Experiments with MRDTL -- A Multi-relational Decision Tree Learning Algorithm

Hector Leiva, Anna Atramentov and Vasant Honavar[1]

Artificial Intelligence Laboratory
Department of Computer Science and
Graduate Program in Bioinformatics and Computational Biology
Iowa State University
Ames, IA 50011, USA
www.cs.iastate.edu/~honavar/aigroup.html

Abstract. We describe experiments with an implementation of Multi-relational decision tree learning (MRDTL) algorithm for induction of decision trees from relational databases using an approach proposed by Knobbe et al. [1999a]. Our results show that the performance of MRDTL is competitive with that of other algorithms for learning classifiers from multiple relations including Progol [Muggleton, 1995], FOIL [Quinlan, 1993], Tilde [Blockeel, 1998]. Preliminary results indicate that MRDTL, when augmented with principled methods for handling missing attribute values, could be competitive with the state-of-the-art algorithms for learning classifiers from multiple relations on real-world data sets such as those used in the KDD Cup 2001 data mining competition [Cheng et al., 2002].

1 Introduction

Machine learning currently offers one of the most cost effective approaches to data-driven knowledge discovery [Mitchell, 1997]. Most learning algorithms (e.g., decision trees, Naïve Bayes) that are used in practice assume that each instance is represented in the form of a tuple of attribute values. The data set in this case can be viewed as a table (relation) where each row corresponds to an example or instance and each column to an attribute. In contrast, relational databases organize the data into several tables for reasons of efficient storage and access. Although in principle, it is possible to reconstruct a single relation by performing a relational join operation on the tables; such an approach is fraught with many difficulties in practice [De Raedt, 1998; Getoor, 2001]. Consequently, the task of learning from relational data or multi-relational learning, has begun to receive significant attention in the literature [Blockeel, 1998; De Raedt, 1998; Knobbe et al., 1999; Friedman et al., 1999; Koller, 1999; Krogel and Wrobel, 2001; Getoor, 2001; Kersting et al., 2000; Pfeffer, 2000; Dzeroski and Lavrac, 2001; Dehaspe and De Raedt, 1997; Dzeroski et al., 2001; Jae-

[1] This research was supported in part by grants from the National Science Foundation (9982341, 9972653), the Carver Foundation, and Pioneer Hi-Bred, Inc. This research has benefited from interactions with Adrian Silvecu, Doina Caragea, Jun Zhang, and Jaime Reinoso-Castillo of the Iowa State University Artificial Intelligence Research Laboratory.

ger, 1997; Karalic and Bratko, 1997]. KDD Cup competition organized in conjunction with the ACM SIGMOD Conference on Knowledge Discovery and Data Mining in 2001 included interesting data sets for multi-relational learning [Cheng et al., 2002]. The work described in this paper draws primarily on a framework for multi-relational learning proposed by Knobbe et al [1999a] that exploits Structured Query Language (SQL) to learn directly from data in a relational database. To the best of our knowledge, there are no experimental results available concerning the performance of the algorithm for induction of decision trees from a relational database proposed by Knobbe et al [1999a; 1999b]. This paper: briefly describes an implementation of this multi-relational decision tree learner (MRDTL); and compares the performance of MRDTL with several other approaches on some representative multi relational data sets including those used in KDD Cup 2001. Our results show that the performance of MRDTL is competitive with that of the state-of-the-art algorithms for learning classifiers from relational databases.

2 Multi-Relational Decision Tree Learning (MRTL)

2.1 Relational Database and Selection Graphs

A relational database consists of a set of *tables*, $\chi = \{X_1, X_2, ..., X_n\}$ and a set of associations between pairs of tables. These associations can be seen as constraints on how the records from one table relate to records in the other table. Both tables and associations are also known as relations. The columns in a table denote the attributes of that table and the rows correspond to records. Each table has a key attribute denoted by $X.K$ that uniquely identifies the records in it. The remaining attributes are either descriptive attributes or foreign key attributes. A foreign key attribute is a key attribute of another table. Foreign keys allow associations between tables to be defined. The nature of this relationship denotes the multiplicity (e.g., many-to-one, one-to-many) of the association. Let $X.A$ denote the descriptive (or foreign key) attribute A of table X. Let $D(X.A)$ denote the domain of $X.A$. In the case of foreign key attributes we need to define their domain type (Dom[F]) and range type (Range[F]). If $X.F$ is a foreign key attribute in table X and happens to be the primary key in table Y, then Dom[F] is X and Range[F] is Y. Most of associations in a relational database correspond to foreign key relations. These relations can be seen as having two directions. One goes from the table where the attribute is primary key to the table where the attribute is foreign and the other one in the reverse way. Let us call them F and F^{-1}, respectively. An object in a relational database can consist of several records fragmented across several tables and connected by associations. The entity-relationship diagram for a part of the mutagenesis database is shown in Fig 1. The figure shows that the database consists of three tables and four relations. A molecule has at least one atom, and at least one bond (this makes a total of at least two atoms per molecule). Table bond describes the type of relation between two atoms; therefore both atoms must be already in their respective table. The latter is represented by the two associations between atom and bond tables.

