PREDICTIVE DATA MINING USING ITEMSET FREQUENCIES

Panov Panče[1], Sašo Džeroski[1], Hendrik Blockeel[2], Suzana Loškovska[3]
[1]Department of Knowledge Technologies, Jozef Stefan Institute, Ljubljana, Slovenia
[2] Department of Computer Science, Katholieke Universiteit Leuven, Heverlee, Belgium
[3] Department of Computer Science, Faculty of Electrical Engineering, Skopje, Macedonia
e-mail: {Pance.Panov,Saso.Dzeroski}@ijs.si,
Hendrik.Blockeel@cs.kuleuven.ac.be,
suze@etf.ukim.edu.mk

ABSTRACT

Predictive models make predictions about values of data using known results from different data, while frequent itemsets describe properties of a subset of the data and are descriptive in nature. In this paper we present a method of building predictive models by using frequency information from frequent itemsets. Modifications were done on three standard algorithms for learning probability models, decision trees and rules. Implementation was done in the WEKA system. We present preliminary results on two datasets and discuss further work to be done in this area.

1 INTRODUCTION

Predictive models make predictions about values of data using known results found from different data.
Frequent sets play an essential role in many data mining tasks that try to find interesting patterns from databases.
The difference between frequent itemsets and models is that frequent itemsets are local in that they typically describe properties of a subset of the data, whereas models are global in that they characterize the entire dataset. In addition, frequent itemsets are typically used for descriptive purposes and models for predictive ones.
Once interesting frequent itemsets have been identified, they can be used as descriptors for building models.
Itemset frequencies can be used for many more things than just for mining association rules. Once the work of finding and storing them is done, we can build any kind of model (in context of inductive queries) quickly.
In this paper, we present a new way of building predictive models by using frequency information from frequent itemsets, instead of dataset itself. Frequency information from the itemsets is used at different places of the machine learning algorithms to calculate the probability distributions which are used to build predictive models.

2 OVERVIEW OF PREDICTIVE MODELS

2.1 Probabilistic models

Probabilistic models describe probabilistic dependencies among variables. One such model is the naive Bayes classifier. Bayesian classification is based on the Bayes theorem. This theorem is used to estimate the probability of an example belonging to each of the possible classes in a classification problem.
An example $X = (x_1, ..., x_n)$ from the dataset is represented as a conjunction of conditions $(A_i = x_i)$ for each of the n attributes A_i. A Bayesian classifier will assign to a new example the class value c_k that maximizes $P(C_k|X)$, i.e., $P(C_k|X) \geq P(C_j|X), j = 1, ..., m$. According to the Bayes theorem $P(C_j|X) = P(X|C_j)P(C_j)/P(X)$.
The key assumption of the naive Bayesian classifier is the assumption of class conditional independence. This allows the term $P(X|C_j)$ to be replaced by the product $\prod_i P(A_i = x_i|C_j)$. Individual conditional probabilities $P(A_i = x_i|C_j)$ are easy to estimate from the training data.
Learning a naive Bayes classifier consists of estimating the prior probabilities $P(C_j) = P(C = c_j)$ and $P(A_i = v_{ik} \mid C_k)$ for each of the possible values c_j for the class C and each of the attribute values v_{ik} of each attribute A_i. $P(C_j)$ is estimated by counting the number of examples $n_j = N(c_j)$ of class c_j and dividing this by n, the total number of training examples, i.e. $P(C_j) = n_j / n$. $P(A_i = v_{ik} \mid C_k)$ can be estimated as $N(A_i = v_{ik} \wedge C = c_j)$ divided by $N(C = c_j)$.

2.2 Decision trees

Decision trees are hierarchical structures, where each internal node contains a test on an attribute, each branch corresponds to an outcome of the test, and each leaf node gives a prediction for the value of the class variable.
Finding the smallest decision tree that would fit a given dataset is computationally expensive so heuristic search is thus employed to build decision trees. Tree constructing proceeds recursively starting with the entire set of training

examples. At each step, an attribute is selected as the root of the subtree and the current training set is split into subsets according to the values of the selected attribute.
Tree construction stops when the examples in a node are sufficiently pure if some other stopping criterion is satisfied. Different measures can be used to select an attribute in the attribute selection step. Quinlan [4] uses information gain, which is the expected reduction in the entropy of the class value caused by knowing the value of a given attribute.
An important mechanism used to prevent trees from over-fitting data is tree pruning. Pruning can be employed during tree construction (pre-pruning) or after the tree has been constructed (post-pruning). The minimum number of examples in branches can be prescribed for pre-pruning and a confidence level in accuracy estimates for leaves for post- pruning.

