Kernels over relational algebra structures

Adam Woźnica[1] and Alexandros Kalousis[1] and Melanie Hilario[1]

University of Geneva, Computer Science Department
Rue General Dufour 24, 1211 Geneva 4, Switzerland
{woznica, kalousis, hilario}@cui.unige.ch

Abstract. In this paper we present a novel and general framework based on concepts of relational algebra for kernel-based learning over relational schema. We exploit the notion of foreign keys to define a new attribute that we call instance-set and we use this type of attributes to define a tree like structured representation of the learning instances. We define kernel functions over relational schemata which are instances of R-Convolution kernels and use them as a basis for a relational instance-based learning algorithm. These kernels can be considered as being defined over typed and unordered trees where elementary kernels are used to compute the graded similarity between nodes. We investigate their formal properties and evaluate the performance of the relational instance-based algorithm on a number of relational benchmark datasets.

1 Introduction

Learning from structured data has recently attracted a great deal of attention within the machine learning community ([1]). The reason for this is that it is in general hard to represent most of real world data as a flat table. Recently it has been also realized that one strength of the kernel-based learning paradigm is its ability to support input spaces whose representation is more general than attribute-value ([2–5]). The latter is mainly due to the fact that the proper definition of a kernel function enables the structured data to be embedded in some linear feature space without the explicit computation of the feature map. This can be achieved as long as we are able to define a function which is both positive definite and appropriate for the problem at hand. The main advantage of this approach is that any propositional algorithm which is based on inner products can be applied on the structured data.

In this paper we bring kernel methods and learning from structured data together. First we propose a novel database oriented approach and define our algorithms and operations over relational schema where learning examples come in the form of interconnected relational tables. There exists a single main relation each tuple of which gives rise to a relational instance that spans through the relations of the relational schema. Second we define a family of kernel functions over relational schemata which are generated in a "syntax-driven" manner in the sense that the input description specifies the kernel's operation. We show that the resulting kernels can be considered as kernels defined on typed, unordered trees and analyze their formal properties. We exploit these kernels to define a relational distance and experiment with and compare a number of kernel-based distance measures.

2 General Description of the Relational Instance Based Learner

Consider a general relational schema that consists of a set of relations $\mathcal{R} = \{R\}$. Each tuple, $R_{i\_}$, of a relation R represents a relationship between a set of values $\{R_{ij}\}$ of the set of attributes $\{R_{\_j}\}$ related via R. The *domain*, $D(R_{\_j})$, of attribute $R_{\_j}$ is the set of values that the attribute assumes in relation R. An attribute $R_{\_j}$ is called a *potential key* of relation R if it assumes a unique value for each instance of the relation. An attribute $X_{\_i}$ of relation X is a *foreign key* if it references a potential key $R_{\_j}$ of relation R and takes values in the domain $D(R_{\_j})$ in which case we will also call the $R_{\_j}$ a *referenced key*. The association between $R_{\_j}$ and $X_{\_i}$ models one-to-many relations, i.e. one element of R can be associated with a set of elements of X. A *link* is a quadruple of the form $l(R, R_{\_k}, X, X_{\_l})$ where either $X_{\_l}$ is a foreign key of X referencing a potential key $R_{\_k}$ of R or vice versa. We will call the set of attributes of a relation R that are not keys (i.e. referenced keys, foreign keys or attributes defined as keys but not referenced) *standard attributes* and denote it with $\{S_{\_j}\}$. The notion of links is critical for our relational learner since it will provide the basis for the new type of attributes, i.e. the instance-set type that lies in the core of our relational representation.

Accessing a relational instance For a given referenced key $R_{\_k}$ of relation R we denote by $L(R, R_{\_k})$ the set of links $l(R, R_{\_k}, X, X_{\_f_k})$ in which $R_{\_k}$ is referenced as a foreign key by $X_{\_f_k}$ of X. We will call the multiset of X relations, denoted as $L(R, R_{\_k})\{1\}$, the *directly dependent relation* of R for $R_{\_k}$. By $L(R, \_) = \cup_k L(R, R_{\_k})$ we note the list of all links in which one of the potential keys of R is referenced as a foreign key by an attribute of another relation. Similarly for a given foreign key $R_{\_f_k}$ of R, $L^{-1}(R, R_{\_f_k})$ will return the link $l(R, R_{\_f_k}, X, X_{\_k})$ where $X_{\_k}$ is the potential key of X referenced by the foreign key $R_{\_f_k}$. We will call relation X the *directly referenced relation* of R for $R_{\_f_k}$ and denoted it as $L^{-1}(R, R_{\_f_k})\{1\}$. If R has more than one foreign keys then by $L^{-1}(R, \_) = \cup_{f_k} L^{-1}(R, R_{\_f_k})$ we denote the set of all links of R defined by the foreign keys of R, and by $L^{-1}(R, \_)\{1\}$ the corresponding multiset of relations to which these foreign keys refer.

