Adaptive Matching Based Kernels for Labelled Graphs

Adam Woźnica, Alexandros Kalousis, and Melanie Hilario

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

Abstract. Several kernels over labelled graphs have been proposed in the literature so far. Most of them are based on the Cross Product (CP) Kernel applied on decompositions of graphs into sub-graphs of specific types. This approach has two main limitations: (i) it is difficult to choose a-priori the appropriate type of sub-graphs for a given problem and (ii) all the sub-graphs of a decomposition participate in the computation of the CP kernel even though many of them might be poorly correlated with the class variable. To tackle these problems we propose a class of graph kernels constructed on the proximity space induced by the graph distances. These graph distances address the aforementioned limitations by learning combinations of different types of graph decompositions and by flexible matching the elements of the decompositions. Experiments performed on a number of graph classification problems demonstrate the effectiveness of the proposed approach.

1 Motivation and Related Work

Support Vector Machines (SVMs), and Kernel Methods in general, became popular due to their very good predictive performance [1]. As most of the real-world data can not be easily represented in an attribute-value format many kernels for various kinds of structured data have been proposed [2]. In particular, graphs are a widely used tool for modelling structured data in data mining / machine learning and many kernels over these complex structures have been proposed so far. These kernels have been mainly applied for predicting the activity of chemical molecules represented by sparse, undirected, labelled and two-dimensional graphs.

However, due to the rich expressiveness of graphs it has been proved that kernels over arbitrarily structured graphs, taking their full structure into account, can be neither computed [3] nor even approximated efficiently [4]. The most popular approach to tackle the above problem is based on decompositions of graphs into sub-graphs of specific types which are compared via sub-kernels. The sub-graph types mainly considered are walks [3, 5–8]. However, other researchers have experimented with shortest paths [9], sub-trees [4, 10], cyclic and tree patterns [11] and more general sub-graphs [12, 13].

A common feature in all the above graph kernels is that the *Cross Product (CP) Kernel* is used between the corresponding multi-sets of decompositions. More precisely, for particular decompositions $\mathcal{G}_1^t$ and $\mathcal{G}_2^t$ of the graphs G_1 and G_2 into sub-structures of type t the above kernels can be written as $K(G_1, G_2) = \sum_{(g_1,g_2)\in\mathcal{G}_1^t\times\mathcal{G}_2^t} k(g_1, g_2)$

M.J. Zaki et al. (Eds.): PAKDD 2010, Part II, LNAI 6119, pp. 374–385, 2010.
© Springer-Verlag Berlin Heidelberg 2010

where k is a kernel over a specific type t of graphs, and the summation over the elements of the multisets takes into account their multiplicity.

One problem with the above kernel is that *all* the possible sub-graphs of a given type are matched by means of a sub-kernel. This might adversely affect the generalization of a large margin classifier since, due to the combinatorial growth of the number of distinct sub-graphs, most of the features in the feature space will be poorly correlated with the target variable [12, 14]. Possible solutions to this problem include down-weighting the contribution of larger sub-graphs [3, 15], using prior knowledge to guide the selection of relevant parts [16], or considering contextual information for limited-size sub-graphs [12]. Yet another solution was proposed in [10] in which only specific elements of the corresponding multi-sets are matched in such a manner that the sum of similarities of the matched elements is maximum. The underlying idea in this kernel is that the actual matching will focus on the most important structural elements, neglecting the sub-structures which are likely to introduce noise to the representation. The idea of using specific pairs of points in a set kernel is promising, however, it is easy to show that this kernel is not positive semi-definite (PSD) in general [17, 18].

More importantly, to the best of our knowledge, all existing graph kernels are currently limited to a *single* type of decomposition which is then, most often, used in the context of the CP kernel. However, in general it is difficult to specify in advance the appropriate type of sub-structures for a given problem. Although, it is in principle possible to simultaneously exploit kernels defined over different representations, this is usually not done because there is a trade-off between the expressivity gained by enlarging the kernel-induced feature space and the increased noise to signal ratio (introduced by irrelevant features). A common intuition is that by decomposing into more complex sub-graphs the expressiveness, and consequently the performance, of resulting kernels increases. This is, however, in contrast with some experimental evidence [12] which suggest that decompositions into rather simple sub-structures perform remarkably well with respect to more complex decompositions on a number of different datasets. A simplified solution to the problem of representation selection is to select the decomposition by cross-validation. However, this approach is problematic since it requires the use of extra data and only one representation can be selected which limits the expressiveness of the resulting method. We can also directly learn to combine graph kernels using multi-kernel learning methods, [19]. Nevertheless, the problem with learning kernel combinations is that the combined elements should, obviously, be valid kernels. However, as we saw previously this type of kernels are based on the CP kernel that requires the complete matching of the components, raising the problems that we described above.

