A unifying framework for relational distance-based learning founded on relational algebra

Alexandros Kalousis[1], Adam Woznica[1], and Melanie Hilario[1]

University of Geneva,
Computer Science Department,
Rue General Dufour, 1211, Geneve, Switzerland
{kalousis,woznica,hilario}@cui.unige.ch

Abstract. We propose a novel unifying framework for relational distance based learning where learning examples are stored in a relational database. The framework is tailored to and fully supports relational algebra representations. We define relational distances whose building blocks are distances between tuples of relations and distances between sets. Unlike all existing distance-based relational systems our framework is not limited to a single relational distance measure, instead it offers a variety among which the user can choose or combine the ones that best match the requirements of the problem at hand, or even define new ones if the existing do not satisfy the application requirements. Moreover to what amounts to model selection the most appropriate relational distance measure can be automatically selected via means of an inner cross validation. We study the properties of the relational distance and show how these are related to the properties of its constituent parts. We evaluate our framework in the context of the classification task. We perform a series of experiments on a number of well known classification datasets used in relational learning, analyze the relative performance of the different relational distance measures induced, and try to highlight their similarities and differences. The experiments show that our framework competes favorably and in some cases is better than state of the art systems used in relational learning. We believe that the fact that the framework is completely defined on the basis of relational algebra and relational algebra operators has the potential to make it, and thus relational learning, much more accessible to the large community of relational databases.

1 Introduction

Distance-based learning is one of the most prominent paradigms used in machine learning. It has been used extensively in the tasks of classification, clustering and regression. In classification it is known under various names such as k-nearest neighbor classification, instance-based learning or lazy learning, (Duda et al., 2001a; Aha et al., 1991; Aha, 1997). In clustering it is probably the most widely used approach, exploited in methods like k-means clustering, hierarchical or

agglomerative clustering, self organized maps, (Duda et al., 2001b). In regression it has been used to perform local regression where regression models are fitted locally within neighbors of the learning examples, (Hastie et al., 2001).

Although intuitively simple distance-based learning has proved its utility in many applications, especially when a solid description of the learning problem is available. Its most important constituents are the definition of an appropriate representation of the learning examples and the definition of a distance measure over that description. Both of them are directly determined by domain knowledge and the application requirements.

Today's machine learning applications become more and more complex and they hardly fit within the typical propositional representations assumed by the typical learning approaches. As a result the field of relational data mining has developed and flourished over the last fifteen years, (Dzeroski & Lavrac, 2001). One of its most prominent representatives is the subfield of Inductive Logic Programming which is usually described as the intersection of machine learning and logic programming. Within the ILP paradigm examples are described in terms of clauses, terms, in general with first order logic concepts or subsets of first order logic. A number of relational distance-based approaches have been developed within the ILP framework, (Nienhuys-Cheng, 1998; Hutchinson, 1997; Horvath et al., 2001; Ramon & Bruynooghe, 1998).

In this paper we adopt a different representational approach from the logic programming approach typically employed within the ILP community. We define relational distances using solely concepts from relational algebra. Relational databases are probably the most common way of storing structured information nowadays. Moreover relational algebra provides a robust data modeling tool with well understood semantics by a large audience. The advantage of this approach is that it can readily tackle any kind of relational problem in which training data are stored within a typical relational database with no need for a change of representation. The above factors increase the chances of acceptance of the system, and in general that of relational learning, by the community of the relational database practitioners.

To define a relational distance within the scope of relational algebra there are two important building blocks. The first is the definition of a distance over the tuples of any given relation, and the second the definition of a distance between sets. Once these are defined a relational distance can be derived by combining them.

All existing relational distance-based approaches are constrained to a single monolithic type of relational distance. Nevertheless it is obvious that there is no single distance measure that is overall better than any other for all types of problems. Typically the analyst should consider different systems to find the one that best matches the problem requirements. To make things even more complicated it can be that sets of different types, e.g. sets of tuples coming from different relations long for different set distance measures. Thus one can imagine that the final relational distance measure is derived by a combination of heterogeneous set distance measures where for each relation the most appropriate

set distance is selected. None of the existing relational distance-based systems offers this type of flexibility.

Our framework overcomes these limitation by clearly disassociating the two types of basic distances, i.e. distances defined on tuples of relations and distances defined on sets. It offers a variety of set distance measures from which the user can either choose a single one to be applied on any type of sets or specify the type of set distance that should be used for sets of a given type, i.e. sets of tuples coming from specific relations. In the latter case the final relational distance will be a heterogeneous relational distance assembled by different elementary components. Furthermore in the absence of domain knowledge that could guide the selection of the appropriate distance and in what, loosely speaking, amounts to model selection the system can automatically select the most appropriate homogeneous relational distance by choosing from the pool of available set distance measures via means of inner cross validation. Finally in the case that none of the existing distances satisfies the requirements of a given domain it is straightforward to define and use new domain dependent distance measures.

The rest of the paper is organized as follows: in section 2 we give a short description of some concepts of relational algebra, and show how we exploit these to traverse a given relational schema and construct our relational instance representations. An important notion here is that of foreign keys which will provide the basis for the definition of a new type of attributes, the set type, that will explicitly model for the concept of sets. In section 3 we go through a review of notions of distance measures and examine in detail various distance measures for sets. In section 4 we show how we can combine a simple distance on tuples of relations and distances on sets of tuples in order to derive a distance between relational instances. We study the formal properties of the relational distance and how they relate to the properties of the constituent distances, and we explain how relations contribute to the actual value of the final distance. In section 5 we systematically experiment with and evaluate the different options offered by the system. We evaluate the derived relational distances in a number of different classification problems and try to shed some light on their similarities and differences; we evaluate the performance of the model selection procedure in the same classification problems; we take a closer look on the performance of the Tanimoto set distance, which we have extended so that it can handle graded similarities, namely because it is the only one that has a tunable parameter; finally in its last subsection, 5.8, we define, use, and study a domain specific distance, based on an extension of the classical edit distance that can account for graded similarities, for the problem of classification of mass spectra. In section 6 we show how our relational learning approach relates with two of the most common learning approaches used in ILP, namely learning from entailment and learning from interpretations. In section 7 we give an overview of the related work and place our framework within that context. We conclude in section 8 where we address the open issues and the future work.

2 Relational Algebra and Relational Instances

The following relational algebra definitions are adapted from Ullman (1982). A relation R_i is a set of tuples, more specifically a subset of some Cartesian product $D_1 \times \ldots \times D_{z_i}$ where each D_l is a domain (i.e., a set of values-data type). The relation schema of a relation R_i is the set of attributes of R_i and we denote it as $R_i(A_1, \ldots, A_{z_i})$. Each attribute A_l has an associated *domain* $dom(A_l) = D_l$. The number of attributes z_i of a relation R_i is called the *arity* of the relation. A tuple R_{i_j} of a relation R_i is a particular row in R_i with $R_{i_j} = (v_{j1}, v_{j2}, \ldots, v_{jz_i})$ and v_{jl} the value of the A_l attribute in the R_{i_j} tuple, v_{jl} can also be denoted as $R_{i_j}.A_l$. A relational database schema is a set of relation schemes $\mathcal{R} = \{R_1, \ldots, R_n\}$.

An attribute $A_k \in R_i(A_1, \ldots, A_{z_i})$ is called a *potential key* of relation R_i if it assumes a unique value for each instance of the relation. An attribute $A_l \in R_j(A_1, \ldots, A_{z_j})$ of relation R_j is a *foreign key* if it references a potential key A_k of relation R_i, in that case we will also call A_k a *referenced key*. The association between A_k and A_l models one-to-many relations, i.e. one element of R_i can be associated with a set of elements of R_j. A *link* is defined on the basis of a foreign key relation and is a quadruple of the form $l(R_i, A_k, R_j, A_l)$ where either A_l is a foreign key of R_j referencing a potential key A_k of R_i or vice versa. The notion of links build on top of the foreign key relations is critical for our relational representation since it will provide the basis for the definition of the set type that lies in the core of our relational representation.

In the next sections we will describe how we can exploit the structures provided by relational algebra in order to define a general representation for relational learning problems.

2.1 Relation Level Access

For a given referenced key A_l of relation R_i we denote by $L(R_i, A_l)$ the set of links $l(R_i, A_l, R_j, A_k)$ in which A_l is referenced as a foreign key by A_k of R_j. By $L(R_i) = \cup_l L(R_i, A_l)$ we denote the set of all links in which one of the potential keys of R_i is referenced as a foreign key by an attribute of another relation. The multiset[1] of R_j relations with which R_i is associated via the $L(R_i)$ links are the *directly dependent* relations of R_i.

Similarly for a given foreign key A_l of R_i, $L^{-1}(R_i, A_l)$ will return the link $l(R_i, A_l, R_j, A_k)$ where A_k is a potential key of R_j referenced by the foreign key A_l of R_i. If R_i has more than one foreign keys then by $L^{-1}(R_i) = \cup_l L^{-1}(R_i, A_l)$ we denote the set of all links of R_i defined by the foreign keys of R_i. The multiset of relations R_j to which R_i is associated via the $L^{-1}(R_i)$ links are the *directly referenced* relations of R_i.

The complete set of links of a given relation R_i is given by $L(R_i) \cup L^{-1}(R_i)$. In one sense a relation R_i is described in terms of its directly dependent and

[1] The term *multiset* is more appropriate than the term *set*, since there can be a relation R_j that appears more than once in the dependent relations, e.g. it has two foreign keys pointing to attributes of R_i.

referenced relations, as these are given by $L(R_i) \cup L^{-1}(R_i)$, which in their turn are also described in terms of their directly dependent and referenced relations in a recursive manner.

2.2 Relational Instance

We will now show how we can use the relational algebra formalism presented so far in order to define the representation of a relational instance.

Before presenting the exact procedure we will now define the new type of attribute that we call *set* attribute and which will directly model for the notion of sets. An attribute of type set is an attribute that, in the general case, takes as value a *set* of relational instances. This type of attribute includes as special cases: classical attributes that can take only a single value, attributes whose value is a set of values and attributes whose value is a set of tuples from a given relation. In the case of the attribute value representation the set actually reduces to a single element set where the element is described by a single attribute; in the case of an attribute that assumes a set of values we have a set of elements that are described also by a single attribute.

As already mentioned the attributes of type set are based on the notion of links. In fact each link in which a relation R_i participates will give rise to an attribute of type set. We will denote the set of attributes of a relation that are of type set with $\mathcal{I}_{L,R_i}$. We will call the set of attributes of a relation R_i that are not keys (i.e. referenced keys, foreign keys or attributes defined as keys but not referenced) *standard* attributes and denote it with $\mathcal{I}_{A,R_i}$.

We will now describe how we can retrieve the description of a relational instance. Each tuple-instance, R_{i_a}, of a relation, R_i, can give rise to a *relational instance*, $R^+_{i_a}$. A relational instance is defined recursively in terms of the instances with which R_{i_a} is related. It is a tree like structure whose root contains R_{i_a}. Each node at the first level of the tree is a set of instances from some relation $R_j \in \mathcal{R}$ related via a link $l(R_i, A_l, R_j, A_k)$ with instance R_{i_a}. In the same way nodes at the d level of the tree are also sets of instances from a given relation. Each of these sets is related with one of the instances found in a set of a node of the $d-1$ level. One can view a relational instance $R^+_{i_a}$ as that snapshot of the dataset that we get when we start from instance R_{i_a} of R_i and retrieve instances by recursively following the links defined in the relational schema.

To get the complete description of $R^+_{i_a}$ one will have to traverse possibly all the relational schema according to the relation associations, i.e., links, defined in the schema. This is done via the recursive application of the function $R^+_t(R_i, R_{i_a}, \cdot, \cdot)$ (algorithm 1), on the associated tuples of R_{i_a} in the relations given by the links $L(R_i)$, $L^{-1}(R_i)$. The function takes as input an instance R_{i_a} of a relation R_i for which it returns its corresponding relational instance $R^+_{i_a}$. The reason we follow links in both directions is that an entity is described both in terms of the entities it refers to, $L^{-1}(R_i)$, but also in terms of the entities they refer to it $L(R_i)$.

More precisely for a given tuple, R_{i_a}, of any relation, R_i, the function $R^+_t(R_i, R_{i_a}, \cdot, \cdot)$ will create a relational instance $R^+_{i_j}$ that will have the same set of stan-

dard attributes $\mathcal{I}_{A,R_i}$ and the same values for these attributes as R_{i_a} has (algorithm 1 lines 14-16). Furthermore for each link $l(R_i, A_l,\ R_j, A_k) \in L(R_i) \cup L^{-1}(R_i)$ it will add in $R^+_{i_j}$ one attribute of type set, constructing like that the set attributes, $\mathcal{I}_{L,R_i}$. The value of an attribute of type set is defined based on the link l with which the attribute is associated and it will be the set of tuples—relational instances— with which R_{i_a} is associated in relation R_j when we follow l (algorithm 1 lines 19-21). This set is retrieved via the application of the function $set(R_{i_a}.A_l, R_j, A_k, \cdot, \cdot)$ (algorithm 1 line 20).

Algorithm 1 Retrieving a tree description of a relational instance.

1: $R^+_t(R_i, R_{i_a}, d, instStack)$
2: $\{R_{i_a}$: instance for which we want to create its relational instance, $R^+_{i_a}.\}$
3: $\{R_i$: the relation to which R_{i_a} belongs to.$\}$
4: $\{d$: the depth of recursion to which we are.$\}$
5: $\{instStack$: instances visited so far in the current recursion path.$\}$
6:
7: **if** $R_{i_a} \in$ instStack **then**
8: return {Came to a loop}
9: **else**
10: stack.push(R_{i_a})
11: **end if**
12:
13: {Get the values of all standard attributes (i.e. attribute-value) for $R^+_{i_a}$}
14: **for** $A_l \in \mathcal{I}_{A,R_i}$ **do**
15: $R^+_{i_a}.A_l \leftarrow R_{i_a}.A_l$
16: **end for**
17:
18: {Recuperate the values of all set attributes for $R^+_{i_a}$}
19: **for all** $l(R_i, A_l, R_j, A_k) \in L(R_i) \cup L^{-1}(R_i)$ **do**
20: $R^+_{i_a}.A_l \leftarrow set(R_{i_a}.A_l, R_j, A_k, d+1, instStack)$
21: **end for**
22: return $R^+_{i_a}$

1: $set(R_{i_a}.A_l, R_j, A_k, d, instStack)$
2: $set = \{R_{j_m} \in R_j : R_{i_a}.A_l == R_{j_m}.A_k\}$
3: $set' = \emptyset$
4: **for all** $R_{j_b} \in set$ **do**
5: $R^+_{j_b} \leftarrow R^+_t(R_j, R_{j_b}, d, instStack)$
6: $set' = set' \cup R^+_{j_b}$
7: **end for**
8: return set'

The *set* function, (given in the second part of algorithm 1), first performs a simple SQL query which returns the set of tuples of relation R_j for which $A_k == R_{i_a}.A_l$, i.e. the set of tuples related with R_{i_a} in the R_j relation (algorithm 1, set

function, line 2). Then it returns the corresponding set of relational instances computed by the R_t^+ function for each of the elements of the initial set.

To summarize a relational instance $R_{i_a}^+$ consists of two parts. The first one corresponds to the set of $\mathcal{I}_{A,R_i}$ attributes of R_{i_a}, while the second one to the attributes of type set, $\mathcal{I}_{L,R_i}$, constructed on the basis of the links in which $R_{i_a a}$ participates.