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 instance, $M_{i\_}$, of the M relation will give rise to one *relational instance*, $M^+_{i\_}$, i.e. an instance that spans the different relations in $\mathcal{R}$. To get the complete description of $M^+_{i\_}$ one will have to traverse possibly the whole relational schema according to the associations defined in the schema. More precisely given instance $M_{i\_}$ we create a relational instance $M^+_{i\_}$ that will have the same set of standard attributes $\{S_{\_j}\}$ and the same values for these attributes as $M_{i\_}$ has. Furthermore each link $l(M, M_{\_k}, R, R_{\_f_k}) \in L(M, \_) \cup L^{-1}(M, \_)$ adds in $M^+_{i\_}$ one attribute of type *instance-set*. The value of an attribute of type instance-set is defined based on the link l and it will be the set of instances (actually also relational instances) with which $M_{i\_}$ is associated in relation R when we follow the link l. By recursive application of this procedure at each level we obtain the complete description of the relational instance $M^+_{i\_}$.

We should note here that the relational instance $M_{i\_}^{+}$ can be seen as a tree like structure whose root contains $M_{i\_}$. Each node at the second level of the tree is an instance from some relations $R \in \mathcal{R}$ related via a link $l(M, M_{\_k}, R, R_{\_f_k})$ with instance $M_{i\_}$. In the same way nodes at the d level of the tree are also instances from a given relation. Each of these instances is related with one of the instances found in nodes of the $d-1$ level. In other words $M_{i\_}^{+}$ is a tree where each node is one tuple from one of the relations that are part of the description of the relational instance and the connections between the nodes are determined by the foreign key associations defined within the relational schema. This means that the resulting tree is *typed* (i.e. each node is of a given type determined by one of the relations within the relational schema) and *unordered* (i.e. the order of instances appearing as children of a given node is not important). To limit the size of the resulting tree and to make the computation of our algorithms less expensive we sometimes prune the tree to a specific depth d.

The presence of two foreign keys, $X_{\_f_{1_k}}, X_{\_f_{2_k}}$, in relation X on the same potential key $R_{\_k}$ of relation R has a side effect. It is now possible to have self-replicating loops when traversing the relational schema in order to get the complete relational description of an instance. In order to avoid that kind of situation we will have to keep track of all the instances of the different relations that appear in a given path of the recursion; the moment an instance appears a second time in the given recursion path the recursion terminates.

Having an adequate way to handle attributes of type instance-set is the heart of the problem that should be tackled in order to come with a relational learning algorithm that could exploit the relational structure that we have sketched thus far. In the next section we define kernel-based distances operating on this type of relational structures.

3 Kernels

A kernel is a symmetric function $k : X \times X \rightarrow \Re$, where X is any set, such that for all $x, y \in X$, $k(x, y) =< \phi(x), \phi(y) >$ where ϕ is a mapping from X to a feature space Φ embedded with an inner product, actually a *pre-Hilbert* space.

We should note here that the definition of kernels does not require that the input space X be a vector space –it can be any kind of set which we can embed in the feature space Φ via the kernel. This property allows us to define kernels on any kind of structures that will embed these structures in a linear space. The attractiveness of kernels lies in the fact that one does not need to explicitly compute the mappings $\phi(x)$ in order to compute the inner products in the feature space.

Examples of kernels defined on vector spaces are the polynomial kernel $k_{P_{p,a}}(x, y) = \frac{(<x,y>+a)^p}{\sqrt{(<x,x>+a)^p}\sqrt{(<y,y>+a)^p}}$ where $a \in \Re, p \in N^+$ and the Gaussian RBF kernel $k_{G_\gamma}(x, y) = e^{-\gamma\|x-y\|^2}$ where $\gamma \in \Re$. This two kernels are the ones we are going to use in our experiments, however any valid elementary kernel could be plugged in so that it depicts the domain knowledge of the relational problem.

Kernels on relational instances In order to define a kernel on the relational instances we will distinguish two parts, $R_{i\boldsymbol{s}}, R_{i\boldsymbol{set}}$, in each relational instance $R_{i\_}$ found in a

relation R. $R_{i\boldsymbol{s}}$ denote the vector of standard attributes $\{S_{\_j}\}$ of R, let $D_s = |\{S_{\_j}\}|$; $R_{i\boldsymbol{set}}$ denotes the vector of attributes that are of type instance-set and for a relation R are given by $L(R,\_) \cup L^{-1}(R,\_)$, let $D_{set} = |L(R,\_) \cup L^{-1}(R,\_)|$.

