Efficient and Scalable Induction of Logic Programs using a Deductive Database System

Michel Ferreira, Ricardo Rocha, Tiago Soares, and Nuno A. Fonseca *

DCC-FC & LIACC
University of Porto, Portugal
{michel,ricroc,tiagosoares,nf}@ncc.up.pt

Extended Abstract

A consequence of ILP systems being implemented in Prolog (e.g., [1–3]) or using Prolog libraries (e.g., [4]) is that, usually, these systems use the Prolog internal database to store and manipulate data. However, in real-world problems, the original data is rarely in Prolog format. In fact, due to the huge amount of information used to characterise these type of problems, the data is often kept in relational database management systems (RDBMS).

A common approach is thus to convert the data in the RDBMS to a format acceptable by the ILP system. A more interesting approach is to link the ILP system to the RDBMS and manipulate the data without converting it [5]. This can be done by **(i)** mapping Prolog predicates to database tables; or by **(ii)** translating logical clauses into SQL statements. Note that both schemes have the advantage of being more scalable since the whole data does not need to be loaded into memory by the ILP system.

If the ILP system is implemented in a first order language like Prolog, then the mapping scheme is a more transparent solution for the designer of the ILP engine. However, this scheme can result in increased communication with the RDBMS since many accesses may be needed to evaluate each single hypothesis [6]. On the other hand, the translating scheme reduces communication at the cost of some expressiveness - the ILP system is no longer able to learn recursive definitions. In this scheme one can devise two ways of translating a hypothesis to an SQL statement: **(i)** transform a hypothesis into an equivalent SQL view; or **(ii)** transform a hypothesis into an equivalent SQL count query. The second alternative is the one usually pursued since it minimizes communication and transfers almost all work to the RDBMS. For each hypothesis, the communication exchanged with the RDBMS is the SQL count query and a number, with the count value, returned as the result.

Several previous implementations have already coupled ILP systems with relational databases, some using the mapping scheme, others translating logical

* This work has been partially supported by MYDDAS (POSC/EIA/59154/2004) and by funds granted to LIACC through the Programa de Financiamento Plurianual, Fundação para a Ciência e Tecnologia (FCT) and Programa POSC. Tiago Soares is funded by FCT PhD grant SFRH/BD/23906/2005.

rules into SQL statements [7–9], and others using both [10]. The level of transparency for the user in these implementations is quite variable, ranging from no transparency (e.g., the user manually defines the views for each literal that may appear in a hypothesis [10]), to completely transparent [9]. Despite the work on linking ILP systems with relational databases (e.g., [7–10]), very little is known about the impact on efficiency of learning directly from a database. It is often assumed that the increase in scalability resulted from using a RDBMS costs a reduction in performance.

In this work we present an *user transparent* approach to couple an ILP system with a relational database by using a Deductive Database (DDB) system. Our proposal uses the DDB engine to transparently translate the generated hypotheses to SQL statements with only minor changes to the implementation of the ILP system. By transferring as much as possible the evaluation of hypotheses to the RDBMS, we show how this coupled environment provides an excellent framework for the efficient and scalable execution of ILP algorithms. By using a DDB system, the ILP system can transparently exploit advanced features of relational databases, such as powerful indexing schemes, query optimization, efficient aggregation and joining algorithms.

The idea of coupling ILP with DDB is not new - the ILP system Warmr has been coupled with a deductive database system to mine association rules [11]. The difference to our work is twofold. First, the tasks addressed are different: Warmr learns associations rules from multiple relations while we are concerned with learning classification rules. Second, Warms loads data into main memory while in our proposal the data remains in the database.

Being able to abstract the Prolog to SQL translation by using the DDB, we concentrate on evaluating several high-level schemes of interaction between the ILP system and the RDBMS, with different distributions of work between the logic system and the database system. In all experiments we used four artificially generated data-sets [12] that allowed us to perform the evaluations while considering different data-set sizes and hypotheses complexity (number of joins in a hypothesis). To do the experiments we used April [13] as the ILP system and MYDDAS [14], which couples the Yap Prolog engine [15] with the MySQL RDBMS, as the DDB system.