Traversing the relational schema in order to retrieve the complete description of a given relational instance can easily produce self replicated loops. For example when we follow twice in the row the same link: if we are at an instance R_{i_j} of relation R_i and we follow the link $l(R_i, A_l, R_j, A_k)$ whose opposite link, i.e., $l(R_j, A_k, R_i, A_l)$ lead us to R_{i_j} then among other instances of relation R_j we will also visit again that instance of R_j that brought us to R_{i_j} whose information has already been accounted for. The same situation can appear when there are two foreign keys, $A_{f_{1_k}}, A_{f_{2_k}}$, in relation R_j on the same potential key A_k of a relation R_i. We have chosen to terminate the acquisition of information when a self replicated loop appears. To do that we keep track of all the instances of the different relations that appear in a given path of the recursion, this is the role of the *instStack* variable in algorithm 1. The moment an instance appears a second time in the given recursion path the recursion terminates (algorithm 1, line 8). In the next section we will give an example of the construction of a relational instance. Within the example we will also provide a short discussion of the issues related to self replicating loops. For the moment let us simply note that while not allowing them seems as a reasonable choice, since these would result in the introduction of redundant information, taking a closer look on the problem makes the other alternative equally plausible also.

Finally to define a classification problem one of the relations in $\mathcal{R}$ should be defined as the *main* relation, M, i.e. the relation on which the classification problem will be defined. Then one of the attributes of this relation should be defined as the class attribute, M_c, i.e. the attribute that defines the classification problem. Each of the instances of the relation M will give rise to their corresponding relational instances which are the ones that will be used during learning.

The use of the links defined on foreign keys guides the acquisition of the information related with a given instance. At each moment we know on which relations we should be looking and which attribute we should be using to retrieve that information. Having an adequate way to handle set attributes is the heart of our relational representation. In the section 3.1 we will see how this can be done using some well known distance measures on sets. We should note that we are not necessarily limited to distance based approaches. If the appropriate operators are defined for set attributes one can imagine relational learners of different paradigms, e.g. decision trees, linear learners etc. In section 6 we will show how the representational paradigm that we have established here relates to some of the most common approaches used in Inductive Logic Programming.

2.3 An example

We will now illustrate the main ideas of our approach with a simple relational example from the proteomics field.

We would like to characterize proteins as belonging to a positive or negative class based on their individual description, the substances with which they interact and the family to which they belong. We have a relational schema containing three tables. One corresponding to descriptions of families, *Families*, with primary key **F.ID**; a second one, *Substances*, with primary key **S.ID**, corresponding to descriptions of substances; and a third for a description of proteins, *Proteins*, with primary key **P.ID**. To describe protein substance interactions we define the table *P-S-Interaction* with two foreign keys *P.ID, S.ID,* pointing respectively to the *Proteins* and *Substances* tables. To declare the family to which a protein belongs we add a foreign key, *F.ID*, to the table of *proteins* pointing to the table of *Families*. The complete description of the example is given in table 1.

Table 1. A simple example database from the proteomics domain. Referenced keys are marked in **bold**, foreign keys in *italics*.

Substances		*Families*		*Proteins*			*P-S-Interaction*		
S.ID	DESCRIPTION	**F.ID**	DESCRIPTION	**P.ID**	*F.ID*	CLASS	*P.ID*	*S.ID*	DESCRIPTION
D_a		F_a		P_A	F_a	+	P_A	D_a	
D_b		F_b		P_B	F_a	-	P_A	D_b	
D_z		F_c		P_C	F_c	+	P_B	D_z	

Lets suppose that we want to access the information related with protein P_A in order to construct the corresponding relational instance. To that end we should traverse the complete relational schema in a recursive manner using the defined links. Looking at the *Proteins* relations we see that there is a foreign key *F.ID* referencing the *Families* relation so we have to move there to recuperate the Description of the family to which the protein P_A belongs to. The description of the family F_A associated with our protein P_A will add *one* attribute of type set to the attributes of the *Proteins* relation. Examining now the *Families* relation we see that it does not have any foreign keys that we could follow to another relation. However its key **F.ID** is referenced by the *F.ID* of the *Proteins* relation; following that link back for the F_A brings us to a set of two instances from the proteins relation, this set of instances now adds one more attribute of type set to the description of F_A family. Nevertheless P_A has already been present in the recursion path so we exclude it from the set, according to the design choice on self replicating loops, and continue only with the P_B protein which via the *F.ID* foreign key will bring us back to the F_A instance of the *Families* relation. But F_A was also already present in the recursion path so we terminate here the recursion.

The relation between *Proteins* and *Families* clearly illustrates why links should be followed in both direction. A protein is described by the family to

which it belongs so in that case we should follow the link from proteins to families, and on the same time a family is described by the proteins that form that family so we should now follow the same link in the opposite direction. It is questionable whether ignoring protein P_A when we come from the family F_A is the perfect choice. Ignoring P_A leaves somehow incomplete the description of F_A, especially if P_A is a typical or important member of the family, on the other hand the information that P_A brings has already been accounted for, although in a shallower level of the recursion.

Back in the *Proteins* relation the key **P.ID** is referenced by a foreign key in table *P-S-Interaction* so now we have to move there and recuperated all the instances of the *P-S-Interaction* table associated with our protein P_A. We see that there are two instances like that. Again this set of instances will contribute one more attribute of type set to the attributes of the *Proteins* relation. While being at relation *P-S-Interaction* we see that there is one foreign key *S.ID* pointing to the *Substances* relation which we should follow in order to get a description of each of the substances interacting with our protein. Again this description will contribute an attribute of type set in table *P-S-Interaction*. When at the *Substances* relation following back the links to which it participates will brings us to instances that have already been accounted for, so the recursion ends here.

The result is that the final description of P_A will consist of two set attributes. The first one describes the family to which P_A belongs to, a family which is described in terms of its protein members except P_A. The second set attribute describes the set of substances with which P_A interacts, substances whose description was retrieved from the *Substances* table. The resulting relational instance is given in figure 1.

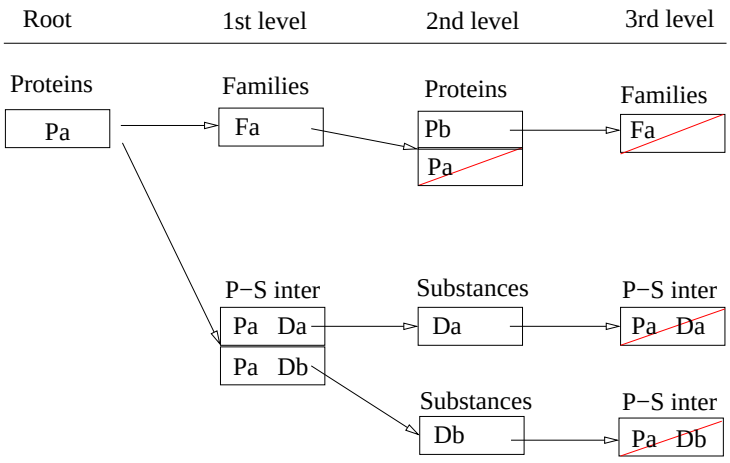

Fig. 1. The relational instance that corresponds to protein Pa. The oblique lines indicate where recursion ends due to the appearance of a self replicating loop.

3 Distance Measures

Before going to the description of distance measures between set of objects we will briefly review some of the terminology and the definitions used to characterize the different measures explored in this study.

A function $d : S \times S \rightarrow R_0^+$ is a *dissimilarity* function on a nonempty set S, if and only if it is *reflexive*, i.e.

$$\forall x \in S : d(x,x) = 0.$$

A function $d : S \times S \rightarrow R^+$ is a *distance* function if it is a dissimilarity function and it is *symmetric*, i.e.

$$\forall x, y \in S : d(x,y) = d(y,x)$$

A function $d : S \times S \rightarrow R^+$ is a *metric* if it is a distance function and

- it is *strict* i.e.
$$\forall x, y \in S : d(x,y) = 0 \Rightarrow x = y$$
- and satisfies the *triangle inequality* i.e.
$$\forall x, y, z \in S : d(x,z) \leq d(x,y) + d(y,z)$$

If d satisfies all but the strictness property it is called a *pseudo-metric*, and if it satisfies all but the triangle inequality it is called a *semi-metric*. The strict and reflexive properties are some times identified collectively as reflexivity:

$$\forall x, y \in S : d(x,y) = 0 \; if \; and \; only \; if \; x = y.$$

For reasons of readability we will call all of the above *distance measures*; when it is necessary we will make clear whether a distance measure is a dissimilarity, distance, metric etc.

3.1 Distance Measures On Sets

The issue that arises when working with sets of objects is how one can use the distance measures defined on S in order to define distance measures on the power set 2^S of S and under what conditions the set distance measure is a distance or a metric function. A number of different measures have been proposed in the literature for defining distances between sets of objects. We will briefly present some of them. For the moment we will not consider distances between relational instances, the elements of S are attribute-value vectors. In section 4 we will see how we can exploit the set distances in order to define distances for relational instances.

Consider two sets $A = \{a_i | a_i \in A, i = [1..|A|]\} \subseteq S$ and $B = \{b_j | b_j \in B, j = [1..|B|]\} \subseteq S$. Let $d(.,.)$ be a distance measure defined on S. The set distance measure D defined on 2^S as:

$$D : 2^S \times 2^S \rightarrow R^+, \; D(A,B) = f(\{d(a_i,b_j) | (a_i,b_j) \in A \times B\})$$

is some function of the pairwise distances, $d(a_i,b_j)$, of the set of all pairs $(a_i,b_j) \in A \times B$. A and B should be non-empty and finite sets.

– *Single Linkage*: The set distance is the minimum distance of all pairwise distances:
$$D_{SL}(A,B) = \min_{ij}\{d(a_i,b_j)\}.$$
It is not a metric or a pseudo-metric even if the underlying distance measure $d(.,.)$ is a metric because it does not satisfy the strict and triangle inequality properties. However if the underlying distance measure is at least a distance function then D_{SL} is also a distance function.
– *Complete Linkage*: The set distance is the maximum of all pairwise distances:
$$D_{CL}(A,B) = \max_{ij}\{d(a_i,b_j)\}.$$
It is not even a dissimilarity function since it does not satisfy the reflexive property.
– *Average Linkage*: The set distance is the average of all pairwise distances:
$$D_{AL}(A,B) = \frac{\sum_{ij} d(a_i,b_j)}{|A||B|}.$$
It is not even a dissimilarity function since it does not satisfy the reflexive property.
– *Sum of Minimum Distances*: discussed in (Eiter & Mannila, 1997) it is the sum of the minimum distances of the elements of the first set to the elements of the second set and vice versa.
$$D_{SMD}(A,B) = \frac{1}{2}(\sum_{a_i}\min_{b_j}\{d(a_i,b_j)\} + \sum_{b_i}\min_{a_j}\{d(b_i,a_j)\})$$
It is not a metric even if the underlying distance measure $d(.,.)$ is a metric because it fails to satisfy the strict and triangle inequality properties. However if $d(.,.)$ is at least a distance function then D_{SMD} is also a distance function since it is reflexive and symmetric.
– *RIBL*: the sum of the minimum distances of the elements of the smaller set to the elements of the larger, (Horvath et al., 2001).
$$D_{RIBL}(A,B) = \begin{cases} \frac{\sum_{a_i}\min_{b_j}\{d(a_i,b_j)\}}{|B|}, & |A| < |B| \\ \frac{\sum_{b_j}\min_{a_i}\{d(a_i,b_j)\}}{|A|}, & |A| \geq |B| \end{cases}$$
A dissimilarity measure since it only satisfies the reflexive property. A simple counter example for the triangle inequality is to consider $A = \{a\}, B = \{b\}, C = \{a,b\}$ the $D_{RIBL}(A,B) > D_{RIBL}(A,C) + D_{RIBL}(C,B) = 1 > 0+0$.
– *Hausdorff:* Discussed in (Eiter & Mannila, 1997) is one of the best known distances measures between sets defined as:
$$D_H(A,B) = \max(\max_{a_i}\{\min_{b_j}\{d(a_i,b_j)\}\}, \max_{b_i}\{\min_{a_j}\{d(b_i,a_j)\}\})$$
If $d(.,.)$ is a metric then D_H is also a metric.

– *Tanimoto*: Discussed in Duda et al. (2001a) it is used when we do not have a notion of graded distance or similarity between the elements of two sets, i.e. two elements are simply the same or different, it is defined as:

$$D_T(A,B) = \frac{|A| + |B| - 2|A \cap B|}{|A| + |B| - |A \cap B|}$$

However we define an extension of it that is able to deal with graded similarities. Two elements a_i, b_j, will be considered identical if $d(a_i, b_j) \leq \theta$ where θ is a user specified threshold. Under this loose definition of identity it is now possible to compute the cardinality of $A \cap B$ with the constraint that each element of a set can be only matched once. This loose definition of identity has as a result that the strictness property and the triangle inequality are no longer satisfied making D_T a distance.

So far the distance measures over sets that we have presented are relative simple ones whose computation is straightforward if one has computed all the pairwise distances among all the pairs of elements defined from the two sets. Another family of more elaborate distance measures is based on the definition of a set of relations $R = \{R_i | R_i \subseteq A \times B\}$ between the two sets. The computation of the distance measure will be based on that $R_i \in R$ that minimizes a distance measure computed on the elements that are part of the relation R_i.

– *Surjections, D_S*: Here the set of relations, R, consists of all the possible surjections of the larger to the smaller set. The distance of the two sets is defined as the minimum sum, overall R_i, of the distances of the pairs of elements that participate in the surjection:

$$D_S(A,B) = \min_{R_i \in R} \sum_{(a_i,b_j) \in R_i} d(a_i, b_j)$$

Eiter and Mannila (1997) give an algorithm for computing it based on graph theory and more precisely on minimum weight perfect matching in bipartite graphs. It does not satisfy the triangle inequality so it is a semi-metric.

– *Linkings, D_L*: The set of relations R is the set of all possible linkings. A linking is a mapping of one set to the other where all elements of each set participate in at least one pair of the mapping. The distance between two sets is given by:

$$D_L(A,B) = \min_{R_i \in R} \sum_{(a_i,b_j) \in R_i} d(a_i, b_j)$$

It was introduced by Eiter and Mannila (1997) and its computation is also based on minimum weight perfect matching. It does not satisfy the triangle inequality so it is a semi-metric.

– *Fair Surjections, D_{FS}*: The set of relations is the set of all fair surjections. A surjection is fair if it maps as evenly as possible the elements of the larger set to the elements of the smaller set. Again the distance of the two sets will

be the minimum sum, overall R_i, of the distances of the pairs of elements that participate in the fair surjection:

$$D_{FS}(A,B) = \min_{R_i \in R} \sum_{(a_i,b_j)\in R_i} d(a_i,b_j)$$

(Eiter & Mannila, 1997) give an algorithm for the computation of the fair surjection distance based on flow networks and the minimum weight maximum flow. It does not satisfy the triangle inequality so it is a semi-metric.

– *Matchings,* D_M: The set of all possible matchings is considered within which the minimum distance is computed. In a matching each element of the two sets is associated with at *most* one element of the other set. It was introduced by Ramon and Bruynooghe (2001) who provided an algorithm for its computation based also on minimum weight maximum flow. They prove that their distance function is a metric when $d(.,.)$ is a metric. For the computation of the D_M they give a general schema that incorporates as special cases the D_S, D_L, D_{FS} distance measures, provided that R is adjusted accordingly to the set of surjections, linkings or fair surjections, and is given by:

$$D_M(A,B) = \min_{R_i \in R}\Big(\sum_{(a_i,b_j)\in R_i} d(a_i,b_j) + (|B - R_i(A)| + |A - R_i^{-1}(B)|) \times \frac{M}{2}\Big)$$

where M is the maximum possible distance between two elements. What actually the second term of the sum does is to add an M/2 penalty for these elements of the A and B that do not participate in the relation R_i. In D_S, D_L, D_{FS} the second term of the sum vanishes because all elements of the two sets participate in the relation R_i.

All of the above set distance measures, with the exception of the Tanimoto measure, reduce to $d(a,b)$ in the trivial case that $A = \{a\}$ and $B = \{b\}$. This is a necessary property if we want the set representation to have as a special case the typical attribute-single-value representation.