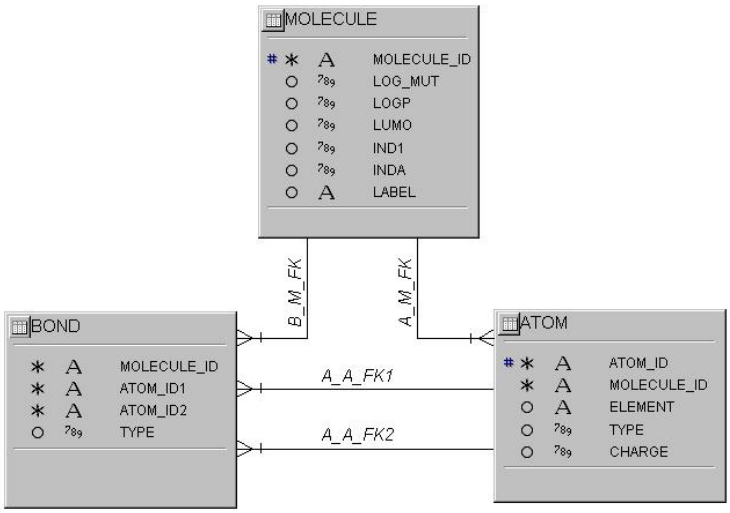

Fig. 1. Entity-relation diagram for Mutagenesis database

In the multi-relational data mining framework [Knobbe et al., 1999], although the data model can consist of several tables, each describing particular objects' features, only one view of the objects is central to the analysis. The user can choose the kind of objects to analyze by selecting one of the tables as target table (this table will be called $T_0 = X_i$ for some $i \in \{1,...,n\}$). The important point here is that each record in the target table will correspond to a single object in the database[2]. Once the target table has been selected, a particular descriptive attribute of that table can be chosen for classification or regression purposes. Such an attribute is known as the target attribute within the target table. Given a schema of a relational database and the physical implementation of it, the goal of multi-relational data mining is to find interesting patterns that not only involve attribute-value descriptions but also structural information denoted by the associations that can explain or predict the target attribute within the target table. Multi-relational patterns can be expressed in a graphical language of selection graphs [Knobbe et al., 1999b]. A selection graph G is a directed graph (N, E), where N is a set of triples (T, C, s) called selection nodes, T is a table in the data model and *C* is a, possibly empty, set of conditions on attributes in *T* of type $T.A \oplus c$; $\oplus$ is one of the usual selection operators =, <=, >, etc. *s* is a flag with possible values *open* and *closed*. *E* is a set of tuples (*p, q, a, e*) called *selection edges*, where *p* and *q* are selection nodes and *a* is an association between *p.T* and *q.T* in the data model. *e* is a flag with possible values *present* and *absent*. The selection graph contains at least one node n_0 that corresponds to the target table T_0. Selection graphs can be repre-

[2] This point is very important in the discussion of the experimental results.

sented as directed labeled graphs. An example is shown in Figure 2 based on the data model for Mutagenesis database shown in Figure 1. The current graph selects those molecules that have at least one atom whose partial charge is less than or equal to -0.392, but for which none of them have charge less than or equal to -0.392 and element equal to 'b' at the same time. Note that the value of *s* is represented by the presence or absence of a cross in the node, representing the value *open* and *closed* respectively. The value for *e*, in turn, is indicated by the presence (*present* value) or absence (*absent* value) of a cross on the corresponding arrow representing the edge. Thus, in this case, the target table is Molecule, and within molecule the target attribute is Label.

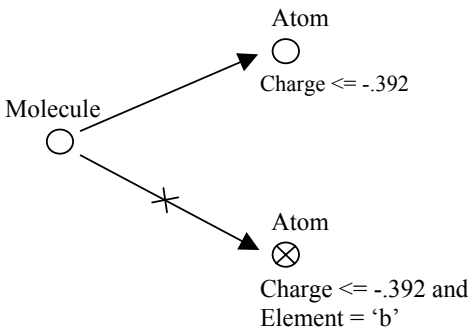

Fig. 1. Example of a selection for Mutagenesis database.

A *present* edge between tables *p* and *q* represents a join between these two tables using as the join attribute that one that is a primary key in *p* and a foreign key in *q*. This edge combined with a list of conditions selects those records that match the list of conditions and belong to the join. On the other hand, an *absent* edge between table p and *q* combined with a list of conditions *q*.C selects those records in *p* that *do not* match the list of conditions. Any subgraph that is pointed by an *absent* edge thus corresponds to a set of negative conditions. Knobbe et al [1999b] describe an algorithm for translating a selection graph into SQL.

