1	4	3
tic-tac-toe	958	9	9	0	2
vote	435	16	16	0	2
waveform	5000	22	1	21	3
wine	178	14	1	13	3

Table 1. Dimensions of test datasets.

Since most of the datasets contain continuous attributes, they had to be discretized. This was done using the K-Means clustering algorithm. This was an arbitrary choice, motivated by [16, 7], whereas the choice of the best discretization algorithm is left for further work.

Results Experiments were performed using the following parameters: *Minimal confidence* was set to 0.9, *Minimal support* to 0.03 and *Maximal rule length* to 5. These setting work well for most of the datasets. However, for some of them the values had to be changed to prevent combinatorial explosion. In such situations the parameter *Minimal support* had to be set to a higher value, e.g. 0.5 or even 0.1. On the other hand, for some datasets the algorithm didn't find any rule and the value of the parameter *Minimal support* was set to 0.01.

The accuracy of the basic APRIORI-C algorithm was tested by 10-fold cross-validation. In Table 2, the accuracy of APRIORI-C is compared with the accuracies of other machine learning algorithms, reported in [18], where all the algorithms were run with their default settings.

Dataset	C4.5	CN2	kNN	Ltree	LBayes	APRIORI-C
australian	0.85	0.83	0.85	**0.87**	0.86	0.86
balance	0.77	0.81	0.80	**0.93**	0.87	0.72
breast-w	0.94	0.94	0.95	0.94	**0.97**	0.92
bridges-td	0.85	0.85	0.85	0.85	**0.88**	0.83
car	0.92	**0.95**	0.94	0.88	0.86	0.85
diabetes	0.74	0.74	0.73	0.75	**0.77**	0.72
echocardiogram	0.65	0.66	0.69	0.64	**0.71**	0.65
german	0.72	0.73	0.70	0.73	**0.75**	0.70
glass	0.70	0.65	0.70	0.68	0.63	**0.77**
hepatitis	0.79	0.80	**0.85**	0.82	**0.85**	0.80
hypothyroid	**0.99**	**0.99**	0.97	**0.99**	0.96	0.95
image	**0.97**	0.86	**0.97**	**0.97**	0.92	0.77
iris	0.95	0.93	0.95	0.97	**0.98**	0.96
tic-tac-toe	0.85	0.98	0.91	0.82	0.70	**0.99**
vote	**0.96**	0.95	0.90	0.95	0.9	**0.96**
waveform	0.76	0.69	0.81	0.85	**0.86**	0.34
wine	0.94	0.93	0.97	0.97	**0.99**	0.88

Table 2. Results of the other algorithms.

It can be observed that APRIORI-C performs poorly on the datasets containing continuous attributes (e.g., datasets *balance*, *waveform* and *wine*). Except for APRIORI-C, all the other algorithms handle continuous attributes by themselves and not in pre-processing. The poor results can be attributed to the suboptimal discretization currently used in APRIORI-C. In datasets with few or no continuous attributes, our algorithm is at least as good as the other algorithms, sometimes even slightly better (e.g., *tic-tac-toe* and *vote*). Notice, however, that the accuracy of each algorithm could substantially improve if the parameters were set to more appropriate values [17].

3 Improvements of APRIORI-C

The most serious problem is the exponential increase in time complexity with regard to the number of attributes. Next, correlations between attributes can seriously affect the performance of the algorithm and the quality of results. To solve this problem, we have employed *feature subset selection*. These methods select a subset of all the available features that should be sufficient for the learner to construct a classifier of a similar accuracy, but in substantially shorter time. The tested variants to pre-processing by feature subset selection are outlined in Section 3.1.