Each of the presented distance measures imposes different semantics on what is important in determining the distance between two sets. For example in D_{SL} it is only the two most similar elements that determine the distance between the two sets, while in D_{CL} it is exactly the opposite, i.e. the two most dissimilar objects determine the set distance. The main limitation that one could see in D_{SL}, D_{CL} and D_H is that they do not take into account the complete information provided by the whole sets but rather focus on a specific pair of elements. A fact that can be quite problematic in the presence of noise or outliers within the sets. D_{RIBL} and D_{SMD} do not focus on the distance of a single pair of elements but on the set of minimum distances of the elements of one set to the elements of the other set providing a more global measure of how similar are the two sets with respect to their most similar elements. This approach though more global could still be problematic if there is an outlier in one of the sets, lets say in set A, that is much closer than all other elements of set A to the elements of the B

Table 2. Characterization of the different distance measures according to the properties they satisfy.

Measure	Reflexive	Symmetric	Strict	Triangle inequality	type
D_{SL}	+	+	-	-	distance
D_{CL}	-	+	-	-	-
D_{AL}	-	+	-	-	-
D_{SMD}	+	+	-	-	distance
D_{RIBL}	+	-	-	-	dissimilarity
D_H	+	+	+	+	metric
D_T	+	+	-	-	distance
D_S	+	+	+	-	semi-metric
D_L	+	+	+	-	semi-metric
D_{FS}	+	+	+	-	semi-metric
D_M	+	+	+	+	metric

set. In that case it will be the minimum distances from that element that will mainly determine the final distance. The problem is more acute for D_{RIBL} since it is not symmetric in its use of minimal distances with respect to the two sets. For D_{SMD} it is less acute since it will only dominate one of the two sum terms. D_{AL} tries to take into account the complete available information by averaging overall the pairwise distances. However the main problem of this measure is that it does not satisfy the reflexive property which in fact means that the distance of a set from itself can be, and most often will be, bigger than zero. This can lead to awkward situations where a set is more similar to another set than it is with itself. D_{CL} also exhibits the same type of pathology. The reason for which we included these two measures is that they are extensively used in computing set distances in clustering. The Tanimoto based distance measure is a measure of the degree of overlap between the two sets under the loose definition of identity that we have introduced. It is less sensitive to the problem of outliers since each element of a set can be matched at most once. The relation based measures are also less sensitive to the existence of outliers since they take a more global view by seeking a mapping between the elements of the two sets which uses all the available information given by the sets. In the case of D_L, D_S and D_{FS} each element of the two sets will participate at least once in the relation and the resulting distance will be the minimum computed overall permissible relations. Like that the distance computation is spread across all the elements of the two sets making it less sensitive to the presence of noise or outliers. In the case of D_M is not required that every element participates in the relation, however even the elements that do not participate are accounted in the distance computation by the penalty term.

Nothing can be said about the general superiority of one distance measure over another. It all depends on the specific problem application and its semantics which should mainly drive the selection of the appropriate measure. For example in some applications it might be that what is most important is the distance

between the two most similar elements of the sets, while for others a more global approach that takes into account all the elements might be required.

3.2 Normalization

If the distance measures are to be used in order to compute distances between sets, i.e., each learning instance corresponds to a single set, there is no need for normalization. However if they are to be incorporated in a relational learning schema as the one described then they should be normalized. Each attribute of a relation, whether standard attribute or set attribute should equally contribute to the distance between two tuples of that relation, if set distances are not normalized in the same interval as the distances over standard attributes then they will dominate the final distance thus distorting its real value and altering the order of similarity of the relational instances. This for example will happen with all set distances which are based on the definition of specific types of mappings where the final distance is a sum of distances, thus its value will depend on the cardinality of the mapping. Moreover the influence of set attributes will increase with the level of recursion again dominating over standard attributes and distorting the real distance.

Of the presented distance measures the ones that are not normalized are D_{SMD}, D_S, D_L, D_{FS} and D_M. To normalize D_{SMD} instead of dividing by two we divide with the sum of elements of the two sets; for D_S, D_L and D_{FS} we normalize by the cardinality of the surjection, linking or fair surjection on which the minimum distance was computed. For all of the above the normalization procedure does not alter their formal properties. For D_M we used its normalized version given by Ramon and Bruynooghe (2001):

$$D_M := \frac{2D_M(A,B)}{D_M(A,B) + (|A| + |B|)/2}.$$

It is these normalized distance measures that will be incorporated in our relational distance based learner.

4 A Relational Distance

Having defined the appropriate operators for distance computation that can handle the attributes of type set we can proceed to the complete description of the relational distance.

The computation of the distance between two relational instances is done in a recursive manner traversing the full tree structures of the relational instances. During the recursion the components of the relational instances are visited in exactly the same order as when these were constructed. The full procedure is given by algorithms 2 and 3. For computational reasons the depth of recursion is controlled by a depth parameter, d. In subsection 4.1 we will see how this parameter affects the estimation of the final relational distance.

The *distance* function, given by algorithm 2, shows how the computation of the distance between any two relational instances, R_{i_a}, R_{i_b}, of a given relation, R_i, is done. In order to compute that distance we use the following formula:

$$d(R_{i_a}, R_{i_b}) = \sqrt{\frac{\sum_{k=1}^{N} diff^2(v_{ak}, v_{bk})}{N}} \tag{1}$$

with v_{ak}, v_{bk}, the values of the R_{i_a}, R_{i_b}, for attribute A_k The sum runs over all standard attributes and all set attributes of the R_i relation, thus $N = |\mathcal{I}_{A,R_i}| + |\mathcal{I}_{L,R_i}|$

What the function $diff$ does depends on the type of the attribute A_k on which it is applied. If A_k is a standard continuous attribute then $diff$ is simply given as:

$$diff(v_{ak}, v_{bk}) = |\frac{v_{ak} - \min(A_k)}{\max(A_k) - \min(A_k)} - \frac{v_{bk} - \min(A_k)}{\max(A_k) - \min(A_k)}| \tag{2}$$

i.e. the distance of the normalized values of the two instances for the given attribute. If it is a standard discrete attribute then:

$$diff(v_{ak}, v_{bk}) = 0 \; if \; v_{ak} == v_{bk}, \; 1 \; otherwise$$

In both cases the values of $diff$ will always be between zero and one.

If A_k is of type set (algorithm 2, line 26) then the values v_{ak}, v_{bk}, are actually the two sets of relational instances that are associated with R_{i_a} and R_{i_b} in relation R_j when we follow the link $l(R_i, A_k, R_j, A_l)$. In order to compute the distance between these two sets we use the function $diff$ given in algorithm 3. If both sets are empty then $diff$ will return a distance of zero, if only one of them is empty then $diff$ will return the maximum possible distance (since we are working with normalized distances this will be one). In the case that none of the sets is empty $diff$ will first compute all the pairwise distances between the elements of the two sets. Remember that these sets are sets of relational instances whose distances have to be computed by using again the *distance* function given by algorithm 2. After all pairwise distances between the relational instances have been computed the $SetDistance$ function is applied on that collection and computes one of the set distance measures, described in section 3.1, according to the user's choice. Since we use normalized set distance measures $diff$ will always return a value between zero and one. The relational distance measure reduces to a normalized version of the euclidean distance when a relation does not contain attributes of type set.

4.1 Properties of the relational distance

The formal properties of the relational distance function, i.e., whether it is a metric, distance etc, depend on the formal properties of the set distance measure that is used. $d(R_{i_a}, R_{i_b})$ can be seen as a function defined on the cartesian product, $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_N$, of N spaces, and it will be a metric if and

Algorithm 2 Distance between two relational instances

distance($R_i, R_{i_a}, R_{i_b}, aStack, bStack, d$)
{R_i : relation in which we find the instances}
{R_{i_a}, R_{i_b}, : the relational instances whose distance we want to calculate}
{aStack, bStack : instances visited so far during the recursion}
{d : current depth of recursion}
if $d > MAX - DEPTH$ **then**
  return NULL
end if
if $R_{i_a} \in aStack$ **then**
  return NULL {A loop has occurred, R_{i_a} could be used to describe R_{i_a}, STOP}
else
  aStack.push(R_{i_a})
end if
if $R_{i_b} \in bStack$ **then**
  return NULL {A loop has occurred, R_{i_b} could be used to describe R_{i_b}, STOP}
else
  bStack.push(R_{i_b})
end if
Distance $\leftarrow 0$
{Standard attributes}
for $A_k \in \mathcal{I}_{A,R_i}$ **do**
  Distance $\leftarrow$ Distance + $\text{diff}^2(v_{ak}, v_{bk})$
end for
{Set attributes}
for $l(R_i, A_k, R_j, A_l) \in L(R_i, \_) \cup L^{-1}(R_i, \_)$ **do**
  Distance $\leftarrow$ Distance + $\text{diff}^2(R_j, v_{ak}, v_{bk}, aStack, bStack, d+1)$
end for
aStack.pop(); bStack.pop();
$N = |\mathcal{I}_{A,R_i}| + |\mathcal{I}_{L,R_i}|$
return $\sqrt{Distance/N}$

Algorithm 3 Distance between sets

diff($R_j, v_a, v_b, aStack, bStack, d$)
{v_a, v_b: sets of relational instances}
if $v_a = \emptyset$ XOR $v_b = \emptyset$ **then**
  return $maxDistance$
else if $v_a = \emptyset$ AND $v_b = \emptyset$ **then**
  return 0
end if
DistanceBag $\leftarrow \emptyset$
for all $(R_{j_k}, R_{j_l}) \in v_a \times v_b$ **do**
  dist = distance($R_j, R_{j_k}, R_{j_l}, aStack, bStack, d$)
  if dist!= NULL **then**
    DistancesBag.add($R_j, R_{j_k}, R_{j_l}, dist$)
  end if
end for
return SetDistance(DistancesBag)

only if each one of the $diff(.,.)$ functions employed in equation 1 is a metric. It is obvious that for a problem involving only two levels i.e., a main relation, M, and a number of relations directly associated with the main relation, the distance $d(M_a, M_b)$ between two instances, M_a, M_b, is a metric if and only if the set distance measure employed in the $diff(.,.)$ function is a metric. For problems involving k levels of recursion we can show that each of the distances defined on the relations found at the $k-1$ level is a metric according to the reasoning given before for the two levels. Applying recursively the same reasoning we can easily see that at the top level, i.e. the main relation, the distance function that we get is also a metric. In general $d(M_a, M_b)$ inherits exactly the properties of the set distance function that it uses, thus the table 2 giving the properties of the set distance functions is the same for the corresponding relational distances that derive from using these set distance functions.

The distance computation between two relational instances, R_{i_a}, R_{i_b}, of a given relation R_i is done in a recursive manner by computing the distances of the relational instances that are associated with R_{i_a}, R_{i_b}, which were determined based on the links in which the relation R_i participates. Each one of the links of a relation adds one more dimension to the distance between two relational instances. With an increasing depth the importance of a link and its corresponding relational instances become smaller. Relations that are closer to the main relation have a higher contribution on the distances of the instances of the main relation.

It is relatively easy to estimate the contribution of a given relation to the final relational distance in association with the recursion depth at which the former is encountered. Consider a simple scenario with n relations $\{R_i | i := 1..n\}$, where each relation R_i is linked to a single relation R_{i+1}, and R_1 is the main relation. Let d be the relational distance between two relational instances of R_1, computed with a recursion depth of n, and $\hat{d}$ its approximation, computed with a recursion depth of $n-1$ (i.e. relation R_n is not included in the distance computation). The approximation error and thus the influence of the R_n relation, in the case of average linkage, can be shown to be bounded as follows:

$$|d_1^2 - \hat{d}_1^2| \leq 1/\prod_{i=1}^{n-1} N_i$$

where N_i is the number of attributes (standard plus set type) of relation R_i. For other types of set distances the bound is not straightforward to compute. Excluding relation R_n might alter the mappings between elements of sets of R_{n-1}, when we compute set distances on R_{n-1}, nevertheless these distances too follow roughly the same law. A similar result was also given by Bisson (1992) where a closely related iterative approach was used for distance computation between complex objects.

A direct consequence of that result is that we can reduce the depth of recursion, thus speeding up classification time, without a significant loss in the accuracy of the distance computation, also the problem of self replicated loops is alleviated since their contribution reduces with the depth of the recursion.

5 Learning Experiments

We will compare the 11 different instantiations of the relational distance measure, given in table 2, on a number of relational problems. The learning task will be always classification. We will use the k-nearest neighbor classification rule to decide the classification of a given instance (Duda et al., 2001a), we have experimented with different values of k (k=1,3,5,10). The relative performance of the different relational distance measures did not vary with k, we will thus report results only for k=1. Some of the learning problems can be simply reduced to direct comparisons of sets of points having thus no recursion (duke, diterpenes, musk). In others (the two different representations of the mutagenicity problem), it will be possible to move beyond a single level comparison of the instances and have many levels of recursion. In the following section we will briefly present each of the datasets used in the study.

5.1 Datasets

Mutagenicity. It was introduced in (Srinivasan et al., 1994). The application task is the prediction of mutagenicity of a set of 188 aromatic and heteroaromatic nitro-compounds (regression friendly version). We defined the learning problem in two different ways. In the first the examined compounds (in the *main* table) consist of atoms (in the *atom* table) which constitute bonds (in the *bound* table) (*version 1*); the median of the number of atoms for each compound is 26. The recursion depth was limited to four. In the second the compounds consist of bonds while bonds consist of atoms (*version 2*); the median of the number of bonds per compound is 28. Here the recursion level was limited to three. Bonds are described by two links to specific entries in the *atom* table and by the type of the bond while atoms are described by their charge (numeric values), type and name.

Diterpene. In the diterpene dataset (Dzeroski et al., 1996) the goal is to identify the type of diterpenoid compound skeletons given their ^{13}C-*NMR*-Spectrum. Each spectrum consists of a set of points, each of the form *(multiplicity, frequency)*. It is a problem of direct comparison of sets. There are 1503 instances, i.e. spectra, and 23 classes. Each instance is described by 20 tuples.

Musk. It was described in (Dietterich et al., 1997); here the goal is to predict the strength of synthetic musk molecules. Version 1 of the dataset contains 47 musk molecules and 45 non-musk molecules. Each example is described by a set of tuples. The cardinality of these sets ranges mostly between two and ten with a maximum value of 40 and a median of four. Each tuple has 166 features. Version 2 contains 63 non-musk and 39 musk molecules. The cardinality of these sets ranges mostly between one and 60 with a maximum value of 1044 and a median of 12. The second version of the problem has sets of much more varied and greater cardinality than version 1. The problem is limited to a direct comparison of sets.

Table 3. Datasets sets used in the comparative study

Dataset	# Instances	# classes	Median of First Level Set Cardinality	# Levels
Diterpene	1503	23	20	One
Duke	41	2	522	One
Musk (ver. 1)	92	2	4	One
Musk (ver. 2)	102	2	12	One
Mutagenicity (ver. 1)	188	2	26	Four
Mutagenicity (ver. 2)	188	2	28	Three

Duke. This is a dataset from the domain of proteomics (Campa et al., 2003). The goal is to classify mass-spectra acquired from blood serum of individuals that have developed lung cancer or healthy individuals. The dataset consists of mass-spectra of 24 diseased and 17 healthy persons. Each spectrum is described by a set of peaks of the form *(mass,intensity)*. The median of the cardinality of the sets is 522. We will consider two versions of this dataset. In the first, *duke-M*, a peak will consist only of its mass, in the second, *duke-MI*, both mass and intensity will be used. The *duke-M* representation corresponds roughly to a binary representation of a spectrum where presence of a given mass indicates presence of the corresponding peak. The most similar spectra will be the ones that have more similar peaks (determined by the mass dimension). In the context of set distances this is a more appropriate representation for a mass spectrometry problem since what is more important is to find an appropriate matching of masses and only then take into account intensity differences. The *duke-MI* representation places on an equal footing both dimensions, it thus violates the semantics of the problem and does not naturally fit within the context of relational distances presented so we will not consider here. In section 5.8 we will describe a distance measure specifically tailored to the mass spectrometry problem with which we can tackle both representations.

