Adaptive Distances on Sets of Vectors

Adam Woźnica and Alexandros Kalousis
University of Geneva, Computer Science Department
7 Route de Drize, Battelle batiment A
1227 Carouge, Switzerland
Email: {Adam.Woznica, Alexandros.Kalousis}@unige.ch

***Abstract*—Recently, there has been a growing interest in learning distances directly from training data. While the previous works focused mainly on adapting distance measures over vectorial data, it is a well-known fact that many real-world data could not be easily represented as fixed length tuples of constants. In this paper we address this limitation and propose a novel class of distance learning techniques for learning problems in which instances are set of vectors; examples of such problems include, among others, automatic image annotation and graph classification. We investigate the behavior of the adaptive set distances on a number of artificial and real-world problems and demonstrate that they improve over the standard set distances.**

***Keywords*-distance learning; adaptive distances; complex objects; sets; graphs;**

I. Introduction

The k-Nearest Neighbor (kNN) algorithm is successfully used to address classification problems and has proved its utility in many real-world applications [4]. Most common kNN classifiers represent training instances as vectors in the $\mathbb{R}^p$ space where the Euclidean metric is used to measure the dissimilarities between examples. This approach has the advantages of simplicity and generality, however, it has two main limitations. First, most of today's machine learning applications hardly fit within the typical propositional representation and in such cases more general representations (e.g. sets or graphs) should be used [23]. Second, the Euclidean metric implies that the input space is isotropic which is rarely valid in practical applications [29].

Since the Euclidean metric is not suitable to many real-world problems encountered in machine learning, many researchers have recently proposed various methods for adjusting parameters of a distance measure directly from the data, either in a fully supervised setting [9], [13], [14], [18], [29] or using side information [16], [34]. All these methods were developed for vectorial data and the distances are usually restricted to the Mahalanobis metric family. [1]

The above studies have established the general utility of distance metric learning for kNN classification when the training instances are represented as vectors. However, so far only few works have addressed the problem of adapting a distance over objects which are not represented in a vectorial form (e.g. sets or graphs). In this context, researchers investigated adaptive distances for restricted class of non numerical objects that can not be easily represented in a vectorial form; the relevant work has focused on learning the family of edit distances over strings and graphs, labeled with elements from a finite alphabet [2], [22], [25]. The other approach that falls into this category is based on kernelizing the existing methods which enables their use on complex data as long as a valid kernel function over the learning objects is defined. For example, the work from [27] extends the RCA algorithm and the algorithm presented in [18] can be seen as the extension of the Xing algorithm [34]. A related approach which could be used to adapt distances over complex objects is based on learning kernel combination, e.g. [19]. This technique is more general than metric learning since any valid kernel combination can be used to compute a pseudo metric in the feature space.

The main problem with the above kernel-based methods is that most of the valid kernels for complex objects, due to the requirement of positive semi-definiteness, are defined as cross product kernels between sets of objects' decompositions, i.e. they are based on averaging [30]. This might adversely affect the generalization of kNN since most of the decompositions, and hence attributes in the induced feature space, will be poorly correlated with the actual class variable [20]. An example of such application is *multiple-instance learning* where the task is to learn a concept given positive and negative sets of instances [1], [8], where a set is labeled negative if all the instances are negative, and is labeled positive if at least one of the instances is positive. For these applications the (adaptive) distances that are not limited to averaging are expected to outperform (adaptive) distances based on averaging [30].

In this work we propose a new class of methods that, in the context of kNN, directly adapts a distance over composite objects for a problem at hand. We will focus on applications in which learning examples are naturally represented in the form of sets of vectors of possibly different cardinalities, and the distances to be adapted are different distances on sets. Even though this representation seem to have a limited expressive power (i.e. sets can not be used to easily model more complex structures), many researchers

[1] The Mahalanobis metric between $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^p$, parametrized by a matrix $\boldsymbol{A} \in \mathbb{R}^{p \times p}$, is defined as $d_{\boldsymbol{A}}(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{A}^T \boldsymbol{A} (\boldsymbol{x}_i - \boldsymbol{x}_j)$.

have modeled general (labeled) graphs as decompositions of graphs into sets of sub-graphs of specific types, e.g. [33]. The set distances we consider in this work belong to the class of mapping based set distances that allow for flexible ways of mapping the sets' elements, in which it is not necessary that all elements are accounted in the mapping; this potentially gives rise to a more expressive class of adaptive distances over complex objects.

We exploit ideas developed previously for adapting a metric to a given task by learning it directly from vectorial data and show how these methods can be used in the context of sets of vectors. It should be stressed that in principle any (semi-)supervised metric learning method can be adapted to set distance learning, as long as in the objective function the access to data is only through a distance function. The learning phase of our method boils down to solving a possibly non-differentiable optimization problem. We also propose a smoothing technique of the non-differentiable cost function so that it allows for application of standard gradient-based method.

The paper is organized as follows. In Section II we formally define a class of mapping based set distance measures defined over sets of vectors. Next, in Section III we propose a framework for learning set distances. Experimental results on both artificial and real-world datasets are reported in Section IV. Related work is discussed in Section V. Finally, we conclude with Section VI where we address open issues.

II. Distances on Sets

We begin with a labeled set of n learning objects $\{(X_1, y_1), (X_2, y_2), \ldots, (X_n, y_n)\} = (\mathcal{X}, \mathcal{Y})$ where $X_i \in 2^{\mathbb{R}^p}$ are sets (possibly with different cardinalities) that consist of p-dimensional vectors. We assume that $y_i \in \{1, 2, \ldots, c\}$. Let also d be a distance metric defined over $\mathbb{R}^p$ and $F \subseteq X_i \times X_j$ a specific mapping of the elements of the one set to the elements of the other. F is often constrained to belong into a specific family of mappings $\mathcal{F}$, denoted as $F \in \mathcal{F}$; possible constraints are that every element is mapped to *exactly one* element, or to *at most one* element, etc. The general form of set distance measures D between X_i and X_j can be written as:

