Analyzing relational databases using rough set based methods

Jakub Wróblewski
Institute of Mathematics, Warsaw University
Warsaw, Poland
jakubw@mimuw.edu.pl

Abstract

One of the most important problems in KDD applications is a size of real-world databases. In practical problems data may contain millions of records in many data tables bounded by relations. On the other hand, most of theoretical works and practical applications concentrate on relatively small data sets collected in single tables. The paper proposes a new approach to the problem of analysis of relational databases. The proposed methodology woks in adaptive way: if a considered data table is not sufficient to create satisfactory set of rules, new attributes (generated by arithmetical operations or based on database relations) are added to information system.

Keywords: Relational databases, new features extraction, rough sets.

1 Introduction

The most of KDD approaches bases on a paradigm of one decision table. On the other hand, most of large data (especially in financial, business or medical domains) have form of relational, multi-table databases. In these cases data mining process is preceded by preprocessing steps (based on domain-dependent knowledge) done interactively by user. After these steps one joint data table is created (see e.g. [2]).

The aim of this paper is to present a classification system based on rough set theory ([5], [7]). In our approach, formalized in section 2 and described in details in section 4 and 5, preprocessing is done automatically, by selecting and adding new features (attributes) into the initial information system. These new features are created basing on information hidden in relations and other tables.

Generation of new attributes, as well as the rule induction algorithm itself, needs a quality measure to determine which attribute (or subset of attributes) is more promising in terms of its usefulness in final classification algorithm. Predictive quality measure, based on rough set and probabilistic principles, is presented in section 3.

2 Relational information system

Let us consider a typical knowledge discovery problem on large, real-life data. Suppose we have a relational database of medical data collected by a hospital. The database contains a table of patients' personal data, their hospitalizations, results of examinations, diagnoses, surgical interventions, a table of doctors, rooms, medical equipment etc. All these tables are connected by different types of relations, e.g. "patient", "doctor" and "equipment" are related in terms of one "examination". Typically we have tens of thousands of patients, tens of examination results for each, hundreds of numerical attributes describing one examination etc. The task is to describe (predict) a class of patients with a high probability of complications. In many cases we have no such an attribute in a database – we should add it as a binary column into "patient" table, basing on other attributes and expert knowledge.

In general we can assume, that any descriptive or predictive problem in KDD is concerned with selected (or added) decision attribute in one of existing or newly created data tables (i.e. concerns one entity in relational database). If so, we have a distinguished table in our database, which can be analyzed using standard data mining tools. But, on the other hand, only a part of information virtually available in database is collected in this table. Most of knowledge is distributed over relations and another tables. In our example, "patient" table (which is in the centre of our considerations, as decision is an attribute of patient) contains only sex, age, name, statistical number and another general information about patient – this is far too less to obtain good results, as the most interesting information are hidden in "examination results", "interventions", "treatment" and other tables. If our data mining methods are designed to work on decision tables (as the most methods are) we should dynamically create such a table basing on another tables and relations.

To formalize these considerations in rough sets language, let us introduce a notion of *relational information system*:

Definition 1 *By* ***relational information system*** *we call a triple $\mathcal{S} = (\mathcal{A}, \mathcal{R}, \mathbb{A}_0)$, where:*

$\mathcal{A}$ *- a family of information systems: $\mathcal{A} = \{\mathbb{A}_0, \mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_k\}$, where $\mathbb{A}_i = (U_i, A_i)$ for $i \geq 1$, and $\mathbb{A}_0 = (U_0, A_0 \cup \{d\})$.*

$\mathcal{R}$ *- a set of relations on objects of information systems: $\mathcal{R} = \{r_{i,j} \subseteq U_i \times U_j : \mathbb{A}_i, \mathbb{A}_j \in \mathcal{A}\}$*

$\mathbb{A}_0$ *- a distinguished decision table, $\mathbb{A}_0 \in \mathcal{A}$.*

From the relational database point of view, set $\mathcal{A}$ corresponds to a set of data tables, and $r_{i,j}$ indicates a relation between tables. Technically, if there are columns used only as a primary or foreign key (and bringing no more information from a domain point of view), they can be omitted. $\mathbb{A}_0$ is a decision table, i.e. it contains a decision attribute d. Our knowledge discovery problem is concerned with objects from this table.