In our improvements to APRIORI-C we have addressed also other problems:

- It is common that in the set of constructed rules, many rule subsets cover almost the same training examples. This problem is called *rule redundancy*.
- The induced classifier is often not understandable and too big to be presented to the user, since it includes too many rules. The user's upper limit

of understanding is a classifier with at most 10-15 rules. Depending on the choice of parameters of the algorithm, the original algorithm often created classifiers containing more than 100 rules, sometimes even more than 500. This problem is called *rule overflooding*.
- The classifier often gives no answer as no rule in the classifier fires. This problem is very serious when dealing with highly unbalanced class distributions.
- When class distribution is unbalanced, the rules classifying to the majority class prevail over rules classifying to the minority class. This causes poor accuracy for the examples of the minority class, which needs to be avoided.

To solve the above problems we have employed rule post-processing to select a subset of induced rules that should have a comparable classification accuracy despite the lower number of rules in the selected rule subset. The tested variants to rule post-processing are outlined in Section 3.2.

3.1 Pre-processing by feature subset selection

Feature subset selection is a pre-processing step in which a small subset of all the available features is selected. By testing four quality measures, proposed in [17, 10–12, 9], it became obvious that feature subset selection needs to be sensitive to the difference in the meaning of attribute values 0 and 1 in association rule learning. In association rule learning, value 1 is favorable for rule construction while value 0 only prohibits the construction of a rule but cannot be used in a constructed rule. The preceding research confirmed that the approaches to feature subset selection that make a distinction between these two values always give better results than the algorithms that are insensitive to this distinction.

The following four approaches to feature subset selection were tested: statistical correlation, odds ratio, and two variants of the RELIEF algorithm.

Statistical correlation The biggest threat for association rule learning algorithm are strong correlations among items, since they cause drastic increase of the number of the supported itemsets. Items were removed by first testing all the pairs of non-target items and removing one of them if the correlation was above the user defined threshold $MaxCorr$. The problem of this algorithm is its complexity ($\mathbf{O}(n^2)$) with respect to the number of attributes, which becomes problematic when analyzing data with several hundreds of items.

Odds Ratio Odds ratio is computed for each feature as follows, for C_1 being the target class, C_2 the union of all the non-target classes and n the number of training examples.

$$OddsRatio(A) = log\frac{odds(A|C_1)}{odds(A|C_2)}, \quad where \;\; odds(A) = \begin{cases} \frac{\frac{1}{n^2}}{1-\frac{1}{n^2}}; & p(A) = 0 \\ \frac{1-\frac{1}{n^2}}{\frac{1}{n^2}}; & p(A) = 1 \\ \frac{p(A)}{1-p(A)}; & p(A) \neq 0 \wedge p(A) \neq 1 \end{cases}$$

All the probabilities are estimated using the relative frequency formula. We sort the attributes in descending order of the computed odds ratio quality estimate and select a number of attributes as specified by the value of the parameter *Ratio of selected attributes*. Given that *Odds ratio* algorithm distinguishes between attribute values 1 and 0, it performs very well in domains where this distinction is crucial for the success of learning [17, 8].

VRELIEF in VRELIEF2 The *RELIEF* algorithm [10] is an excellent algorithm for feature subset selection. In our problem, however, it performs poorly since it makes no distinction between attribute values 1 and 0. Therefore we have modified the distance function to build a new algorithm *VRELIEF*.

The RELIEF algorithm is outlined below, where q is a list of quality estimations, $SampleSize$ is the number of randomly selected examples (selected by the user) and $d_{rx,i}$ is the value of the distance function between examples r and x for attribute A_i.

for each $attributeA_i$ **do** $q[Ai] = 0$
for j = 1 to *Sample size* **do**
    randomly select an example r
    find *nearest hit* t and *nearest miss* s
    for each $attributeA_i$ **do**
        q[A_i]:=q[A_i]+$\frac{d_{rs,i}-d_{rt,i}}{Samplesize}$
sort attributes according to the measure q
select the best attributes

The standard RELIEF algorithm uses the following distance function, where $X_{m,i}$ denotes the value of attribute m for example i.

$$d_{ab,i} = \begin{cases} 0;\ X_{a,i} = X_{b,i} \\ 1;\ X_{a,i} \neq X_{b,i} \end{cases}$$

Since the algorithm makes no distinction between values 0 and 1, we have changed the distance function so that it assigns higher quality estimation to an attribute having value 1 for the examples belonging to the same class. This new algorithm is called VRELIEF. The computation of the distance for training examples, labeled a and b, and values of the tested attribute for these two examples is shown at the left-hand side of Figure 3.