Let $R_{i\_} = (R_{i\boldsymbol{s}}, R_{i\boldsymbol{set}}) \in X = X_{\{S_{\_j}\}} \times X_{set}$ where $X_{set} = X_{set_1} \times X_{set_2}, \times ... \times X_{set_{D_{set}}}$ and $R_{i\boldsymbol{s}} \in X_{\{S_{\_j}\}}, R_{i\boldsymbol{set}} \in X_{set}$. Given this formalism we defined two relational kernels: *direct sum kernel* ($K_\Sigma(.,.)$) and the kernel which is derived by direct application of the *R-Convolution* kernel ([3]) on the set X ($K_\Re(.,.)$). Since these kernels are defined over multi-relational instances they are computed following the same recursion path as the retrieval of a multi-relational instance.

The *direct sum kernel* is obtained by exploiting the fact that the direct sum of kernels is itself a kernel , [6], which would give the following kernel on the set X (if $|\{S_{\_j}\}| \neq 0$):

$$K_\Sigma(R_{i\_}, R_{j\_}) = k_s(R_{i\boldsymbol{s}}, R_{j\boldsymbol{s}}) + \sum_{l=1}^{D_{set}} K_{set}(R_{iset_l}, R_{jset_l}) \tag{1}$$

where $k_s(.,.)$ can be any type of elementary kernel defined on the set $\{S_{\_j}\}$ of the standard attributes of R and $K_{set}(.,.)$ is a kernel between sets which will be defined in section 3. If $|\{S_{\_j}\}| = 0$ then the kernel defined over standard attributes vanishes and we obtain $K_\Sigma(R_{i\_}, R_{j\_}) = \sum_{l=1}^{D_{set}} K_{set}(R_{iset_l}, R_{jset_l})$. It is obvious that the value of $K_\Sigma(.,.)$ is affected by the number of attributes that are of type instance-set since it contains a sum of kernels defined on these attributes. If we were working with a single-table propositional problem that would not pose a problem. However in the multi-relational context this is a problem since the kernel on the main relation is based on recursive computations of kernels in relations at the next levels which can have varying D_s and D_{set}. In order to factor out that effect among different relations we use a normalized version of K_Σ (if $|\{S_{\_j}\}| \neq 0$) defined as:

$$K_\Sigma(R_{i\_}, R_{j\_}) = \frac{K_\Sigma(R_{i\_}, R_{j\_})}{1 + D_{set}} \tag{2}$$

If $|\{S_{\_j}\}| = 0$ we have $K_\Sigma(R_{i\_}, R_{j\_}) = \frac{K_\Sigma(R_{i\_}, R_{j\_})}{D_{set}}$. The resulting kernel is a valid kernel since $\frac{1}{1+D_{set}} > 0$ ($\frac{1}{D_{set}} > 0$) .

An alternative kernel is derived by the direct application of the *R-Convolution* kernel as described in [3]. The main idea in the R-Convolution kernel is that composite objects consist of simpler parts that are connected via a relation $\Re$. Kernels on the composite objects can be computed by combining kernels defined on their constituent parts. Let $x \in X$ be a composite object and $\boldsymbol{x} = x_1, ..., x_D \in X_1 \times ... \times X_D$ its constituent parts. Then we can represent the relation $\boldsymbol{x}$ *are the parts of* x by the relation $\Re$ on the set $X_1 \times X_2 \times ... \times X_D \times X$ where $\Re(\boldsymbol{x}, x)$ is true iff $\boldsymbol{x}$ are the parts of x. Let $\Re^{-1}(x) = \{\boldsymbol{x} : \Re(\boldsymbol{x}, x)\}$, a composite object can have more than one decomposing possibilities. Then the R-Convolution kernel is defined as:

$$K_{conv}(x, y) = \sum_{\boldsymbol{x} \in \Re^{-1}(x), \boldsymbol{y} \in R^{-1}(y)} \prod_{d=1}^{D} K_d(x_d, y_d) \tag{3}$$

Since we defined only one way to decompose a relational instance $R_{i\_}$ the sum in the equation 3 vanishes and we obtain the product of kernels defined over attributes of type instance-set and the kernels defined on standard attributes (only if standard attributes are present). In case $|\{S_{\_j}\}| \neq 0$ the resulting R-Convolution kernel is defined as

$$K_{\Re}(R_{i\_}, R_{j\_}) = k_s(R_{i\boldsymbol{s}}, R_{j\boldsymbol{s}}) \prod_{l=1}^{D_{set}} K_{set}(R_{iset_l}, R_{jset_l}) \tag{4}$$

otherwise, i.e. the relation does not have standard attributes, we obtain: $K_{\Re}(R_{i\_}, R_{j\_}) = \prod_{l=1}^{D_{set}} K_{set}(R_{iset_l}, R_{jset_l})$. Again it is obvious that the value of $K_{\Re}(.,.)$ is affected by the number of attributes that are of type instance-set since it contains a product of kernels defined on these attributes. In order to factor out the effect of the varying number of attributes across different relations we opted for:

$$K_{\Re}(R_{i\_}, R_{j\_}) = \frac{K_{\Re}(R_{i\_}, R_{j\_})}{\sqrt{K_{\Re}(R_{i\_}, R_{i\_}) K_{\Re}(R_{j\_}, R_{j\_})}} \tag{5}$$