Definition 2 *We will say, that a decision table $\mathbb{B}$ is* ***induced*** *from a relational information system $\mathcal{S} = (\mathcal{A}, \mathcal{R}, \mathbb{A}_0)$ using a set of operations (transitions) $\mathcal{T}_\mathcal{S}$ if:*

$$\exists_{t_1,t_2,...,t_n \in \mathcal{T}_\mathcal{S}} : \mathbb{B} = t_1(t_2(...t_n(\mathbb{A}_0))...) \quad (1)$$

A set of possible transitions may be defined based on domain-dependent expert knowledge as well as on interpretation of relations in database.

Definition 3 *Let $\mathbb{B} = (U, B \cup \{d\})$. Set $\mathcal{T}_\mathcal{S}$ of possible operations on decision tables contains only transitions expanding decision table with a new attribute $\overline{a}$, i.e. if $t \in \mathcal{T}_\mathcal{S}$ then:*

$$t(\mathbb{B}) = t((U, B \cup \{d\})) = (U, B \cup \{\overline{a}\} \cup \{d\}) \quad (2)$$

where $\overline{a}$ is defined in one of the following ways:

1. *$\overline{a}(u) = a(u')$, where $a \in A_i \in \mathbb{A}_i \in \mathcal{A}$ and $(u, u') \in r_{0,i}$ and $r_{0,i}$ describes 1:1 or n:1 relation (i.e. $r_{0,i}$ is a function).*
2. *$\overline{a}(u) = F(\{a(u')\}_{u' \in r_{0,i}(u)})$, where $a \in A_i \in \mathbb{A}_i \in \mathcal{A}$ and F is an aggregation function (e.g. sum, average, min, max, number of null values, number of values satisfying a condition etc.) working on objects from U_i, and $r_{0,i}$ is an arbitrary non-empty relation.*
3. *$\overline{a}(u) = F_1(\{F_2(...\{F_k(\{a_k(u_k)\}_{u_k \in r_{k-1,k}(u_{k-1})})\}_{u_{k-1} \in r_{k-2,k-1}(u_{k-2})} ...)\}_{u_1 \in r_{0,1}(u)})$, where $a_k \in A_k \in \mathbb{A}_k$ and $F_1, ..., F_k$ is a set of aggregation functions like above, and there exists a sequence of tables $\mathbb{A}_1, ...\mathbb{A}_k \in \mathcal{A}$ such that relations $r_{0,1}$, $r_{1,2}$, ... $r_{k-1,k}$ are non-empty.*
4. *$\overline{a}(u) = G(b_1(u), ..., b_n(u))$, where $b_1, ...b_n \in B$ and G is an arbitrary function chosen from a certain class of functions (e.g. from a class of linear combinations).*

One may notice, that operations of type 1 and 2 are special cases of operations of type 3. However, from the practical point of view, this is reasonable to distinguish them.

Let us consider a medical database and a relational information system $\mathcal{S}$ based on it as an example. We have a table of "patients" with an additional decision attribute "complications" – this is our $\mathbb{A}_0$ table. In the first step we may try to generate a classification system basing only on $\mathbb{A}_0$, but the quality of such a system will probably be poor. Now we should generate $\mathbb{B}$ induced from $\mathcal{S}$ by adding new attributes to $\mathbb{A}_0$. Suppose that our database contains a table "environment" containing (among other) degree of air pollution in different cities. In this case we can add new attribute "pollution in permanent residence" to a table of "patients" (this is type 1 addition). Suppose we have a table "blood analysis" with an attribute "cholesterol" – in this case we can add new attribute "maximal cholesterol amount during last year" (this is type 2 addition). We may have also a table of "physicians" related to "examinations" and add new attribute "does patient had any examination with Dr. Black?" (this is type 3 addition). It is easy to see that a number of possible combinations of tables, attributes, operations and conditions is potentially infinite.