To tackle the above problems we propose a class of adaptive graph kernels built on proximity spaces [20]. The proximity spaces are induced by learned weighted combinations of a given set distance measure on a number of decompositions of different sub-graph types, and constructed on the basis of a representation set, typically the full[1] training set. The weighted combinations are learned from the data, exploiting the NCA method [21] developed for learning Mahalanobis metrics for vectors. By learning the

[1] It is also possible to reduce the size of the representation set relying on feature selection in the proximity space. However, this approach is not discussed in this paper.

weights of the different decompositions we hope to obtain a combination of graph representations that is expressive enough for the task at hand and does not overfit. Using the set distance measures on the graph decompositions allows for flexible ways of mapping the elements of these decompositions, namely it is not necessary to use all the elements (sub-graphs) in the mapping. We originally explored the use of proximity spaces to define a graph kernel in [22]; however, there we were limited to a fixed decomposition and did not consider learning the combinations of the different types of substructures.

In this paper we will experiment with, and learn combinations of, two types of decompositions: walks and unordered trees of various lengths and heights, respectively. The former were shown to be very effective in chemical domains and achieved state-off-the-art results [3, 5–8]. Decomposition kernels based on trees were first proposed in [4], however, experimental evaluation was not performed. Kernels based on trees were examined in [10]. Walks and trees can be easily compared by a graded similarity (e.g. standard Euclidean metric for walks). Nevertheless, most of the kernels on graphs use the Kronecker Delta Kernel (i.e. $k_\delta(x, y) = 1$ if $x = y, k_\delta(x, y) = 0$ otherwise) on sub-parts. As a result, the ability to find partial similarities is lost and the expressivity of these kernels is reduced [7]. A graded similarity on walks was considered in the graph kernel presented in [7], however, it suffers from high computational complexity since it requires taking powers of the adjacency matrix of the direct product graph, leading to huge runtime and memory requirements [9]. Graded similarities on trees were also used in [10]. Finally, we note that our framework is not limited only to walks and trees and can be applied on decompositions of any type.

This paper is organized as follows. In Sect. 2 we define all the necessary notions of graphs. In Sect. 3 we define the adaptive kernels on graphs in the proximity space induced by the set distance measures. Experimental results are reported in Sect. 4. Finally, Sect. 5 concludes the work.

2 Primer of Graph Theory

An *undirected* graph $G = (\mathcal{V}, \mathcal{E})$ is described by a finite set of *vertices* $\mathcal{V} = \{v_1, \ldots, v_k\}$ and a finite set of *edges* $\mathcal{E}$, where $\mathcal{E} = \{\{v_i, v_j\} : v_i, v_j \in \mathcal{V}\}$; for *directed* graphs we set $\mathcal{E} = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$. We shall also use the following notation for edges: $e_{ij} = \{v_i, v_j\}$ (or $e_{ij} = (v_i, v_j)$). For *labelled* graphs there is additionally a set of vertex labels $\mathcal{L}_\mathcal{V}$ and edge labels $\mathcal{L}_\mathcal{E}$ together with functions $l_\mathcal{V} : \mathcal{V} \to \mathcal{L}_\mathcal{V}$ and $l_\mathcal{E} : \mathcal{E} \to \mathcal{L}_\mathcal{E}$ that assign labels to vertices and edges, respectively. We will use $lab(x)$ to denote, in a more general form the label of x; whether $lab(x) = l_\mathcal{V}(x)$ or $lab(x) = l_\mathcal{E}(x)$ will be clear from the type of the x argument, i.e. vertex or edge. In this work we assume that the labels are vectors, i.e. $\mathcal{L}_\mathcal{V} \subseteq \mathbb{R}^p$ and $\mathcal{L}_\mathcal{E} \subseteq \mathbb{R}^n$ for some values of m and n. By $dim(lab(x))$ we will denote the dimensionality of the label vector $lab(x)$ which obviously will be the dimensionality of either $\mathcal{L}_\mathcal{V}$ or $\mathcal{L}_\mathcal{E}$, depending again on x.

Some special types of graphs that are relevant to our work are walks and trees. A *walk*, W, in a graph G is a sequence of vertices and edges, $W = [v_1, e_{12}, v_2, \ldots, e_{s,s+1}, v_{s+1}]$, such that $v_j \in \mathcal{V}$ for $1 \leq j \leq s+1$ and $e_{ij} \in \mathcal{E}$ for $1 \leq i \leq s$. Walks can also end to an edge, e.g. $W = [v_1, e_{12}, v_2, \ldots, e_{s,s+1}]$. The length l of a walk, W, is the number of vertices and edges in W. We denote the set of all walks of length l in a

graph G by $\mathcal{W}(G)^l$. A *tree* T with vertices $\mathcal{V}$ and edges $\mathcal{E}$ of a graph G is a sub-graph of G which is connected and acyclic. The root node of a tree is denoted as $root(T)$. In this work we will only consider directed trees where the order is given from the root node to the leafs. The height h of a tree T is the length of the longest walk from the root node to any of the leaf nodes (similar to walks we allow that the trees have edges as leafs). We denote the set of all directed trees of height h in a graph G by $\mathcal{T}(G)^h$. We define the neighborhood of a node, v, in a tree as $\delta(v) = \{e : e = (v, u) \in \mathcal{E}\}$, and the neighborhood of an edge, e, as $\delta(e) = \{u : e = (v, u) \in \mathcal{E}\}$, note that in fact the neighborhood of an edge is a one element set, containing a single node.