5.2 Experimental Setup

We estimate accuracy using ten-fold stratified cross-validation and control for the statistical significance of observed differences for all pairs of distance measures using McNemar's test (sig. level=0.05). In order to have a more global picture, than the one provided by the basic pairwise comparisons, of the relative merits of the different distance measures, we establish a ranking schema based on the results of the pairwise comparisons. More precisely for a given dataset if distance measure a is significantly better than b then a is credited with one point and b with zero points; if there is no significant difference then both are credited with half point. Since we are comparing 11 different relational distance measures it is obvious that if there is a distance measure that is better than all the others it will be credited with ten points, if it is worse than all the others it will be credited with zero points while if all distances are equivalent they will be all

credited with five points. The results are presented collectively in table 4 (top ranked distance measures will be emphasized), the detailed significance results are given in appendix A.

5.3 Classification performances of the relational distances

From the overall results, table 4, it is obvious that there is no set distance measure which is the overall winner, something that was to be expected. To give an overall picture of the performance of the distance measures we averaged their rankings over the different datasets and report their average ranks in table 5. There are however some distance measures that perform consistently well over a series of problems. For example D_{SMD} is ranked on the top in four (duke, musk ver. 1,2, mutagenesis ver. 1) out of the six examined problems and in one (mutagenesis ver. 2) it is very close to the top performing; in terms of its average rank it takes the top position. What makes this good performance even more interesting is that it is one of the simplest distance measures with only quadratic complexity.

What came as a surprise was the low performance of the matching based distance measure, D_M, in terms of its average rank it was placed only on the eighth position among the 11 different measures. Quite astonishing was its bad performance on the musk dataset for which previously it was reported to have a very good performance (Ramon & Bruynooghe, 2001). We experimented also with its unnormalized version for which indeed it performed well on the musk problem; for the other datasets there were no differences between the normalized and unnormalized versions (in the above paper it is not clear which version the authors used, i.e., normalized or unnormalized). Another important difference from the results reported in (Ramon & Bruynooghe, 2001) is the performance of the D_{FS} set distance measure on the musk (version-1) and diterpenes datasets; more precisely Ramon and Bruynooghe (2001) report errors of 51% and 13% for the two datasets while our error estimation gives much lower errors, 16.30% and 4.44% respectively. These big differences cannot be explained by a different experimental setup since they used ten-fold cross validation. In an initial phase we had similar error estimates only to discover later that this was due to a limitation of the library used to solve the min weight maximum flow problem.

In contrast to D_M the three other relation based distance set measures, D_L, D_S, and D_{FS}, exhibit a rather good predictive performance. In terms of their average ranking they take the second, fourth and fifth position respectively. Their main difference from D_M is the fact that each element of a set must be a part of the mapping between the two sets. No element is left outside as it is the case with D_M. Another difference is the way that D_{FS}, D_L, and D_S are normalized, i.e. by the cardinality of the relation over which they were computed. D_{FS}, D_L, and D_S are rather similar to D_{SMD} to these respects, i.e. the normalization[2] and the fact that all the elements are accounted for in the distance computation. Note

[2] In D_{SMD} normalizing by the sum of the cardinalities of the two sets is equivalent to normalizing by the cardinality of the relation defined by D_{SMD}.

that these four distance measures were ranked to the top five positions when we average their rankings over all the datasets, table 5.

D_{RIBL} differs from D_{FS}, D_L, and D_S, in the same way that D_M does. D_{RIBL} does not use all the elements of the two sets in the distance computation, many of the elements of the larger set can be left aside. The normalization of the distance is not done by the cardinality of the relation that D_{RIBL} imposes between the two sets but by the cardinality of the larger set. If we take a look on the ranking of D_{RIBL} and D_M averaged over all the datasets we see that they are ranked next to each other, with a small difference in their average ranks, with D_{RIBL} being ranked on the ninth position and D_M on the eighth as already mentioned.

Another distance measure that is similar to D_{FS}, D_S, D_L, and D_{SMD}, with respect to normalization and accounting for all the elements, is D_{AL} which nevertheless was ranked usually on the last positions. However D_{AL} exhibits a fundamental flaw, it does not satisfy the reflexivity property, which can explain its bad behavior.

A rather separate family of distance measures are the ones that base their distance only on a single pair of elements, namely D_H, D_{SL} and D_{CL}. The performance of the former two with respect to their rankings averaged over all the datasets is very similar, in fact they are ranked next to each other taking the sixth and seventh position. D_{CL} though has a very bad performance which can be explained by the fact that it does not satisfy the reflexivity property.

The Tanimoto based distance is quite different from all the above. Although it constructs a mapping between the elements of the two sets based on the threshold parameter[3] it does not compute the distance between the two sets based on the distances included in the mapping. It uses the mapping to compute a degree of fuzzy overlap between the two sets. Some clarifications should be given here on its parameter setting. When we performed the experiments we have chosen the value of the threshold to be equal to 0.01 since we considered that to be a sensible choice. We did that apriori and without any tests. However experimenting afterward with the three datasets (duke, musk ver. 1, 2) in which it did not performed well we found parameter settings for which it performed much better, a more informative way of selecting the value of the threshold would result in better performance. Nevertheless we have chosen to report results only on this initial setting so that the results are not biased. In terms of its average rank it performs quite well taking the third position over all distance measures.

No general statement can be done about the superiority of one distance measure over another. There are though some distance measures that exhibited a good and stable performance over the small number of datasets that we examined here. For example D_{SMD}, the good performance of which coupled with its quadratic computational complexity make it a good first choice. However in general everything depends on the type of application and its underlying assumptions which should mainly guide the selection of the distance measure. In

[3] It does not account for all the elements of both sets. To that aspect is more similar to D_{RIBL} and D_M.

the absence of such assumptions the obvious question is how distance selection could be done. In section 5.6 we will see how we can achieve that.

Table 4. Error and rank results on the datasets

Set Distance	*duke*	*musk, (version 1)*	*musk, (version 2)*
D_{SL}	41.46 (4.5)	*15.21 (7.0)*	28.43 (5.0)
D_{AL}	41.46 (4.5)	*17.39 (7.0)*	31.37 (5.0)
D_{CL}	56.09 (3.0)	32.60 (2.5)	31.37 (5.0)
D_{SMD}	*19.51 (7.0)*	*17.39 (7.0)*	*24.50 (6.0)*
D_{H}	39.02 (4.5)	*18.47 (7.0)*	*23.52 (6.0)*
D_{RIBL}	29.26 (5.5)	50.00 (1.0)	40.19 (3.5)
$D_{T} - \theta$ 0.01	39.02 (5.0)	48.91 (1.5)	38.23 (5.0)
D_{L}	24.39 (5.5)	*15.21 (7.0)*	26.47 (5.0)
D_{S}	31.70 (5.5)	*16.30 (7.0)*	26.47 (5.0)
D_{M}	36.58 (5.0)	50.00 (1.0)	40.19 (3.5)
D_{FS}	39.02 (5.0)	*16.30 (7.0)*	*24.50 (6.0)*
Def. Error	41.46	48.91	38.23

	mutagenesis version 1	*mutagenesis version 2*	*diterpene*
D_{SL}	25.53 (4.5)	25.00 (28.19 2.5) (4.0)	48.23 (2.0)
D_{AL}	44.68 (0.5)	44.68 (35.63 0.5) (1.0)	60.61 (1.0)
D_{CL}	50.53 (0.5)	59.57 (23.40 5.0) (0.0)	80.10 (0.0)
D_{SMD}	*16.48 (8.0)*	15.95 (18.08 5.5) (8.0)	04.85 (6.0)
D_{H}	23.40 (4.5)	23.40 (22.34 4.0) (4.0)	15.10 (3.0)
D_{RIBL}	27.65 (4.0)	24.46 (21.80 5.0) (4.0)	04.85 (6.0)
$D_{T} - \theta$ 0.01	*14.89 (8.0)*	*12.76 (12.23 9.5) (8.5)*	03.19 (9.0)
D_{L}	*17.55 (8.0)*	16.48 (15.95 7.0) (8.0)	05.45 (6.0)
D_{S}	18.08 (7.0)	19.14 (20.21 5.5) (6.0)	04.32 (6.5)
D_{M}	25.53 (4.0)	27.12 (23.93 4.0) (3.5)	*03.26 (9.5)*
D_{FS}	17.02 (6.0)	15.95 (17.02 6.5) (8.0)	04.44 (6.0)
Def. Error	33.51	33.51	70.19

5.4 Families of distance measures

Overall it seems that a strong qualitative division of the distance measures arises along the following dimensions:

1. how they account for the elements of the two sets, possible options are:
   - account for all the elements of the two sets
   - account for subsets of the two sets
   - base distance only on a single pair of elements
2. whether they normalize by the cardinality of the mapping relation that they establish,
3. and finally whether they respect the reflexivity property.

Table 5. Average ranks of set distances over all the datasets

D_{CL}	D_{AL}	D_{RIBL}	D_M	D_{SL}	D_H	D_S	D_T	D_{FS}	D_L	D_{SMD}
1.83	3.16	4.00	4.41	4.50	4.83	6.16	6.16	6.33	6.58	7.00

This division is very closely reflected on the performances of the measures as these are given by their average rankings. Measures that share common features have a very similar average ranking among the different problems.

Going one step further we have tried to cluster the different distance measures based on their ranking on each dataset. Each measure was described by a six element vector where each feature of the vector corresponds to the ranking of the given measure in one of the datasets. We used an agglomerative hierarchical clustering algorithm (Duda et al., 2001a), together with *Ward*'s minimum-variance method to determine which clusters should be merged at each agglomeration step. Clusters represent groups of distance measures that have similar relative performance. The resulting dendrogram is given in figure 2.

We can see that this performance based clustering reflects completely the conceptual differences among the distance measures. The two measures that are based on a single pair of instances and respect the reflexivity property, D_H, D_{SL}, are the first to be merged. The next two mergings take place between the measures that exploit all the instances and normalize by the cardinality of the relation that they induce, D_{FS}, D_S, D_{SMD}, and D_L, creating two subclusters, $\{D_{FS}, D_S\}$ and $\{D_{SMD}, D_L\}$ which are merged later. The first subcluster contains the surjection based set distances which intuitively makes sense since they both use surjection in order to match elements of the two sets, with D_{FS} imposing the extra constraint that surjections should be fair. The next merge is that of D_M and D_{RIBL} later merged with D_T, creating a cluster of distance measures that do not account for all the elements of the two sets. In fact if we ask for four clusters the division we get is: $\{D_{FS}, D_S, D_{SMD}, D_L\}$, $\{D_{SL}, D_H\}$, $\{D_T, D_{RIBL}, D_M\}$, $\{D_{AL}, D_{CL}\}$, which are very naturally described by the three dimensions given above (the last cluster contains the measures that violate the reflexivity property). Given though the small number of datasets examined it is an open question whether this performance based clustering would persist and reflect in the same manner the qualitative differences of the measures if more datasets are considered.

Loosely speaking a set distance measure computes a distance in a high dimensional space where the number of dimensions equals the product of cardinalities of the two sets whose distance is computed. Each dimension is defined by a specific pair of elements of the two sets. Depending on the set distance measure used all, some, or a single dimension, are considered in order to compute the final distance. One can view that as a weighted distance computation where the dimensions that are not used have a weight of zero. This view is directly reflected on the first dimension of the qualitative characterization of the set dis-

tance measures. Normalizing then by the cardinality of the relation that is used to compute the set distance is simply averaging over the dimensions that are accounted in the distance computation.

The clustering of the distance measures according to their relative performances and the characterization of the clusters according to the dimensions given in the beginning of this section is strongly connected to the different types of problems that we can face. Relational classification problems can be characterized along the first two qualitative dimensions, e.g. whether all, subsets or a specific pair should determine classification, for example multi-instance classification problems fall in the last category. If we have a way to characterize a classification problem along these dimensions then we would expect the associated cluster of distance measures to have the best performance for that problem.

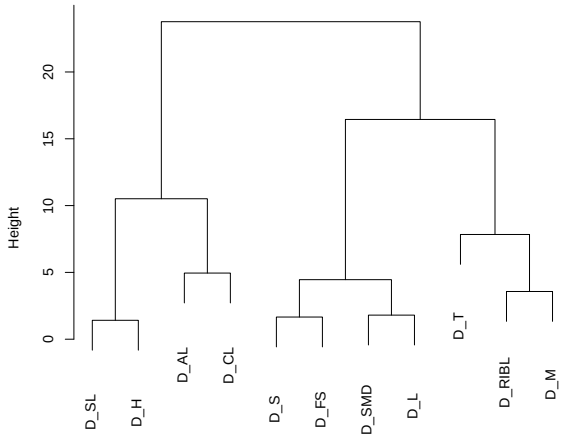

Fig. 2. Clustering the relational distances with respect to their rank performance among the datasets examined.

5.5 Comparison with other systems

The right choice of distance measure results to a classification performance that compares favorably with the performances of state of the art relational learning systems. In table 6 we give the performance of some well known relational systems along with the performance of the best distance measure (under the entry *Best-distance*). In the table we only report results on the same problem

formulation, with the same error estimation procedure (always ten-fold cross-validation). In case that more than one results were available we always took the best.

The results on diterpenes are taken from (Ramon & Bruynooghe, 2001). For musk 1 the TILDE result was the best result reported in (Blockeel & De Raedt, 1997), while for matchings the result is from (Ramon & Bruynooghe, 2001). For the mutagenesis the results are taken from (Blockeel & De Raedt, 1997) on the B_2 formulation of the problem that corresponds to our version 2 of mutagenesis and from (Ramon, 2002) for the matchings. For the KES and DES systems the results are from (Gartner et al., 2004).

Especially for mutagenesis there is a variety of other results reported in the literature showing even better performance. Nevertheless these have either been achieved on a different problem formulation that contained more information, or used a different evaluation strategy like leave-one-out cross validation. For these reasons they are not directly comparable and we thus do not include them in the comparison. However for completeness we will mention few of them. G-NET (Anglano et al., 1998) achieved an error of 8.8% on the merge of the friendly and unfriendly datasets using as input information on atoms and bonds, global properties of the molecules (five attributes, e.g. hydrophobicity) and chemical structures present in the molecules, e.g. benzenic rings. On the same formulation as our version 2 a series of kernel based methods specifically designed to tackle graph classification problems have exhibited very good results but were evaluated using leave-one-out cross validation, e.g. (Kashima et al., 2003) with an error of 14.9%, and (Mahe et al., 2004) with an error of 9% (note here that both results are the best of an extensive testing of different parameter values).

For the mutagenesis version 2 *Best-distance* is better than all the other reported performances, while for diterpenes it is only second after *DES* a kernel-based distance measure. For version 1 of musk it ranks third while for version 2 the performance is quite worse than the reported result. One possible reason for the quite mediocre performance on version 2, although the same type of application with version 1, could be the very skewed distribution of set cardinalities which range from one to more than one thousand, figure 3 shows the cardinality distribution for the two problems.

It is obvious that the above comparison between the *Best-distance* and the other relational learners is biased since the former is the result of extensive experimentation[4] and of an aposterior selection of the best. The comparison would have been fair if the selection was done apriori without looking on the performance on the test set. Actually this is related to the problem of selecting among different models—parametrizations of a learning algorithm— or among different learning algorithms the most appropriate for the problem at hand. In the next section we will see how we can perform distance selection in the absence

[4] Although the same could be argued for most of the results of the other relational learners given in table 6 since when multiple results where available we reported on the best.

of any assumptions about the problem at hand. The selection strategy will also provide a basis for fair comparison with previous work.