Since composition of operations from $\mathcal{T}_{\mathcal{S}}$ is associative and commutative, we can define the following notion:

Definition 4 *By **inductive closure** $\mathbb{B}^*$ of relational information system $\mathcal{S}$ we will denote a (potentially infinite) decision table $\mathbb{B}^* = (U, B^* \cup \{d\})$ such that for any $\mathbb{B} = (U, B \cup \{d\})$ induced from $\mathcal{S}$ we have $B \subset B^*$.*

The table $\mathbb{B}^*$ contains all information one can induce from $\mathcal{S}$ using the given set of transitions.

Remark If a set of attributes R is a reduct for decision table $\mathbb{B}$, this is also a reduct for $\mathbb{B}^*$.

There is no need (nor possibility) to analyze $\mathbb{B}^*$, since one can create an efficient classification algorithm based on properly selected subset $\mathbb{B}$. Therefore one should use a quality measure to determine which new attribute (or set of attributes) is worth to be included.

3 Predictive quality measure of sets of attributes

There are many ways of attributes' subsets evaluation. Rough sets theory [5] [7] bases on a notion of **reduct** – a minimal subset of attributes discerning objects in terms of its decision value. One can say, that the notion of reduct itself induces a quality measure: if a subset is a reduct, its quality is good (because we are able to construct complete rule set basing on it). On the other hand, there are "better" and "worse" reducts: some of them generates more general sets of rules than others. In many applications reducts are optimized to be short (i.e. to contain as less attributes as possible, see [9], [4]). Other authors consider a reduct measure based on its stability in terms of data table's subsets (dynamic reducts, see [1]). Another approach is based on a number of rules generated by a reduct (if this number is small, rules covers more objects and are more general; see [9]). In all these cases we can induce a quality measure of arbitrary subset of attributes by setting it to 0 when a subset is not a reduct, or to a reduct quality measure value otherwise. However, in practical applications classification algorithms based on exact reducts are often too specific and there are too many not recognized testing objects. Fortunately, solutions based on number of rules (or rules coverage) can be easily adopted to the case of approximate reducts and rules.

In [8] a quality measure for new attribute $\overline{a}$ defined as a linear combination of another attributes was proposed. The measure can be easily generalized to the case of arbitrary set of attributes. For a given $R = \{a_1, ...a_n\} \subseteq A$ we have a partition $\{U_1, U_2, ...U_k\}$, where $\bigcup\limits_{i=1...k} U_i = U$, such that U_i are the abstract classes of indiscernibility relation $IND_{\mathbb{A}}(R)$. This partition generates a set of rules $\{r_1, r_2, ...r_k\}$, where $r_i = ((a_1 = v_{1,i}) \wedge ... \wedge (a_n = v_{n,i}) \implies d = d_i)$ and $U_i = \{u \in U : (a_1(u) = v_{1,i}) \wedge ... \wedge (a_n(u) = v_{n,i})\}$ and d_i is the most frequent decision value in U_i. The quality measure used in [8] has a form:

$$Q(R) = \sum_{i=1}^{k} \|U_i\|^2 \tag{3}$$

Let $q(r) \in [0, 1]$ be any quality measure of rules. Quality measure Q_{qual} of subsets of attributes induced by a rule measure q can be defined as:

$$Q_{qual}(R) = \frac{1}{k} \sum_{i=1}^{k} q(r_k)^2 \tag{4}$$

where $\{r_1, ...r_k\}$ is a set of rules concerned with partition generated by R. Unfortunately, experiments on several data sets show that neither

(3) nor (4) evaluates subsets of attributes well enough.