The algorithm outputs a generic query of the form:

```
select distinct T0.primary_key
from table_list
where join_list
and condition_list
```

The graph in Figure 2 translates to the following SQL statement:

```
select distinct T0.mol_id
from Molecule T0, Atom T1
where T0.mol_id = T1.mol_id
and T1.charge <= -0.392
and T0.mol_id not in
        (select T2.mol_id
         from Atom T2
         where T2.charge <= -0.392
         and T2.element = 'b')
```

2.2 Multi-relational Decision Tree Learning (MRDTL) Algorithm

Multi-relational decision tree learning algorithm constructs a decision tree whose nodes are multi-relational patterns i.e., selection graphs. The algorithm described by Knobbe et al [1999b] which we have called MRDTL is an extension of the logical decision tree induction algorithm called TILDE proposed by Blockeel [1998]. TILDE uses first order logic clauses to represent decisions (nodes) in the tree. The data are represented in first order logic rather than a collection of records in a relational database. MRDTL extends TILDE's approach to deal with records in relational databases. Essentially, MRDTL, like the propositional version of the decision tree algorithm [Quinlan, 1993], adds decision nodes to the tree through a process of successive refinement until some termination criterion is met (e.g., correct classification of instances in the training set). The choice of decision node to be added at each step is guided by a suitable impurity measure (e.g., information gain) Thus, MRDTL starts with a single node at the root of the tree, which represents the set of all objects of interest in the relational database. This node corresponds to the target table T_0 together with the specified target attribute. The pseudocode for the algorithm is shown in Figure 3.

The function *optimal_refinement* considers every possible refinement that can be made to the current pattern G with respect to the database D and selects, in a greedy fashion, the optimal refinement (i.e., one that maximizes information gain). $\overline{G}$ denotes the complement of the selection graph (i.e., it selects objects from the database that are not selected by G).

```
Tree_induction(T: tree, D: database,
               G: selection graph)
  R := optimal_refinement(G, D)
  if stopping_criteria(G)
     T := leaf(G)
  else
     T_left  := R(G)
     T_right := R_complement(G)
     tree_induction(T_left, D, G)
     tree_induction(T_right, D, Ḡ)
  T := node(T_left, T_right, R)
```

Fig. 1. General outline of a decision tree learning algorithm.

Refinements

As noted above, once the target table and the target attribute have been selected (i.e. the kind of objects central to the analysis have been completely defined) a number of possible *refinements* can be made to the initial node that represents T_0 and successive

nodes in order to find a hypothesis to be consistent with the data in the training database.

Knobbe et al [1999a] describe several possible ways of refining selection graphs. These include specialization of the selection graph by adding a condition, adding an edge and a node, adding an edge between two already existing nodes as well as refinements that are designed to deal with the multiplicity of the associations in a relational database and the usefulness of look ahead in search. Space does not permit a detailed description of the refinement operators here. Our implementation essentially uses the operations described in [Blockeel, 1998; Knobbe et al., 1999b] with one important modification which allows us to consider any attribute within any table in the relational database to be a target attribute. We briefly describe this modification in what follows.

Consider the refinement `**Add *present* edge and *open* node'** which instantiates an association in the data model as a *present* edge together with its corresponding table represented as an *open* node and adds these to *G*. We find it useful to distinguish between two cases of this refinement. Generally, the associations in relational databases are one (from table *p*) to many (in table *q*). In this case, the key attribute in table *p* is a foreign attribute in table *q*. These associations can be seen as having two directions: *forward and backward*. Thus, we can add a present edge from the node representing the table where attribute *F* is primary key to the new open node that represents the table where *F* is foreign key (as proposed in Knobbe et al. [1999b]). This corresponds to adding an edge and a node in the *forward* direction. However, sometimes it is necessary to add an edge and a node in the *backward* direction. That is, given the current node where *F* is a foreign key, add a present edge to the node representing the node where *F* is primary key. This is especially necessary in learning tasks when the objects to be classified can have multiple class labels for a given assignment of values to their attributes. This would be the case when class labels are not mutually exclusive [Caragea et al., 2002].

Sometimes while refining a selection graph, the addition of the optimal refinement may result in no information gain. An example of this scenario occurs when adding an edge from an existing node to a new one if it so happens that the multiplicity of the association (represented by the edge) between the two tables is one to many and each record in the first table has at least one corresponding record in the other one. One approach to dealing with this scenario is to provide some sort of look ahead capability to the learner. In the TILDE system [Blockeel, 1998], such a look ahead capability allows several refinement steps at once.