3 Kernels on Graphs

We denote the decomposition of a graph, G, into the *multi-set* of sub-graphs of type t by $\mathcal{G}^t$. We focus on decompositions into walks of various lengths l, i.e. $\mathcal{G}^t = \mathcal{W}(G)^l$, and (directed) trees of various heights h, i.e. $\mathcal{G}^t = \mathcal{T}(G)^h$. More generally, a graph G can be represented by a tuple of m different decompositions as $\boldsymbol{\mathcal{G}} = (\mathcal{G}^{t_1}, \ldots, \mathcal{G}^{t_m})^T$.

In this section we will define a class of adaptive kernels for labelled graphs. The construction of these kernels is based on set distance measures which will be used to compute the distances on the components of $\boldsymbol{\mathcal{G}}$. Since it is difficult to select a priori the appropriate types of sub-structures, i.e. components of $\boldsymbol{\mathcal{G}}$, for a given problem, we propose a method which learns a weighted (quadratic) combination of a fixed set distance on the different components of $\boldsymbol{\mathcal{G}}$. Finally, we use the learned weighted combination of the set distance to induce a proximity space on which we define the final kernel.

Distances on Sets. The central idea in the set distance measures that we consider is the definition of a mapping of the elements of one set to the elements of the other such that the final distance is determined on the basis of specific pairs of elements from the two sets. More precisely, consider two nonempty and finite sets $A = \{a\} \subseteq \mathcal{X}$ and $B = \{b\} \subseteq \mathcal{X}$. Let $d(\cdot, \cdot)$ be a distance measure defined on $\mathcal{X}$. The set distance measure d_{set} is defined on $2^{\mathcal{X}}$ as $d_{\text{set}}(A, B) = \frac{\sum_{(a,b)\in F} d(a,b)}{|F|}$ i.e. it is a normalized sum of pairwise distances over specific pairs which are defined by $F \subseteq A \times B$; different F correspond to different set distance measures. Within this framework we can define *Average Linkage* (d_{AL}), *Single Linkage* (d_{SL}), *Complete Linkage* (d_{CL}), *Sum of Minimum Distances* (d_{SMD}), *Hausdorff* (d_{H}), *RIBL* (d_{RIBL}), *Tanimoto* (d_{T}), *Surjections* (d_{S}), *Fair Surjections* (d_{FS}), *Linkings* (d_{L}) and *Matchings* (d_{M}) distance measures. Detailed description of these distance measures can be found in [23].

Distances Between Sets of Decompositions. We will now show how to use the above set distance measures to compute distances between sets of decompositions of graphs to sub-graphs of specific types. As already mentioned, we will focus on decompositions into walks and (directed) trees of various lengths and heights. In the latter, we consider *unordered trees* where the order among the siblings at a given height is not important. Both types of decompositions are obtained from a depth first exploration emanating from each node in a graph. We note that the walks and trees we consider in this work are non-standard in the sense that a walk can end in an edge and a tree can have edges as leafs.

In our decompositions we do not allow repetitions of a node within a walk or a tree, i.e. we do not allow cycles, in order to avoid the problem of *tottering* [6]. We exclude from the graph decomposition walks of the form $W = [v_1, e_{12}, v_2, \ldots, e_{s,s+1}, v_{s+1}]$ with $v_i = v_{i+2}$ for some i since these are likely to introduce redundancy. In the case of trees we do not allow such walks in the trees' descriptions. Even though tottering-free graph representations do not seem to bring an improvement to the predictive performance [6, 10], they have the advantage of inducing decompositions smaller in size. This has a direct influence on the computational complexity of the set distance measures applied over this representation.

The main difficulty in applying the set distance measures from the previous section is to define the distance, d, between the elements of the sets, i.e. walks or trees. To define a distance measure over walks we exploit the (normalized) Euclidean metric. More precisely, the distance of two walks W_i and W_j of equal length l is defined as the Euclidean metric between the labels of the elements of the walks $d^2_W(W_i, W_j) = \frac{\sum_{k=1}^{l} d^2_{\text{lab}}(lab(W_i[k]), lab(W_j[k]))}{N}$ where $W[k]$ is the k-th element of walk W, and $d_{\text{lab}}(\cdot,\cdot)$ is the Euclidean metric between the labels of corresponding walks. Obviously, $W_i[k]$ and $W_j[k]$ are of the same type and they will be either edges or vertices. The N in the denominator is a normalization factor and corresponds to the sum of the dimensionalities of the label vectors of the l elements that make up the paths, obviously $N = \sum_{k=1}^{l} dim(lab(W_i[k]) = \sum_{k=1}^{l} dim(lab(W_j[k])$. We have $0 \leq d^2_{\text{W}}(\cdot,\cdot) \leq 1$.