The final goal of a process of selection a set of attributes is to create a set of rules based on these attributes. Any quality measure of a set of attributes should reflect to quality of classification algorithm generated by it, i.e. to ability of proper classification of unseen objects. Although we obviously do not know such a quality during a training phase, we can estimate it. First, let us introduce a quality measure of final classification algorithm (CA) on a test data:

Definition 5 *Let CA be a classification algorithm. Let $\mathbb{A}_{tst} = (U_{tst}, A \cup \{d\})$ be a testing decision table. Let $P(CA, \mathbb{A}_{tst}, i)$ be a number of properly classified test objects belonging to i-th decision class, $N(CA, \mathbb{A}_{tst}, i)$ – a number of test objects belonging to i-th decision class but classified to another one, n – number of decision classes. Let* ***sensitivity*** *and* ***coverage*** *of CA be defined as:*

$Sens(CA, \mathbb{A}_{tst}) =$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{P(CA, \mathbb{A}_{tst}, i)}{P(CA, \mathbb{A}_{tst}, i) + N(CA, \mathbb{A}_{tst}, i)}}$$

$$Cov(CA, \mathbb{A}_{tst}) = \frac{\sum_{i=1}^{n} P(CA, \mathbb{A}_{tst}, i)}{\|U_{test}\|}$$

Then the quality of CA on $\mathbb{A}_{tst}$ is defined as:

$$Q(CA, \mathbb{A}_{tst}) = Sens(CA, \mathbb{A}_{tst}) \times Cov(CA, \mathbb{A}_{tst}) \quad (5)$$

The measure defined above takes into account two aspects of classification algorithm: its sensitivity (ability of distinguish between decision classes) and coverage (number of objects classified properly). The first parameter is a generalization of sensitivity definition used in ROC (Relative Operating Characteristic) analysis [6].

The sensitivity of final classification algorithm can be estimated by calculating sensitivity of rule set on training data. However, the second coefficient (coverage) is harder to estimate as procedure described above creates a rule set covering the whole training table. Let $n(r_i)$ be a number of training objects classified properly by rule r_i, generated basing on single object (and its abstract class). Then, by excluding the base object, we obtain $\frac{n(r_i)-1}{\|U\|-1}$ as an estimation of probability, that a random object from the testing set will be classified properly by r_i (it corresponds to an observation, that rules covering only one object on training set are practically useless on test table). Now, treating covering by different rules to be independent (which is not true for training table), we obtain an estimated probability for testing object to be covered by at least one rule which classifies it properly:

$$P_{cov} = 1 - \prod_{i=1}^{k}\left(1 - \frac{n(r_i) - 1}{\|U\| - 1}\right) \quad (6)$$

where k – number of rules in CA. Now, we are ready to introduce a *predictive measure* of subset R of attributes:

$Q_{pred}(R) =$

$$= \sqrt[n]{\prod_{i=1}^{n} \frac{P(\{r_1...r_k\}, \mathbb{A}, i)}{P(\{r_1...r_k\}, \mathbb{A}, i) + N(\{r_1...r_k\}, \mathbb{A}, i)}} \times P_{cov} \quad (7)$$

where $\mathbb{A}$ – training decision table. Experiments show, that Q_{pred} is good estimator of final classification algorithm quality on a real data.

4 General adaptive scheme

As was shown in section 2, in case of relational database the most of useful information about objects to be classified may be distributed in many tables. Let us consider an adaptive process of collecting knowledge about data:

1. Let $\mathbb{A}_0$ will be a base decision table of a relational information system $\mathcal{S}$. Let $\mathbb{B} = \mathbb{A}_0$.

2. Create a classification algorithm CA for table $\mathbb{B}$ and calculate its quality (using e.g. (5)) on test data. If there is no test table available, we should create one as a separate part of training data.

3. If quality of CA is satisfactory, stop. If not, try to gain additional information.

4. Let $\mathbb{B} \leftarrow t_1(...t_k(\mathbb{B})...)$, i.e. expand the table by adding k new attributes (use one of possible transitions). Use attribute's quality measure (see section 7) to choose the most promising operations.