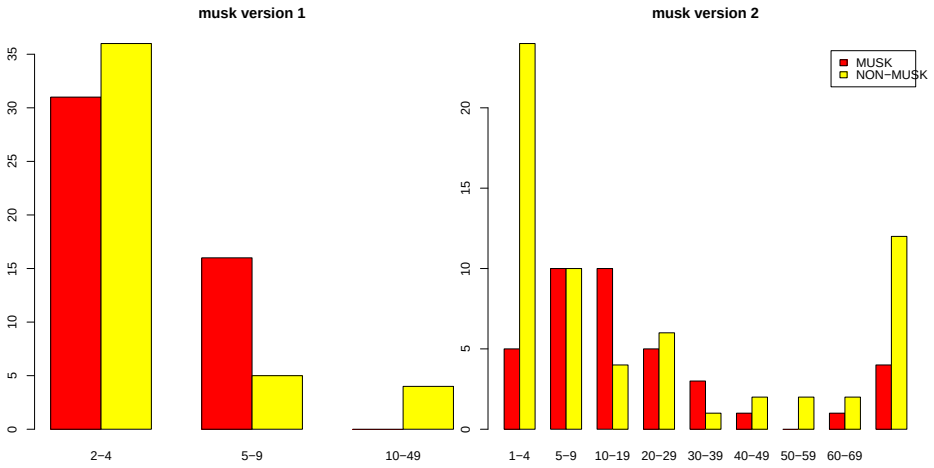

Fig. 3. Distributions of set cardinalities for the two versions of musk

5.6 Selecting relational distances

The problem of selecting the appropriate model among a set of candidate models or selecting the appropriate classification algorithm from a set of classification algorithms has received considerable attention. One of the strategies most often used to tackle it, which has been proved quite efficient in practice, is the use of cross-validation on the training set (Schaffer, 1993). Here the error of all classification algorithms is estimated on the training set using cross-validation, the algorithm selected for application is the one that has the lowest estimated error. Once a specific algorithm is selected it is retrained on the complete training set and the induced classification model is tested on the test data.

We adopt the same strategy in order to select the appropriate relational distance. That is, all relational distances are cross-validated on the training set and we choose for application the one that minimizes the estimated error. To measure the classification performance of the distance selection strategy we used the same ten-fold stratified cross-validation as in the previous experiments. For each of the training folds there is now an inner ten-fold stratified cross-validation over all the relational distances that performs the distance selection. The results of this automatic distance selection are also given in table 6 under the entry *CV-distance*. For all the problems the performance of *CV-distance* almost always matches that of the *Best-distance* with the exception of the two musk problems. For the latter this is probably due to the small sample size and the fact the there are a number of distances that have a similar performance resulting in different

selections over the different folds. For mutagenesis its performance is better than any of the reported of the other relational systems while for diterpenes it is again second after *DES*.

The performance of our system when it exploits the different available relational distances to perform distance selection compares favorably with other state of the art relational systems. At least to our knowledge this is the first relational system that allows for automatic selection of the appropriate learning bias[5] for a given classification problem. Moreover, although we do not demonstrate it here, the system allows the user to declare which set distance should be used for a given relation according to domain knowledge. If different set distances are used for different relations the resulting relational distance will be heterogeneous; again to the best of our knowledge this is the only system that offers such a flexibility.

Table 6. Error results of the CV-distance and other relational systems

System	*duke*	*musk, (version 1)*	*musk, (version 2)*
Progol			
RIBL			
Tilde		13.00	
Foil			
KES		19.00	14.50
DES			
Matchings		12.00	
$Best - distance$	19.51	15.21	23.52
$CV - distance$	19.51	18.47	25.49

	mutagenesis (version 1)	*mutagenesis (version 2)*	*diterpene*
Progol		19.00	
RIBL			08.80
Tilde		21.00	09.60
Foil		39.00	21.70
KES			05.30
DES			02.90
Matchings		17.00	06.50
$Best - distance$	14.89	12.76	03.19
$CV - distance$	15.95	12.76	03.45

[5] Different types of distances consider different matchings between the elements of the sets thus introducing different learning biases.

5.7 A closer look at Tanimoto distance

To be able to use the Tanimoto distance we had to adapt it so that it can work with graded similarities, this adaptation included the definition of a threshold parameter, θ. We will take a closer look on how the value of this parameter affects classification error. In order to do that we estimated the classification error when θ was taking the following values: 0.001..0.009 with a step of 0.001; 0.01..0.09 with a step of 0.01; 0.1 to 0.5 with a step of 0.1. The results are depicted graphically in figure 4. For three of the datasets, mutagenesis (version 1 and 2) and diterpene, the default value of the parameter is very close to the best achieved performance from which it has no significant difference (table 7). However for the two versions of musk and duke this is not the case. Better performance can be achieved if one carefully tunes the parameter value, this could be done using cross-validation for parameter tuning.

Examining figure 4 we can observe a rough pattern over all the datasets. Classification performance improves as we move away from very low values of θ, it then reaches some kind of plateau in which it gets its best values, and deteriorates as we move to higher values of θ. It is obvious that the position of the plateau depends on the classification problem. This kind of pattern is quite reasonable if one thinks how the Tanimoto based distance works. For very low values of the θ few elements of the two sets, if any at all, will be matched. As the value increases more and more elements are matched to reach at some point an optimal matching level that depends on the problem at hand. As θ continues to increase even more elements are matched; at $\theta = 1.0$ all elements are matched, this results in a Tanimoto distance that only depends on the cardinality of the two sets being compared, $D_T = \frac{|A|+|B|-2min\{|A|,|B|\}}{|A|+|B|-min\{|A|,|B|\}}$ (in this case $|A \cap B| = min\{|A|, |B|\}$, since we allow each element to be matched only once). If we assume that set A is the minimum cardinality set then $D_T = \frac{|B|-|A|}{|B|}$, which simply describes how larger is the cardinality of $|B|$ compared to the cardinality of $|A|$.

Table 7. Effect of parameter tuning on classification performance of the Tanimoto based distance. The sign in the sig column indicates whether the performance of the best threshold value was significantly better than that of the default, +, or there was no significant difference.

dataset	best θ	best error	def θ error	sig
duke	0.03,0.04,0.05	0.2941	0.3902	=
musk, (version 1)	0.2	0.2608	0.4891	+
musk, (version 2)	0.09	0.1960	0.3823	+
mutagenesis, (version 1)	0.09	0.1329	0.1489	=
mutagenesis, (version 2)	0.02,0.06,0.07,0.08	0.1223	0.1276	=
diterpene	0.009	0.0299	0.0319	=

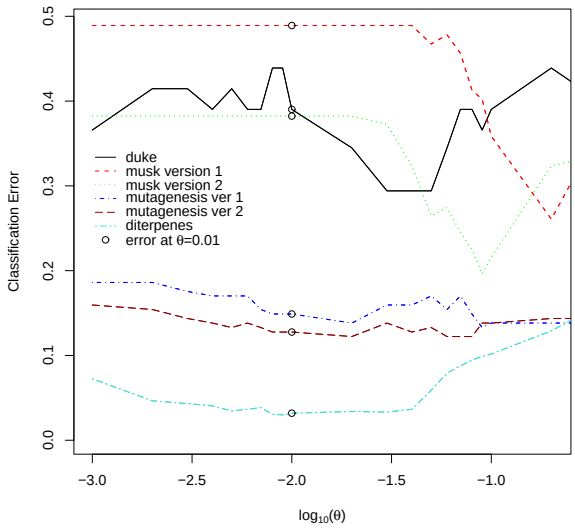

Fig. 4. Behavior of the Tanimoto distance with respect to its θ parameter.

5.8 Designing domain specific distances

In section 5.1 we limited experimentation on the mass spectrometry problem only on the *duke-M* representation since the *duke-MI* representation did not naturally fit into the context of the relational distances presented thus far. In this section we will present a new distance specifically designed to tackle problems from mass spectrometry which explicitly takes into account domain knowledge.

As already mentioned the reason for which we did not experiment with the *duke-MI* representation was that it uses on an equal basis the mass and intensity dimensions, a fact that does not conform to the semantics of the mass spectrometry problem. Nevertheless there is one more point that makes the relational distances based on sets not that appropriate for the mass spectrometry problem (even for the *duke-M* representation). In fact the peaks extracted from a mass spectrum are not really sets of points, they should be rather treated as a sequence or as a list of points where order is determined by the mass value. A distance measure that matches peaks in a manner that respects order would be more appropriate and would reflect closer the semantics of the problem compared to a set distance that does not consider order when matching peaks.

A distance measure appropriate for sequences or lists of symbols is the alignment-based edit distance (Durbin et al., 1998). In edit distance we are given a set of basic edit operations on lists (replace, insert, delete) together with a cost function that tells us how expensive each operation is. The edit distance of two lists is the cost of the lowest cost sequence of such operations that turns the

first list into the second one. The edit distance does not only give a distance of the two lists but it also provides the alignment-matching of the elements of the two lists, a fact that will prove quite useful in the case of the mass spectrometry problem.

The standard edit distance operates on sequences or lists that contain symbols, i.e. discrete values from a restricted alphabet. However this is not the case with the mass spectrometry problem. Here a list will consist either of continuous values (*duke-M* representation), or of complex objects, $((mass, intensity)$ pairs of the *duke-MI* representation). In order to adapt the edit distance so that it can operate on lists that do not necessarily consist of discrete symbols we simply have to change the cost function that it uses. Replacing the cost function with a distance between general complex objects the edit distance can then be applied to any list of complex objects. We will give now a more formal presentation of the alignment-based edit distance that we are going to use.

Given two sequences $S_1 = [s_{1_1}, ..., s_{1_n}]$, $S_2 = [s_{1_1}, ..., s_{1_m}]$, where $s_{i_j} \in \Omega$ (here Ω can be any space and not necessarily a finite symbol alphabet as it is normally the case with edit distance), and $d(.,.)$ a distance function in Ω. An *alignment* A of S_1 and S_2 is a set of two sequences S'_1 and S'_2 of equal length, l, $l \geq \max(n, m)$, constructed from the initial sequences by insertion of gaps, $-$. Aligning two elements there are only three possibilities:

- the i-th element of S_1 is aligned to a gap (insert operation),
- the i-th element of S_1 is aligned to the k-th element of S_2 (replace operation),
- the k-th element of S_2 is aligned to a gap (delete operation).

The cost of an alignment is simply the sum of the cost of all operations used to derive the alignment:

$$c(A) = \sum_{i=1}^{l} c(s'_{1_i}, s'_{2_i}),$$

where the cost of the replace operation is $c(x, y) = d(x, y),\ x, y \in \Omega$, and the cost of the insert and delete operations is a constant α, i.e., $c(x, -) = c(-, y) = \alpha$, also known as the *gap penalty*.

The alignment-based edit distance, $D_E(S_1, S_2)$, of two sequences, S_1, S_2, is then simply the minimum cost overall possible alignments of the two sequences:

$$D_E(S_1, S_2) = argmin_A c(A),$$

which is equivalent to the cost of the lowest cost sequence of operations that turns the first list into the second. Using dynamic programming (Durbin et al., 1998) it is easy to compute the edit distance with a complexity of $O(nm)$.

A mass spectrum S can be seen as a sequence of peaks, $S = [s_1, ..., s_n]$, where a peak $s_i = (s_i.M, s_i.I)$ is described by its mass, M, and its intensity, I. We can now use the alignment-based edit distance on the M representation to compute not only the distance, $D_{E_M}(S_1, S_2)$, between two mass spectra, S_1, S_2, but also their peak alignment-matching, A. The distance function $d(s_1, s_2)$ used by D_{E_M} to compute the distance of two peaks is simply the one given in formula 1

limited only on the mass dimension thus N=1 and consequently $d(s_1, s_2) = diff(s_1.M, s_2.M)$, i.e. simply the normalized difference of the two mass values. So the D_{E_M} mass spectra distance simply ignores the intensity dimension. The gap penalty α is set to one.

To compute the distance of two spectra S_1, S_2, on the *MI* representation we will use the peak alignment, A, computed on the mass dimension by the edit distance. The final distance is computed using only the intensity dimension of the peaks; we will denote this version of mass spectra distance as $D_{E_{MI}}(S_1, S_2)$. More formally:

$$D_{E_{MI}}(S_1, S_2) = \sum_{i=1}^{l} d_I(s'_{1_i}, s'_{2_i}),\ where\ (s'_{1_i}, s'_{2_i}) \in A\ and$$
$$d_I(s'_{1_i}, s'_{2_i}) := \frac{|s'_{1_i}.I - s'_{2_i}.I|}{s'_{1_i}.I + s'_{2_i}.I},\ iff\ s'_{1_i}, s'_{2_i} \neq -,\ else$$
$$d_I(s'_{1_i}, s'_{2_i}) := 1$$

We can think of the $D_{E_{MI}}$ distance as working in a new representation where the masses of the peaks become attributes and the values of these attributes are the corresponding intensities. Like that we respect the semantics of the mass spectrometry problem and do not place the two dimensions on an equal footing. We use the mass dimension only to find a matching among peaks and the intensity dimension to compute the final distance. We use the $d_I(\cdot, \cdot)$ function to compute the distance between intensities, and not the $diff(\cdot, \cdot)$, because the range of values of intensities depends on the mass scale. Peaks in small mass ranges have higher intensity values than peaks in high masses. $diff(\cdot, \cdot)$ would perform a normalization over all possible intensity values thus lower mass peaks would consistently have a higher contribution in the distance, the normalization used in $d_I(\cdot, \cdot)$ avoids that.

The proposed extension of the edit distance matches better the properties of the mass spectrometry problem since in matching two objects, here peaks, it respects order. More precisely given peaks a_1, b_1, of spectrum S_1 and peaks a_2, b_2, of spectrum S_2, with $a_1.M < b_1.M$ and $a_2.M < b_2.M$, then edit distance will never produce an alignment that violates order, i.e. the matching $a_1 \rightarrow b_2,\ b_1 \rightarrow a_2$ is not possible with the edit distance. This constraint is not respected by the set distances.

We will also incorporate one more mass spectrometry domain constraint that we have not considered thus far. Namely that two peaks can be matched only if their respective masses have a relative distance that is smaller than *err*[6]. To account for this constraint in edit distance we have to modify the $diff(m_1, m_2)$ distance function in the following way:

$$diff(s_1.M, s_2.M) := 2\alpha\ if\ \frac{|s_1.M - s_2.M|}{s_1.M + s_2.M} > err\ else\ diff(s_1.M, s_2.M) \quad (3)$$

[6] *err* is a constant that reflects the measurement error of the mass spectrometry apparatus, here *err=0.3%*. The relative distance of two masses m_1, m_2, is defined as $|m_1 - m_2|/(m_1 + m_2)$.

Like that masses with relative mass distance greater than *err* are never going to be matched by the edit distance (matching them would incur a higher cost than matching each one of them with a gap).

The comparison with the set distances is unfair since there we did not incorporate the constraints on the peak matchings based on the *err* variable. To provide an equal comparison ground we will adapt the D_{SMD} set distance (this was the one that achieved the best results on the mass spectrometry problem) and try to take into account the domain constraints described above; we will denote this new version as D_{SMD_M}. More precisely for the M representation we will modify the distance function used to compute the distances between two masses as:

$$diff(s_1.M, s_2.M) := 1.0 \; if \; \frac{|s_1.M - s_2.M|}{s_1.M + s_2.M} > err \; else \; diff(s_1.M, s_2.M) \quad (4)$$

The D_{SMD} set distance will then be computed in the standard way on the collection of the pairwise distances of the masses of the two spectra.

We will also explore an extension of the D_{SMD} for the *MI* representation, that is on the same spirit as the $D_{E_{MI}}$ distance, and denote it $D_{SMD_{MI}}$. Here the matching of the peaks of two spectra will be determined on the basis of the minimum mass distances, a peak from spectrum S_1 is matched to its minimum distance peak from S_2 spectrum and vice versa. The distance of two matched peaks will be computed using only the intensity dimension, i.e. using the $d_I(\cdot, \cdot)$ distance function given above. Peaks of a given spectrum for which there was no peak from the other spectrum within *err* distance will be considered unmatched and will contribute a maximum distance of one. Once the distances for all peaks of the two spectra have been computed the D_{SMD} is computed in the standard way.

We will now present the results of the mass spectrometry specific distances on the two representations of the duke problem, *duke-M, duke-MI*. The classification errors are given in table 8, estimated using the same ten-fold stratified cross-validation separation as all the previous experiments, and the statistical significance results in table 9.