Our results indicate that the execution time of ILP algorithms can be effectively reduced and that the size of the problems solved can be significantly increased due to a non-memory storage of the data-sets. Best performance is achieved when we use a scheme that transforms the hypotheses into an equivalent SQL count query.

ILP systems often use some kind of input/output mode declarations to supply information concerning the arguments of each predicate that may appear in the hypotheses [16, 17]. These declarations specify if an argument (attribute) of a predicate (table) is intended to be a constant, an input or an output argument. Although the mode declarations are usually provided by the user, they can be also automatically extracted from the background knowledge. This mode information provided to the ILP system is used to optimize query execution in the

RDBMS by automatically creating indexes in the tables. The rationale is that all hypotheses (queries) generated will be mode conform and, thus, the projection operations in the queries will always be performed using the constant and/or the input attributes. Our performance study also shows that the automatic index creation, when used with the count scheme, reduces the execution speed significantly [6].

References

1. Raedt, L.D., Laer, W.V.: Inductive Constraint Logic. In: International Conference on Algorithmic Learning Theory, Springer-Verlag (1995) 80–94
2. Raedt, L.D., Dehaspe, L.: Clausal Discovery. Machine Learning **26** (1997) 99–146
3. Srinivasan, A.: The Aleph Manual. (2003) Available from `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph`.
4. Muggleton, S., Firth, J.: Relational Rule Induction with CProgol4.4: A Tutorial Introduction. In: Relational Data Mining. Springer-Verlag (2001) 160–188
5. Blockeel, H., Raedt, L.D.: Relational Knowledge Discovery in Databases. In: International Workshop on Inductive Logic Programming. Number 1314 in LNAI, Springer-Verlag (1996) 199–211
6. Soares, T., Ferreira, M., Rocha, R., Fonseca, N.A.: On Applying Deductive Databases to Inductive Logic Programming: a Performance Study. In: Colloquium on Implementation of Constraint and LOgic Programming Systems. (2006) To appear.
7. Brockhausen, P., Morik, K.: Direct Access of an ILP Algorithm to a Database Management System. In: MLnet Familiarization Workshop on Data Mining with Inductive Logic Programing. (1996) 95–100
8. Morik, K.: Knowledge Discovery in Databases - an Inductive Logic Programming Approach. In: Foundations of Computer Science: Potential - Theory - Cognition, Springer-Verlag (1997) 429–436
9. Bockhorst, J., Ong, I.M.: FOIL-D: Efficiently Scaling FOIL for Multi-Relational Data Mining of Large Datasets. In: International Conference on Inductive Logic Programming. (2004) 63–79
10. Weber, I.: Discovery of First-Order Regularities in a Relational Database Using Offline Candidate Determination. In: International Workshop on Inductive Logic Programming, Springer-Verlag (1997) 288–295
11. Dehaspe, L., Toironen, H.: Discovery of Relational Association Rules. In: Relational Data Mining. Springer-Verlag (2000) 189–208
12. Botta, M., Giordana, A., Saitta, L., Sebag, M.: Relational Learning as Search in a Critical Region. Journal of Machine Learning Research **4** (2003) 431–463
13. Fonseca, N.A., Silva, F., Camacho, R.: April - An Inductive Logic Programming System. In: European Conference on Logics in Artificial Intelligence. Number 4160 in LNAI, Springer-Verlag (2006) To appear.
14. Soares, T., Ferreira, M., Rocha, R.: The MYDDAS Programmer's Manual. Technical Report DCC-2005-10, Department of Computer Science, University of Porto (2005)
15. Santos Costa, V., Damas, L., Reis, R., Azevedo, R.: (YAP User's Manual) Available from `http://www.ncc.up.pt/~vsc/Yap`.
16. Muggleton, S.: Inverse Entailment and Progol. New Generation Computing, Special Issue on Inductive Logic Programming **13** (1995) 245–286
17. Blockeel, H., Raedt, L.D.: Top-Down Induction of First-Order Logical Decision Trees. Artificial Intelligence **101** (1998) 285–297