Our implementation of MRDTL includes such look ahead capability which is employed when none of the refinements result in a positive information gain although the termination criterion of the algorithm is not met. The basic idea is to evaluate the impact of adding a new edge with its corresponding open node plus some condition on an attribute of the new node. In general, we can consider adding several edges and nodes as part of the look ahead process. An example of a refinement based on such look ahead is shown in Figure 4. The successive refinements performed in this case are to add an edge from Molecule to Atom with the corresponding node for Atom table and a condition (charge <= -.392) on table Atom. Because of the computational cost of performing look-ahead, we limit look ahead to 2-steps in our current implementation of MRDTL. The TILDE system relies on the user to provide some informa-

tion in order to determine when look ahead is needed [Blockeel, 1998]. Our implementation of MRDTL automatically determines from the relational schema and the current database instance when look ahead might be needed.

Computing Information Gain Associated with a Refinement

Each candidate refinement is evaluated in terms of the split of the data induced by the refinement with respect to the target attribute, as in the case of the prepositional ver-

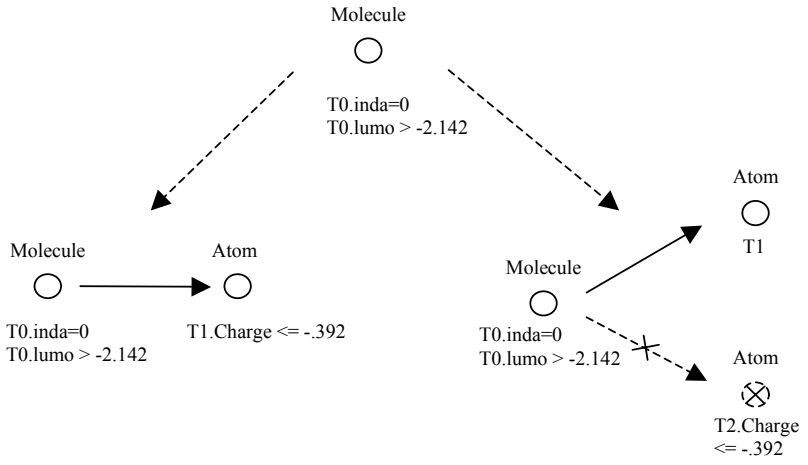

Fig. 1. Example of look-ahead refinement.

sion of the decision tree learning algorithm [Quinlan, 1993]. Splits based on numerical attributes are handled using a technique similar to that of C4.5 algorithm [Quinlan, 1993] with modifications proposed in [Fayyad and Irani, 1992, Quinlan, 1996]. Our current implementation of MRDTL uses existing SQL operations to obtain the counts needed for calculating information gain.

Classifying Instances

The hypothesis resulting from the relational induction of decision trees described can be viewed as a set of SQL queries associated with the selection graphs that correspond to the leaves of the decision tree. Each selection graph (query) has a class label associated with it. If the corresponding node is not a pure node, (i.e., it does not unambiguously classify the training instances that match the query), the label associated with the node can be based on the classification of the majority of training instances that match the corresponding selection graph. Alternatively, we can use probabilistic assignment of labels based on the distribution of class labels among the training instances that match the corresponding selection graph. The complementary nature of the different branches of a decision tree ensures that a given instance will not be assigned conflicting labels. It is also worth noting that it is not necessary to traverse the entire tree in order to classify a new instance; all the constraints on a certain path are

stored in the selection graph associated with the corresponding leaf node. Instances that do not match the selection graphs associated with any of the leaf nodes in the tree are assigned *unknown* label (and are counted as incorrectly classified when evaluating the accuracy of the tree on test data).

Handling Missing Values
The current implementation of MRDTL uses an extra value for each attribute to denote values that may be missing. However, we can easily incorporate a variety of more sophisticated approaches for handling missing values that have been proposed in the machine learning literature [Liu, 1997; Zheng and Low, 1999; Ortega and Numao, 1999; Mitchell, 1997; Quinlan, 1993; Witten and Frank, 2000].

Pruning
The current implementation of MRDTL does not incorporate any sophisticated pruning techniques although a variety of pruning methods developed in the decision tree literature [Witten and Frank, 2000; Mitchell, 1997] can be easily incorporated into the system.

2.3 Description of MRDTL Software

We have implemented MRDTL in Java using Oracle relational database. However, the system can be easily ported to other relational database systems. The front-end of the system is a rudimentary GUI which allows the user to connect to the database to be mined and define in an easy way the numerical and nominal attributes as well as delete those attributes that are not central or of interest to the classification task in question. It also enables the user to specify the target table and attribute. Finally, an error percentage can be provided by the user. This error percentage can be used to terminate refinement of a path in the decision tree if the error criterion is satisfied by the current node (this serves as a crude pruning method). The rest of the information used by the algorithm is extracted from the relational schema of the database. The result of learning (i.e., the selection graphs associated with the decision tree) is stored into the relational database for future use.

3 Experimental Results

Our experiments focused on two data sets – the mutagenesis database available from MLNet which has been widely used in Inductive Logic Programming (ILP) research [Sreenivasan, 1996]; and the data for prediction of gene localization from KDD Cup 2001 [Cheng et al., 2002].