Table 8. Error results of the mass spectrometry domain specific distances on the two representations of the Duke mass spectrometry problem.

D_{E_M}	D_{SMD_M}	$D_{E_{MI}}$	$D_{SMD_{MI}}$
19.51(2.0)	41.46(1.0)	31.70(1.5)	31.70(1.5)

The best performance is achieved by D_{E_M} with an error of 19.51%; actually this was the performance that the best set distance measure (D_{SMD}) achieved on the duke dataset. D_{E_M} is significantly better than D_{SMD_M}, the latter has a performance that is not different from that of the default classifier.

Table 9. Significance results of the mass spectrometry domain specific distances on the two representations of the Duke mass spectrometry problem.

	D_{E_M}	$D_{E_{MI}}$	D_{SMD_M}	$D_{SMD_{MI}}$
D_{E_M}		=	+	=
$D_{E_{MI}}$	=		=	=
D_{SMD_M}	-	=		=
$D_{SMD_{MI}}$	=	=	=	

In what concerns the distances that exploit the intensity dimension, i.e. operate on the MI representation, they are not significantly different than their counterparts on the M representation. In that sense it seems, at least for the distances examined here and the given dataset, that the intensity dimension does not bring in any additional discriminatory information. However these results should be taken cautiously due to the very small number of instances in the duke dataset (only 42).

To further verify or refute the above observations we experimented with one more dataset from the domain of mass-spectrometry namely the ovarian cancer dataset ((Petricoin et al., 2002) version 8-07-02). This is also a two class problem with a total number of 253 instances 162 of which correspond to diseased and 91 to healthy individuals. This time all the different domain specific distances exhibited very good performance (roughly 5% to 7% classification error, table 10) with no significant performance differences. For the ovarian dataset too the choice of representation, i.e. *M* or *MI*, does not seem to affect the performance of the distance measures and the introduction of intensity does not seem to add discriminatory information.

Table 10. Error results of the mass spectrometry domain specific distances on the two representations of the Ovarian mass spectrometry problem. $Err = 0.3\%$.

D_{E_M}	D_{SMD_M}	$D_{E_{MI}}$	$D_{SMD_{MI}}$
05.92(1.5)	05.13(1.5)	05.92(1.5)	06.71(1.5)

To see whether the domain specific distances improve classification performance over the standard set distances we evaluated all standard set distances on the M representation (table 11). Again the D_{SMD} set distance was found to be by far the best performing with an error of 11.85%. All the domain specific distances are in the case of ovarian significantly better than standard D_{SMD} set distance.

Two are the main findings of the above experiments with the domain specific distances for mass spectrometry datasets. First the edit distance measure was on the top rank for both datasets examined, this does not come as a surprise since the edit distance is the only distance that completely respects the

domain constraints computing a well defined mapping of peaks among different mass spectra. Second the incorporation of the intensity dimension on the distance measures does not appear to add discriminatory information, this seems to be a rather puzzling observation since one would expect the extra information introduced to increase the discriminatory power.

Trying to understand why incorporating intensity does not improve classification accuracy we took a closer look on how the two versions of the edit distance, D_{E_M} and $D_{E_{MI}}$, relate on the ovarian dataset. In order to do that we computed the Spearman's rank correlation coefficient of the rankings of the instances, that the two representations define, on the basis of their distances from a fixed instance and repeated that for all available instances. It turns out that the ranks induced by the two representations are highly correlated, the average Spearman's rank correlation coefficient over all instances is 0.58. Moreover for 30% of the instances they find the same most similar instance.

Both instantiations of the edit distance can be seen as consisting of two components:

$$\sum a_i + \sum b_i \tag{5}$$

where $\sum a_i$ is the contribution of the distances of the matched peaks and $\sum b_i$ the contribution of distances of the unmatched peaks. No matter whether we are working with D_{E_M} or $D_{E_{MI}}$, the latter term is simply the count of the unmatched peaks, and obviously is exactly the same for both versions of the D_E distance. The two distances differ in the computation of the $\sum a_i$ term.

In the D_{E_M} distance the term $\sum a_i$ is a sum of very small quantities, due to the thresholding introduced in formula 3, so the final distance is completely determined by the $\sum b_i$ term. Graph a) in figure 5 gives the distribution of values of the a_i and b_i terms in formula 5 for the D_{E_M} distance and graph b) the values of the two sum terms; all graphs have been computed for a specific pair of instances but the picture is very similar among all the other pairs. It is obvious that D_{E_M} can be considered as a sum of binary distances, either one when a peak is unmatched or very close to zero when two peaks are matched.

The computation of the $\sum a_i$ term in the $D_{E_{MI}}$ distance is very different. Here the distances between matched peaks are computed on the intensity dimension and there is no thresholding which results in a_i terms that are smoother and take values in the interval $[0, 1)$, graph c) in figure 5 gives the distribution of values of the a_i and b_i terms for the $D_{E_{MI}}$ distance. In that sense $D_{E_{MI}}$ has, in principle, higher discriminative power since it is constructed on the basis of finer terms. Nevertheless the $\sum b_i$ still dominates the final distance, as we can see in graph d).

The dominance of the $\sum b_i$ term explains why the two different versions of the edit distance have such a high correlation both in terms of the Spearman's rank correlation coefficient and the percentage of times that it returns the same instance as the most similar instance. To understand that let us see under which conditions the D_{E_M} and $D_{E_{MI}}$ would reach a different conclusion. Let x be a given instance, we want to determine which of the instances y and z is most

similar to x according D_{E_M} and $D_{E_{MI}}$. The decision of D_{E_M} is based on the sign of:

$$\begin{aligned} D_{E_M}(x,y) - D_{E_M}(x,z) &= \sum a_{i_{(x,y)}} + \sum b_{i_{(x,y)}} - (\sum a_{i_{(x,y)}} + \sum b_{i_{(x,z)}}) \\ &\approx \sum b_{i_{(x,y)}} - \sum b_{i_{(x,z)}} \end{aligned}$$

since as we saw above the a_i terms are very small in the case of D_{E_M}. It is easy to see that in order for $D_{E_{MI}}$ to give a different result this difference should be smaller than $\sum a_{i_{(x,z)}} - \sum a_{i_{(x,y)}}$ (where the a_i terms now are computed on the intensity values). In figure 6 we give the distribution of times that for a given instance x the two distances give the same decision on which of z, y, is most similar to x, where z, y, are all possible pairs of instances from the dataset. The distribution is computed by letting x be each instance in the dataset. As we can see in the big majority of cases the two distances agree, i.e. the $\sum a_{i_{(x,z)}} - \sum a_{i_{(x,y)}}$ difference does not overtake $\sum b_{i_{(x,y)}} - \sum b_{i_{(x,z)}}$. This explains why although the intensity brings extra discriminatory information this does not result in an increase in predictive accuracy.

When the difference $\sum b_{i_{(x,y)}} - \sum b_{i_{(x,z)}}$ is small, i.e. there is roughly the same number of unmatched peaks between x and y, and x and z, then $D_{E_{MI}}$ can give different predictions; however this does not happen in the datasets that we have examined, what is more important is simply the count of the unmatched peaks. It remains to be confirmed by experimenting with different pathologies whether this is a general trend in mass spectrometry or specific to the pathologies examined here.

Table 11. Error performance of standard set distance measures on the *ovarian-M* dataset.

Set Distance	Error (rank)
D_{SL}	37.94 (3.0)
D_{AL}	35.96 (3.5)
D_{CL}	31.22 (3.5)
D_{SMD}	11.85 (9.0)
D_H	28.45 (4.5)
D_{RIBL}	17.39 (8.5)
D_T	15.01 (8.5)
D_L	20.55 (8.0)
D_S	37.94 (3.0)
D_M	50.98 (0.0)
D_{FS}	35.96 (3.5)

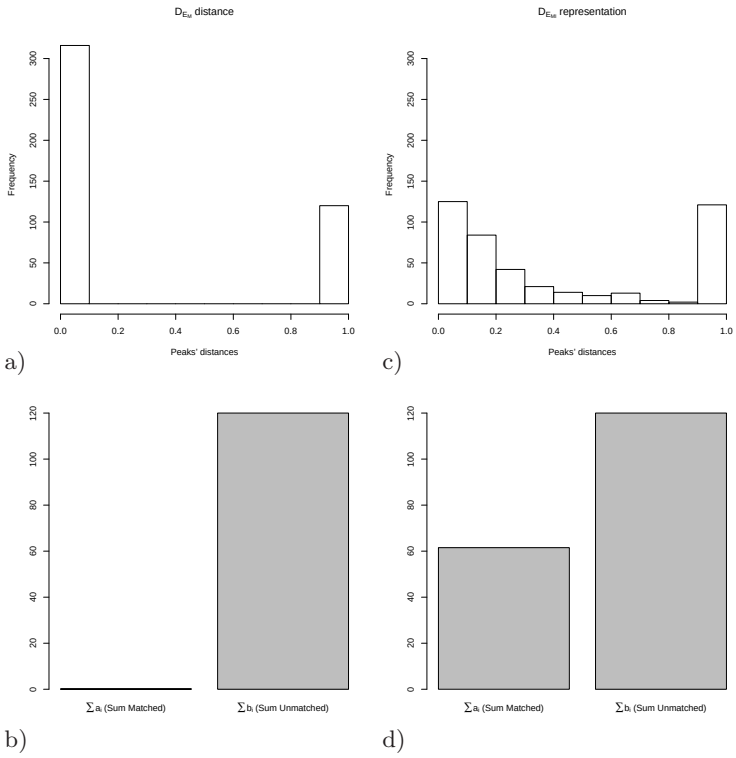

Fig. 5. For a given pair of instances of the ovarian dataset we give: the distribution of values of the a_i and b_i terms for the D_{E_M} and $D_{E_{MI}}$ distances, graphs a) and c); the values of $\sum a_i$ and $\sum b_i$ for D_{E_M} and $D_{E_{MI}}$, graphs b) and d).

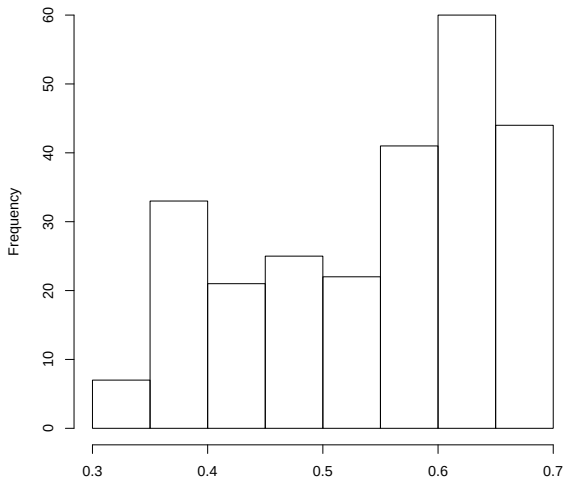

Fig. 6. For each instance, x, compute the percentage of times that the two distances, i.e. D_E on M and MI, gave the same decision over which instance y, z, is most similar to x, where y, z, are all pairs of instances on the dataset excluding x. The histogram is computed by letting x be each instance in the dataset.

6 Relation to other representations used in relational learning

There is a straight forward mapping of the notions of relational algebra to these of logic programming, (Dzeroski & Lavrac, 2001):

- a relation name R_i is a predicate symbol R_i,
- a relation R_i, i.e. a set of tuples, corresponds to a predicate R_i defined extensionally as a collection of ground facts,
- an attribute A_l of the relation R_i is an argument of the predicate R_i,
- a tuple R_{i_j} corresponds to the j^{th} ground fact built on the predicate R_i,

With respect to the above definitions and mappings the main difference of relational algebra from logic programming is the direct ability of the former to define types, i.e. each attribute of a relation has a given type defined by its domain, and associations between tuples based on the notion of foreign keys.

Two of the most popular learning paradigms in relational learning are learning from interpretations and learning from entailment (Muggleton, 1995; De Raedt, 1997; Dzeroski & Lavrac, 2001; Blockeel, 1998a). We will briefly sketch the representation differences between the two settings. In relational learning knowledge is usually separated to two parts, knowledge concerning the training examples and background knowledge, B, i.e. knowledge which is common to many examples. The differences of the two approaches are in the way knowledge is represented and in the notion of coverage that they use. We will focus mainly on the the knowledge representation.

In learning from interpretations each example is described by a set of ground facts, e. The interpretation that represents the example is the set of all the ground facts that are entailed by $e \cap B$, i.e. the minimal Herbrand model of $e \cap B$. Usually each example comes in the form of a Prolog program; background knowledge is also represented in the form of a Prolog program. In learning from interpretations there is a clear separation of examples and background knowledge. There is also a clear separation between examples since one of its main assumptions is that these are independent of each other, i.e. there are no associations between them; for example a coverage test would examine only a single example at the time.

In learning from entailment learning examples are in general represented by clauses but in practice are most often represented by a single fact, everything else even information concerning an individual example is placed within B, usually represented as a Prolog program. Thus unlike learning from interpretations there is no clear separation of the information concerning the examples and the background knowledge. Furthermore no assumption is done about the independence of examples, so training examples can be linked to other training examples. This on the one hand makes it possible to learn more complex concepts, like recursive concepts, but considerably increases the computational burden since every time the complete information has to be considered, i.e. *all* training examples together with the background knowledge. For example a coverage test on a single example would have in principle to consider not only the background knowledge but also all other examples.

To place the representation constructed by $R_t^+(\cdot)$ in the context of the representations constructed by the two aforementioned approaches let us first consider a second alternative in acquiring the complete information related with an instance R_{i_j} that avoids the use of recursion. Let us define the function set' as:

$$set'(R_{i_j}) = \{R_{j_m} \in R_j : \exists R_j \in \mathcal{R},\ \exists A_l \in \mathcal{I}_{k,R_i},\ \exists A_k \in \mathcal{I}_{k,R_j},\ R_{j_m}.A_k == R_{i_j}.A_l\}$$

This function will return the set of instances found in any relation of the relational database schema that have a key attribute that shares a value with one of the key attributes of R_{i_j}. The produced set is the set of instances associated with R_{i_j} at level one. To get all instances associated with R_{i_j} in a depth d we simply have to apply the set function to all the instances retrieved at the $d-1$ depth. To get the complete set of instances we simply get the union of all sets for each depth up to the maximum depth (algorithm 4, function $R_f^+(\cdot)$). The union guarantees that each instance appears at most once. We will first examine the similarities and differences of the example representation constructed by $R_f^+(\cdot)$ with respect to existing representational paradigms in relational learning and then consider in more detail the differences of $R_t^+(\cdot)$ from $R_f^+(\cdot)$.

Algorithm 4 Retrieving a flat description of a relational instance.

1: $R_f^+(R_{i_j})$
2: $\{S$ the complete set of instances associated with $R_{i_j}\}$
3: $\{S'$ the set of instances associated with R_{i_j} at level $d\}$
4: $S = S' = R_{i_j}$
5: **for** d=1 to MAX-DEPTH **do**
6: $\quad S' = \cup_{x \in S'} set'(x)$
7: $\quad S = S \cup S'$
8: **end for**
9: return S

Algorithm 4 is equivalent to the *INTERPRETATIONS* algorithm given by Blockeel (1998a) in order to retrieve the interpretation of an instance from the relations of a relational database[7] in the context of learning from interpretations. In fact the set of instances returned by $R_f^+(R_{i_j})$ is the interpretation that corresponds to R_{i_j} including also the related part of the background knowledge since we make no distinction between relations that belong or do not belong to the background knowledge. This is exactly the minimal Herbrand model of $e \cap B$ as it is used in learning from interpretations. There is however an important difference with respect to learning from interpretations: we do not make any assumption about the independence of learning examples. The collected set of instances describing a given learning example, R_{i_j}, could very well incorporate information about other learning examples with which R_{i_j} is associated. In that sense the representation is closer to the representation used in learning from entailment where learning examples can be described in terms of their associations

[7] Excluding these relations that are part of the background knowledge

with other learning examples and their properties, but unlike learning from entailment learning operators are not going to be applied to the complete database but only to that part which is relevant to R_{i_j} and is contained in $R_f^+(R_{i_j})$.