$$D(X_i, X_j) = \mathcal{A}\{d(F) \mid F \in \mathcal{F}\} \tag{1}$$

where $d(F)$ denotes a set of pairwise distances d applied over the pairs of elements defined by F and $\mathcal{A}$ is an aggregation function defined over the set of $d(F)$. Typical aggregation functions include the possibly nested functions $\min$, $\max$ and average.[2] Depending on the definition of $\mathcal{F}$ and $\mathcal{A}$, we can define within this framework several set distance measures [10], [24]. Simple and well-known examples include the *Single Linkage* (D^{SL}) and *Complete Linkage* (D^{CL}) set distance measures, where the mapping family $\mathcal{F}$ constraints F to map one element to exactly one element and $\mathcal{A}$ are $\min$ and $\max$, respectively. The *Average Linkage* (D^{AL}) set distance[3] is obtained by allowing everything to be mapped with everything (i.e. $F = X_i \times X_j$) and setting $\mathcal{A}$ to average. By defining $\mathcal{F}$ to allow mapping of each element to *all* the other elements, and representing $\mathcal{A}$ by a specific combinations of $\max$, $\min$ and average we obtain the *Sum of Minimum Distances* ($\mathrm{D}^{\mathrm{SMD}}$) and the *Hausdorff* metric (D^{H}), both discussed e.g. in [10]. The computational complexity of all the D^{SL}, D^{CL}, D^{AL}, $\mathrm{D}^{\mathrm{SMD}}$ and D^{H} set distances is $O(m^2)$, where m is the cardinality of sets. Finally, other more elaborate instantiations of D include e.g. Surjections and Matchings which are obtained by respectively limiting $\mathcal{F}$ to surjections and matchings, and setting $\mathcal{A}$ to $\min$ [24]; these set distances have computation complexity of $O(m^3)$ and were not considered in this study.

The reason why we focused on the above matching based set distances is that they are not based on averaging as only specific examples are taken into account. As a result, these set distances are suited to the set classification problems considered in Section IV where only specific instances are important for classification and the proportion of these important instances to the total number of instances in a set might be low. Finally, we note that it is hard in general to specify apriori which set distance is best suited for a given learning problem; we tackled this problem in [32] by adaptively combining a number of predefined set distances.

One potential problem with some of the above set distances is that they are not smooth; this is due to the $\max$ and $\min$ functions that could appear in the definitions of the functions $\mathcal{A}$ in (1). As we will see in the next section this could be problematic for the task of adapting these distances for a problem at hand. In this work we examine a simple modification of the functions $\mathcal{A}$ so that the resulting set distances become "well-behaved"; more precisely, we replace functions $\max$ and $\min$ with their smooth alternatives. More formally, for two subsets $A \subseteq X_i$ and $B \subseteq X_j$, let $\mathcal{D}$ be a subset of set of all pairwise distances, i.e. $\mathcal{D} = \{d(x_i, x_j) \mid (x_i, x_j) \in A \times B\}$. Then the $\min(\mathcal{D})$ and $\max(\mathcal{D})$ functions, that return respectively the minimum and maximum values of $\mathcal{D}$, can be approximated as:

$$\text{softmin}(\mathcal{D}) =$$

$$\sum_{(x_i,x_j)\in A\times B} \{d(x_i, x_j) * \frac{\exp(-d(x_i, x_j)/\gamma)}{\sum_{(x_k,x_l)\in A\times B} \exp(-d(x_k, x_l)/\gamma)}\}$$

and

$$\text{softmax}(\mathcal{D}) =$$

[2] We note that the most "inner" function in the definition of $\mathcal{A}$ is always applied over $2^{\mathbb{R}^p}$.

[3] It is important to note that D^{AL} is equivalent to computing the standard Euclidean distance over the mean (average) vectors computed for each set.

$$\sum_{(x_i,x_j)\in A\times B} \{d(x_i,x_j) * \frac{\exp(d(x_i,x_j)/\delta)}{\sum_{(x_k,x_l)\in A\times B}\exp(d(x_k,x_l)/\delta)}\}$$

where $\gamma, \delta \in \mathbb{R}^+$. Note that as $\gamma \to 0$ ($\delta \to 0$), the behavior of the normalized exponential in $\mathrm{softmin}(\mathcal{D})$ (and $\mathrm{softmax}(\mathcal{D})$) tends to assign 1 for the smallest (largest) distances, and 0 for all the other distances. In Section IV-B we will give details on how these parameters were selected in our experiments.

III. Learning Distances on Sets

As it is clear from the previous section the examined distances on sets assume a given family of mappings $\mathcal{F}$ between the elements of the sets together with an aggregation function $\mathcal{A}$, and use some optimization procedure (sometimes a trivial one) to compute the distance between sets. This assumption corresponds to a form of a learning bias, and more precisely to the selection of a representation bias, since it determines what, and how many, are the "features" of the sets. Ideally, both $\mathcal{F}$ and $\mathcal{A}$ should be chosen for a given problem in a manner that matches the specific characteristics of the problem; however, in [32] we have shown that this is difficult in practice. In this section we will present the main contribution of this paper and show how this representational bias can be relaxed by introducing more flexible set distances.

We start by noting that the core of the problem lies in representing both $\mathcal{F}$ and $\mathcal{A}$ in a structured way so that the search for "optimal" values of these two elements can be performed efficiently. However, this has turned out to be difficult since the possible parametrization of the problem is not structured, rendering the learning process difficult as we have to address issues such as orderings of the elements of the sets and varying cardinalities. Instead, the idea presented in this paper is based on the observation that for a given $\mathcal{F}$, the distance between any two sets is fully determined by the set of pairwise distances, which are computed using d. As a result, by changing d, or equivalently by changing the representation of the elements of the sets, we expect to acquire more adapted set distances as we can actually adjust the input arguments of $\mathcal{A}$ and hence influence the way the overall mapping-based set distance is computed. In what follows, we restrict d to be the Mahalanobis metric parametrized by a matrix $\boldsymbol{A}$ which we will denote by $d_{\boldsymbol{A}}$; in this setting, the problem of learning $d_{\boldsymbol{A}}$, and hence the mapping based set distances, is cast to the problem of finding a suitable matrix $\boldsymbol{A}$. The parametrization over $\boldsymbol{A}$ is completely structured, rendering the learning process efficient. However, this comes at the cost of reduced flexibility since we will have to be constrained within a fixed mapping family $\mathcal{F}$. As we will see in the experimental part, this indeed brings an improvement over standard set distances. A similar parametrization was explored in [7] and used to find optimal matchings between the vertices of two graphs. In the remainder of this work, we will denote the adaptive set distances by $D_{\boldsymbol{A}}$.