3.1 Experiments with the Mutagenesis Dataset

The entity-relation diagram for the part of the Mutagenesis database used in this study is shown in Figure 1. The data set consists of 230 molecules divided into two subsets: 188 molecules for which linear regression yields good results and 42 molecules that are regression-unfriendly. This database contains descriptions of molecules and the

characteristic to be predicted is their mutagenic activity (ability to cause DNA to mutate) represented by attribute *label* in molecule table. This target attribute has two possible values: *active* for those molecules with positive levels of mutagenicity, and *inactive* for those with zero or negative levels of mutagenicity. Table 4.1 shows the class distribution for the 230 compounds [Srinivasan, et al., 1996].

Table 1. Class distribution for Mutagenesis database

Compounds	Active	Inactive	Total
Regression friendly	125	63	188
Regression unfriendly	13	29	42
Total	138	92	230

A recent study using this database [Srinivasan et al., 1999] recognizes five levels of background knowledge about mutagenesis which can provide richer descriptions of the examples. In this study we used only the first three levels of background knowledge in order to compare the performance of MRDTL with other methods for which experimental results are available in the literature.

As mentioned earlier, the data can be split into two disjoint subsets. To avoid bias against linear regression [Srinivasan et al., 1996] and to make our results comparable to those in the literature we treat them as two separate subsets. The classification results obtained from the theory inferred by the learner implemented here are averages over *k*-fold cross-validation; with $k = 10$ for the regression friendly data set and with $k = 41$ (i.e., leave one out crossvalidation) for the regression unfriendly data.

3.1.1 Experimental Results on Mutagenesis Data Set

Table 3 shows comparison of the performance of our current implementation of MRDTL with that of Progol (based on results reported by Blockeel [1998] and [Srinivasan et al., 1999]), FOIL (based on results reported by Blockeel [1998], and Tilde (based on results reported in Blockeel [1998]). The results are based on 10-fold crossvalidation on the regression-friendly subset of the mutagenesis data. Note that the running times are shown mainly to give a general feel for the speed of the algorithms in question. The exact values of the running times are not comparable because of the differences in hardware and software platforms used in the different studies. The results shown in Table 4 suggest that the performance of MRDTL is competitive with that of Progol, FOIL, and Tilde.

Results shown in Table 2 and Table 4 show that the richer description of examples obtained using background knowledge does in fact improve the accuracy of the resulting classifiers. In the case of MRDTL, the improvement obtained by using *B1* as opposed to *B0* is quite pronounced but the improvement resulting from the use of B2 as opposed to *B1* is marginal. It is also clear that different algorithms benefit to varying degrees from the use of different levels of background knowledge. For instance, use of *B1* as opposed to *B0* results in no improvement in FOIL's performance on the re-

gression friendly mutagenesis data (see Table 2). This can be explained by the differences in the representational and inductive biases of the algorithms in question.

Table 2. Accuracy and running time comparisons of Progol, FOIL, Tilde and MRDTL on the set of regression friendly compounds of Mutagenesis database. The numbers represent averages based on ten-fold cross-validation. Entries marked with "--" indicate that the corresponding results are not available at present.

Systems	**Accuracy (%)**					**Time (secs.)**				
	B0	*B1*	*B2*	*B3*	*B4*	*B0*	*B1*	*B2*	*B3*	*B4*
Progol[3]	79	86	86	88	89	8695	4627	4974	6530	9587
Progol[4]	76	81	83	88	--	117k	64k	42k	41k	--
FOIL[5]	61	61	83	82	--	4950	9138	0.5	0.5	--
Tilde[6]	75	79	85	86	--	41	170	142	352	--
MRDTL	67	87	88	--	--	0.85	332	221	--	--

Table 3. Comparison of size of decision trees obtained with the different levels of background knowledge for MRDTL.

System	**Number of nodes**		
	B0	*B1*	*B2*
MRDTL	1	53	51

It is interesting to note from Tables 3 and 4 that MRDTL yields a decision tree with one node when *B0* is used as background knowledge. In this case, in the current implementation of MRDTL, the instances are assigned the class label of the majority class. (Alternatively, a probabilistic assignment of class labels based on the class distribution at that node could be used). Thus, the classification accuracy (67%) obtained with a single node decision tree can be considered a lower bound on accuracy against which we can compare the different algorithms. Ideally, none of the algorithms should have accuracy lower than 67% on the regression friendly subset of the mutagenesis data. We find that this is in fact true for all of the algorithms studied with the exception of FOIL which has an accuracy of 61% (when *B0* and *B1* are used as background knowledge).

[3]Results for this row were extracted from [Srinivasan et al., 1999].
[4] Results for this row were extracted from [Blockeel, 1998].
[5] Results for FOIL were taken from [Blockeel, 1998].
[6] Results for Tilde were taken from [Blockeel, 1998].

Table 4. Accuracy, running time, and decision tree size obtained on the MRDTL on the set of 42 regression unfriendly compounds of Mutagenesis database. The numbers represent averages based on 42-fold (i.e., leave-one out) cross-validation.