The representations constructed by $R_t^+(R_{i_j})$ and $R_f^+(R_{i_j})$ contain exactly the same set of instances; the union of all instances found in the $R_t^+(R_{i_j})$ gives us $R_f^+(R_{i_j})$. Nevertheless contrary to the flat representation assumed by $R_f^+(R_{i_j})$, $R_t^+(R_{i_j})$ provides a structured representation in which an instance of a relation can be present more than once inside the tree, the only restriction is that it can appear at most once in a given path from the root to a leaf. The multiple appearances are not redundant but describe different local aspects of the relational instance with each appearance occurring in a different local context. The advantage of the tree representation is that it provides readily information about the structural properties of the learning example, i.e. how the various tuples-ground facts that constitute a relational instance are related to each other.

7 Related Work

One of the most important differences between our system and most of the work on relational learning is the chosen representation language. Almost all relational learning approaches use a first order language or some subset of it (Dzeroski & Lavrac, 2001). We have chosen as representation language the relational algebra. We see two main advantages in that choice, namely the modeling of relational problems is straight forward and the underlying concepts of the system are more intuitive for database users. The relational algebra representation formalism is directly applicable to any kind of relational problem that is represented via a relational database; there is no need for conversion to another representation formalism, as it is done for example in learning from interpretations (Blockeel, 1998b), or in RIBL2.0 (Horvath et al., 2001) where examples have to be represented in the form of ground atoms.

Using relational algebra there is no need for type definitions commonly used in learning in first order languages to define relations between predicate arguments that reduce the size of the hypothesis search space (e.g. only predicate arguments of the same type can be unified or compared). For example in Progol-4.4 (Dzeroski & Lavrac, 2001) types are used to describe different categories of objects that could be matched meaningfully. Similar structures are also used in RIBL2.0 and STILL (Sebag & Rouveirol, 2000). RIBL2.0 defines together with standard types like number, constant, the type object which corresponds to identifiers representing objects. These identifiers are then used to collect all the information relevant with a given object. i.e. they have a similar role as keys in the relational algebra context. However RIBL2.0 will try to match any arguments that are of type object independently of whether they can be semantically matched, since it does not make a distinction between objects of different types but it is only based on the identifiers (e.g. cars and persons could be matched, the only thing that prevents this from happening is that the analyst should have taken care of guarantying different identifiers), thus introducing an extra

computational load when collecting the relevant information. In STILL they distinguish two types of arguments in predicates: relational and valued. Relational arguments have a similar role as arguments of type object in RIBL2.0, but here they directly indicate which arguments of the predicates constituting an example description correspond to the same object, thus modeling relations between predicate arguments and establishing a direct parallel to the notion of foreign keys.

Relational algebra provides a robust data modeling tool that automatically offers typed representations and models naturally and explicitly relations between attributes (predicate arguments in first order terminology) via foreign keys without any need for extra machinery. Exploiting the foreign keys we know exactly at which relations we should be looking for relevant information and retrieving it is just a matter of an SQL query, there is no need for predicate matching or argument matching within the predicates. Foreign keys explicitly model what can be matched with what, any other matching is not allowed. All data modeling assumptions are made explicit providing a clear view of the relations that appear in the problem. All these result in more efficient implementations, algorithmic concepts that are much closer to the way of thinking and easier to understand by typical database users, and a better understanding of the problem that should be solved. Thus we cover what we see as an important gap in the current work on multirelational learning bringing it closer to the database community. Quoting Domingos (2003) '... by looking directly at "unpacked" databases, Multi-Relational Data Mining is closer to the "real world" of programmers formulating SQL queries than traditional KDD. This means that it has the potential for wider use than the latter, but only if we address the problem of expressing MRDM algorithms and operations in terms that are intuitive to SQL programmers and OLAP users'. It is our feeling that the current work makes one step to that direction. Knobbe et al. (2000) tried to address that issue by using UML (unified modeling language) as a declarative bias language. A user can model the specifications of a problem using UML which can then be translated, by ILP specific wrappers, to a variety of logic-based languages in order to be used as input by ILP engines. UML has a close relation to relational database modeling and can be used to represent the relational schema of a given database. A relational database should thus be converted to the UML description and then using the wrappers to the input format of the tool that will be used for learning. The limited attention that database-oriented representations have received was also stressed in (Lachiche & Flach, 2000; Flach & Lachiche, 2004); the authors emphasize the need to sufficiently understand the underlying database representation in order to convert it to an ILP representation. They consider four different representations: the Entity-Relationship model, the relational model, a term-based representation, and their own flattened datalog individual-centred representation (ISP), and study their relations. The building blocks of ISP are individuals, structural predicates and properties. They show that there exists a direct mapping of the concepts used in a restricted form of entity-relationship models to the ISP representation, primary

keys are candidates for being declared as individuals, structural properties are used to represent foreign keys, and attributes of relations correspond to properties. Nevertheless ISP is yet another representation formalism distinct from the actual entity-relationship and relational models. A notable exception to the sparseness of work on learning directly from relational databases is (Perlich & Provost, 2006). They follow a propositionalization approach and extract first-order features using aggregation operators on distributions defined on identifier attributes (discrete attributes whose definition domain has very high cardinality, which usually, but not necessarily, correspond to key attributes).

Relational algebra offers a very natural approach for dealing with sets which is not readily available in first order logics being thus quite suitable for applications where the building blocks of examples are sets. Using first order terminology we perform learning over function free representations. However one could argue that relational algebra constraints the representational power of the system; application domains where examples are described by lists, trees or graphs can not be adequately tackled within the relational algebra framework. Nevertheless this is not precise; for example the representational language as described so far is able to represent with no problem unordered trees, i.e., trees in which the children of a given node have no order. To represent such a tree within the relational algebra framework we only need a single relation where each tuple in the relation corresponds to a node in the tree and contains a pointer (in the form of a foreign key referring to the same relation) to its parent. Moreover in Woznica et al. (2005a) we have shown how the relational algebra formalism can be easily extended in order to represent lists of general relational instances and defined an appropriate distance based on an extension of the edit distance along the same line that we followed here in order to tackle the representation requirements of the mass spectrometry problem. From the ability to represent lists follows also naturally the ability to represent ordered trees.

A number of systems that perform distance based learning over first order representations and languages have appeared on the literature. The most directly related with our work are KBG (Bisson, 1992), RIBL2.0 (Horvath et al., 2001), FORC and RDBC (Kirsten et al., 2001) with the latter two being based on the relational distance of RIBL2.0. In all of them the representation language is that of first order logic and the distance is computed recursively by taking into account all the information directly or indirectly related with two given instances and information that is found in deeper levels of the recursion has a lesser impact on the computation of the final distance. RIBL2.0 is much closer to our system since it is used in a classification context the other three systems are clustering systems.

In terms of expressive power RIBL2.0 can handle terms and lists which is not possible within our approach (although as mentioned previously we have shown how we can model lists in the context of relational algebra in Woznica et al. (2005a)). Unlike RIBL2.0 we are not constrained to a specific distance measure but we can induce a variety of relational distance measures depending on the distance measure that we use in order to compute the distance of the new type

of attributes that we have introduced, i.e. the set attributes. This provides much greater flexibility since it is generally accepted that there is no distance measure that is adequate for all kinds of applications, a fact which also came clear out in the experimental comparison. One now is able to choose among a number of distance measures or even plug-in easily the one that is most appropriate for the application at hand. Moreover in what can be seen as model selection the most appropriate relational distance measure can be selected via the use of internal cross-validation. Finally RIBL2.0's distance measure is a dissimilarity measure while the properties of the finally induced relational distance of our system depend directly, as we have shown, on the set distance measure that is used.

Other related work on distances in relational domains include DISTILL, (Sebag, 1997) , where a set of disjunctive hypotheses, possibly redundant, are constructed using disjunctive version spaces. These hypotheses are used in a subsequent step to map examples onto a new space where each dimension of that space corresponds to a given disjunctive hypothesis and its value is the number of disjunctions that subsume a given example. The distance between two relational instances is simply the euclidean distance in that space. The induced distance is a pseudo-metric. This approach has some commonalities with kernel approaches, of which more later in the same section, where examples are also mapped in a new space in which a semi-metric can be induced on the basis of the kernel. Hutchinson (1997) defines pseudo-metrics between ground terms based on their distance from their least general generalization. These are then coupled with the Hausdorff distance in order to compute distances between clauses, which are nothing more than disjunctions of positive or negative atoms. Nienhuys-Cheng (1997) proposed a metric on ground terms computed recursively on the different levels of the structure of the terms. Subterms that are found deeper in the structure have a lower influence on the distance between the terms. Combining this metric with the Hausdorff distance he produced a metric over sets of ground terms, i.e. interpretations, which can be used to represent the learning examples. Both of the above do not account for variables or identifiers (identifiers correspond to foreign key relations in the context of relational algebra).

Ramon and Bruynooghe (1998) extended the above approaches so that they are able to handle variables and identifiers and proposed a general framework for the definition of distances between first order logic objects in three levels. In the first level a distance between atoms is required, then exploiting that distance a distance between models or clauses can be defined (models are sets of atoms and clauses sets of literals, i.e. positive or negative atoms). In the last level one has to account for the common terms, i.e. variables or identifiers, that appear within each set and appropriately adjust the final distance. This is done by selecting over all the computed distances between the two sets of atoms or literals under all possible renaming substitutions[8] of the variables and identifiers the one that is minimal. The main problem here comes from the computational complexity of

[8] The renaming substitution amounts to trying to find the best match between the variables and identifiers of the two sets

the renaming substitutions which is exponential on the number of the variables or identifiers that are present within the sets.

More concretely if learning examples are represented via interpretations, i.e. sets of ground atoms, the distance between the sets of ground atoms will be computed using some set distance measure over the distance defined on atoms. In order to take into account the presence of identifiers this distance should be computed for each of the possible renaming substitutions of the identifiers. The final distance of two learning examples will be the minimal distance over all the renaming substitutions of their identifiers. To place the interpretation approach in the context of relational algebra in order to construct the interpretation of a given relational instance we would have to flatten the tree structure of the relational instance to its corresponding set of atoms, each tuple of a relation present within the tree structure of the relational instance will result in one atom of the interpretation. The main difference between the recursive computation of the relational distance as it is done in RIBL2.0 and our system, and the approach of Ramon and Bruynooghe (1998) is that in the latter all atoms of an interpretation contribute equally to the final distance between two relational examples while in the former the contribution of a given tuple-atom reduces with the depth in which it is found in the recursion. The recursive computation of the distance is an approximate computation whose quality increases with the depth of the recursion while the approach of Ramon and Bruynooghe (1998) is an exact computation; however the cost that one has to pay is the computation of the optimal matching of identifiers which is exponential to the number of identifiers. In fact because of that one has often to resolve to an approximate computation as it is done by Ramon (2002) chapter 5 for the mutagenicity problem.

Probably the most important difference from all of the previous work on relational distances is that we are not limited to a single distance measure. All previous systems implement a single relational distance measure. Instead our system offers a variety of distance measures among which the user can choose the one that better matches the problem at hand. In the case that such knowledge is not available distance selection based on cross validation can be employed. Furthermore the system is implemented in such a way that it is straightforward to incorporate new distances. Last but not least the presence of different distances allows for their simultaneous use in a single relational problem. For example the user can specify that for sets of tuples coming from one relation a given set distance measure should be used, while for sets of tuples of another relation a different distance should be used. This results in a final relational distance that is heterogeneous and matches better the problem requirements, none of the previous work offers for that flexibility.

Another way of defining distances is by exploiting kernels over structured domains. A kernel over a structured domain can induce a distance measure which is actually a pseudo-metric. The basic idea here also is that a structured object can be decomposed to its components on which kernels are computed and then combined in such a way so that the final combination is a kernel over the structured domain. Gartner et al. (2004) introduced kernels defined over

structured data. The representation language that they used was that of typed λ-calculus which allows for the natural representation of sets and multisets, a main difference from first order terms. The proposed approach allows also for the modeling of more complex structures like lists, trees or graphs. Individuals are represented as terms of the typed λ-calculus formal logic. The composition of individuals from their parts is expressed in terms of a fixed type structure that is made up of function types, product types and type constructors. Function types are used to represent types corresponding to sets and multisets, product types represent types corresponding to fixed size tuples and type constructors for arbitrary structured objects such as lists, trees, graphs. Each type defines a set of terms that represent instances of that type. Kernels are then defined on each of the functions, product types and type constructors, along with a kernel defined on the data constructors associated with each type constructor.

There is a growing literature on kernels defined over structured objects. Most of the work on that area is based on a decomposition of the structured objects to subparts (Haussler, 1999), the computation of kernels on these subparts, and the combination of these kernels in a way that the resulting function is still a kernel, i.e., positive definite. The combination is defined by taking the sum of kernels over all the possible pairs of subparts of the two structured objects. When the structured objects are sets of vectors this corresponds to the sum of kernels on all pairs of elements of the two sets (Gartner et al., 2004; Woznica et al., 2005b). Compared to distances over sets one limitation of kernels on sets is that they consider all pairs of elements (in order to guarantee positive definiteness), while as we have seen in distances over sets it is possible to define mappings of elements resulting in greater flexibility. Nevertheless the advantage of kernels is that they can be used within a support vector machine which can potentially result in higher classification accuracy and speed compared to the nearest neighbor classifier; the latter due to the small number of support vectors that will be established resulting in a small number of kernel computations. For a survey of the related work on kernels on structured objects the reader is referred to (Gartner, 2003).

8 Conclusions

We proposed a novel and general framework for learning over relational schemata that builds directly over the concepts of relational algebra, unlike the existing work which was mainly logic programming oriented. Our algorithm works in a recursive manner traversing the relations of the relational schema in order to gather the relevant information. The recursion among the different relations is guided by the concept of links which are based on the notion of foreign keys. Foreign keys provide a natural and intuitive way to provide a declarative bias and render unnecessary type definitions extensively used in inductive logic programming. At each moment they provide direct access to the relevant information providing increased efficiency. The system is directly operational on any classification problem represented via means of a classical relational database, without

any need for conversion as it is the case with almost all first order based systems. The algorithmic concepts are expressed in terms of relational algebra terminology bringing the underlying notions much closer to the intuition of the database community thus increasing the potential of a wider use by less expert users.

To be able to work with this type of relational structures we defined a new attribute type that we call set attribute. It corresponds to attributes whose values can be whole sets of instances and is build on top of the foreign keys notion. We adopted the view that each association in which a relation participates via a foreign key link adds one more attribute to the set of the standard attributes of that relation.

To come with a full fledged relational learner we had to define appropriate operators for the new attribute type. A very natural choice was to turn to distance operators defined over sets of instances. The algorithm is not constrained to a specific distance measure over the attributes of type set. Different distance measures for this type of attributes give rise to different instantiations of our algorithm. Classification performance critically depends on the choice of the distance measure which should be guided by domain knowledge. The right choice of measure gives classification results that compare favorably with these of some of the best known relational learners reported in literature. Moreover by exploiting a cross validation based distance selection, to what amounts to model selection, we could achieve performances that matched or were very close to the performance of the best distance measure in a given domain. Finally the user has the possibility to specify which set distance measure should be used for sets of tuples of a given relation resulting in relational distance measures which are possibly heterogeneous.

The formalization of the relational distance measures on the basis of the set type allowed us to study their formal properties and show that these are directly determined by the properties of the distance measure used to compute distances between attributes of type set. We have shown that the finally induced distance measure is a metric if and only if the set distance measures that we use are metrics.

We proceeded to a qualitative characterization of the various instantiations of the relational distance measure on the basis of the characteristics of the set distance measure that they employed. This characterization was supported by the empirical results. Distance sets that were similar with respect to their qualitative characterization were also very similar in terms of their relative performance on the relational benchmarks examined. However it remains to be seen whether these similarities would carry on if a greater number of benchmarks is considered.