It is worth noting that this kernel can be also derived by exploiting the fact that kernels are closed under *tensor product*, [6], i.e. if $K_1(x, y)$ is a kernel on $X \times X$ and $K_2(u, v)$ is a kernel on $U \times U$ then $K_1 \otimes K_2((x, u), (y, v)) = K_1(x, y)K_2(u, v)$ is a valid kernel on $(X \times U) \times (X \times U)$.

These two kernels, $K_{\Re}(.,.), K_{\Sigma}(.,.)$, are the ones with which we are going to experiment and on which we are going to base our distance computations. Having a kernel it is straightforward to compute the distance in the feature space Φ in which the kernel computes the inner product as $d(\phi(x), \phi(y)) = \sqrt{k(x, x) - 2k(x, y) + k(y, y)}$. This is the final distance that we will be using to perform classification.

Kernels on Sets To complete the definition of the kernel on the relational structure we defined here a kernel over sets of instances. This kernel can easily be derived by the definition of R-Convolution kernels by letting $\Re$ in the equation 3 be $\boldsymbol{x} \in \Re^{-1}(x) \Leftrightarrow \boldsymbol{x} \in x$. Consequently we obtain:

$$K_{set}(X, Y) = \sum_{\boldsymbol{x} \in X, \boldsymbol{y} \in Y} K_{\Sigma|\Re}(\boldsymbol{x}, \boldsymbol{y}) \tag{6}$$

where $K_{\Sigma|\Re}(.,.)$ is either $K_{\Sigma}(.,.)$ or $K_{\Re}(.,.)$. The kernel from equation 3 is sometimes refereed as the *cross product kernel*. The computation of the final kernel is based on recursive alternating applications of $K_{\Sigma}(.,.)$ or $K_{\Re}(.,.)$ and $K_{set}(.,.)$.

The procedure of computing the kernel on the variables of type instance-set indicates that if the cardinalities of the sets vary considerably, sets with larger cardinality will dominate the solution. This leads us to the issue of normalization of the sum on the right side of the equation 6, so that we obtain:

$$K_{norm}(X, Y) = \frac{K_{set}(X, Y)}{f_{norm}(X) f_{norm}(Y)} \tag{7}$$

where $f_{norm}(x)$ is a normalization function which is nonnegative and takes non-zero values. Different choices of $f_{norm}(x)$ give rise to different normalization methods, [7]. By putting $f_{norm}(X) = |X|$ we obtain the *Averaging* normalization method ($k_{\Sigma_A}(.,.)$). The obtained function is a kernel since the explicit representation of the feature space can be constructed (equation 8). Defining $f_{norm}(X) = \sqrt{k_{set}(X,X)}$ we get the *Normalization in the feature space* ($k_{\Sigma_{FS}}(.,.)$). Again the obtained function is a valid kernel (the explicit representation of the feature space is given in equation 9).

Relational kernels as kernels over trees We already mentioned in the section 2 that a relational instance can be considered as a tree-like structure where each node is one tuple from one of the relations that are part of the description of the relational instance and connections between nodes are determined by the foreign key associations defined within the relational schema. This makes the relational kernel a kernel over trees. The input trees are *typed* which results in the definition of a graded similarity. The similarity of nodes of different type, i.e. nodes coming from different relations, is zero. The similarity of nodes of the same type is determined on the basis of the attributes found on the relation associated with the given type. At the same time input trees are *unordered* which means that the order of comparison of the descendants is not important, the only constraint being that only descendants of the same type can be compared. In other words the subtree comparison is meaningful only between subtrees that are rooted on nodes that come from the same relation.

Feature space induced by relational kernels In order to get a new insight into the behavior of the relational kernels defined so far we will specify the feature space associated with them. We start with the definition of the feature space induced by the kernel over sets from equation 6. Lets assume $\Phi_{\Sigma|\Re}$ (i.e. Φ_{Σ} or $\Phi_{\Re}$) is an embedding function into a feature space $F_{\Sigma|\Re}$ (F_{Σ} or $F_{\Re}$) for the kernel $K_{\Sigma|\Re}$ (K_{Σ} or $K_{\Re}$) from equation 6 so that $K_{\Sigma|\Re}(\boldsymbol{x},\boldsymbol{y}) =< \Phi_{\Sigma|\Re}(\boldsymbol{x}), \Phi_{\Sigma|\Re}(\boldsymbol{y}) >$. It is easy to show that the feature space induced by this kernel is given by:

$$\Phi_{set}(X) = \sum_{x \in X} \Phi_{\Sigma|\Re}(x) \in F_{\Sigma|\Re}$$

Similarly the feature space induced by kernel from equation 7 where $f_{norm}(X) = |X|$ is given by

$$\Phi_{set}(X) = \frac{\sum_{x \in X} \Phi_{\Sigma|\Re}(x)}{|X|}. \tag{8}$$

It is clear that this normalization method amounts to computing the inner product, in the feature space induced by the elementary kernels, between the two centroids of the corresponding sets. In case $f_{norm}(X) = \sqrt{K_{set}(X,X)}$ the feature space is given by

$$\Phi_{set}(X) = \frac{\sum_{x \in X} \Phi_{\Sigma|\Re}(x)}{\| \sum_{x \in X} \Phi_{\Sigma|\Re}(x) \|}. \tag{9}$$

So this normalization method computes the cosine of the angle between the two normalized resultants of the vectors of the two sets.

Now we define the feature space associated with the direct sum (K_Σ) and the R-Convolution ($K_\Re$) kernels. Lets assume that Φ_Σ ($\Phi_\Re$) is an embedding function into a feature space F_Σ ($F_\Re$) for kernel K_Σ ($K_\Re$). Let also $\phi_1, \phi_{set_1}, \ldots, \phi_{set_{|D_{set}|}}$ be embedding functions into feature spaces $F_s, F_{set_1}, \ldots, F_{set_{|D_{set}|}}$ of the kernels $k_s, k_{set_1}, \ldots, k_{set_{|D_{set}|}}$ which constitute the K_Σ and $K_\Re$ kernels, respectively. It is easy to show that $F_\Sigma = F_s \oplus F_{set_1} \oplus \cdots \oplus F_{set_{|D_{set}|}}$ and $F_\Re = F_s \otimes F_{set_1} \otimes \cdots \otimes F_{set_{|D_{set}|}}$ where $\oplus$ denotes the direct sum and $\otimes$ denotes the tensor product of vector spaces. In other words the $F_\Re$ is constructed by computing all the possible products of all the dimensions of its constituent spaces, where each product becomes a new dimension of $F_\Re$. In contrast the F_Σ is constructed by a simple concatenation of the dimensions of its constituent spaces. It is obvious that $dim(F_\Sigma) = dim(F_s) + \sum_{i=1}^{|D_{set}|} dim(F_{set_i})$ and $dim(F_\Re) = dim(F_s) \prod_{i=1}^{|D_{set}|} dim(F_{set_i})$. In order to get an explicit feature space representation induced by the relational kernel one has to recursively combine the feature spaces induced by the kernel on sets and the direct sum or the R-Convolution kernels.

An important result is that the dimensionality of the feature space of the R-Convolution kernel is much higher than that of the direct sum kernel (this result holds if the elementary kernels, thus also the kernel over sets, induce a feature space of finite dimensionality, otherwise they are both of infinite dimension). This means that instance based learning in the feature space induced by the R-Convolution kernel should be more difficult than in this induced by the direct sum kernel. On the other hand the R-Convolution kernel is more expressive since it accounts for feature interactions by means of the products.

Time complexity Here we analyze the time complexity of the relational kernel defined above. Let $TrI = \{TrI_1, TrI_2, \ldots, TrI_n\}$ be a set of tree representations of the relational instances in a given relational schema. Let also TrR be a tree representation (with the depth d) of the relational schema at the "relation" level where each node is a relation and the connections between the nodes are again determined by the foreign key associations. In case there are loops in the relational schema the depth is limited to an arbitrary value so that a valid tree is constructed. It is worth noting that depths of each tree in TrI are at most as big as d. Having defined TrI and TrR let BF_I be the maximal out-degree of all nodes in all trees in the set TrI while BF_R be the maximal out-degree of all nodes in the TrR. BF_I can be considered as the maximum cardinality over all the sets which are part of a description of all possible relational instances in the relational schema. On the other hand BF_R is the maximum number of links we can follow from any relation in the relational schema. The computation of the relational kernel between two tree representation of relational instances is proportional to $O((BF_I{}^2)^{d-1}) = O(BF_I{}^{2(d-1)})$ (this because the computation of the kernel on sets is proportional to $O(BF_I{}^2)$ and at level d there are $BF_I{}^d$ sets to compare). Here we assume that the root of a tree is at level 1. The overall time complexity is proportional to $O(BF_R{}^{d-1} BF_I{}^{2(d-1)})$. Of course this is the pessimistic estimate of the time complexity and more useful characterization would be acquired if the average branching factors were used. The complexity is dominated by BF_I since $BF_R << BF_I$.