Background	Accuracy	Time	#Nodes
B0	70 %	0.6 secs.	1
B1	81 %	86 secs.	24
B2	81 %	60 secs	22

Another point worth noting is that MRDTL results in smaller trees when *B2* is used as background knowledge as compared to *B1* and the execution time in the case of B2 is lower than that for *B1*. With exception of Progol, the minimum execution time is obtained with *B0*, followed by *B2* and *B3*.

The results from leave one out method for the second set of compounds are shown in Table 4 for MRDTL. Essentially, MRDTL's behavior on the regression unfriendly subset of the Mutagenesis data shows the same general pattern as that in the case of regression-friendly subset. Use of background knowledge *B0* yields a single node decision tree which classifies instances according to the label of the majority class. This can be corroborated on the basis of the distribution of instances shown in Table 1: A majority (70%) of the regression unfriendly instances are inactive compounds.

3.2 KDD Cup 2001 Data

Several interesting data sets that were used in the KDD Cup data mining competition held in conjunction with the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2001) are available on line [Cheng et al., 2002]. These data sets are drawn from genomics and drug design applications. The three tasks that KDD Cup 2001 involved were: prediction of molecular bioactivity for drug design, prediction of gene/protein function and localization. Of these, the data sets for the latter are in relational form. However, in the case gene/protein function prediction, an instance may have several class labels. MRDTL, much like its propositional counterpart C4.5, assumes that each instance can be assigned to only one of several non-overlapping classes, we in its current form, decided to focus our experiments on the gene/protein localization task where each instance has a single class label. We will use the name *Gene Localization database* to refer to the data for the gene/protein localization task.

KDD Cup 2001 provided two variants of the data set for the gene/protein localization task. The first version consists of a single table with 2960 attributes. The second consists of two tables with 13 attributes in total. We used a *normalized* version of the data set with 2 tables. The names of the two original tables are *genes_relation* and *interactions_relation*. The *genes_relation* table contains 862 different genes but there could be more than one row in the table for each gene. The attribute *gene_id* identifies a gene is uniquely. Since our current implementation of MRDTL requires that the target table must have a primary key, it was necessary to normalize the *genes_relation* table before we can use it as the target table. This normalization was achieved by cre-

ating the tables named *gene, interaction, and composition* as follows: Attributes in the *genes_relation* table that did not have unique values for each gene were placed in the *composition* table and the rest of the attributes were placed in the *gene* table. The *gene_id* attribute is a primary key in the *gene* table and as a foreign key in the *composition* table. The *interaction* table is identical to the original *interactions_relation* table. This represents one of several ways of normalizing the original table. This renormalization of the relational database given is different from the one described in Cheng et al. [2002]. It is possible that the particular renormalization used might have an impact on the performance of different algorithms.

The entity-relation diagram for the renormalized version of the *Gene Localization database* is shown in Figure 5. Thus, for the gene/protein localization task, the target table is *gene* and the target attribute is *localization*. From this point of view, the training set consists of 862 genes and the test set 381. (If we want to predict *function*, the number of instances in the training set will be 4346, that is, the number of records in *composition* table). The experiments described here focused on building a classifier for predicting the localization of proteins by assigning the corresponding instance to one of 15 possible localizations.

For this task, we followed the general procedure used in the KDD Cup competition namely, construct a classifier using all of the training data and test the resulting classifier on the test set provided [Cheng et al., 2002].

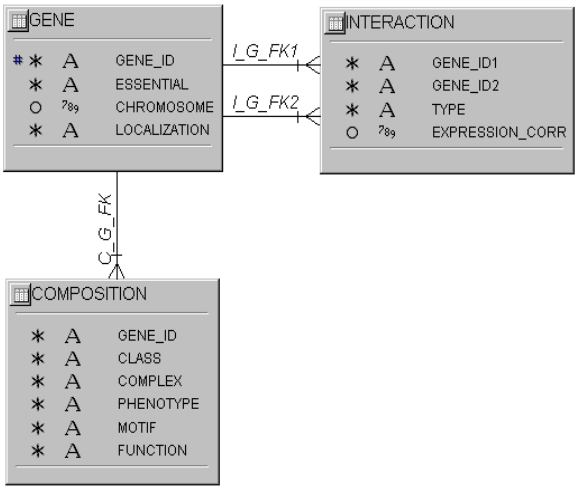

Fig. 2. Entity-relation diagram for *Gene Localization* data set

Gene localization task presents significant challenges because many attribute values in the training instances corresponding to the 862 training genes are missing. Initial experiments using a special value (*missing*) to encode a missing value for an attribute resulted in classifiers whose accuracy is around 50 % on the test data.