2.3 Predictive rules

Predictive rules are a popular alternative to decision trees. We will use the word rule here to denote patterns of the form "IF antecedent THEN consequent". The antecedent, or precondition, of a rule is a series of tests on attributes just like the tests on nodes in decision trees, and the consequent, or a conclusion, gives the class or classes that apply to instances covered by that rule. Generally, the preconditions are logically ANDed and all tests must succeed in order the rule to fire.
For a classification problem with several class values, a set of rules is constructed for each class. When rules for class c_i are constructed, examples of this class are referred to as positive, and examples from all the other classes as negative. The algorithm for constructing rules works as follows. We first construct a rule that correctly classifies some examples. We then remove the positive examples covered by the rule from the training set and repeat the process until no more examples remain. To construct a single rule that classifies examples into class c_i, we start with a rule with an empty antecedent an the selected class c_i as a consequent. The antecedent of this rule is satisfied by all examples in the training set, and not only those of the selected class. We than progressively refine the antecedent by adding conditions to it, until only examples of class c_i satisfy the antecedent. To allow for handling imperfect data, we may construct a set of rules which is imprecise, i.e. does not classify all examples in the training set correctly.

3 FREQUENT ITEMSETS

Given a set *R*, a *0/1* relation *r* over *R* is a collection (multiset) of subsets of *R*. The elements of *R* are called items, and the elements of *r* are called rows/transaction. The number of rows in *r* is denoted by $|r|$.

Let $X \subseteq R$ be a set of items. The set *X* matches a row $t \in r$, if $X \subseteq t$. The set of rows in *r* matched by *X* is denoted by $M(X,r)$, i.e., $M(X,r) = \{t \in r | X \subseteq t\}$, also called the cover of *X*. The frequency of *X* in *r*, denoted by $fr(X,r)$ is $\frac{|M(X,r)|}{|r|}$. Given a frequency threshold $min\_fr \in [0,1]$, the set *X* is frequent if $fr(X,r) \geq min\_fr$.

4 BUILDING PREDICTIVE MODELS USING ITEMSET FREQUENCIES

All of the above described algorithms for building predictive models use probability estimates based on counts. Here we present a new way of estimating probabilities by using frequency information from frequent itemsets. To build the models we first have to generate the frequent itemsets with a given *min_fr* threshold. For the calculation of probability estimates we need collection of frequent itemsets containing only combinations of attribute values and itemsets containing combinations of attribute values along with a class value. The size of the collections of frequent itemsets being used depends on the algorithm for building the predictive models.
A problem arises when we need a frequency of an itemset *X* that is infrequent i.e. $fr(X,r) \leq min\_fr$. In this case we only know that the frequency of this itemset is lower than minimal frequency and we have to guess the frequency of the itemset to get the probability distributions.

4.1 Modifications of Naive Bayes

In section 2.1 we described the original algorithm for learning a naive Bayes classifier that uses counts to estimate the probabilities. The modification of this algorithm includes the change of counts with frequencies of itemsets of size 1 and 2. We need itemsets of size 2 that contain only pairs of attribute value and class value i.e. $A_i = v_{ik} \wedge C = c_j$, for all attributes $A_i, i = 1,...,n$ and all classes $c_j, j = 1,...,m$, and itemsets of size 1 containing class values. The conditional probabilities can be calculated using equation 4.1.

$$P(A_i = v_{ik} \mid C_k) = \frac{fr(A_i = v_{ik} \wedge C = c_j)}{fr(C = c_j)} \quad \text{(Eq. 4.1)}$$

If we need frequencies of itemsets that are infrequent we estimate the frequency as *min_fr/2*. As we can see the estimate depends on the value of minimal frequency. If the threshold is too large, the number of infrequent itemsets is increasing and the probability distribution becomes more uniform. For a specific value of the threshold the model becomes invariant of the changes in the minimal frequency and probabilities are uniformly distributed.

4.2 Modification of the algorithm for inducting decision trees

In section 2.2 we described the algorithm for induction of decision trees using information theory heuristics. The modification of this algorithm includes the usage of frequent itemsets to estimate the probability distributions for all nodes of the tree. The probability distributions are needed to calculate information gain and gain ratio parameters for choosing the appropriate attribute to split the tree. In this case we need itemsets of size 1 to size h+1, where h is the height of the tree.