Lets now define a distance, $d_{\text{T}}(T_i, T_j)$, between two trees, T_i, T_j; we should note here that unlike walks, trees do not have to be of the same height. Let x, y, be two elements of the two trees found at the same height h. These elements will be either two vertices, u, v, or two edges e_i, e_j. Then the (squared) distance between x and y, $d^2_{\text{t}}(x, y)$, is recursively defined as:

$$\begin{cases} \frac{d^2_{\text{lab}}(lab(x),lab(y)) + d^2_{\text{M}}(\delta(x),\delta(y))}{N'} & \text{if } \delta(x) \neq \emptyset \wedge \delta(y) \neq \emptyset \\ \frac{d^2_{\text{lab}}(lab(x),lab(y))}{N'} & \text{if } \delta(x) = \emptyset \wedge \delta(y) = \emptyset \\ \frac{d^2_{\text{lab}}(lab(x),lab(y)) + 1}{N'} & \text{if } \delta(x) = \emptyset \otimes \delta(y) = \emptyset \end{cases}$$

where $\otimes$ is the logical XOR; $d_{\text{lab}}(lab(x), lab(y))$ is the Euclidean metric between the labels of the x, y; $\delta(x)$ is the neighbourhood function, defined in Sect. 2, that returns either: the set of edges to which a vertex connects to as a starting vertex, if x is of type vertex, or the vertex to which an edge arrives if x is of type edge; finally d_{M} is the matching set distance measure between the sets of elements that are found in the neighborhoods of x and y. The N' in the denominator is also a normalization factor that corresponds to the dimensionality of the label vectors of x and y plus one to account for the set distance dimension, i.e. $N' = dim(lab(x)) + 1 = dim(lab(y)) + 1$. The matching distance d_{M} will establish the best mapping among the elements of $\delta(x), \delta(y)$, letting outside, and penalizing for, sub-structures that cannot be matched. In a discrete case scenario it is equivalent to a frequency based distance, i.e. for each of the discrete sub-structures in $\delta(x), \delta(y)$, it will count how many times it appears and the final distance will be the sum of differences of these frequencies. Note here that $d_{\text{t}}(x, y)$ is a recursive distance requiring the computation of set distances between the elements of

the trees that are associated to x, y, at the $h+1$ height. The final distance between two trees, T_i, T_j, is given by $d_{\mathrm{T}}(T_i, T_j) = d_{\mathrm{t}}(root(T_i), root(T_j))$.

Combining Different Decompositions. Here we show how to combine different decompositions of labelled graphs in order to produce a final weighted graph distance. Recall that two graphs G_1 and G_2 can be represented by m different decompositions as $(\mathcal{G}_1^{t_1}, \ldots, \mathcal{G}_1^{t_m})^T$ and $(\mathcal{G}_2^{t_1}, \ldots, \mathcal{G}_2^{t_m})^T$ of types $t = t_1, \ldots, t_m$. For a fixed set distance measure d_{set} and the given m decompositions we define the vector $\boldsymbol{d}(G_1, G_2) = (d_{\mathrm{set}}(\mathcal{G}_1^{t_1}, \mathcal{G}_2^{t_1}), \ldots, d_{\mathrm{set}}(\mathcal{G}_1^{t_m}, \mathcal{G}_2^{t_m}))^T$ which consists of the distances over the different decompositions. Then the weighted combination of these decompositions is defined as

$$d^2_{\boldsymbol{A}}(G_1, G_2) = \boldsymbol{d}(G_1, G_2)^T \boldsymbol{A}^T \boldsymbol{A}\, \boldsymbol{d}(G_1, G_2) \tag{1}$$

where $\boldsymbol{A}$ is a $m \times m$ matrix. It should be noted that for any matrix $\boldsymbol{A}$ the matrix $\boldsymbol{A}^T\boldsymbol{A}$ is PSD, and hence $d_{\boldsymbol{A}}$ is a pseudo-metric. Here we propose a method for learning the matrix $\boldsymbol{A}$ of equation (1) directly from the data by casting the problem as an optimization task.

To learn the matrix $\boldsymbol{A}$ we use the Neighborhood Component Analysis (NCA) method from [21] which was originally developed for metric learning over vectorial data.[2] This method attempts to directly optimize a continuous version of the leave-one-out (LOO) error of the kNN algorithm on the training data. First, a conditional distribution is introduced which for each example G_i selects another example G_j as its neighbor with some probability $p_{\boldsymbol{A}}(j|i)$, and inherits its class label from the point it selects. The probability $p_{\boldsymbol{A}}(j|i)$ is based on the softmax of the $d^2_{\boldsymbol{A}}$ distance given by $p_{\boldsymbol{A}}(j|i) = \frac{e^{-d^2_{\boldsymbol{A}}(G_i, G_j)}}{\sum_{k \neq i} e^{-d^2_{\boldsymbol{A}}(G_i, G_k)}}$, $p(i|i) = 0$. We denote C_i as the set of points that share the same class with G_i, i.e. $C_i = \{j \mid class(G_i) = class(G_j)\}$. The objective function to be maximized, as described in [21], is $\mathcal{F}_{\boldsymbol{A}} = \sum_i \log(\sum_{j \in C_i} p_{\boldsymbol{A}}(j|i))$.

