International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 26, No. 5 (2012) 1260005 (22 pages)

DOI: 10.1142/S0218001412600051

World Scientific
www.worldscientific.com

BRIDGING STRUCTURE AND FEATURE REPRESENTATIONS IN GRAPH MATCHING

WAN-JUI LEE*, VERONIKA CHEPLYGINA,
DAVID M. J. TAX, MARCO LOOG and ROBERT P. W. DUIN
Pattern Recognition Laboratory
Delft University of Technology, The Netherlands
**w.j.lee@tudelft.nl*

Received 30 May 2011
Accepted 19 April 2012
Published 19 October 2012

Structures and features are opposite approaches in building representations for object recognition. Bridging the two is an essential problem in pattern recognition as the two opposite types of information are fundamentally different. As dissimilarities can be computed for both the dissimilarity representation can be used to combine the two. Attributed graphs contain structural as well as feature-based information. Neglecting the attributes yields a pure structural description. Isolating the features and neglecting the structure represents objects by a bag of features. In this paper we will show that weighted combinations of dissimilarities may perform better than these two extremes, indicating that these two types of information are essentially different and strengthen each other. In addition we present two more advanced integrations than weighted combining and show that these may improve the classification performances even further.

Keywords: Graph edit distance; graph kernel; multiple instance learning.

1. Introduction

Most techniques from statistics, machine learning and pattern recognition that have been developed, such as dimension reduction, clustering, classification and regression tasks, have been applied to numerical data. These techniques have been applied with the assumption that data can be represented in a vector space where each feature forms a dimension of the space. However, for structural data, a data object includes not only the numerical feature values but also the inter-relationships between features. Structural information is essential in areas such as bioinformatics and social network analysis. Here, objects such as chemical compounds or network structures include not only numerical feature values, but also inter-relationships that can be modeled by graphs. Traditional machine learning and pattern recognition techniques for statistical data, however, cannot be applied

directly to structural data because graph data cannot be embedded into a vector space in a straightforward manner.

How to utilize these two sources of information at the same time is a crucial issue. A naive way of thinking is to assume these sources to be independent, and apply statistical pattern recognition approaches to only features and structural pattern recognition approaches to only structures to obtain two (dis)similarity measures. These two results can then be combined with mathematical operations such as sum or product to derive a joint (dis)similarity measure.

However, as we have mentioned earlier, structure is usually the inter-relationships of features. Therefore, it might take more information into account if these two sources can be considered at the same time for comparing objects. More importantly, if two objects have very similar features, it is not possible to distinguish them, but if these objects have different structures, then the probability of separating them is much higher, and vice versa.

In structural pattern recognition,[3,7,10,13,28] one of the most fundamental issues is the lack of techniques to embed the structural data objects into one and the same vector space. Currently, most problems can only be solved by deriving pairwise dissimilarities between such objects and deciding on the class of a new, unseen structural object based on the class of its nearest neighbor.[14] Most techniques from statistics, machine learning and pattern recognition[15,17,25,27,48] can analyze data sets that can typically be represented in a fixed-dimensional numerical vector space by means of so-called feature vectors. Unfortunately, this means that an essential part of these versatile and powerful techniques cannot be applied directly to the non-vectorial, structural data. In consequence, being able to turn structural data into a proper vectorial representation, the principal machine learning and pattern recognition tools could be utilized to the benefit of structural pattern recognition problems. Currently, in real-world applications, structural data is often simplified into numerical data by neglecting the structures and discarding inconsistent information. Methods such as graph edit distance or graph kernels use both structure and feature representations, but usually put more emphasis on structures. One can also combine these two sources of information by assuming their independence and have an average sum of two individual distance measurements. To utilize both feature and structure representations in a flexible and smooth setting, we propose the modified graph edit distance and the modified shortest path graph kernel.