Let $A_i = v_{ik}, ..., A_l = v_{lm}$ be a path of attribute tests from the root to one node of the tree, and let q be the number of attributes being tested. To calculate the probability distribution of classes for a specific node of the tree we have to know the frequency of the itemsets of size q containing combination of attributes $fr(A_i = v_{ik} \wedge ... \wedge A_l = v_{lm})$ and the frequency of itemsets of size q+1 containing combination of attributes including a class value $fr(A_i = v_{ik} \wedge ... \wedge A_l = v_{lm} \wedge C = c_k)$ for all class values c_k. According to the former, the probabilities for a specific node can be calculated using equation 4.2.

$$P(C = c_k \mid A_i = v_{ij} \wedge ... \wedge A_l = v_{lm}) = \frac{fr(A_i = v_{ij} \wedge ... \wedge A_l = v_{lm} \wedge C = c_k)}{fr(A_i = v_{ij} \wedge ... \wedge A_l = v_{lm})} \quad \text{(Eq.4.2)}$$

If we need frequencies of itemsets that are infrequent we estimate frequency as $min\_fr/(2 \cdot s)$, where s is the size of the itemset. As we can see, the estimate of frequency of infrequent itemsets decreases as the size of the itemsets becomes higher.

4.3 Modification of the algorithm for learning predictive rules

In section 2.3 we described the original algorithm for learning predictive rules that uses counts to estimate the accuracy of the rules being built. The modification of the algorithm includes the use of frequent itemsets to estimate the accuracy of all rules. In this case we need itemsets of size 1 to size h+1, where h is the number of attribute tests used in the antecedent of the rule..

Let $A_i = v_{ik} \wedge ... \wedge A_l = v_{lm}$ be an antecedent of the rule and let q be the number of attributes being tested in the antecedent. If the consequent of one rule is determined, i.e., $C = c_k$, to calculate the accuracy of the rule we need to know the frequency of the itemsets of size q containing combination of attributes $fr(A_i = v_{ik} \wedge ... \wedge A_l = v_{lm})$ and the frequency of itemsets of size q+1 containing combination of attributes including a class value $fr(A_i = v_{ik} \wedge ... \wedge A_l = v_{lm} \wedge C = c_k)$. According to the former, the accuracy for a specific rule can be calculated using equation 4.3.

$$Accuracy = \frac{fr(A_i = v_{ij} \wedge ... \wedge A_l = v_{lm} \wedge C = c_k)}{fr(A_i = v_{ij} \wedge ... \wedge A_l = v_{lm})} \quad \text{(Eq.4.3)}$$

If we need frequencies of itemsets that are infrequent we estimate frequency as $min\_fr/(2 \cdot s)$, where s is the size of the itemset.

5 IMPLEMENTATION

All of the above described algorithms and their modifications were implemented in the WEKA system. WEKA provides implementations of a wide range of learning algorithms that can easily be applied to any dataset. It also includes a variety of tools for pre-processing and post-processing the data. In most data mining applications, the machine learning component is just a small part of a far larger software system. If the intention is to write a new data mining algorithm we can access the programs in WEKA from inside of our own code and we can solve the machine learning subproblem with a minimum of additional programming. This was the reason why we used this system for testing the modifications of the standard algorithms for machine learning.

For frequent itemset generation we used the Apriori algorithm, that was already implemented in the WEKA system. The problem with this implementation is that it is limited to small datasets because of its inefficiency handling large number of attributes and large memory usage. Further work will include the use of faster and more efficient algorithms and the use of condensed representations of itemsets so that we could handle large datasets.

Modifications on predictive models were done on three algorithms: Naive Bayes, J48 - JAVA implementation of C 4.5 (release 8) and JRIP - JAVA implementation of RIPPER.

6 PRELIMINARY RESULTS

For evaluation purposes we used only nominal datasets to avoid the process of attribute discretization. In this preliminary phase we generated and compared only unpruned trees and rules. However, pruning can be used in J48, but it was not tested in this implementation. There is a problem with RIPPER pruning of rules because of the re-sampling of examples from the training set. If we implement this pruning, our algorithms would become inefficient because in every pruning phase we would have to generate new frequent itemsets.

Evaluation of the algorithms was done by using 10-fold cross validation on two datasets (more are in progress) from the UCI Repository.

We tested the algorithms for different values of minimal frequency of the frequent itemsets. The results of the evaluations are presented in tables 1 and 2.

Min freq.	Accuracy Naïve Bayes	Accuracy J48	Size of tree	Number of leaves	Accuracy JRIP	Number of rules
Unpruned	0,9011	0,963218	19	37	0,954023	10
Pruned	/	0,91326	6	10	0.954022	4
0,1	0,8920	0,956322	25	49	0,921839	4
0,15	0,8920	0,954023	22	43	0,873563	3
0,2	0,8828	0,905747	19	37	0,852874	2
0,25	0,8851	0,931034	9	17	0,88046	3
0,3	0,8805	0,845977	4	7	0,896552	3
0,4	0,8805	0,82069	2	3	0,845977	2
0,5	0,8805	0,613793	1	1	0,567816	1