Despite the significant increase of the accuracy, the experimental results show the possibility of additional improvements. Namely, the algorithm often selects attributes that cause unnecessary increase in time complexity. We tried to overcome this deficiency by changing the distance function so that attributes that do not increase accuracy but increase complexity receive lower quality estimation. The new algorithm is called VRELIEF2. The computation of its distance function is shown at the right-hand side of Table 3.

Results Algorithms *Odds Ratio*, *VRELIEF* and *VRELIEF2* were tested with values of the parameter *Ratio of selected attributes* 0.2, 0.4, 0.6 and 0.8. Algorithm *Statistical Correlation* was tested with values of the parameter $MaxCorr$ 0.2, 0.4, 0.6 and 0.8.

Class membership		$value_a/value_b$ 1/1	0/1	1/0	0/0	$value_a/value_b$ 1/1	0/1	1/0	0/0
Class(a)=+	Class(b)=+	1	0	0	-1	2	0	0	-2
Class(a)=+	Class(b)=-	0	-1	1	0	-1	-2	2	0
Class(a)=-	Class(b)=+	0	1	-1	0	-1	2	-2	0
Class(a)=-	Class(b)=-	-1	0	0	1	-2	-1	-1	2

Table 3. Values of the distance function for the VRELIEF algorithm (left) and VRELIEF2 algorithm (right).

The accuracy was tested with 10-fold cross-validation. For each value of the parameter, statistical t-test was computed to confirm whether the best algorithm is significantly better than the other two. The data about the complexity was processed in a similar way to determine the best algorithm.

The summarized results are as follows:

- The *Odds Ratio* algorithm is a safe bet for nearly every value of the *Ratio of selected attributes* parameter. Other algorithms often come close, but never surpass it. The values around 0.4 of the *Ratio of selected attributes* parameter are usually enough to retain the same accuracy.
- The *Statistical correlation* algorithm is often useful for cleaning the dataset as it only reduces the time complexity and retains the quality of the results. The values around 0.5 of the $MaxCorr$ parameter are usually sufficient to drastically decrease the time complexity while retaining the same accuracy. It is also worth noting that this algorithm can easily be used in combination with other algorithms.
- The *VRELIEF* and *VRELIEF2* algorithms produce worse results in terms of accuracy when compared to *Odds Ratio* but they are better at decreasing the time complexity. Also, there is no statistically significant difference in performance between *VRELIEF* and *VRELIEF2* algorithm.

3.2 Post-processing by rule subset selection

As mentioned at the beginning of Section 3 the original algorithm's way of combining rules (i.e., building a classifier) is not always good enough. Let us recall the abovementioned problems of rule redundancy, rule overflooding, incapability of classifying examples and poor accuracy in problems with unbalanced class distribution. Below are some remedies to these problems.

The *default* rule assignment After the algorithm selects the rules to be used in a classifier, there will probably remain some unclassified examples in the set L. Such unclassified examples bring majority of false classifications. The preceding research confirmed that the probability of the false classification is much lower than the probability of the disability to classify an example.

Therefore we want to use the information that is left in the form of unclassified examples. This should help us decide what to do when a classifier does not

give a satisfactory answer. The simple approach is to add a rule at the end of the list of rules in a classifier that will always unconditionally classify an example to the majority class of the unclassified examples. This approach is also used in the CN2 algorithm [6].

Selecting rules: Use N best rules Figure 2 shows the new algorithm that should solve the problem of rule overflooding. As it can be clearly seen, the algorithm uses the covering approach for rule selection. The number of rules to be selected is specified by the user-defined value of parameter N.