We view the problem of finding a suitable matrix $\boldsymbol{A}$ as an optimization problem where the three main constituents are the definition of a cost function, $\mathcal{F}_A$, a set distance, $D_{\boldsymbol{A}}$, and an optimization method. More precisely, we define the problem of set distance learning as:

$$\min_{\boldsymbol{A}} \mathcal{F}_{\boldsymbol{A}}((\mathcal{X},\mathcal{Y}), D_{\boldsymbol{A}}) \tag{2}$$

possibly subject to some constraints. Depending on the actual form of the function $\mathcal{F}_{\boldsymbol{A}}$ and the set distance $D_{\boldsymbol{A}}$ in (2), and an optimization method, different instantiations of the algorithm can be obtained. It should be mentioned that it is sometimes advantageous to change (2) so that the optimization is performed over $\boldsymbol{L} = \boldsymbol{A}^T\boldsymbol{A}$, however, this procedure was not examined in this study.

The characteristics of $\mathcal{F}_{\boldsymbol{A}}$ and $D_{\boldsymbol{A}}$ directly determine the choice of the appropriate optimization method. When the objective function in (2) is not differentiable (D^{SL}, D^{CL}, D^{H} and $\mathrm{D}^{\mathrm{SMD}}$) we use a standard sub-gradient descent algorithms to find its minimum [3]. In this procedure the current k-iterate estimate of a solution $\boldsymbol{A}^{(k)}$ is replaced by $\boldsymbol{A}^{(k)} - \alpha_k \boldsymbol{g}^{(k)}$ ($\alpha_k > 0$ is the k-th step size and $\boldsymbol{g}^{(k)}$ is *any* sub-gradient of the cost function at $\boldsymbol{A}^{(k)}$); since this is not a descent method, we keep track of the best solutions obtained so far. We experimented with different choices of α_k; in this work we report the results only for $\alpha_k = 1/||\boldsymbol{g}^{(k)}||^2$. The above sub-gradient method, although conceptually simple, might come at the price of harmed computational performance. Finally, all the differentiable optimization problems (i.e. for set distances that use the $\mathrm{softmax}$ or $\mathrm{softmin}$ functions, and D^{AL}) were solved using a gradient-based procedure, where the Polack-Ribiere flavor of conjugate gradients is used to compute search directions [3]; a line search using quadratic and cubic polynomial approximations and the Wolfe-Powell stopping criteria is used together with the slope ratio method for guessing initial step sizes [6].

One possible problem with the optimization task of (2) is that for full matrices $\boldsymbol{A}$ the number of parameters to estimate is p^2. This can be problematic when p is large compared to the number of instances in the training set, and can lead to overfitting. We explore two ways to overcome this problem. In the first one we add a penalization term to (2). We set this to $\lambda\|\boldsymbol{A}\|_{\mathcal{F}}^2$ where $\|\boldsymbol{A}\|_{\mathcal{F}}$ is the Frobenious norm of $\boldsymbol{A}$, and $\lambda > 0$ is a regularization parameter. Alternatively, we restrict $\boldsymbol{A}$ to be diagonal resulting in a weighted combination of features (this restriction is in fact a simple form of regularization since it reduces the effective number of parameters from p^2 to p). It should be noted that the regularization technique based on diagonal matrices, although faster than the one based on full matrices, is also less expressive since it does not account for interactions between different attributes. However, as we will see in the

experimental part, for the datasets we experimented with it achieves better performance than the method based on the full matrices.

A. Cost Function

In this subsection we define one specific instantiation of the above framework. More precisely, the cost function $\mathcal{F}_{\boldsymbol{A}}$ from (2) we will focus on is motivated by the cost function used in the Neighborhood Component Analysis (NCA) method [14] that was originally developed for metric learning over vectorial data. We should emphasize that in principle any metric learning method can be adapted for set distance learning, as long as in the objective function the access to data is only through a distance function.

The NCA method attempts to directly optimize a continuous version of the leave-one-out error of the kNN algorithm on the training data. Its cost function is based on stochastic neighbor assignments in the weighted feature space. These are based on a conditional distribution which for an example i selects another example j as its neighbor, with some probability $p_{\boldsymbol{A}}(j|i)$, and inherits its class label from the point it selects. In the context of set distance measures, we define the probability $p_{\boldsymbol{A}}(j|i)$ as the softmin of the $D_{\boldsymbol{A}}$ set distances:

$$p_{\boldsymbol{A}}(j|i) = \frac{\exp(-D_{\boldsymbol{A}}(X_i, X_j))}{\sum_{k \neq i} \exp(-D_{\boldsymbol{A}}(X_i, X_k))} \ , \ p_{\boldsymbol{A}}(i|i) = 0.$$

Under this stochastic selection rule the probability $p_{\boldsymbol{A}}(i)$ of correctly classifying X_i is given by

$$\sum_{j \in C_i} p_{\boldsymbol{A}}(j|i)$$

where $C_i = \{j | y_i = y_j\}$. The optimization problem is defined as:

$$\min_{\boldsymbol{A}} \mathcal{F}_{\boldsymbol{A}} = \sum_i \log(\sum_{j \in C_i} p_{\boldsymbol{A}}(j|i)). \tag{3}$$

The computational complexity of this method scales as $O(n^2 m^2 p)$.

The main advantage of NCA is that it makes no assumptions about the shape of the class conditional distributions, i.e. whether they are uni- or multi-modal [14]. However, its objective function is not convex and hence there is no guarantee that a (sub-)gradient based method will converge to a global optima. There are existing learning techniques, such as Large Margin Nearest Neighbor (LMNN) [29], that are convex for vectorial data; however, these are not anymore convex when the D^{SL}, $\mathrm{D}^{\mathrm{SMD}}$ and D^{H} set distances are used. This is a result of the fact that these set distances are based on the $\min$ function which does not preserve convexity[4]. We also experimented with the above algorithm (results are not reported in this paper); however, in terms of predictive performance there is an advantage of NCA. In Section IV-B1, we will discuss the issue of non-convexity of NCA in more details.

[4]On the other hand, LMNN is convex for D^{CL} and D^{AL}.

