Mining Relational Databases

Frédéric Moal, Teddy Turmeaux, and Christel Vrain

LIFO, Université d'Orléans,
rue Léonard de Vinci, BP 6759,
45067 Orléans cedex 02, France
{moal,turmeaux,cv}@lifo.univ-orleans.fr

Abstract. In this paper, we propose a classification system to induce an intentional definition of a relation from examples, when background knowledge is stored in a relational database composed of several tables and views. Refinement operators have been defined to integrate in a uniform way different induction tools learning numeric and symbolic constraints. The particularity of our approach is to use integrity constraints over the database (keys and foreign keys) to explore the hypotheses space. Moreover new attributes can be introduced, relying on the aggregation operator "group by".

1 Introduction

Nowadays, most of the data sources are stored in large relational database systems in which information is expressed in a given normal form and attributes are often distributed over several tables. Current data mining systems [2] often require data to be stored in a single relation and therefore relational applications have to be flattened into a single table, losing interesting information about the structure of the database and leading to huge, intractable relation. On the other hand, Inductive Logic Programming (**ILP** [4]) is devoted to relational learning in a logical framework, and some algorithms have been successfully applied to Data Mining tasks [5]. However, characteristics of Data Mining tasks differ from usual ILP ones: the search space is more restricted (no recursive rules), and the data size is much higher.

In this paper, we present a classification system that is able to handle relational databases and to use information about the structure of the database to guide the search of good hypotheses. For efficiency reasons, we have chosen a separate and conquer strategy relying on a hill climbing heuristic, as in the system FOIL [6]. We have defined new refinement operators based on the schema of the database and on integrity constraints. Moreover, a new quality criterion is used to avoid the generation of too specific rules at the end of the process. Finally, the classical ILP expressiveness has been extended by introducing the *group-by* SQL operator.

The paper is organized as follows. Section 2 formalizes the problem in the framework of relational algebra. Section 3 is devoted to our system, its quality criterion and its refinement operators. In Section 4, we present experiments on

D.A. Zighed, J. Komorowski, and J. Żytkow (Eds.): PKDD 2000, LNAI 1910, pp. 536–541, 2000.

a classical problem for relational learners (mutagenesis [5]), and we conclude on perspectives in Section 5.

2 Relational Algebra

The Data Mining task we address here is a classification task. It can be seen as the search of a "good" query over the database, the answer of which covers many positive examples and rejects many negative examples. A natural way to express queries is relational algebra [1], that can easily be translated into SQL.

2.1 Notations

The database schema is denoted by $\mathcal{D}$, with $\mathcal{D} = \{R_1(U_1), \ldots, R_n(U_n)\}$ where R_i is a relation and U_i is the set of attributes of the relation R_i.

To avoid naming conflicts in expressions, attributes are referenced by their positions in the relation. For instance, in the relation $person(id, name, age)$, $person[2]$ stands for the attribute *name*. Moreover, we use uppercase letters for the schema $\mathcal{D}$ and its relations R_i, and the same lowercase letters for their corresponding instances; an instance d of the database $\mathcal{D} = (R_1, \ldots, R_n)$ is a set $\{r_1, \ldots, r_n\}$ where r_i is an instance, i.e. a set of tuples, of the relation R_i.

We consider the following relational algebra operators:

- $R \bowtie_{i=j} S$ denotes the join of the tuples of r and s on the ith attribute of R and the jth attribute of S;
- $\sigma_C(R)$ denotes the selection of the tuples t of r that satisfy the constraint C, a propositional formula over attributes of R;
- $\pi_i(R)$ denotes the projection of r on its ith attribute;
- $R \cup S$ denotes the union of the tuples of r and s.

A relation is defined *in intension* by a relational algebra expression, it is defined *in extension* by the list of its tuples. If E denotes a relational algebra expression over the relations of $\mathcal{D}$, and if d is an instance of the database, then $E(d)$ is the set of tuples of d satisfying E.

In this paper, we focus on two usual classes of integrity constraints: keys and foreign key links. A *key* $R_1[i]$ means that given a value for the ith attribute of the relation r_1, there exists a single tuple in r_1 with that value. A *foreign key link* links an attribute in a relation to a key attribute in another relation: $R_1[i] \rightarrow R_2[j]$ means that for each tuple with a given value for the ith attribute of the relation r_1, there exists a single tuple in the relation r_2 with this value for the jth attribute.