This prompted us to investigate incorporation of more sophisticated approaches to handling missing values that have been proposed in the literature. Replacing missing values by the most common value of the attribute for the class during training resulted in an accuracy of around 85%. This shows that providing reasonable guesses for missing values can significantly enhance the performance of MRDTL on real world data sets. However, in practice, since the class labels for test data are unknown, it is not possible to replace a missing attribute value by the most frequent value for the class during testing. Hence, there is a need for incorporating more sophisticated techniques for handling missing values (e.g., predicting missing values based on the values of other attributes). Work in progress is aimed at extending the current implementation of MRDTL to include principled approaches for dealing with missing values that can be applied in a realistic setting.

4 Summary and Discussion

Learning from relational databases is an important problem in machine learning. In this paper, we have described an implementation of Multi-relational decision tree learning (MRDTL) algorithm, based on the techniques proposed by Knobbe et al [1999a]. Results of our experiments on the widely used benchmark Mutagenesis data set indicate that MRDTL offers a promising alternative to existing algorithms such as Progol [Muggleton, 1995], FOIL [Quinlan, 1993], Tilde [Blockeel, 1998]. Preliminary results of our experiments with the protein/gene localization task (based on the data from the KDD Cup 2001 competition) indicate that MRDTL, if equipped with principled approaches to handling missing attribute values, can be an effective algorithm for learning from relational databases in real-world applications.

Work in progress is aimed at:

- Incorporation of sophisticated methods for handling missing attribute values into MRDTL
- Incorporation of sophisticated pruning methods or complexity regularization techniques into MRDTL to minimize overfitting and improve generalization
- More extensive experimental evaluation of MRDTL on real-world data sets
- Development of ontology-guided multi-relational decision tree learning algorithms to generate classifiers at multiple levels of abstraction (based on the recently developed prepositional decision tree counterparts of such algorithms [Zhang et al., 2002]
- Development of variants of MRDTL for classification tasks where the classes are not disjoint, based on the recently developed propositional decision tree counterparts of such algorithms [Caragea et al., 2002]
- Development of variants of MRDTL that can learn from heterogeneous, distributed, autonomous data sources based on recently developed techniques for distributed learning [Caragea et al., 2001a; Caragea et al., 2001b] and ontology-based data integration [Honavar et al., 2001; Honavar et al., 2002; Reinoso-Castillo, 2002].
- Application of multi-relational data mining algorithms to data-driven knowledge discovery problems in bioinformatics and computational biology.

Bibliography

[Blockeel, 1998] Blockeel, H. Top-down induction of first order logical decision trees. PhD dissertation, Department of Computer Science, Katholieke Universiteit Leuven, 1998.

[Caragea et al., 2001a] Caragea, D., Silvescu, A., and Honavar, V. **Invited** Chapter. Towards a Theoretical Framework for Analysis and Synthesis of Agents That Learn from Distributed Dynamic Data Sources. In: *Emerging Neural Architectures Based on Neuroscience*. Berlin: Springer-Verlag, 2001.

[Caragea et al., 2001b] Caragea, D., Silvescu, A., and Honavar, V. Learning Classification Trees from Distributed Data. Technical Report TR-2001-10, Department of Computer Science, Iowa State University, Ames, Iowa 50011-1040, 2001.

[Caragea, 2002] Caragea, D., Silvescu, A., and Honavar, V. Learning Decision Tree Classifiers When the Classes are not Disjoint. In preparation, 2002.

[Cheng et al., 2002] Cheng, J., Krogel, M., Sese, J., Hatzis, C., Morishita, S, Hayashi, H., and Page, D. KDD Cup 2001 Report. *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations*, vol. 3, issue 2, January 2002. http://www.acm.org/sigs/sigkdd/explorations/

[Dehaspe et al., 1997] Dehaspe, L., and De Raedt, L.. Mining association rule sin multiple relations. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, vol. 1297of *Lecture Notes in Artificial* Intelligence, Springer-Verlag, 1997.

[De Raedt, 1998] De Raedt, L. Attribute-value learning versus Inductive Logic Programming: the Missing Links (Extended Abstract). In *Proceedings of the 8th International Conference on Inductive Logic Programming*, volume 1446 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 1998.

[Dzeroski and De Raedt, 2001] Dzeroski, S., De Raedt, L., and Driessens, K.. Relational Reinforcement Learning. Report CW 311, Department of Computer Science, K. U. Leuven, 2001.

[Dzeroski and Lavrac, 2001] Dzeroski, S. and Lavrac, N, editors. Relational Data Mining. Springer-Verlag, 2001.

[Fayyad and Irani, 1992] Fayyad, U. M., and Irani, K. B. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, vol. 8, 1992.

[Friedman et al., 1999] Friedman, N., Getoor, L., Koller, D., and Pfeffer, A. Learning probabilistic relational models. In *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, 1999.

[Getoor, 2001] Getoor, L. Multi-relational data mining using probabilistic relational models: research summary. In: A. J. Knobbe, and D. M. G. van der Wallen, editors. *Proceedings of the First Workshop in Multi-relational Data Mining*, 2001.