We will compare the computational complexity of the cross product kernel with that of the inner product computed directly in the feature space. Lets assume that the elementary kernel is a polynomial kernel with the exponent p (without the bias towards lower order monomial) and input space is $\Re^N$. The computation of the cross product kernel between two finite sets A and B is proportional to $O(|A||B|(N+p))$ (one has to compute $|A||B|$ elementary kernels and each of them can be computed in time proportional to $O(N+p)$) whereas computing directly the inner product in the feature space induced by this kernel is proportional to $O(2\binom{p+N-1}{p}(|A|+|B|))$ (each point has to be mapped to $\binom{p+N-1}{p}$-dimensional feature space $(|A|+|B|)$ times and the computation of the inner product in the feature space is again proportional to $\binom{p+N-1}{p}$). For example if we put $N=10$, $p=2$ and $|A|=|B|=100$ then the computation of the cross product kernel requires approximately five times more operations than the inner product in the feature space, i.e. in some cases it is better to explicitly map the instances to the feature space and compute the inner product.

4 Experiments

We will compare the selected kernel-based distance measures on a number of relational problems: musk - version 1, diterpenes and mutagenesis. In the diterpene dataset [8] the goal is to identify the type of diterpenoid compound skeletons given their ^{13}*C-NMR*-Spectrum. The musk dataset was described in [9]; here the goal is to predict the strength of synthetic musk molecules. We worked with version 1 of the dataset which contains 47 musk molecules and 45 similar non-musk molecules. The Mutagenesis dataset was introduced in [10]. The application task is the prediction of mutagenicity of a set of 230 aromatic and heteroaromatic nitro-compounds. We worked with the "regression friendly" version of the dataset. We defined two different versions of the learning problem. In *version 1* the examined compounds (in the *main* relation) consist of atoms (in the *atom* relation) which constitute bonds (in the *bound* relation). The recursion depth was limited to four. In *version 2* the compounds consist of bonds while bonds consists of atoms and the recursion level was limited to three. Bonds are described by two links to specific entries in the *atom* relation and by the type of the bond while atoms are described by their charge (numeric values), type and name (e.g. *N, F, S, O*, etc.). In both versions the recursion depth was limited because of the time complexity of the algorithm. All the results are given in table 1.

For diterpenes and musk datasets the computation of relational kernel can be simply reduced to computing kernels on sets of vectors requiring thus no recursion. In these cases the $K_\Sigma(.,.)$ and $K_\Re(.,.)$ relational kernels are equivalent (up to a normalization term) so we report results only for the former. In the mutagenicity problem it will be possible to move beyond a single level comparison of the instances and have many levels of recursion. We report results for different set normalization schemes; the subscript $_A$ will denote averaging and the subscript $_{FS}$ feature space normalization. In all experiments we limited ourselves to normalized polynomial $k_{P_{p,a}}(.,.)$ and Gaussian RBF $k_{G_\gamma}(.,.)$ elementary kernels. Here we give results for $p=2,3, a=1$ ($k_{P_{p,a}}(.,.)$) and for $\gamma = 0.01, 0.001$ ($k_{G_\gamma}(.,.)$), however more experiments with different parameter setting were performed.

In the experiments we want to explore the effect of different elementary kernels, the effect of different kernel set normalizations, as well as the relative performance of the $K_{\Sigma}(.,.)$ and $K_{\Re}(.,.)$ kernels. We will experiment with one number of nearest neighbors $K = 1$ (again more experiments with different K were performed).

We estimate accuracy using ten-fold cross-validation and control for the statistical significance of observed differences using McNemar's test (sig. level=0.05). We also establish a ranking schema of the different kernel-based distance measures, based on their relative performance as determined by the results of the significance tests, as follows: in a given dataset if kernel-based 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.

Table 1. Accuracy and rank results on the benchmark datasets

Elementary kernel	DITERPENES	MUSK (VERSION 1)	MUTAGENESIS (VERSION 1)	MUTAGENESIS (VERSION 2)
Relational kernel	K_{Σ_A}		K_{Σ_A}	
$k_{P_{p=2,a=1}}$	91.22 (5.5)	85.87 (3.5)	78.72 (3.5)	82.45 (3.5)
$k_{P_{p=3,a=1}}$	91.75 (6)	88.04 (4.5)	79.79 (3.5)	82.98 (3.5)
$k_{G_{\gamma=0.01}}$	86.69 (2.5)	83.70 (3.5)	81.38 (3.5)	83.51 (3.5)
$k_{G_{\gamma=0.001}}$	83.30 (0.5)	81.52 (2.5)	80.32 (3.5)	84.04 (3.5)
Relational kernel	$K_{\Sigma_{FS}}$		$K_{\Re_A}$	
$k_{P_{p=2,a=1}}$	90.82 (4.5)	85.87 (3.5)	79.79 (3.5)	82.98 (3.5)
$k_{P_{p=3,a=1}}$	91.68 (6)	88.04 (4.5)	79.79 (3.5)	81.38 (3.5)
$k_{G_{\gamma=0.01}}$	86.76 (2.5)	83.70 (3.5)	81.38 (3.5)	82.45 (3.5)
$k_{G_{\gamma=0.001}}$	83.03 (0.5)	81.52 (2.5)	80.85 (3.5)	80.32 (3.5)
Default Accuracy	29.81	51.09	66.49	