2.2 The Mutagenesis Problem

The aim of this problem [5] is to discriminate between molecules, depending on their mutagenic activity (*active* or *inactive*). The dataset consists of generic 2D chemical compound descriptions, described by:

- a measure of hydrophobicity of the molecule, denoted by $LogP$;
- a measure of the energy of the lowest unoccupied molecular orbital in the molecule, denoted by $Lumo$;
- a boolean attribute $I1$, identifying compounds with 3 or more benzyl rings;
- a boolean attribute Ia, identifying a subclass of compounds (acenthryles);
- the atoms that occur in a molecule, with their elements (carbon, zinc, ...), Quanta type and partial charge;
- the bonds between atoms, with their bond type (aromatic, ...).

This information is represented by the following relational model:

$$\begin{aligned}\mathcal{D} = \{\ & Compound(DrugId, Lumo, Logp, I1, Ia)\\ & Atom(AtomId, DrugId, Element, Quanta, Charge),\\ & Bond(AtomId1, AtomId2, BondType)\ \}\end{aligned}$$

with the keys $\{Compound[1], Atom[1], Bond[1,2]\}$ and the foreign key links $\{Atom[2] \rightarrow Compound[1], Bond[1] \rightarrow Atom[1], Bond[2] \rightarrow Atom[1]\}$.

2.3 Formulation of the Problem

Given a database $\mathcal{D}$ and an instance d of $\mathcal{D}$, and given a concept to learn defined *in extension* by two relations on the same attribute, a positive example relation e^+ and a negative one e^-, find an *intentional* definition E over the database $\mathcal{D}$ that is complete ($e^+ \subseteq E(d)$) and consistent ($e^- \cap E(d) = \emptyset$).

When these requirements are too strong, the aim is only to find a definition which covers many positive examples and rejects many negative examples.

As explained in Section 3 and due to our search strategy and to our definitions of operators, the relational expressions that are learned in our system are written: $\pi_a(\dots \sigma_{C_{i_3}}(\sigma_{C_{i_2}}(\sigma_{C_{i_1}}(R_{i_1}) \bowtie R_{i_2}) \bowtie R_{i_3}) \dots)$. A projection must be applied on the learned hypothesis to restrict the relation to the attribute defining the examples.

3 Architecture of the System

The underlying algorithm is a refinement of the classical *separate-and-conquer* algorithm: the basic idea is to search for a rule that covers a part of the positive examples, to remove these examples, and to start again with the remaining positive examples. In our system, two parameters are given defining respectively the minimum rate of covered positive examples (MinPosRate) and the maximum rate of covered negative examples (MaxNegRate). Each iteration builds a hypothesis that is the best refinement of the current one according to a quality criterion.

3.1 Quality Criterion

Algorithms based on a *separate-and-conquer* strategy lead to overly specific rules at the end of the process. To overcome this problem, we propose a new quality

evaluation function, which takes into account the number of rules that have already been learned.

The quality of a refinement h' of a hypothesis h is therefore computed by the formula: $t_{pos}^n * t_{neg}^2$, where t_{pos} (resp. t_{neg}) is the ratio of the number of positive (resp. negative) examples covered (resp. rejected) by h' out of the number of positive (resp. negative) examples covered by h, and n is the number of the iteration. With such an expression, when n increases, low values for t_{pos} penalize the quality function. This solution reduces the number of rules in the final solution; on the other hand, the last generated rules are longer, since more refinements are necessary.

3.2 Refinement Operators

Our approach is a top-down one: the system starts from a general rule that covers all the examples and iteratively specializes it so that it covers less and less tuples representing negative examples. Given a hypothesis h, a refinement operator generates hypotheses h' that are more specific than h.

Classical Operators

Selection refinement $\mathcal{O}_S$: Given a hypothesis $h = \pi_a(R)$, $\mathcal{O}_S$ produces a set of hypotheses $\mathcal{O}_S(h) = \{\pi_a(\sigma_C(R))\}$, where C is a propositional formula over the attributes of R. The constraint C is obtained by using classical algorithms in discrimination tasks as for example finding a discriminant hyperplane.

Join refinement $\mathcal{O}_J$: Given a current hypothesis $h = \pi_a(R)$, where R is a composition of selections and joins on relations $R_1, \ldots, R_n$, the operator generates a set of hypotheses h' such as: $h' = \{\pi_a(R \bowtie_{i=j} S)\}$, with $S(A_1, \ldots, A_n) \in \mathcal{D}$, i references the kth attribute of a relation R_m in $R_1 \ldots, R_n$, and $R_m[k]$ and $S[j]$ are connected with a foreign key link. The use of keys and foreign key links restricts the possible joins and thus enables to prune the search space efficiently.

For instance, on the mutagenesis database, the hypothesis $\pi_2(Atom)$ is refined to $\pi_2(Atom \bowtie_{2=1} Compound)$, using the link with Compound, or to $\pi_2(Atom \bowtie_{1=1} Bond)$ or $\pi_2(Atom \bowtie_{1=2} Bond)$ using the two links between *Atom* and *Bond*.