It is clear that for full matrices $\boldsymbol{A}$ the number of parameters to estimate is m^2. This could be problematic in cases where m is large with respect to the number of instances in the training database. One possible solution to overcome this problem is to add a soft constraint to the above objective function which results in the following regularized objective function $\mathcal{F}^{\mathrm{reg}}_{\boldsymbol{A}} = \sum_i \log(\sum_{j \in C_i} p_{\boldsymbol{A}}(j|i)) + \lambda \|\boldsymbol{A}\|^2_{\mathcal{F}}$ where $\|\boldsymbol{A}\|_{\mathcal{F}}$ denotes a Frobenious norm of matrix $\boldsymbol{A}$ and $\lambda > 0$ is a regularization parameter. The other solution is to restrict matrix $\boldsymbol{A}$ to be diagonal resulting in a weighted combination of distances for different decompositions (we denote NCA where optimization is performed over a diagonal matrix by $\mathrm{NCA}_{\mathrm{diag}}$). The main advantage of the NCA algorithm is that it is non-parametric, making no assumptions about the shape of the class conditional distributions [21]. Its main problem is that the objective function is not convex, hence some care should be taken to avoid local optima.

Adaptive Matching Based Kernels. Here we exploit the adaptive combinations of decompositions presented in Sect. 3, and the notion of proximity spaces, to define a class of adaptive matching based kernels over labelled graphs. The proximity space is

[2] We also experimented with other metric learning methods; however, due to the lack of space, these results are not reported in this paper.

defined by an instantiation of $d_{\boldsymbol{A}}$ from equation (1) and a representation set (i.e. set of prototypes) of learning instances. More precisely, consider a graph G which is decomposed in m different ways as $\boldsymbol{\mathcal{G}} = (\mathcal{G}^{t_1}, \ldots, \mathcal{G}^{t_m})^T$. For a given representation set $S = \{G_1, \ldots, G_p\}$ (we assume that each graph is uniquely identified by its index) and a distance measure $d_{\boldsymbol{A}}$ we define a mapping $\boldsymbol{d}_{\boldsymbol{A}}(G, S) = (d_{\boldsymbol{A}}(G, G_1), \ldots, d_{\boldsymbol{A}}(G, G_p))^T$. Since distance measures $d_{\boldsymbol{A}}$ are non-negative, all the data examples are projected as points to a non-negative orthotope of that vector space. The dimensionality of this space is controlled by the size of the set S. In this setting the adaptive graph kernel in the proximity space between graphs G_1 and G_2 is defined as $k_{d_{\boldsymbol{A}}}(G_1, G_2) = k(\boldsymbol{d}_{\boldsymbol{A}}(G_1, S), \boldsymbol{d}_{\boldsymbol{A}}(G_2, S))$, where k denotes an elementary kernel in the proximity space; k can be any standard kernel on vectorial data.

We mention that the complexity of computing the adaptive matching kernel between two graphs (if the weights of different decompositions are known) can be shown to be at most $O\{|S|m^2|G|^3(\alpha^{3l} + \alpha^{3h})\}$ where $|S|$ is the cardinality of the representation set, $|G|$ is the number of vertices in graph G, α is the branching factor which can be upper bounded by a small constant (usually 4), and $m = l + h$ is the number of decompositions. The complexity of learning the weight is at most $O\{m^2(n^2|G|^3 + n^3)\}$ where n is the number of graphs in the training set.

4 Experiments

We will experiment with two graph classification problems: Mutagenesis and Carcinogenicity. The application task in the Mutagenesis dataset is the prediction of mutagenicity of a set of 188 aromatic and hetero aromatic nitro-compounds which constitute the "regression friendly" version of this dataset. The other classification problem comes from the Predictive Toxicology Challenge and is defined over carcinogenicity properties of chemical compounds. This dataset lists the bioassays of 417 chemical compounds for four type of rodents: male rats (MR), male mice (MM), female rats (FR) and female mice (FM) which give rise to four independent classification problems. We transformed the original dataset (with 8 classes) into a binary problem by ignoring EE (equivocal evidence), E (equivocal) and IS (inadequate study) classes, grouping SE (some evidence), CE (clear evidence) and P (positive) in the positive class and N (negative) and NE (no evidence) in the negative one. After this transformation the MR, MM, FR and FM datasets had 344, 336, 351 and 349 instances, respectively. References to these datasets can be found in [22].

In the experiments we want to examine a number of issues related to our graph kernels. More precisely, first, for different set distance measures we will explore the effect of the length of walks (and the height of trees) on the performance of $k_{d_{\text{set}}}$, i.e. we fix the walk length to a specific l for $l = 1, \ldots, 11$ (and the height of a tree is fixed to a specific h, $h = 2, \ldots, 5$) and we use only the corresponding decomposition to construct the final kernel; we do *not* combine decompositions. We will compare the performance of the above simplified kernels with $k_{d_{\boldsymbol{A}}}$ where we combine all the different $m = 11 + 4$ decompositions. As the kernel k on the proximity space we will be always using the linear kernel in order to make a fair comparison between the algorithms and to avoid the situation where an implicit mapping given by a nonlinear kernel will influence the