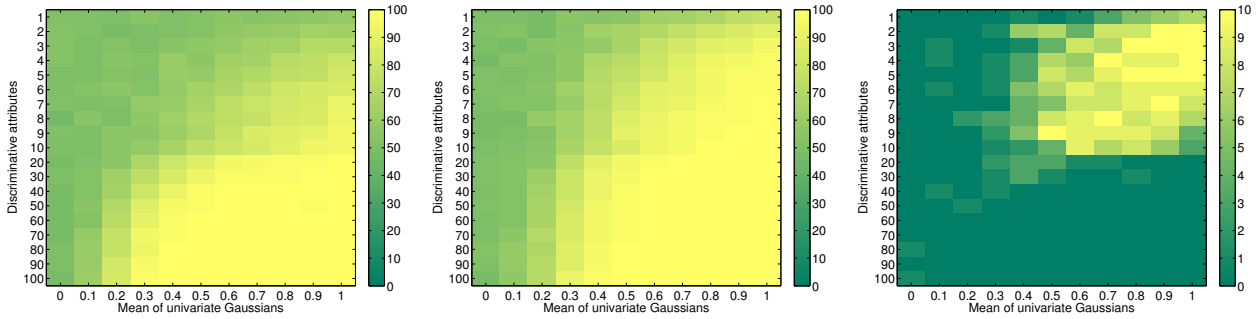
IV. Experiments

We evaluated the performance of the proposed approach on a number of artificial and real-world datasets where learning objects are represented as sets of vectors. The different instantiations of the set distance learning methods are compared in the context of kNN; the main goal is to examine whether we can increase the predictive performance of kNN by learning set distances. We experiment with two versions of the adaptive set distances, with full and diagonal matrices $\boldsymbol{A}$, which we will call $D^f_{\boldsymbol{A}}$ and $D^d_{\boldsymbol{A}}$, respectively. We report results for both non-differentiable distances and distances that were first smoothed by applying the $\mathrm{softmin}$ and/or $\mathrm{softmax}$ functions. In $D^f_{\boldsymbol{A}}$ we only report the results for $\lambda = 10$ (we comment on the influence of λ to the behavior of $D^f_{\boldsymbol{A}}$ in Section IV-B1). We experimented with different values of k ($k = 1, 3, 10$); as the relative performance of the instantiations of kNN did not vary with k, we report results only for $k = 1$. We estimate accuracy using 10-fold cross-validation and control for the statistical significance of observed differences using McNemar's test (sig. level of 0.05).

A. Artificial Datasets

The goal of these experiments is to investigate the impact of attribute relevance on the relative performance of standard and adaptive set distances, for artificially generated set classification problems. The artificial datasets are composed of 100 randomly generated sets; each of which has from 1 to 10 vectors, with 5 elements on average. Each set is randomly assigned a binary label. The sets' elements are 100-dimensional vectors where each dimension is sampled from a uni-variate Gaussian distribution (with 0 mean and a standard deviation of 1), except for the first l ($l = 1, \ldots, 100$) dimensions which are sampled from a uni-variate Gaussian distribution, with a standard deviation of 1 and with a mean of μ (for sets with a positive label) or $-\mu$ (for negative sets); we varied the μ parameter from 0 to 1, with a step of 0.1. The intuition behind this generative process is that attributes sampled from Gaussians with non-zero mean contain discriminatory information. The larger the magnitude of μ the more discriminatory is an attribute; moreover, the more discriminatory attributes a given dataset has, the easier is the corresponding classification problem.

In Figure 1 we graphically present the experimental results. In each plot the x-axis corresponds to the μ value of the uni-variate Gaussian distribution ($\mu = 0, 0.1, \ldots, 1$), and the y-axis corresponds to the number of attributes which are sampled from a Gaussian distribution with μ (or $-\mu$) mean. The cells of the two first plots represent the mean accuracies of the standard and the adaptive kNN algorithms averaged over the same randomly generated datasets (the lighter the



(a) standard set distances (b) adaptive set distances (c) significance tests

Figure 1. Experimental results on the artificial datasets (for D^{SL}).

Table I
SIMPLE STATISTICS ON THE EXAMINED REAL-WORLD DATASETS. n, m, p AND c DENOTE RESPECTIVELY THE NUMBER OF TRAINING INSTANCES, MEAN SET CARDINALITY, DIMENSIONALITY OF VECTORS AND THE NUMBER OF CLASSES.

Datasets	n	m	p	c
musk1	92	5.17	167	2
musk2	102	64.69	167	2
muta	188	27.89	101	2
fm	349	25.62	51	2
mm	336	25.40	51	2
tiger	200	6.1	230	2
fox	200	6.95	230	2
elephant	200	6.6	230	2

color the higher the accuracy). The cells of the third plot give the number of times that the accuracy of the adaptive kNN was significantly better than that of the standard kNN according to a McNemar test of statistical significance with significance level of 0.05; since for each configuration we repeat the experiments 10 times the maximum possible value of a cell is 10. In this experiment we report results for $D^d_{\boldsymbol{A}}$ adaptive distances ($D^f_{\boldsymbol{A}}$ has a similar behavior) and for D^{SL} (similar trend holds for the other set distances); we focus here only on the non-smooth set distances.

As expected, by looking at plots (a) and (b), we observe that in general the predictive accuracy depends on: (i) the value of μ (the higher in magnitude the value, the better) and (ii) the number of discriminatory attributes (the more discriminatory attributes, the better). Moreover, in plot (c) we can see that the biggest advantage of the set distance learning is in cases where the attributes that discriminate well are few and are drown within many irrelevant attributes (the values in the upper-right corner of plot (c) are close to 10). The above results clearly suggest that the set distance learning methods are effective for the artificial problems we examined.

B. Real-World Datasets

The empirical results reported here are from two main domains: chemoinformatics and automatic image annotation. In the first group we experimented with musk, mutagenesis and carcinogenicity datasets. The musk dataset [8] is a standard multi-instance benchmark dataset. We experimented with both version 1 and version 2 of this dataset. We also experimented with the "regression friendly" version of the Mutagenesis dataset [26], and represent each molecule as a set of bonds together with the two adjacent atoms. The next classification problem comes from the Predictive Toxicology Challenge and is defined over carcinogenicity properties of chemical compounds [15]. This dataset lists the bioassays of 417 chemical compounds for four type of rodents; the actual representation of training instances is the same as in mutagenesis. We present results only for the fm and mm versions of the problem. We transformed the original dataset (with eight classes) into a binary problem as in [20]. Finally, for automatic image annotation we experimented with three datasets where the goal is to detect specific kinds of animals in images. Images are represented as sets of segments (or blobs); each segment is characterized by color, texture and shape [1], [12]. We experimented with the tiger, elephant and fox datasets. All the above datasets were first preprocessed by (i) normalizing the numeric attributes, and (ii) converting nominal attributes into binary numeric attributes. Simple statistics on the above datasets are given in Table I.