5 Results

To compare the different elementary kernels we fix a dataset and average the ranks of k_P and k_G, ignoring their parameter settings.There is an advantage of the polynomial over the Gaussian RBF elementary kernel for musk 1 and diterpenes datasets. For musk 1 the average rank of polynomial kernels is 4 (for Gaussian RBF 3) while for diterpenes is 5.5 (1.5). For both formulations of mutagenesis the average rank of polynomial kernels is 3.5 (3.5).

We performed more experiments than those listed in table 1, where we systematically varied the parameters of the elementary kernels ($p = 1, 2, 3, 4, a = 0, 1$ for polynomial and $\gamma = 0.1, 0.001, 0.0001, 0.00001$ for Gaussian RBF). We also varied the number of number of nearest neighbours $K = 1, 3, 10$. This did not seem to have a significant effect on predictive performance, which was quite stable among different values of the parameters, providing possible indication that the relational kernel is not very sensitive to the parameter settings of the elementary kernels and thus no extensive search is required over the parameters space. The stability among different parameters of the elementary kernels might be a result of the fact that the normalization (in equations 2, 5, 7) plays an important role in the construction of our relational kernel.

This normalization can factor out the effect of possible outliers, different cardinalities of sets which finally amounts to the above mentioned stability. In addition, for mutagenesis dataset, the stability could be an indication that the structural properties of the relational instances are more important for the classification than the properties of instances which constitute nodes.

The different normalization methods for kernels over sets also do not appear to have an influence on the final results. For diterpenes *Averaging* had an average rank of 3.625 over the different elementary kernels and *Feature space normalization* an average rank of 3.375. For musk 1 the corresponding figures were 3.5 and 3.5. One explanation for this might be that the two denominators in equation 8 and 9 are correlated, which makes sense since sets of higher cardinality will have probably a higher $\| \sum_{x \in X} \Phi_{\Sigma|\Re}(x) \|$, at least for the datasets we examined. However, this depends on the problem at hand, there could other datasets where this correlation does not hold.

The final dimension of comparison is the relative performance of $K_{\Sigma}(.,.)$ and $K_{\Re}(.,.)$. Here again it did not have a big influence on the final results: for both formulations of the mutagenesis problem $K_{\Sigma}(.,.)$ and $K_{\Re}(.,.)$ had an average rank of 3.5. This is a rather surprising fact since as we have shown before the feature space induced by R-Convolution kernel is of much higher dimensionality than the feature space induced by the direct sum kernel. This means that the instance based learning with the R-Convolution kernel should be harder than the one using direct sum kernel. On the other hand the R-Convolution kernel appears to be more expressive than the direct sum kernel since it accounts for feature interaction. The trade-off between hardness of learning in space of higher dimensionality and the higher expressiveness might explain similar performance of the R-Convolution and the direct sum kernel.

To situate the performance of our relational learner to other relational learning systems we give the best results reported in the literature on the same benchmark datasets. All the results denote the accuracy and all have been estimated with ten fold cross-validation. The best result for the musk 1 dataset is 92.40 % (IAPR algorihtm) and it was reported in [9]. In comparison our best kernel gave 88.04 % of accuracy. For the diterpenes dataset the best accuracy was achieved using the *DeS* algorithm which comes from [4]. The authors got 97.10 % whereas our best kernel gave 91.82 % of accuracy. For the mutagenesis 2 dataset we obtained 83.51 % of accuracy while the best result from the literature was 81.00 % and was taken from [11] on the B_2 formulation of the problem that corresponds to our version 2 of mutagenesis. From the results reported above we can see that our kernel-based learner compares favorably with the results achieved by special-purpose algorithms applied to structured data. On the other hand our relational kernel gave competitive results with standard elementary kernels which indicates that there is space for improvement if more elaborate elementary kernels better suited for the problem at hand are used.

6 Related work

The most relevant kernel in our context is the R-Convolution kernel which was mentioned in section 3. To our best knowledge our kernel is the first time the original R-

Convolution kernel, [3], was applied to the type of relational structures we considered here.