[Honavar et al., 2002] Honavar, V., Andorf, C., Caragea, D., Dobbs, D., Reinoso-Castillo, J., Silvescu, A. **Invited** Chapter. Algorithmic and Systems Solutions for Computer Assisted Knowledge Acquisition in Bioinformatics and Computational Biology. In: *Computational Biology and Genome Informatics*. Wu, C., Wang,P., and Wang, J. (Ed.) World Scientific, 2002.

[Honavar et al., 2001] Honavar, V., Silvescu, A., Reinoso-Castillo, J., Andorf, C., and Dobbs, D. Ontology-Driven Information Extraction and Knowledge Acquisition from Heterogeneous, Distributed Biological Data Sources. In: *Proceedings of the IJCAI-2001 Workshop on Knowledge Discovery from Heterogeneous, Distributed, Autonomous, Dynamic Data and Knowledge Sources*, 2001.

[Jaeger, 1997] Jaeger, M. Relational Bayesian networks. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence (UAI-1997)*, 1997.

[Karalic and Bratko, 1997] Karalic and Bratko, I. First order regression, *Machine Learning* 26, 1997.

[Kersting and De Raedt, 2000] Kersting, K., and De Raedt, L. Bayesian Logic Programs. In *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 2000.

[Knobbe et al., 1999a] Knobbe, J., Blockeel, H., Siebes, A., and Van der Wallen, D. M. G. Multi-relational Data Mining. In *Proceedings of Benelearn '99*, 1999.

[Knobbe et al, 1999b] Knobbe, J., Siebes, A., and Van der Wallen, D. M. G. Multi-relational decision tree induction. In *Proceedings of the 3rdEuropean Conference on Principles and Practice of Knowledge Discovery in Databases,* PKDD '99, 1999.

[Koller, 1999] Koller, D. Probabilistic Relational Models. In S. Dzeroski and P. Flach, editors, *Proceedings of 9th International Workshop on Inductive Logic Programming (ILP-99)*, Springer, 1999.

[Krogel and Wrobel, 2001] Krogel, M., and Wrobel, S. Transformation-Based Learning Using Multirelational Aggregation. In Céline Rouveirol and Michèle Sebag, editors, *Proceedings of the 11th International Conference on Inductive Logic Programming*, vol. 2157 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, 2001.

[Liu et al., 1997] Liu, W. Z., White, A. P., White, S. G., Thompson, S. G., and Bramer, M. A. Techniques for dealing with missing values in classifications. In X. Liu, P. Cohen and M. Berthold, editors, *Advances on Intelligent Data Analysis*, Springer-Verlag, 1997.

[Mitchell, 1997] Mitchell, T. Machine Learning. McGraw Hill, 1997.

[Muggleton, 1995] Muggleton, S. Inverse entailment and progol. *New Generation Computing*, 1995.

[Ortega, 1999] Ortega, O., and Numao, M. Ordered estimation of missing values. In *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, Springer-Verlag, 1999.

[Pfeffer, 2000] Pfeffer, A. A Bayesian Language for Cumulative Learning. In *Proceedings of AAAI 2000 Workshop on Learning Statististical Models from Relational Data*, AAAI Press, 2000.

[Quinlan, 1993] Quinlan, J. R. C4.5: Programs for Machine Learning. San Mateo: Morgan Kaufmann, 1993.

[Quinlan, 1996] Quinlan, J. R. Improved Use of Continuous Attributes in C4.5. *Journal of Artificial Intelligence Research*, vol. 4, 1996.

[Reinoso-Castillo, 2002] Reinoso-Castillo, J. Ontology-Driven Query-Centric Federated Solution for Information Extraction and Integration from Autonomous, Heterogeneous, Distributed Data Sources. Masters Thesis. Department of Computer Science. Iowa State University. 2002.

[Srinivisan et al., 1996] Srinivasan, A., Muggleton, S., Sternberg, M., and King, R. Theories for mutagenicity: a study in first-order and feature-based induction. *Artificial Intelligence*, 85, 1996.

[Srinivisan et al., 1999] Srinivasan, A., King, R. D., and Muggleton, S.. The role of background knowledge: using a problem from chemistry to examine the performance of an ILP program. Technical Report PRG-TR-08-99, Oxford University Computing Laboratory, Oxford, 1999.

[Witten and Frank, 2000] Witten, I. H, and Frank E. Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.' Morgan Kaufmann Publishers, San Francisco, CA, 2000.

[Zhang et al., 2002] Zhang, J., Silvescu, A., and Honavar, V. Ontology-Driven Induction of Decision Trees at Multiple Levels of Abstraction. In: Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA-2002). Kananaskis, Alberta, Canada, 2002.

[Zheng and Low, 1999] Zheng, Z., and Low, B.T. Classifying unseen cases with many missing values. In *Proceedings of the 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, Springer-Verlag, 1999.