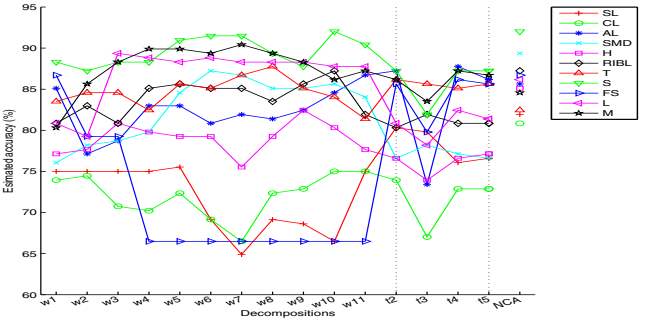


Fig. 1. Estimated accuracy of different kernels in the proximity space ($k_{d_{\text{set}}}$) vs. different decompositions for different set distance measures in the Mutagenesis dataset. wl denotes walks of length l whereas th denotes trees of height h. The last values in the plot denote the performance of $k_{d_{\boldsymbol{A}}}$ where the different decompositions are combined.

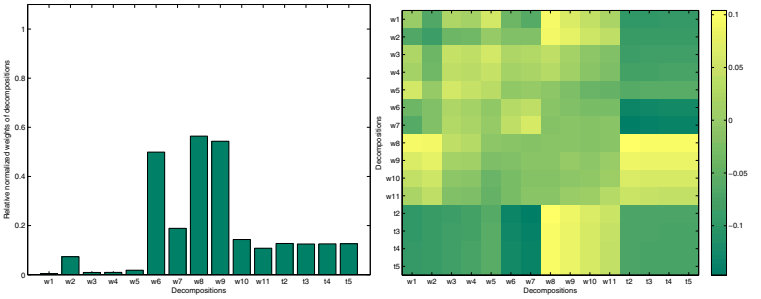
results. Second, we will examine how the performance of the kernel $k_{d_{\boldsymbol{A}}}$ compares with the following two set kernels based on averaging: (i) the direct sum kernel [1] based of the cross product kernels applied on the different $m = 11 + 4$ decompositions with the linear kernel (in case of walks) and the tree kernel[3] (for trees) as kernels on the sets' elements ($k_{\Sigma,\text{CP}}$), and (ii) the linear kernel in the proximity space induced by $d_{\boldsymbol{A}}$, also over the combination of the different $m = 11 + 4$ decompositions, where d_{AL} set distance measure is applied over the decompositions. Finally, we will compare our graph kernels with other graph kernels form the literature.

We use the SVM algorithm where the parameter C is optimized in an inner 10-fold cross-validation loop over the set $C = \{0.1, 1, 10, 50\}$. In all the experiments, unless stated otherwise, we use the regularized version of the NCA algorithm over full matrices $\boldsymbol{A}$, where the regularization parameter λ is internally 10-fold cross-validated over $\lambda = \{0, 0.1, 1, 10\}$. In some experiments we also use the version of NCA, denoted as NCA_{diag}, where $\boldsymbol{A}$ is limited to a diagonal matrix. Note that in the experiments we have 11 different instantiations of the $k_{d_{\boldsymbol{A}}}$ kernel, each one corresponding to one of the 11 set distance measures. In all the experiments accuracy is estimated using stratified 10-fold cross-validation and controlled for the statistical significance of observed differences using McNemar's test, with significance level of 0.05.

4.1 Results and Analysis

Performance of Individual vs. Combined Decompositions. We first examine the influence of the length of the walks and heights of the trees to the predictive performance of SVM. The results for the Mutagenesis dataset are presented in Fig. 1. From the results it is clear that in Mutagenesis the optimal decomposition depends on the actual set distance measure. For example, for d_{SMD} the highest predictive accuracy is obtained

[3] The definition of the tree kernel is based on recursive and alternating computations of the linear and cross product kernels [1]; the way these kernels are combined to compute the final tree kernel is similar to the definition of the tree distance from Sect. 3.



(a) $\mathrm{NCA}_{\mathrm{diag}}$ (b) NCA, $\lambda = 0$

Fig. 2. Relative importance of different decompositions (FM dataset) for the d_{SMD} set distance measure both for diagonal and full matrices $\boldsymbol{A}$. wl denotes walks of length l whereas th denotes trees of height h. It is represented as normalized weights $\boldsymbol{A}$.

for walks of length 6, whereas for d_{S} the best decomposition is into walks of length 10. Additionally, with the exception of d_{SL}, decompositions in walks in general outperform decompositions into trees. For the other examined datasets the optimal decompositions are different. Thus we have no way to know a-priori which type of decomposition is appropriate. By combining them using NCA (results also given in Fig. 1) we get a classification performance that is in most cases as good as that of the single decompositions. Indeed, in Mutagenesis the performance of NCA was significantly better in 68 cases, in 92 cases the differences were not significantly meaningful, and in 5 cases it was significantly worse.[4] In FM the corresponding values are 7, 158 and 0; in FR: 5, 147 and 22; in MM: 12, 153 and 0; in MR: 81, 83 and 1. We should stress here that it is not fair to compare NCA with the results of the best decomposition, since this estimate is optimistically biased since we need to have the results of the cross-validation to determine which is the best decomposition. In a fair comparison NCA should be compared to a model selection strategy in which the best decomposition is selected using an inner cross-validation. However an inner cross-validation-based model selection strategy does not make full use of the data, thus its estimates might not be reliable, and it is computationally expensive.