Experiments on these datasets have 3 goals. First, we study the ability of the set distance learning techniques to improve the predictive performance of kNN (for non-smooth distances). Second, we analyze the issue of non-convexity (for non-smooth distances). Finally, we examine the impact of the smoothing of set distances to the predictive performance.

It is important to note that in this paper, except for state-of-the-art results, we do not to report results on kernel functions applied on structured data such as sets or graphs, even though the limited expressive power of such kernels

Table II
ACCURACY AND SIGNIFICANCE TEST RESULTS OF KNN (WITH STANDARD AND ADAPTIVE SET DISTANCES) IN THE BENCHMARK DATASETS. THE "+" SIGN NEXT TO PERFORMANCE OF NCA STANDS FOR A SIGNIFICANT WIN OF THIS METHOD V.S. STANDARD KNN, "-" FOR A SIGNIFICANT LOSS, AND A MISSING SIGN FOR NO SIGNIFICANT DIFFERENCE.

set distance	method	musk1	musk2	muta	fm	mm	tiger	fox	elephant
D^{AL}	D	76.4	71.0	57.6	58.6	63.0	72.0	52.0	76.0
	$D^f_{\boldsymbol{A}}$	75.9	71.5	58.2	50.1-	53.5-	74.0	56.5	79.5
	$D^d_{\boldsymbol{A}}$	78.0	76.5	68.4+	40.4-	48.1-	78.5+	59.5+	77.0
D^{SL}	D	81.1	77.2	42.2	41.6	39.3	76.0	56.5	71.5
	$D^f_{\boldsymbol{A}}$	78.4	69.5	41.6	41.6	39.3	73.5	57.5	72.5
	$D^d_{\boldsymbol{A}}$	82.4	68.5-	42.8	41.6	39.3	79.0	61.5+	72.5
D^{CL}	D	67.5	71.5	39.4	43.2	40.9	67.5	54.5	64.5
	$D^f_{\boldsymbol{A}}$	71.7	65.5	56.3+	47.3	46.6+	68.5	47.0	55.5
	$D^d_{\boldsymbol{A}}$	77.3+	79.5+	58.2+	52.1+	58.1+	72.5	51.5	74.0+
D^{H}	D	81.3	78.0	75.5	46.8	48.3	74.0	51.5	74.5
	$D^f_{\boldsymbol{A}}$	83.5	73.3	77.5	57.5+	57.2+	74.0	55.5	65.5
	$D^d_{\boldsymbol{A}}$	78.8	77.2	80.6+	57.2+	53.8+	76.5	65.5+	75.5
$\mathrm{D}^{\mathrm{SMD}}$	D	83.3	75.2	79.4	56.6	58.2	77.5	60.0	79.5
	$D^f_{\boldsymbol{A}}$	77.1	73.3	81.4+	58.5	58.3	76.0	59.0	77.5
	$D^d_{\boldsymbol{A}}$	81.3	73.3	80.5	56.6	58.7	85.0+	62.0	85.5+

was on the first place the reason why we decided to work directly with the more flexible set distances. This is motivated by the fact that kernels are most commonly used in the context of large margin classifiers like SVM and hence the results will not be directly comparable with that obtained from kNN (SVM and kNN have different inductive biases). We mention that kernels can be in principle converted to distances and then used in the context of kNN. However, the results on kernel based distances are not reported in this work since the set distance that are induced from the cross product kernel with a linear elementary kernel[5] have exactly the same semantics as D^{AL}; we also experimentally verified that they have very similar performance. The same argument also holds for other existing set kernels that are based on averaging.

1) Results and Analysis:

Adaptive Set Distance vs. Standard Set Distances.: The results (with the significance test results) of comparison between $D^f_{\boldsymbol{A}}$, $D^d_{\boldsymbol{A}}$ and standard set distances D are presented in Table II. From these results we conclude that in terms of predictive performance, $D^d_{\boldsymbol{A}}$ has an advantage over the standard set distances. Except for D^{AL} in the mm and fm datasets, $D^d_{\boldsymbol{A}}$ is never significantly worse and sometimes it is significantly better than standard set distances. Its biggest advantage is for D^{CL}, D^{H} and $\mathrm{D}^{\mathrm{SMD}}$, and in graph classification and automatic image annotation datasets. Overall it is significantly better (worse) that standard kNN in 16 (2) cases. On the other hand, $D^f_{\boldsymbol{A}}$ (with $\lambda = 10$) does not fare so well. To explain the worse performance of $D^f_{\boldsymbol{A}}$, we conducted experiments for different values of the regularization parameter λ (these results are not reported in this paper). The main observation was that in general λ had an impact on the classification performance; more precisely, we observed that by increasing λ we also increase the predictive performance. This suggests that $D^f_{\boldsymbol{A}}$ might be prone to overfitting, whereas by limiting the number of free parameters to p, as it is the case of $D^d_{\boldsymbol{A}}$, overtraining is avoided. $D^f_{\boldsymbol{A}}$ is significantly better (worse) than standard kNN in 5 (2) cases.

The next observation is that, with the exception of carcinogenicity datasets, the set distances that are not based on averaging outperform D^{AL} (although the appropriate mapping depends on the application and ideally should be guided by domain knowledge). As already noted in Section II (Footnote 3), D^{AL} is equivalent to computing the standard Euclidean distance over the mean vector computed for each set. As a result, we conclude that in order to learn distance over composite objects such as sets it is necessary to exploit the internal structure of these objects; by simply aggregating elements of the sets (e.g. by using the average function) the information that could be exploited by our set distance learning technique is simply lost. Finally, as discussed in Section IV-B D^{AL} has the same semantics as most of the existing set kernels that are based on averaging. Hence, the set distances other than D^{AL} are more appropriate for the examined problems than set kernel such as the cross product kernel.