5. Use attributes' subsets measure to decrease the number of attributes in $\mathbb{B}$.

6. Continue from 2.

The main problem one will face using this scheme is a huge number of possible transitions (new attributes). However, since the process of attributes' quality calculation is faster than creation of classification algorithm, one can try to find good examples of attributes in reasonable time. Moreover, one can use additional adaptive technique and utilize information about good examples of new attributes to create similar ones.

5 Results of experiments

5.1 Predictive subset measure

We have used several data table from UCI repository [3] to test predictive subset quality (7). Tests was performed as follows: first, an order (permutation) of attributes $\{\sigma_1, \sigma_2, ...\sigma_n\}$ was generated randomly. Then, a number of sets $A_1 = \{a_{\sigma_1}\}$, $A_2 = \{a_{\sigma_1}, a_{\sigma_2}\}$, ... $A_k = \{a_{\sigma_1}, a_{\sigma_2}, ...a_{\sigma_k}\}$ was evaluated using a quality measure (7). Then, for all sets $A_1, ...A_k$ a rule set was generated and verified on test data. The experiment is successful when a subset with maximal value of predictive measure generates a set of rules with good quality on test data (i.e. one of the three best values of quality). Experiments was performed for 10 random permutations.

We have obtained the following results (percent of successful experiments): *Breast cancer* - 70%, *Diabetes* - 100%, *Sat images* - 50%, *German credit* - 60%, *Letter recognition* - 80%. We have used CV-5 to generate test tables (except Sat images and Letter recognition problems).

These results suggests, that predictive quality measure may be useful in rule set generating. One may generate (randomly or using e.g. genetic algorithm – see [4], [9]) a set of permutations and sequences of corresponding subsets $A_1, ...A_k$. Then one can evaluate predictive quality measure and choose the optimal subset from each sequence (which shows – in most cases – to produce optimal final results). Classification algorithm is based on rules generated by these optimal subsets of attributes.

5.2 Adaptive classification algorithm working on relational database

We have used *PKDD'99 Challenge* data set [2] (financial domain) for experiments. The original data have form of relational database containing data tables: *client* (5369 records), *account* (4500 records), *disposition* (5369 records), *permanent order* (6471 records), *loan* (682 records), *credit card* (892 records), *transaction* (about one million records), *demographic data* (77 records). The goal we focus on was to describe a class of clients which have problems with loan payments.

The main advantage of using adaptive algorithm described in this paper is to minimize data preprocessing done by expert (algorithm should find itself interesting data). However, for optimization purpose, some manual filtering of data was done. First, while there is only 682 clients which have any loan, we filter the table to delete all uninteresting clients and data associated with them. Moreover, since only about 80 clients have any problems with loans, the data was additionally subsampled, so in final experimental set there was about 230 "good" clients and 80 "bad" ones. The rest of data tables was deflated accordingly, so e.g. *transactions* table had as little as 83000 records. Then, a new column "loan ok" (based on data from loan table) was added into *client* table as a decision attribute.

Then the adaptive algorithm described in section 4 was used to create and test new features (attributes) based on relations. First, four randomly created new attributes was evaluated (as one-attribute subsets) using predictive quality measure and the best was added to initial data table (containing only client's data, i.e. birthdate, owner/user indicator and decision). When initial table was expanded by 20 newly created attributes, the next phase began. Then, several random permutations was generated and optimal subset of attributes (see section 3) was created and evaluated using predictive quality measure. Then the best one was chosen and rules based on it was added to the rule set. This process was continued until predicted coverage of the rule set exceeds 98%. Finally the rule set was tested on test table (using CV-5 scheme). The set of possible operations contains sum, average, minimal, maximal value of corresponding records in case of 1:n type relation, as well as SQL operations count and exists; all these operations can be limited by additional criteria.

When the initial data table is tested without adding any new attributes, results are bad: 16% correct answers, 4% bad answers, 80% not classified. This is rather not surprising since there is no useful information about client in initial table. After expanding the table by 20 new attributes, there was 77% correct answers, 18% bad answers, 5% not classified (these bad answers contains both positive and negative cases). Moreover, analysis shows, that only a few attributes was used in final rule set, so the rest can be omitted.