[4] proposed a framework that allows the application of kernel methods to different types of structured data e.g. sets, trees, graphs, lists. The representation formalism used was that of typed λ-calculus. The representation framework allows for the modeling of arbitrary complex objects which however is not at all a trivial task. Under this framework the authors explicitly defined kernels on sets and multisets. In [4] elementary (atomic) kernels are defined for each attribute separately, while our kernel assumes elementary kernels defined on the level of relations thus treating relations as indivisible objects. In [4] a kernel over tuples of objects is always defined as a direct sum of its constituent parts whereas in our framework one is able to use either the direct sum kernel or the R-Convolution, which have different representational powers. Finally in [4] only the cross product kernel can be used to define kernels on sets whereas under our framework any valid kernel on sets can be used.

The kernels described in [5] and in [12] can be considered as specialized R-Convolution kernels where instances are considered to be labeled ordered directed trees. The idea of these kernels is based on the notion of a number of common subtrees in a tree i.e. the kernel function is the inner product in the space which describes the number of occurrences of all possible subtrees. The main difference between [5] and [12] is that the former is applicable only to trees where no node shares its label with any of its siblings. [12] overcomes this limitation by defining the substructures of a tree as a tree such that there is a descendants order preserving mapping from vertices in the substructure to vertices in the tree. There are many differences between our kernel and kernels defined in [12]. First the trees considered in [12] are labeled trees, i.e. each node is characterized by a discrete label so two nodes are either the same or different, there is no graded similarity. In our case however nodes are not labeled but typed which results in the definition of a graded similarity. Second the trees in [12] are ordered whereas in our case there is no order restriction, the only restriction imposed is that comparison is performed only between subtrees rooted at nodes of the same type, i.e. same relation, and only descendants of the same type can be compared. The difference in time complexity between our kernel and the kernels of [5, 12] comes mainly from the fact that the input trees for our kernel are unordered whereas their kernels operate on ordered trees, thus increasing the number of possible comparisons.

7 Discussion and Future Work

Bringing together kernel methods and structured data is an important direction for practical machine learning research. In this paper we proposed a kernel based relational instance based learner which, contrary to most of the previous relational approaches that rely on different forms of typed logic, builds on notions from relational algebra. 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 [13] '... 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 in that direction.

Our kernel functions can be considered as instances of the R-Convolution kernel in the sense that we define a kernel on a composite object by means of kernel on the parts of objects. On the other hand our kernels could be also seen as being defined over typed and unordered trees. Since in other areas of computational biology many problems can be described using similar structures we believe that our kernel could also useful there.

Central to the whole approach was the definition of appropriate kernels on the new type of attributes i.e. the instance-set type. We believe that there is still a lot to be gained in classification performance if more refined kernels are used for this type of attributes. We have followed a rather simple approach where the kernel between two sets was simply the sum of all the pairwise kernels defined over all the pairs of elements of the two sets. A more elaborate approach would take into account only the kernels computed over specific pairs of elements based on some mapping relation of one set to the other defined on the feature space. That mapping relation can be based on the notions of distance computation between sets given in [14].

References

1. Dzeroski, S., Lavrac, N.: Relational Data Mining. Springer-Verlag New York, Inc. (2001)
2. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explor. Newsl. **5** (2003) 49–58
3. Haussler, D.: Convolution kernels on discrete structures. Technical report, UC Santa Cruz (1999)
4. Gaertner, T., Lloyd, J., Flach, P.: Kernels and distances for structured data. Machine Learning (2004)
5. Collins, M., Duffy, N.: Convolution kernels for natural language. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: Advances in Neural Information Processing Systems 14, Cambridge, MA, MIT Press (2002)
6. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA (2002)
7. Gaertner, T., Flach, P., Kowalczyk, A., Smola, A.: Multi-instance kernels. In Sammut, C., ed.: ICML02, Morgan Kaufmann (2002)
8. Dzeroski, S., Schulze-Kremer, S., Heidtke, K.R., Siems, K., Wettschereck, D.: Applying ILP to diterpene structure elucidation from ^{13}c NMR spectra. In: Inductive Logic Programming Workshop. (1996) 41–54
9. Dietterich, T.G., Lathrop, R.H., Lozano-Perez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence **89** (1997) 31–71
10. Srinivasan, A., Muggleton, S., King, R., Sternberg, M.: Mutagenesis: ILP experiments in a non-determinate biological domain. In Wrobel, S., ed.: Proceedings of the 4th International Workshop on Inductive Logic Programming. Volume 237. (1994) 217–232
11. Bloedorn, E., Michalski, R.: Data driven constructive induction. IEEE Intelligent Systems **13** (1998) 30–37
12. Kashima, H., Koyanagi, T.: Kernels for semi-structured data. In: ICML 2002. (2002)
13. Domingos, P.: Prospects and challenges of multi-relational data mining. SIGKDD, Explorations **5** (2003) 80–83
14. Ramon, J., Bruynooghe, M.: A polynomial time computable metric between point sets. Acta Informatica **37** (2001) 765–780