The algorithm first selects the best rule (the rule having the highest support), then eliminates all the covered examples, sorts the remaining rules and again selects the best rule. This is repeated until the number of selected rules reaches the value of the parameter N or until there are no more rules to select or until there are no more examples to cover. When either one of these conditions is not met, the algorithm stops and returns the classifier in a form of sorted list of rules. Before that it constructs the default rule.

$C = \{\}$
put all the rules into C
put all the examples into L
sort the rules according to the support and confidence in L
while $\mid C \mid < N$ and $\mid L \mid > 0$ and $\mid T \mid > 0$ **do**
    remove best rule r from T and put it into C
    remove the examples, covered by rule r, from L
end while
build default rule
result $= C$

Fig. 2. The USE N BEST RULES algorithm.

This algorithm solves the first two problems. It also gives classifiers that execute faster. But it remains vulnerable to the fourth problem, the problem of the *unbalanced distribution*.

Selecting rules: Use N best rules for each class The algorithm was updated to always select N rules **for each class** (if so many rules exist for each class). This way the rules for minority class will also find their way into classifier. The algorithm is presented in Figure 3.

Results Each algorithm was tested with several values of parameter N. These numbers are 1, 2, 5, 10, 15 and 20. To avoid unnecessary details we will just list the main observations, confirmed also by statistical tests:

```
C = {}
for each class c_i do
    put all rules for class c_i into T
    put all learning examples into L
    C_i = {}
    while | C_i |< N and | L |> 0 and | T |> 0 do
            sort the rules according to the support and confidence in L
            remove best rule r from T and insert it into C_i
            remove examples, covered by rule r, from L
    end while
    C = C \ C_i
end for
sort C according to the support and confidence on all the learning data
build default rule
result = C
```

Fig. 3. The USE N BEST RULES FOR EACH CLASS algorithm.

- Algorithms USE N BEST RULES and USE N BEST RULES FOR EACH CLASS do not differ in accuracy. Only when there is significant difference in distributions among classes, the USE N BEST RULES FOR EACH CLASS algorithm performs better.
- Both algorithms increase their accuracy significantly when using the *default* rule. This increase gets smaller with increasing value of the parameter N, but it still remains noticeable.
- Both algorithms achieve good results in terms of both accuracy and representation when value of parameter N reaches value 10. Results are comparable to original algorithm.

The results show that the best choice for such algorithm is one of the algorithms USE N BEST RULES and USE N BEST RULES FOR EACH CLASS with the value of parameter N set to 10. It is also preferable to use the *default* rule.

4 COIL challenge

The algorithm was also applied to COIL challenge data. The data described 4000 households that received certain advertisement and around 6% of them responded to it. There were 86 attributes available, all of them discrete. Our group submitted the results of some other algorithms due to slightly higher accuracy.

The submitted models were first tested by their accuracy. There was no apparent correlation between the type of the algorithm used and the resulting accuracy. The classifier that was produced by APRIORI-C would have achieved average accuracy.

Second test was performed by marketing expert, who evaluated different models by their ability to be used in practice. This test favored rule-based models due to their ease of interpretation. Almost all of the rules (or, better yet, conclusions) were also found by our team using APRIORI-C algorithm and some common sense reasoning. Therefore we strongly feel that using APRIORI-C algorithm in practical situations would prove beneficial.

But, as always in data mining, the quality of the result depends on the input data, data-cleaning process, (a variety of) good algorithms and inventiveness of people using them.

5 Conclusions and further work

In this paper we have presented new algorithms from the field of association rule learning that successfully solve the problems of classification. The main portion of work is done by the APRIORI-C algorithm, which is a modified version of the well-known APRIORI algorithm. The changes, described in this paper, make it more suitable for building rules for classification, as they lower its time and space complexity. Despite that the algorithm still exhaustively searches the rule space.

When selecting the attributes for learning, the user can use one of the several tested algorithms. The best candidate is *Odds Ratio*, but other algorithms can also be used in unusual situations. We also have several candidate algorithms for selecting rules to be used in a classifier. The tests favor the USE N BEST RULES algorithm with the values of the parameter N set to 10.