To situate the performance of our method we provide in Table III the best results reported in the literature on the same benchmark datasets.[6] The best results for musk1, musk2, muta, fm and mm are taken from [30]. The results in tiger and fox are from [12]; the best result in elephant

[5]A non-linear kernel may further influence the results by accounting for feature interactions.

[6]For graph datasets we only cite works which use similar features to describe atoms and bonds. In particular, our results in fm and mm not directly comparable with the ones reported in [11].

Table III
COMPARISON WITH STATE-OF-THE ART RESULTS.

	musk1	musk2	muta	fm	mm	tiger	fox	elephant
Best results from literature	94.7	92.2	92.0	65.0	68.0	86.0	65.0	83.5
Best adaptive set distance	83.5	79.5	81.4	58.5	58.7	85.0	65.5	85.5

was reported in [5], [12]. It is important to note that all these state-of-the-art algorithms are in fact multiple-instance approaches, namely SVM-based multiple-instance approaches using a specialized kernel applied to either to set or graph based data. The values in the "Best adaptive set distance" row correspond to the best set adaptive distance for each dataset, selected over all the examined mappings. It is obvious that the latter results are optimistically biased since they are selected after extensive experimentation with various set distance measures. However, the same could be argued for the results of all the other related kernels given in Table III since in all the cases multiple results where available and we reported on the best. From the results is it clear that the performances of our adaptive distances in musk1, musk2, muta, fm and mm are lower than that of state-of-the-art methods for the respective datasets[7]; for automatic image annotation the obtained results are similar to the best results from the literature.

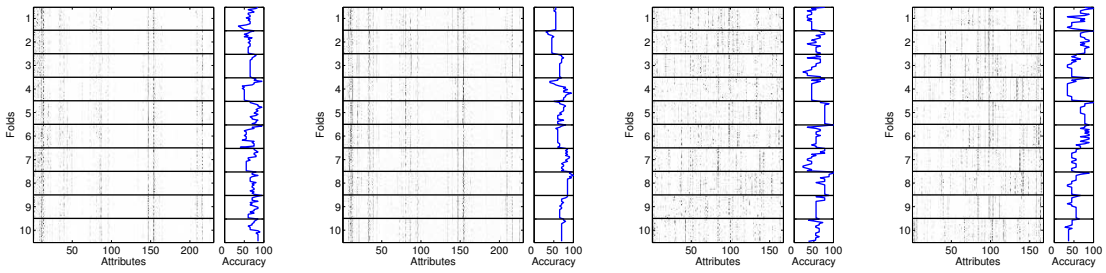
Finally, we note that the running times of learning the diagonal distance over full datasets varied from several seconds (in musk1) to approximately 10 minutes (FM) whereas the corresponding running times of learning the full distances varied from several minutes (in musk, muta, fm and mm) to approximately 1 hour (automatic image annotation); these running times are not reported in Table II. Based on these running times and the predictive performances we believe that the proposed methods based on diagonal matrices are better choice than the ones based on full matrices, and are worth their cost.

Non-convexity of the Optimization Problem.: As already mentioned, the optimization problem of NCA is not convex and hence some care should be taken to avoid local optima during training. To get additional insight into this problem, we visualized in Fig. 2 the results of the NCA optimization process in the tiger and musk1 datasets and for the D^{H} and D^{CL} non-smooth set distances. In each of the graphs, the x-axis corresponds to the individual features of sets' vectors. The y-axis is separated into ten rows, each one corresponding to one of the cross-validation folds. Within each row, we find 20 rows (not visibly separated) corresponding to 20 solutions of the optimization processes with randomly generated initial solutions $\boldsymbol{A}$. Additionally, for each model we report on the right side the accuracy computed for the current data split and the initial $\boldsymbol{A}$. By examining the variability *within* a given data split, we are able to get an insight into the "ruggedness" of the corresponding objective function $\mathcal{F}_{\boldsymbol{A}}$, and the impact of the solutions to the accuracy; a convex $\mathcal{F}_{\boldsymbol{A}}$ would result in only vertical lines (corresponding to individual features) within the different data splits. Additionally, by analysing the variability *between* different splits, we get an insight into the stability of the method with respect to the training data; a perfectly stable method, where $\mathcal{F}_{\boldsymbol{A}}$ is convex, would have only vertical lines everywhere.

In Fig. 2 we can see that indeed the set distance learning methods can produce different models, indicating that the corresponding cost function of NCA has many different local optimas. To quantify this diversity, we computed the average (normalized) Euclidean distances between all the pairs of weight vectors within each data split, averaged over the splits. Surprisingly, we observe that the models produced in a given dataset split are similar; for D^{H} the mean Euclidean distances are 0.055 in tiger and 0.076 in musk1, and for D^{CL} the corresponding values are 0.055 (in tiger) and 0.062 (in musk1). This suggests that even though NCA is a non-convex procedure that produces different models, these models are rather similar. Nevertheless, the produced models give rise to different predictive performances as measured by standard deviations of accuracies computed on each fold, averaged over the number of folds. According to this measure NCA produces more stable predictions in tiger (averaged standard deviation is 5.6 and 9.3 for D^{CL} and D^{H}, respectively) than in musk1 (10.0 and 10.7 for D^{CL} and D^{H}).

Soft Versions of the Set Distances.: Finally, we experimented with the soft versions of the standard set distances. As the soft set distances are based on the $\mathrm{softmin}$ and $\mathrm{softmax}$ functions defined in Section II, which approximate respectively the $\min$ and $\max$ functions, we first assessed how good are these approximations. More precisely, we examined the relative performance of the standard and soft set distances, together with their prediction agreement (measured as a fraction of times kNN produce the same prediction). The key problem in applying the smooth set distances is to select the values of the parameters γ and δ; the "correct" values of these parameters depend on the magnitude of the elements $d_{\boldsymbol{A}}(x_i, x_j)$ in $\mathcal{D}$. After some preliminary experiments, the γ and δ parameters were set to $p \cdot 0.1$ and $p \cdot 0.001$, where p is the length of vectors. The results only for D^{CL} and $\mathrm{D}^{\mathrm{SMD}}$ (due to the lack of space) are

[7]We hypothesize that in these datasets the poor performance is due to the non-appropriateness of the inductive bias of kNN for these datasets. The other possible explanation for the poor performance of set distances in musk is given in [28].