Graph edit distance[9,40,42,43,45,46] is one of the most simple and intuitive approaches to compute the distances between graphs. The idea is to find the minimal cost for transforming one graph to the other. To define the cost for the graph edit distance, one has to set weights for nodes and edges. However, there is no adjusting between feature and structure representations. To have a better understanding of the utilization of both types of information, in this work we propose the modified graph edit distance which includes an extra parameter which controls the impact of these two different sources of information. As a result, the modified graph distance

can compute dissimilarity matrices for using only pure structures, the combination of both structures and feature representations and pure feature representations.

Graph kernels[20,22,26] often express the similarity of two graphs by summing over the similarities between all subparts, such as walks, trees or paths, between these graphs. Similar graphs should have similar subparts and therefore a high similarity value, while different graphs should have less subparts that match. For instance, the shortest path kernel[4] sums the similarities between shortest paths between any two connected nodes in the graphs. This similarity is measured using both the nodes and edges of the paths involved. Here again, there is no way to tune how much impact these sources of information have on the similarity of the path. Here we propose a modified shortest path kernel where an extra parameter controls this impact.

The rest of the paper is organized as follows. In Sec. 2, an overview of multiple instance learning, graph edit distance and graph kernels is given. A small toy example is given in Sec. 3 to show the necessity of utilizing both structure and feature representations. The modified procedures of graph edit distance and shortest path kernel which include an extra parameter to control the impact of structure and feature information are described in Sec. 4. Simulation results are presented in Sec. 5. Finally, conclusions and discussions are given in Sec. 6.

2. Related Work

Graphs often have different numbers of nodes, and therefore are difficult to represent by a fixed-size vector. Hence, it is very challenging to handle even only the features in graphs with classical statistical pattern recognition techniques. By considering only the features on the nodes of a graph, each graph becomes a set of feature points, and the size of this set varies. Multiple-instance learning (MIL)[2,12,16,32,47,50] is one of the most well-known approaches for finding classifiers for bags of instances, where the number of instances in each bag can be different. Therefore, it is suitable for computing distances between graphs if only features are considered.

There are also many techniques for computing distances between graphs, and they can be roughly divided into three groups: graph spectra, graph edit distance and graph kernels. The most well-known graph spectral are the sets of eigenvectors and eigenvalues derived by performing an eigendecomposition of the adjacency matrix or a derived representation of a graph. There are different variants of such spectra,[11,30,38,51] which are also much used in embedding and comparing graphs. All these spectral methods, however, suffer from an elementary shortcoming as they are only capable of embedding graphs with no features on the nodes. Graph edit distance and graph kernels on the other hand can use both structure and feature representations, and therefore we focus on developing modified procedures for these two approaches to include a mechanism for adjusting the impact between the feature and structure representations in this work.

In the following, we give brief introductions on MIL, graph edit distance and graph kernel.

2.1. *Multiple instance learning*

In MIL it is assumed that an object is represented by a structureless collection of feature vectors, or, in MIL terminology, a bag of instances.[16] Objects are assumed to come from a positive or negative class, and typically it is assumed that objects from the positive class contain at least one instance from a so-called *concept*. The task of a classifier is then to identify if one of the instances belong to the concept, and label the object then to the positive class. Many MIL algorithms therefore contain an optimization strategy to search for the most informative instance per bag, and create a model of the concept. The original model proposed by Ref. 16 was an axis-parallel rectangle that was grown and shrunk to best cover the area of the concept. It is applied to a drug discovery problem where molecules have to be distinguished based on their shape into active and inactive molecules. It appears that this rectangular model fits well with the molecule shape classification, but it is less successful in other applications.