Table 1. Results on Vote dataset

Min freq.	Accuracy Naïve Bayes	Accuracy J48	Size of tree	Number of leaves	Accuracy JRIP	Number of rules
Unpruned	0,7168	0,695804	152	179	0,734266	5
Pruned	/	0,755245	4	6	0,70979	3
0,02	0,7203	0,70979	38	48	0,744755	5
0,05	0,7308	0,723776	46	89	0,748252	2
0,08	0,7343	0,716783	35	47	0,702797	1
0,1	0,7448	0,706294	20	29	0,702797	1
0,15	0,7517	0,706294	7	11	0,702797	1
0,2	0,7308	0,723776	5	8	0,702797	1
0,25	0,7308	0,723776	5	8	0,702797	1
0,3	0,734266	0,723776	3	5	0,702797	1
0,35	0,716783	0,716783	3	5	0,702797	1
0,4	0,713287	0,702797	1	1	0,702797	1

Table 2. Results on Breast Cancer dataset

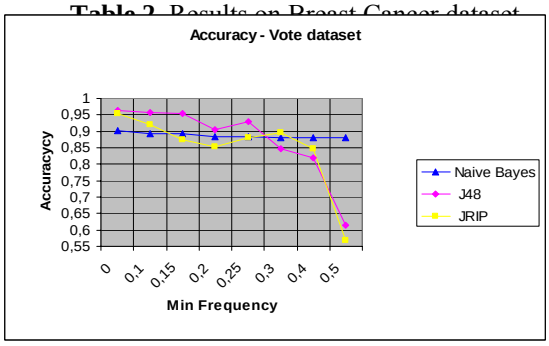

Figure 1. Accuracy comparison for Vote dataset

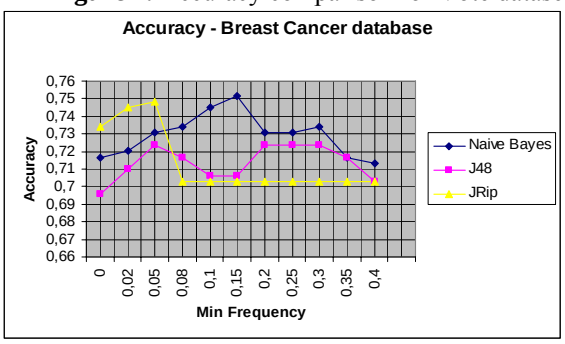

Figure 2. Accuracy comparison for Breast Cancer dataset

7 CONCLUSION AND FURTHER WORK

As we can see from the tables 1 and 2 the accuracy of the models depends on the choice of minimal frequency. The accuracy of the models doesn't decrease linearly as was expected at the beginning. From the results we can conclude that for small values of minimal frequency we get similar results as if we used some pruning method to handle the noisy data. This is so because by increasing the minimal frequency we get combinations of attributes that dominate in the dataset and the models are based mostly on them. As the frequency threshold increases at 0,5 (0,4) we have models that predict only the majority class because there is not enough information in the generated frequent itemsets.

This paper presents preliminary work done in this topic and is a starting point for further work. We want to use faster and more efficient algorithms, that can handle large datasets, to generate frequent itemsets. For very large datasets it would be convenient to use condensed representations of itemsets. Tests have to be made on large number of datasets in various domains so that we can see how the minimum frequency parameter affects the results and to see what are the limitations of the algorithms that can be used, and constraints that can be imposed. By doing the tests we will have more global conclusions about the usefulness of this approach in building predictive models.

References

[1] I. H. Witten, E. Frank. Data mining – Practical Machine Learning Tools and Techniques, Second Edition. Morgan Kaufmann Publishers 2005

[2] Margaret H. Dunham. Data Mining – Introductory and Advanced Topics, Prentice Hall, 2003

[3] Sašo Džeroski. Data Mining in a Nutshell, In Relational Data Mining,, S.Džeroski and N. Lavrač, editors, Springer, Berlin, 2001.

[4] J. R. Quinlan: Induction of decision trees, Machine Learning, no.1, pg.81-106, 1986

[5] William W. Cohen: Fast Effective Rule Induction, Proceedings of the Twelfth International Conference on Machine Learning, 1995

[6] Rakesh Agrawal and Ramakrishnan Sirkant: Fast Algorithms For Mining Association Rules, Proceedings of the 20th International Conference on Very Large Data Bases, 1994

[7] Rakesh Agrawal, Tomasz Imielinski and Arun Swami: Mining Association Rules between Sets of Items in Large Databases, Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, May 1993

[8] Bart Goethals: Course notes in Knowledge Discovery in Databases – Search for Frequent Patterns, http://www.cs.helsinki.fi/u/goethals/dmcourse/chap12.pdf

[9] J. F. Boulicaut, A. Bykowski and C. Rigotti: Approximation of Frequency Queries by Means of Free-Sets, PKDD 2000.