New Operators

Composition: The application of these two operators allows to search the hypotheses space. Nevertheless, as pointed out in [7], when using a hill-climbing algorithm which chooses the best hypothesis at each step, some literals which bring no discrimination power are not introduced while they could be very useful to introduce new discriminant features.

In our system, the two operators are composed: first, the *join* refinement is applied, and for each refinement the *selection* refinement is applied. The *composition* refinement operator is then $\mathcal{O}_C = \mathcal{O}_S \circ \mathcal{O}_J$. Join refinements are evaluated

according to the discriminant power of the new attributes they introduce, and the best composition is chosen.

Aggregate refinement operator. It refines a hypothesis $h = \pi_a(R)$, where R is a composition of selections and joins on relations $R_1, \ldots, R_n$ into $\pi_a(R \bowtie_{i=j} Group(S, j))$, such that $S(A_1, \ldots, A_j, \ldots, A_n) \in \mathcal{D}$, i references the kth attribute of a relation R_m in $R_1, \ldots, R_n$, and there is a foreign link $S[i] \rightarrow R_m[k]$. This definition can be easily extended to group a relation over more than one attribute. The only restriction is that the attributes used in the join appear in these grouping attributes.

4 Experiments and Discussion

The mutagenesis database consists of generic chemical compound descriptions, as shown in Section 2.2. The complete database is composed of 230 compounds which have been divided into two distinct subsets. We focus on the subset identified as "Regression friendly" to compare our results with some results of Attribute-based algorithms. This subset is composed of 125 active molecules (positive examples) and 63 inactive molecules (negative examples).

The goal is to find discriminant rules for the active molecules. All the results are obtained using a 10-fold cross-validation, where every molecule is used nine times in the training set and once in the test set.

Typically, the accuracy is the number of well classified examples divided by the total number of examples. To compute the real accuracy of our system on this problem, we adopt the following procedure: for each of the 10 cross-validation run, 1/10 of the examples composed the test set and 9/10 are used as the learning set. This learning set is then used to determine the parameters MaxNegRate and MinPosRate. for various values of the parameters, we randomly select 80% of this learning set to induce relational expressions, testing the accuracy on the 20% remainder. The parameters that lead to the best accuracy over these 20% are then used with the complete learning set. The accuracy of the learned hypothesis is computed over the initial testing set.

The predictive accuracies of various systems, extracted from [5], are presented in Table 1. Progol [3] is an Inductive Logic Programming system, and the other are attribute-based algorithms. The "default" algorithm classifies all the examples to the majority class (active).

Table 1 states that our system is competitive in terms of predictive accuracy, despite the hill-climbing strategy we use. This strategy is interesting because it tests much less hypotheses than more exhaustive search strategies and this is a very important feature to deal with large databases.

5 Conclusion

In this paper, we have proposed a new system to deal with classification tasks in the framework of relational databases. We have developed new refinement operators based on the database schema and integrity constraints.

Table 1. Predictive Accuracies (from a 10-fold cross-validation).

Method	Estimated Accuracy
Our System	0.89
Progol	0.88
Linear Regression	0.89
Neural Network	0.89
Decision Tree	0.88
Default	0.66

The results of our experiments seem to confirm the interest of our approach, in terms of accuracy and computational cost. The low computational cost allows the user to iteratively refine its search with different parameters and biases. In our opinion, such an iteration is a necessary stage in all Data Mining tasks. However, these results must be validated with other databases.

The use of the database structure for the design of refinement operators offers some perspectives: the study of this approach in the framework of Object Databases which allows to express finer structural constraints; the extension of the tuples expressiveness with Constraints Databases[8].

Acknowledgements: This work is partially supported by "Région Centre".

References

[1] P. Atzeni and V. De Antonellis. *Relational Database Theory.* Benjamin/Cummings Publ. Comp., Redwood City, California, 1993.

[2] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining.* MIT Press, Mento Park, 1996.

[3] S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.

[4] S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19 & 20:629–680, May 1994.

[5] S. Muggleton, A. Srinivasan, R. King, and M. Sternberg. Biochemical knowledge discovery using Inductive Logic Programming. In H. Motoda, editor, *Proc. of the first Conference on Discovery Science*, Berlin, 1998. Springer-Verlag.

[6] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[7] G. Silverstein and M. Pazzani. Relational cliches: constraining constructive induction during relational learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, Los Altos, CA, 1989. Kaufmann.

[8] T. Turmeaux and C. Vrain. Learning in constraint databases. In *Discovery Science, Second International Conference*, volume 1721 of *LNAI*, pages 196–207, Berlin, december 1999. Springer.