(b) tiger, D^{H} (d) tiger, D^{CL} (f) musk1, D^{H} (h) musk1, D^{CL}

Figure 2. Stability of $\mathrm{NCA}_{\mathrm{diag}}$ in tiger and musk1. The ploted weights in each of the row are the normalized (by the Frobenius norm) diagonals of $\boldsymbol{A}^T\boldsymbol{A}$.

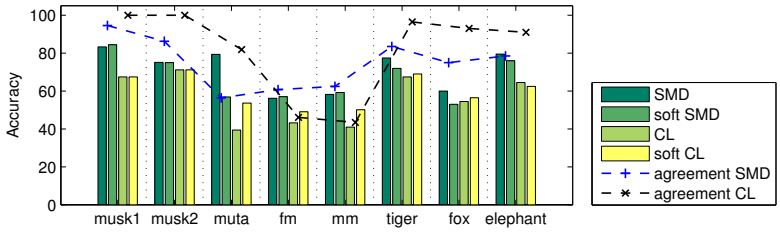


Figure 3. Accuracy of standard vs. soft set distances ($\mathrm{D}^{\mathrm{SMD}}$ and D^{CL}). The dashed lines show the agreement (in percentage) of predictions between these two versions of set distances.

visualized in Figure 3, where bars and dashed lines denote the performance of the two set distances and the agreement of predictions, respectively. From the plot we can see that even though the relative performances are very similar (with the exception of $\mathrm{D}^{\mathrm{SMD}}$ for mutagenesis which is the only case where the difference between performances is statistically significant), the agreement of predictions depends on the dataset, being the highest for musk and automatic image annotation datasets. For the graph classification datasets the two version of set distance produce different predictions, indicating that in this case our smooth approximation is questionable.

In Table IV we report the performances of the soft distances $D^f_{\boldsymbol{A}}$ (due to the lack of space only for D^{CL} and $\mathrm{D}^{\mathrm{SMD}}$). In all the cases these performances are similar to the ones of standard set distances (the performances are not statistically significant). Surprisingly, it also holds in the graph classification datasets, for which, as we saw in Figure 3, the soft and standard set distances produced very different predictions. After some analysis of the soft distances, we conjecture that the reason of why the smoothed distanced did not perform well is that it is hard to set the smoothing parameters that would be meaningful for all sets. More precisely, the magnitude of pairwise Mahalanobis distances between all elements of any two sets will be very different depending of the sets; for some sets softmin (softmax) will be able to "identify" the minimum (maximum) distances, whereas for other sets these functions will be "too smooth".

Table IV
ACCURACY OF TWO ADAPTIVE SOFT SET DISTANCES.

	musk1	musk2	muta	fm	mm	tiger	fox	elephant
$\mathrm{D}^{\mathrm{SMD}}$	78.8	76.5	80.6	58.6	58.1	83.0	63.0	84.5
D^{CL}	77.3	77.2	58.2	52.1	58.1	71.0	52.5	75.5

To summarize, as the soft set distances give rise to a much more complicated algorithm (i.e. the gradient gets more complicated to derive) and it is difficult to select "good" values of parameters γ and δ, it is clear that it is more advantageous to work directly with non-differentiable optimization problems.

V. Related Work

The most relevant work which was independently developed by R. Jin et al. [17] focuses on the problem of learning a distance metric from multiple-instance multiple-label data where each example is represented by a set of vectors and is labeled by multiple labels. The cost function used in [17] combine the idea of *Rayleigh ratio* and clustering to minimize the distance between each set and its assigned classes and maximize the distance between classes (each class is represented as a collection of its potential centroids). The authors exploit D^{SL} to measure the distance between pairs of sets. We mention that the setup where each set can be assigned to multiple classes simultaneously is more challenging than the one considered here. However, to render the optimization task more tractable the authors

employed a rather simple cost function (the LDA algorithm that is also based on the Rayleigh ratio, was reported a poor performance in comparison with more involved metric learning techniques [14]) and limited themselves to D^{SL} that, as we saw in Section IV-B, was generally outperformed by other set distance measures. Notwithstanding the above simplifications, to solve the optimization problem the authors proposed a rather involved procedure that is based on alternating optimization over several groups of variables where each step requires solving a difficult and non-convex optimization problem.

Another line of relevant research focuses on learning (stochastic) edit distances between strings [22], [25], trees [2] and graphs [21].[8] More precisely, most of the approaches that fall into this category estimate the costs of basic operations between elements of a finite alphabet (including an empty symbol), and consider probabilities that one object is transformed into another (the class of such transformations can be seen as a restricted mapping family $\mathcal{F}$). Most of these methods aim to optimize likelihood of predefined pairs of objects that should be similar under the new edit distance; variants of the EM algorithm are subsequently used to find (locally) optimal elementary costs. The main difference between these approaches and the method proposed in this paper is that the former algorithms assume that the elements of strings (or nodes in trees and graphs) come from a finite alphabet, and hence the number of parameters to estimate is finite. On the other hand, we assume that the set elements are vectors in an Euclidean space which makes it more difficult to represent the learning problem in a structured way that is amenable to efficient optimization techniques. As a result, we transform the difficult problem of directly learning the mapping $\mathcal{F}$ (and an aggregation function $\mathcal{A}$), to the easier problem of learning the representation of elements of the sets.

In [32] we developed a framework for adaptively *combining* (complex) distance measures. While both the present work and [32] deal with learning distances for complex objects, their formulations and expressiveness are different. In [32] we learn optimal combinations of different distances applied on composite objects, e.g. combinations of different set distances; there the learned parameters correspond to the importance of different distances. In the present work we stay within a specific type of set distance each time and optimize it by learning the way the elements of the sets are matched. We achieve this by weighting the attributes of the vectors that constitute sets. The difference in expressiveness between these two approaches could be easily demonstrated on the artificial datasets from Section IV-A; on these problems our approach from [32] will fail, since it will not be able to down-weight the irrelevant attributes. With this respect the approach presented in this paper significantly differs from [32] as it allows to tackle the set classification problems where the attributes of vectors that constitute sets are not equally important.

Finally, in [31] we proposed a class of kernels over sets which directly exploit the set distance measures such that the computation is based only on specific pairs of elements. This allows for incorporating various semantics into set kernels and lending the power of regularization to learning in structural domains where natural distance functions exist. These kernels we used in the context of SVM. The main difference from [31] and the work reported in this paper is that in the former study we did not learn the importance of individual attributes of the vectors. Other works that apply kernels or distances for set classification problems include e.g. [28].