Our future work will primarily concentrate on discretization algorithms. Their poor quality is the main reason for the low accuracy in the domains with many continuous attributes. We will test the existing algorithms and, should such need arise, invent new ones. We personally feel that the preprocessing step of discretization is fairly independent of the learning method used, but the difference between greedy and exhaustive methods could prove the opposite.

Acknowledgements

The work reported in this paper was supported by the Slovenian Ministry of Education, Science and Sport, and the IST-1999-11495 project Data Mining and Decision Support for Business Competitiveness: A European Virtual Enterprise.

References

1. K. Ali, S. Manganaris and R. Shrikant. Partial Classification using Association Rules. In Proceedings of the Third International Conference on Knowledge Discovery and Data Mining KDD 1997, Newport Beach, California.
2. R. Agrawal, J.Gehrke, D. Gunopulos and P. Raghavan. Authomatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In Proceedings of ACM SIGMOD Conference on Management of Data, 1998.

3. R. Agrawal, T. Imielinski and R. Shrikant. Mining Association Rules between Sets of Items in Large Databases. In Proceedings of ACM SIGMOD Conference on Management of Data, Washington, D.C., 1993. pp. 207-216.
4. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In Proc. of the 20th Int'l Conference on Very Large Databases, Santiago, Chile, September 1994. pp. 207-216.
5. R. J. Bayardo Jr., R. Agrawal and D. Gunopulos. Constraint-Based Rule Mining in large, Dense Databases. In Proceedings of the 15th International Conference on Data Engineering, 1999. pp. 188-197.
6. P. Clark and T. Niblett. The CN2 induction algorithm. Machine Learning 3: 261-283, 1989.
7. P. S. Bradley, U. M. Fayyad. Refining Initial Points for K-Means Clustering. In Proceedings of International Conference on Machine Learning ICML'98. pp. 91-99.
8. V. Jovanoski, N. Lavrač. Feature Subset Selection in Association Rules Learning Systems. In Proceedings of Slovenian Electrical and Computer Science Conference ERK'99, 1999. pp. 301-304.
9. D. Koller, M. Sahami. Toward Optimal Feature Selection. In ICML-96: Proceedings of the Thirteenth International Conference on Machine Learning, pp. 284-292, San Francisco, CA: Morgan Kaufmann.
10. I. Kononenko. On Biases in Estimating Multi-Valued Attributes. In Proceedings of the 14th International Joint Conference on Artificial Intelligence IJCAI-95, pp. 1034-1040, 1995.
11. I. Kononenko, S. J. Hong. Attribute Selection for Modelling. In Future Generation Computer Systems 13 (1997) 181-195, 1997.
12. I. Kononenko. Estimating Attributes: Analysis and Extensions of RELIEF. In De Raedt, L. and Bergadano, F., editors, Machine Learning: ECML-94, pages 171-182. Springer Verlag.
13. B. Liu, W. Hsu and Y. Ma. Integrating Classification and Association Rule Mining. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining KDD'98, New York, USA, 1998.
14. H. Mannila and H. Toivonen. Discovering Generalized Episodes using Minimal Occurences, In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining KDD96, 1996.
15. N. Megiddo and R. Shrikant. Discovering Predictive Association Rules. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining KDD'98, New York, USA, 1998.
16. M. Meila, D. Heckerman. An Experimantal Comparison of Several Clustering and Initialisation Methods. Technical report MSR-TR-98-06, Microsoft research, 1998.
17. D. Mladenić. Feature Subset Selection in Text-Learning. In Proceedings of 10th European Conference on Machine Learning ECML'98. pp. 121-129.
18. L. Todorovski and S. Džeroski. Combining Multiple Models with Meta Decision Trees. In Proc. ECML2000 Workshop on Meta-learning: Building Automatic Advice Strategies for Model Selection.
19. M. J. Zaki, S. Parthasarathy, M. Ogihara and W. Li. New Algorithms for Fast Discovery of Association Rules. Technical Report 651, The University of Rochester, Computer Science Department, Rochester, New York, 1997.