Example of final (active) attributes set (in pseudo-SQL):

value of payments from loan - i.e. the value of monthly payments;

value of type from client - a client is an owner or user of account?

value of crimes_in_95 from demographic_data - number of committed crimes in 1995 in the district a client lives in;

average amount from permanent_order - average amount of all client's permanent orders;

sum amount from permanent_order where amount > 8094

sum amount from permanent_order where amount > 7240 - these "magical" criteria was generated randomly and shows to be somehow meaningful in terms of predictive quality measure.

These 6 attributes generates 296 rules with efficiency about 77% on test data.

The whole process described above takes about 3 minutes on typical PC (Celeron 400). Note, that main decision table contains only 310 records, but the largest table used in calculations (*transaction*) contains more than 83000 records.

6 Conclusions and future work

An adaptive process of analysis large relational databases by discovering new features based on relations was proposed and implemented. To formalize this approach basing on rough set theory, a new notion of relational information system was introduced. An adaptive classification system based on predictive attributes' subsets quality measure was successfully used to analyze *PKDD'99 Challenge* database. It is worth noting that analysis needs only a little preprocessing by user; there is no need to create joint tables etc. since all operations are performed in SQL-like manner.

Some elements of the classification algorithms need more future research. New attributes are created randomly at the moment – there is no adaptive optimization (described in section 4) implemented yet. More tests on different relational data tables would be interesting too. Another future technical improvement is to implement SQL connection to external relational databases instead of working with the whole data set in memory.

Acknowledgements

This work was partially supported by ESPRIT Project 20288 CRIT-2.

References

[1] J. Bazan (1998). A Comparison of Dynamic and non-Dynamic Rough Set Methods for Extracting Laws from Decision Tables. In: L. Polkowski, A. Skowron (eds.). *Rough Sets in Knowledge Discovery.* Physica Verlag, Heidelberg 1998.

[2] P. Berka (1999). Workshop notes on discovery challenge. PKDD'99 Conference, Prague, Czech Republic. Data avaliable at http://lisp.vse.cz/pkdd99/chall.htm.

[3] D. Michie, D.J. Spiegelhalter, C.C. Taylor (1994). Machine Learning, Neural and Sta-

tistical Classification. Ellis Horwood Limited 1994. Data avaliable at: www.ics.uci.edu/˜mlearn/MLRepository.html.

[4] S.H. Nguyen, A. Skowron, P. Synak, J. Wróblewski (1997). Knowledge Discovery in Databases: Rough Set Approach. *Proc. of The Seventh International Fuzzy Systems Association World Congress*, vol. II, IFSA'97, Prague, Czech Republic 1997, pp. 204–209.

[5] Z. Pawlak (1991). Rough sets – Theoretical aspects of reasoning about data. Kluwer Academic Publishers, Dordrecht 1991.

[6] F. Provost, T. Fawcett, R. Kohsvi (1998). The case against accuracy estimation for comparing induction algorithms. In: *Proc. of IMLC'98*, Madison, WI 1998.

[7] A. Skowron, C. Rauszer (1992). The discernibility matrices and functions in information systems. In: R. Słowiński (ed.), *Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory*, Kluwer Academic Publishers, Dordrecht 1992, pp. 311–362.

[8] D. Ślęzak, J. Wróblewski (1999). Classification Algorithms Based on Linear Combinations of Features. *Proc. of PKDD'99*, Prague, Czech Republic. Springer-Verlag (LNAI 1704), Berlin Heidelberg 1999, pp. 548–553.

[9] J. Wróblewski (1998). Genetic algorithms in decomposition and classification problem. In: Polkowski, L., Skowron, A. (eds.), *Rough Sets in Knowledge Discovery 2*. Physica Verlag, Heidelberg 1998, pp. 471–487.