The results of the optimization process that learns the optimal combination of decompositions can be graphically presented providing insight to the relative importance of different decompositions. In Fig. 2 we give an example of such a visualization for the FM dataset and the d_{SMD} set distance measure. In the left graph of that figure the optimization was performed for diagonal matrices using $\mathrm{NCA}_{\mathrm{diag}}$. The different elements in x-axis are the elements of the diagonal of the matrix $\boldsymbol{A}$ which correspond to a decomposition into walks and trees whereas the y-axis represents the weights, normalized by a Frobenious norm of $\boldsymbol{A}$, returned by the optimization method. What we see from the graph is that in FM the highest weights are assigned to walks of lengths longer than 6 and all tress, independent of their heights. The right graph of Fig. 2 provides the visualization of the optimization process for NCA, where now the matrix $\boldsymbol{A}$ is a full

[4] The total number of comparisons is 165 = 11 walk decompositions x 11 set distances + 4 tree decompositions x 11 set distances.

Table 1. Accuracy and significance test results of SVM in the graph datasets (the + sign stands for a significant win of the first algorithm in the pair, - for a significant loss and = for no significant difference). The sign in the first parenthesis corresponds to the comparison of SVM vs. SVM with $k_{d_{\boldsymbol{A}}}$ with d_{AL}, while the sign in the second parenthesis compares SVM vs. SVM with $k_{\Sigma,\mathrm{CP}}$.

Set Distance	Mutagenesis	FM	FR	MM	MR
d_{SL}	81.91 (=)(=)	65.04 (=)(=)	67.23 (=)(=)	66.96 (=)(=)	63.95 (=)(+)
d_{CL}	80.85 (=)(=)	60.46 (=)(=)	63.24 (-)(=)	65.18 (=)(=)	67.73 (+)(+)
d_{SMD}	89.36 (=)(+)	62.17 (=)(=)	67.52 (=)(=)	66.37 (=)(=)	65.99 (=)(+)
d_{H}	85.11 (=)(=)	64.18 (=)(=)	66.09 (=)(=)	66.07 (=)(=)	65.11 (=)(+)
d_{RIBL}	87.23 (=)(=)	64.76 (=)(=)	67.52 (=)(=)	66.66 (=)(=)	67.15 (+)(+)
d_{T}	82.45 (=)(=)	64.76 (=)(=)	67.81 (=)(=)	68.45 (=)(+)	68.02 (+)(+)
d_{S}	92.02 (+)(+)	64.18 (=)(=)	67.52 (=)(=)	66.96 (=)(=)	61.34 (=)(=)
d_{FS}	86.70 (=)(=)	61.60 (=)(=)	64.10 (=)(=)	63.99 (=)(=)	61.34 (=)(=)
d_{L}	86.17 (=)(=)	64.18 (=)(=)	65.81 (=)(=)	64.88 (=)(=)	66.28 (+)(+)
d_{M}	84.57 (=)(=)	64.46 (=)(=)	66.66 (=)(=)	65.48 (=)(=)	67.73 (+)(+)
d_{AL}	85.64	62.46	66.38	66.07	60.46
$k_{\Sigma,\mathrm{CP}}$	84.04	62.46	64.96	63.39	57.85

matrix ($\lambda = 0$). One surprising observation is that for NCA the decompositions into trees and long walks are assigned low weights. At the same time combinations of trees with long walks and combinations of shorter walks are of high importance.

Performance of Matchings Strategies. The next dimension of comparison is the relative performance of the instantiations of the $k_{d_{\boldsymbol{A}}}$ kernel for different set distances and the following two set kernels based on averaging: (i) the direct sum kernel [1] based on the cross product kernels applied on the different 15 decompositions ($k_{\Sigma,\mathrm{CP}}$), and (ii) the linear kernel in the proximity space induced by $d_{\boldsymbol{A}}$, also over the combination of the different 15 decompositions, with the d_{AL} set distance measure.

The results are presented in Table 1. The relative performance of kernels based on specific pairs of elements and kernels based on averaging depends on the actual application. For Mutagenesis and MR there is an advantage of the kernels based on specific pairs of elements, while for the remaining datasets the performances are similar. Overall, the choice of the appropriate way of matching the elements of two sets depends on the application and ideally should be guided by domain knowledge, if such exists. Nevertheless, the relative performance of the different kernels provides valuable information about the type of problem we are facing. For example, by examining Mutagenesis and MR we see that averaging performs poorly, indicating that the local structure of the molecules is important, while in the remaining datasets both global and local structures seem to be equally informative.

Comparison with Other Graph Kernels. In Table 2 we provide the best results reported in the literature on the same benchmark datasets.[5] Additionally, we report the performance of kernels corresponding to d_{SMD} and d_{T}, both achieving stable good

[5] We only cite works which use similar features to describe atoms and bonds. In particular our results for Carcinogenicity are not directly comparable with the ones reported in [10].