[8]Edit distance between two objects is defined as minimum cost insertion, deletion and substitution operations required to transform one object into another.

VI. Conclusions and Future Work

In this paper we presented a framework that, in the context of kNN, allows to learn distances over complex objects. We focused on applications in which learning examples are represented as sets of vectors, and the distances to be learned are different distances on sets; sets of vectors are expressive data types, and allow for modeling of more complex objects such as labeled graphs. We defined the set distance learning problem as a (possibly non-differentiable) optimization task. We demonstrated the effectiveness of our framework on a number of artificial and real-world problems where the learning instances are represented as sets and graphs. From the experimental evidence we observed that: (i) in almost all the datasets there exist a mapping function with better predictive performance than that averaging-based set distances, (ii) adaptive distances $D^d_{\boldsymbol{A}}$ usually outperform standard distances (if we know the "right" mapping family), (iii) adaptive distances $D^f_{\boldsymbol{A}}$ do not fare so well, (iv) some care should be taken to avoid local optimal during training, and finally (v) the smoothing operation does not work well in practice.

The future research direction is to examine other ways to learn set distances from (1). More precisely, we plan to directly learn $\mathcal{F}$ and $\mathcal{A}$ from (1). As already mentioned, this has turned out to be more difficult; the main challenge here to represent these elements in a structured way, so that the resulting optimization task could be performed efficiently. We believe that learning $\mathcal{F}$ and $\mathcal{A}$ is a very promising research direction with many potential applications. The presented work is a first step in this direction.

Acknowledgment

This work was partially funded by the European Commission through EU projects DebugIT (FP7-217139) and e-LICO (FP7-231519). The support of the Swiss NSF (Grant 200021-122283/1) is also gratefully acknowledged.

REFERENCES

[1] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *NIPS*, 2002.

[2] M. Bernard, L. Boyer, A. Habrard, and M. Sebban. Learning probabilistic models of tree edit distance. *Pattern Recognition*, 41(8):2611–2629, August 2008.

[3] D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] M. B. Blaschko and T. Hofmann. Conformal multi-instance kernels. In *NIPS 2006 Workshop on Learning to Compare Examples*, pages 1–6, 12 2006.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge Univ. Press, 2004.

[7] C. Caetano, L. Cheng, Q.V. Le, and A.J. Smola. Learning graph matching. In *ICCV*, 2007.

[8] Thomas G. Dietterich, Richard H. Lathrop, and Tomas Lozano-Perez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1-2):31–71, 1997.

[9] Carlotta Domeniconi and Dimitrios Gunopulos. Adaptive nearest neighbor classification using support vector machines. In *NIPS*. MIT Press, 2002.

[10] T Eiter and H. Mannila. Distance measures for point sets and their computation. *Acta Informatica*, 34(2):109–133, 1997.

[11] Holger Fröhich, Jörg Wegner, Florian Sieker, and Andreas Zell. Optimal assignment kernels for attributed molecular graphs. In *ICML*, 2005.

[12] P. V. Gehler and O. Chapelle. Deterministic annealing for multiple-instance learning. In *AISTATS*, 2007.

[13] Amir Globerson and Sam Roweis. Metric learning by collapsing classes. In *NIPS 18*. MIT Press, 2006.

[14] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov. Neighbourhood component analysis. In *NIPS*, 2005.

[15] C. Helma, R. D. King, S. Kramer, and A. Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17:107–108, 2001.

[16] Tomer Hertz, Aharon Bar-Hillel, and Daphna Weinshall. Boosting margin based distance functions for clustering. In *ICML*, 2004.

[17] R. Jin, S.J. Wang, and Z.H. Zhou. Learning a distance metric from multi-instance multi-label data. In *CVPR*, 2009.

[18] J.T. Kwok and I.W. Tsang. Learning with idealized kernels. In *ICML*, 2003.

[19] G. Lanckriet, N. Cristianini, P. L. Bartlett, L. El Ghaoui, and M. Jordan. Learning the kernel matrix with semi-definite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[20] Sauro Menchetti, Fabrizio Costa, and Paolo Frasconi. Weighted decomposition kernels. In *ICML*, 2005.

[21] M. Neuhaus and H. Bunke. A probabilistic approach to learning costs for graph edit distance. In *ICPR*, pages III: 389–393, 2004.

[22] Jose Oncina and Marc Sebban. Learning stochastic edit distance: Application in handwritten character recognition. *Pattern Recognition*, 39(9):1575 – 1587, 2006.

[23] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[24] Jan Ramon and Maurice Bruynooghe. A polynomial time computable metric between point sets. *Acta Informatica*, 37(10):765–780, 2001.

[25] Eric Sven Ristad and Peter N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(5):522–532, May 1998.

[26] A. Srinivasan, S. Muggleton, R.D. King, and M.J.E. Sternberg. Mutagenesis: ILP experiments in a non-determinate biological domain. In S. Wrobel, editor, *Proceedings of the 4th International Workshop on Inductive Logic Programming*, volume 237, pages 217–232, 1994.

[27] Ivor W. Tsang, Pak-Ming Cheung, and James T. Kwok. Kernel relevant component analysis for distance metric learning. In *IJCNN*, 2005.

[28] Jun Wang and Jean-Daniel Zucker. Solving the multiple-instance problem: A lazy learning approach. In *ICML '00: Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[29] Kilian Q. Weinberger and Lawrence K. Saul. Fast solvers and efficient implementations for distance metric learning. In *ICML*, 2008.

[30] Adam Woźnica. *Distance and Kernel Based Learning over Composite Representations*. PhD thesis, University of Geneva, 2008.

[31] Adam Woźnica, Alexandros Kalousis, and Melanie Hilario. Distances and (indefinite) kernels for sets of objects. In *ICDM*, Hong Kong, 2006.

[32] Adam Woźnica, Alexandros Kalousis, and Melanie Hilario. Learning to combine distances for complex representations. In *ICML*, 2007.

[33] Adam Woźnica, Alexandros Kalousis, and Melanie Hilario. Adaptive matching based kernels for labeled graphs. In *PAKDD*, 2010.

[34] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning with application to clustering with side-information. In *NIPS*. MIT Press, 2003.