Finally for one of the application problems that we examined, mass spectrometry, we defined domain specific distances that closely match the domain requirements. These distances where based on an extension of the well known edit distance to problems where there is available a graded similarity. In fact in (Woznica et al., 2005a) we have followed the same idea and extended the relational representation presented here so that it can also handle lists defined on

elements of a given relation and not only sets, thus significantly increasing the representational abilities of our system.

With regards to future work it is straight forward to exploit the structure of the relational instances with other learning paradigms like for example decision trees. In this case the tests of the decision tree could be performed on any of the attributes of a relation that is a part of the relational instance, however some extensions should be considered in order to account for the multiple values that one attribute might assume. Currently we are still focusing on distance based approaches where we use convolution based kernels in order to define distances over the relational structures (Woznica et al., 2005b). Convolution based kernels provide an elegant and easy way for the treatment of relational structures, nevertheless as already mentioned they have a reduced flexibility in the case of kernels on sets since they usually average over all possible mappings of the elements of two sets. We are investigating ways to exploit the increased flexibility of the set distances to define new kernels on sets.

One of the limitations of the current system, and in fact of any classification approach that is based on the nearest neighbors classification rule, is the long classification time. In the case of relational learning classification time is even longer due to the complex relational structures that have to be traversed in order to compute the final distance. One approach to tackle that problem is by reducing the recursion depth. As we show the quality of the distance computation does not considerably reduce when we limit the recursion depth. We would like to develop a more systematic approach to choose the minimum recursion depth will retaining maximum discriminatory power. Nevertheless the main bulk of the computational burden comes from the number of instances that are contained in the training set. One approach that can significantly reduce classification time is to use a prototype based nearest neighbor algorithm where only few instances will be used to perform classification, of course the main challenge here will be how to select these prototypes. An alternative is the use of kernels on relational structures coupled with a support vector machine algorithm, as we have done in (Woznica et al., 2006). The application of the support vector machine produces classification models which are based on a small number of prototypes, support vectors, in which case only few kernel computations are required in order to classify a new relational instance, thus considerably speeding up the classification time.

We would like to examine in detail whether the recursive computation of the relational distance which actually assigns less importance to relations that are found in the deeper levels of recursion is actually a better strategy than considering all the relations that form a multi-relational instance in a flat manner with equal contribution to the final distance.

A careful study of the performance of the basic set distance measures is needed in order to see how these are affected by the properties of the sets between which the distance is computed, namely the distribution of the cardinalities of these sets. Loosely speaking a set distance measure computes a distance in a high dimensional space where the number of dimensions is equal to the product

of the cardinalities of the two sets. Depending on the set distance measure used all, some or a single dimension are used to compute the distance.

Instead of model selection we could perform model combination where we learn the importance of each distance measure for a given domain by using for example some form of stacked generalization. Moreover as we have seen the system allows also for the specification of a different set distance measure for each of the attributes of type set so that it can reflect better the properties of that attribute. Instead of performing the distance selection globally it would be interesting to derive a strategy that does that on a local level, i.e. for each attribute of type set, without on the same time increasing overwhelmingly the computational burden.

A Detailed Significance Results

	duke										
	D_{FS}	D_L	D_M	D_{AL}	D_{CL}	D_H	D_{RIBL}	D_{SL}	D_{SMD}	$D_T - \theta$ 0.01	D_S
D_{FS}		=	=	=	=	=	=	=	=	=	=
D_L	=		=	=	+	=	=	=	=	=	=
D_M	=	=		=	=	=	=	=	=	=	=
D_{AL}	=	=	=		=	=	=	=	-	=	=
D_{CL}	=	-	=	=		=	-	=	-	=	-
D_H	=	=	=	=	=		=	=	-	=	=
D_{RIBL}	=	=	=	=	+	=		=	=	=	=
D_{SL}	=	=	=	=	=	=	=		-	=	=
D_{SMD}	=	=	=	+	+	+	=	+		=	=
$D_T - \theta$ 0.01	=	=	=	=	=	=	=	=	=		=
D_S	=	=	=	=	+	=	=	=	=	=	

	musk (version 1)										
	D_{FS}	D_L	D_M	D_{AL}	D_{CL}	D_H	D_{RIBL}	D_{SL}	D_{SMD}	$D_T - \theta$ 0.01	D_S
D_{FS}		=	+	=	+	=	+	=	=	+	=
D_L	=		+	=	+	=	+	=	=	+	=
D_M	-	-		-	-	-	=	-	-	=	-
D_{AL}	=	=	+		+	=	+	=	=	+	=
D_{CL}	-	-	+	-		-	+	-	-	=	-
D_H	=	=	+	=	+		+	=	=	+	=
D_{RIBL}	-	-	=	-	-	-		-	-	=	-
D_{SL}	=	=	+	=	+	=	+		=	+	=
D_{SMD}	=	=	+	=	+	=	+	=		+	=
$D_T - \theta$ 0.01	-	-	=	-	=	-	=	-	-		-
D_S	=	=	+	=	+	=	+	=	=	+	

	musk (version 2)										
	D_{FS}	D_L	D_M	D_{AL}	D_{CL}	D_H	D_{RIBL}	D_{SL}	D_{SMD}	$D_T - \theta$ 0.01	D_S
D_{FS}		=	+	=	=	=	+	=	=	=	=
D_L	=		=	=	=	=	=	=	=	=	=
D_M	-	=		=	=	-	=	=	-	=	=
D_{AL}	=	=	=		=	=	=	=	=	=	=
D_{CL}	=	=	=	=		=	=	=	=	=	=
D_H	=	=	+	=	=		+	=	=	=	=
D_{RIBL}	-	=	=	=	=	-		=	-	=	=
D_{SL}	=	=	=	=	=	=	=		=	=	=
D_{SMD}	=	=	+	=	=	=	+	=		=	=
$D_T - \theta$ 0.01	=	=	=	=	=	=	=	=	=		=
D_S	=	=	=	=	=	=	=	=	=	=	

Table 12. Detailed results of McNemar's test of statistical significance. For a given cell the +/- signs indicate that the predictive performance of the distance associated with the corresponding row is significantly better/worse than that of the distance associated with the given column, the = sign indicates that the performance difference is not significant.

	mutagenicity (version 1)										
	D_{FS}	D_L	D_M	D_{AL}	D_{CL}	D_H	D_{RIBL}	D_{SL}	D_{SMD}	$D_T - \theta$ 0.01	D_S
D_{FS}		=	=	+	+	=	=	=	=	=	=
D_L	=		+	+	+	+	+	+	=	=	=
D_M	=	-		+	+	=	=	=	-	-	-
D_{AL}	-	-	-		=	-	-	-	-	-	-
D_{CL}	-	-	-	=		-	-	-	-	-	-
D_H	=	-	=	+	+		=	=	-	-	=
D_{RIBL}	=	-	=	+	+	=		=	-	-	-
D_{SL}	=	-	=	+	+	=	=		-	-	=
D_{SMD}	=	=	+	+	+	+	+	+		=	=
$D_T - \theta$ 0.01	=	=	+	+	+	+	+	+	=		=
D_S	=	=	+	+	+	=	+	=	=	=	

	mutagenicity (version 2)										
	D_{FS}	D_L	D_M	D_{AL}	D_{CL}	D_H	D_{RIBL}	D_{SL}	D_{SMD}	$D_T - \theta$ 0.01	D_S
D_{FS}		=	+	+	+	+	+	+	=	=	=
D_L	=		+	+	+	+	+	+	=	=	=
D_M	-	-		+	+	=	=	=	-	-	-
D_{AL}	-	-	-		+	-	-	-	-	-	-
D_{CL}	-	-	-	-		-	-	-	-	-	-
D_H	-	-	=	+	+		=	=	-	-	=
D_{RIBL}	-	-	=	+	+	=		=	-	-	=
D_{SL}	-	-	=	+	+	=	=		-	-	=
D_{SMD}	=	=	+	+	+	+	+	+		=	=
$D_T - \theta$ 0.01	=	=	+	+	+	+	+	+	=		+
D_S	=	=	+	+	1	=	=	=	=	-	

	diterpenes										
	D_{FS}	D_L	D_M	D_{AL}	D_{CL}	D_H	D_{RIBL}	D_{SL}	D_{SMD}	$D_T - \theta$ 0.01	D_S
D_{FS}		=	-	+	+	+	=	+	=	-	=
D_L	=		-	+	+	+	=	+	=	-	=
D_M	+	+		+	+	+	+	+	+	=	+
D_{AL}	-	-	-		+	-	-	-	-	-	-
D_{CL}	-	-	-	-		-	-	-	-	-	-
D_H	-	-	-	+	+		-	+	-	-	-
D_{RIBL}	=	=	-	+	+	+		+	=	-	=
D_{SL}	-	-	-	+	+	-	-		-	-	-
D_{SMD}	=	=	-	+	+	+	=	+		-	=
$D_T - \theta$ 0.01	+	+	=	+	+	+	+	+	+		=
D_S	=	=	-	+	+	+	=	+	=	=	

Table 13. Same as the previous table.

Bibliography

Aha, D. W. (1997). Editorial. *Artificial Intelligence Review*, *11*, 1–6. Special Issue on Lazy Learning.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance based learning algorithms. *Machine Learning*, *6*, 37–66.

Anglano, C., Giordana, A., Bello, G., & Saitta, L. (1998). An experimental evaluation of coevolutive concept learning. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 19–27).

Bisson, G. (1992). Conceptual clustering with a similarity measure. *Proceedings of the 10th European Conference on Artificial Intelligence* (pp. 458–462).

Blockeel, H. (1998a). *Top-down induction of logical decision trees*. Doctoral dissertation, Department of Computer Science, K.U.Leuven. Chapter 4, First Order Logic Representations.

Blockeel, H. (1998b). *Top-down induction of logical decision trees*. Doctoral dissertation, Department of Computer Science, K.U.Leuven.

Blockeel, H., & De Raedt, L. (1997). *Top-down induction of logical decision trees* (Technical Report). Department of Computer Science, K.U.Leuven.

Campa, M., Fitzgerald, M., & Patz, E. (2003). Editorial: Proteomics 9/2003. *Proteomics*, *9*.

De Raedt, L. (1997). Logical settings for concept-learning. *Artificial Intelligence*, *95*, 187–201.

Dietterich, T. G., Lathrop, R. H., & Lozano-Perez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*, 31–71.

Domingos, P. (2003). Prospects and challenges of multi-relational data mining. *SIGKDD, Explorations*, *5*, 80–83.

Duda, R., Hart, P., & Stork, D. (2001a). *Pattern classification and scene analysis*. John Willey and Sons.

Duda, R., Hart, P., & Stork, D. (2001b). *Pattern classification and scene analysis*, chapter Unsupervised Learning and Clustering. John Willey and Sons.

Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological sequence analysis, probabilistic models of proteins and nucleid acides*. Cambridge University Press.

Dzeroski, S., & Lavrac, N. (2001). *Relational data mining*. Springer.

Dzeroski, S., Schulze-Kremer, S., Heidtke, K. R., Siems, K., & Wettschereck, D. (1996). Applying ILP to diterpene structure elucidation from 13 c NMR spectra. *Inductive Logic Programming Workshop* (pp. 41–54).

Eiter, T., & Mannila, H. (1997). Distance measures for point sets and their computation. *Acta Informatica*, *34*, 109–133.

Flach, P., & Lachiche, N. (2004). Naive bayesian classification of structured data. *Machine Learning*, *57*, 233–269.

Gartner, T. (2003). Kernel-based multi-relational data mining. *SIGKDD Explorations*, *5*, 49–58.

Gartner, T., Lloyd, J., & Flach, P. (2004). Kernels and distances for structured data. *Machine Learning*. To appear.

Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. Springer.

Haussler, D. (1999). *Convolution kernels on discrete structures* (Technical Report). University of California, Santa Cruz.

Horvath, T., Wrobel, S., & Bohnebeck, U. (2001). Relational instance-based learning with lists and terms. *Machine Learning*, *43*, 53–80.

Hutchinson, A. (1997). Metrics on terms and clauses. *Proceedings of the Ninth European Conference on Machine Learning* (pp. 138–145).

Kashima, H., Tsuda, K., & Inokuchi, A. (2003). Marginalized kernels between labeled graphs. *Machine Learning, Proceedings of the Twentieth International Conference* (pp. 321–328).

Kirsten, M., Wrobel, S., & Horvath, T. (2001). *Relational data mining*, chapter Distance Based Approaches to Relational Learning and Clustering, 212–232. Springer.

Knobbe, A. J., Siebes, A., Blockeel, H., & van der Wallen, D. (2000). Multi-relational data mining, using uml for ilp. *Proceedings of the fourth European Conference on Principles of Data Mining and Knowledge Discovery* (pp. 1–12).

Lachiche, N., & Flach, P. (2000). A first-order representation for knowledge discovery and bayesian classification on relational data. *Proceedings of Workshop on Data Mining, Decision Support, Meta-learning and ILP : Forum for Practical Problem Presentation and Prospective Solutions (PKDD2000)*.

Mahe, P., Ueda, N., Akutsu, T., Perret, J.-L., & Vert, J.-P. (2004). Extensions of marginalized graph kernels. *Machine Learning, Proceedings of the Twenty-first International Conference*. ACM.

Muggleton, S. (1995). Inverse entailment and progol. *New Generation Computing*, *13*, 245–286.

Nienhuys-Cheng, S.-H. (1997). Distances between herbrand interpretations: a measure for aproximations to a target concept. *Inductive Logic Programming, Seventh International Workshop* (pp. 213–226).

Nienhuys-Cheng, S.-H. (1998). Distances and limits on Herbrand interpretations. *Inductive Logic Programming, Eigth International Workshop* (pp. 250–260).

Perlich, C., & Provost, F. (2006). Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, *62*, 65–105.

Petricoin, E., Ardekani, A., Hitt, B., Levine, P., Fusaro, V., Steinberg, S., Mills, G., Simone, C., Fishman, D., Kohn, E., & Liotta, L. (2002). Use of proteomic patterns in serum to identify ovarian cancer. *The Lancet*, *395*, 572–577.

Ramon, J. (2002). *Clustering and instance based learning in first order logic*. Doctoral dissertation, Department of Computer Science, K.U.Leuven, Leuven.

Ramon, J., & Bruynooghe, M. (1998). A framework for defining distanes between first-order logic objects. *Inductive Logic Programming, Eight International Workshop* (pp. 271–280).

Ramon, J., & Bruynooghe, M. (2001). A polynomial time computable metric between point sets. *Acta Informatica*, *37*, 765–780.

Schaffer, C. (1993). Selecting a classification method by cross validation. *Machine Learning*, *13*, 135–143.

Sebag, M. (1997). Distance induction in first order logic. *Proceedings of Inductive Logic Programming Conference*.

Sebag, M., & Rouveirol, C. (2000). Resource-bounded relational reasoning: Induction and deduction through stochastic matching. *Machine Learning*, *38*, 41–62.

Srinivasan, A., Muggleton, S., King, R., & Sternberg, M. (1994). Mutagenesis: ILP experiments in a non-determinate biological domain. *Proceedings of the 4th International Workshop on Inductive Logic Programming* (pp. 217–232).

Ullman, J. (1982). *Principles of databse systems*. Computer Science Press.

Woznica, A., Kalousis, A., & Hilario, M. (2005a). Distance-based learning over extended relational algebra structures. *Proceedings of the 15th International Conference on Inductive Logic Programming (ILP-2005) (late breaking papers)*.

Woznica, A., Kalousis, A., & Hilario, M. (2005b). Kernels over relational algebra structures. *Proceedings of the Ninth Pasific Asia Conference on Knowledge Discovery, PAKDD*. Springer.

Woznica, A., Kalousis, A., & Hilario, M. (2006). Kernels on lists and sets over relational algebra: an application to classification of protein fingerprints. *The 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining.*.