Table 2. Comparison with other related graph kernels

Graph kernels	Mutagenesis	FM	FR	MR	MM
Best kernel from [5]	85.1	63.4	66.1	58.4	64.3
Best kernel from [8]	91.5	64.5	66.9	65.7	66.4
Based on d_{SMD}	89.4	62.2	67.5	66.4	66.0
Based on d_{T}	82.4	64.8	67.8	68.4	68.0
Our best kernel	92.0	65.0	67.8	68.5	68.0

results across the different datasets. The values in the "Our best kernel" row correspond to the best kernel 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 2 since in all the cases multiple results where available and we reported on the best. The corresponding results show that if the mapping is carefully selected for a problem at hand, then the performance of the corresponding kernel is better than the performance of the existing graph kernels. At the same time kernels corresponding to d_{SMD} and d_{T} achieve stable results which are close to the state-of-the-art for the considered datasets.

5 Conclusions and Future Work

It has been argued in [7] that a "good" kernel for graphs should fulfil at least the following requirements: (i) should be a good similarity measure for graphs, (ii) its computational time should be possible in polynomial time, (iii) should be positive semi-definite and (iv) should be applicable for various graphs. In this paper we proposed a class of adaptive kernels for graphs which are based on walks and trees, that are computable in polynomial time, that are PSD, and are applicable to a wide range of graphs. A distinctive feature of our kernels is that they allow for (i) combinations of different types of decompositions and (ii) specific types of mappings between sub-parts. The effectiveness of the approach was demonstrated on problems of activity prediction of chemical molecules. Finally, the ideas presented in this work are not limited to graphs and can be directly exploited to define kernels over not only special kinds of graphs like sequences and trees but also over instances described using first- (or higher-)order logic [24].

In the future work we would like to examine decompositions of graphs into substructures other than walks and trees (e.g. the cyclic patterns from [11]). Moreover, we will apply the methods developed in this work on larger datasets, e.g. the HIV dataset described in [12].

Acknowledgments. 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. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
2. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explor. Newsl. 5(1), 49–58 (2003)
3. Gärtner, T., Flach, P., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In: Proceedings of COLT 16 and the 7th Kernel Workshop (2003)
4. Ramon, J., Gärtner, T.: Expressivity versus efficiency of graph kernels. In: First International Workshop on Mining Graphs, Trees and Sequences (held with ECML/PKDD'03) (2003)
5. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized kernels between labeled graphs. In: ICML, Washington, DC (2003)
6. Mahé, P., Ueda, N., Akutsu, T., Perret, J.L., Vert, J.P.: Extensions of marginalized graph kernels. In: ICML (2004)
7. Borgwardt, K.M., Ong, C.S., Schönauer, S., Vishwanathan, S., Smola, A.J., Kriegel, H.P.: Protein function prediction via graph kernels. Bioinformatics 21(1), i47–i56 (2005)
8. Ralaivola, L., Swamidass, S.J., Saigo, H., Baldi, P.: Graph kernels for chemical informatics. Neural Networks, 1093–1110 (2005)
9. Borgwardt, K.M., Kriegel, H.P.: Shortest-path kernels on graphs. In: ICDM (2005)
10. Fröhich, H., Wegner, J., Sieker, F., Zell, A.: Optimal assignment kernels for attributed molecular graphs. In: ICML (2005)
11. Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: KDD (2004)
12. Menchetti, S., Costa, F., Frasconi, P.: Weighted decomposition kernels. In: ICML (2005)
13. Sherashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., Borgwardt, K.: Efficient graphlet kernels for large graph comparison. In: 12th AISTATS (2009)
14. Ben-David, S., Eiron, N., Simon, H.: Limitations of learning via embeddings in euclidean half spaces. Journal of Machine Learning Research 3, 441–461 (2002)
15. Collins, M., Duffy, N.: Convolution kernels for natural language. In: NIPS (2002)
16. Cumby, C., Roth, D.: On Kernel Methods for Relational Learning. In: ICML (2003)
17. Woźnica, A., Kalousis, A., Hilario, M.: Distances and (indefinite) kernels for sets of objects. In: ICDM (2006)
18. Vert, J.P.: The optimal assignment kernel is not positive definite (2008)
19. Lanckriet, G., Cristianini, N., Bartlett, P.L., Ghaoui, L.E., Jordan, M.: Learning the kernel matrix with semi-definite programming. Journal of Machine Learning Research 5 (2004)
20. Pekalska, E., Duin, R.: The Dissimilarity Representation for Pattern Recognition. In: Foundations and Applications. World Scientific Publishing Company, Singapore (2005)
21. Goldberger, J., Roweis, S., Hinton, G., Salakhutdinov, R.: Neighbourhood component analysis. In: NIPS. MIT Press, Cambridge (2005)
22. Woźnica, A., Kalousis, A., Hilario, M.: Matching based kernels for labeled graphs. In: Mining and Learning with Graphs (MLG 2006), with ECML/PKDD, Berlin, Germany (2006)
23. Ramon, J., Bruynooghe, M.: A polynomial time computable metric between point sets. Acta Informatica 37(10), 765–780 (2001)
24. Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Mach. Learn. 57(3), 205–232 (2004)