A probabilistic description of the MIL problem was given by Ref. 32. The concept is modeled by a general probabilistic model (typically an axis-parallel Gaussian is used). Unfortunately, the optimization of the parameters requires a computationally expensive maximization of an likelihood that is adapted to include the constraint that at least one of the instances in a positive bag has a high concept probability. Newer methods often avoid the modeling of the concept by a density model, and try to separate concept instances from background instances using a discriminative approach.[1,49]

In time, more and more classification problems are identified as MIL problems, but the assumption of the presence of a single concept often does not hold. For many applications the overall distribution of instance is actually informative, and therefore training on all instances is feasible, or the application of general bag similarities.[21]

2.2. *Graph definitions*

A graph is a set of nodes connected by edges in its most general form. Consider the graph $G = (V, E, \mu, \nu)$ with

— the node set $V = \{v_1, v_2, \ldots, v_n\}$
— the edge set $E = \{e_1, e_2, \ldots, e_m\} \subset V \times V$
— the node labeling function $\mu : V \to L$
— the edge labeling function $\nu : E \to L$,

where n is the total number of nodes in the graph and is usually called the graph size, and m is the total number of edges in the graph. Edges, $e_1, e_2, \ldots, e_m$, are given by pairs of nodes $(v_i, v_j) \in E$, where v_i denotes the source node and v_j is the target node of a directed edge for directed graphs. By adding a reverse edge (v_j, v_i) for each edge, the undirected graphs can also be easily modeled. The node and edge labeling functions can assign either a set of integers $\in R^n$, or a set of symbolic labels

($\ell = a, b, c, \ldots$). Furthermore, by assigning all the nodes the same label, one can obtain the unlabeled graphs. Similarly, by assigning all the edges the same label, one can derive unweighted graphs.

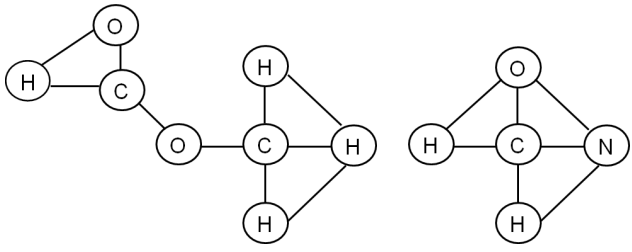
In a graph, a walk of length l is defined as a sequence of nodes $(v_1, v_2, \ldots, v_{l+1})$ where $(v_i, v_{i+1}) \in E, 1 \leq i \leq l$. A path is a walk $(v_1, v_2, \ldots, v_{l+1})$ such that $v_i \neq v_j \leftrightarrow i \neq j$.

2.3. *Graph edit distance*

A common way to define the dissimilarity of two graphs is to measure the minimal distortion that is needed for transforming one graph into the other. Graph edit distance is one of the most flexible ways for measuring dissimilarity between pairs of graphs. A standard set of edit distance operations to define distortion includes insertion, deletion and substitution of both nodes and edges. The substitution of two nodes v_i and v_j is denoted by $(v_i \rightarrow v_j)$, the deletion of node v_i is by $(v_i \rightarrow \phi)$, and the insertion of node v_j is by $(\phi \rightarrow v_j)$. Similar notations are also applied to edges. For each edit operation, a cost is given for measuring the strength of the corresponding operation and is to define whether an edit operation represents a strong modification of the graph. Given two graphs, the source graph G_1 and the target graph G_2, the idea is to delete some nodes and edges from G_1, relabel some of the remaining nodes and edges and probably insert some nodes and edges, such that G_1 is completely transformed into G_2. The dissimilarity between the two is given by the cost of a sequence of edit operation (so-called edit path) that transforms G_1 into G_2. For a pair of graphs G_1 and G_2, there could be a number of different edit paths transforming G_1 into G_2, and the minimum cost edit path between two graphs is the edit distance of two graphs.

To find the optimal graph edit distance, the A^* searching tree[6,24] is often used. However, the computational complexity for such algorithm is exponential in the number of nodes of the involved graphs. To cope better with the problem, in Ref. 40 a suboptimal method is proposed. The idea is to decompose graphs into sets of subgraphs consist of only one node and its connecting edges. The graph matching problem is then reduced to the problem of finding the optimal match between the sets of subgraphs by a bipartite matching procedure. This method aims for the suboptimal solutions and therefore generally returns an approximate graph edit distance.

The process of graph matching can be seen as an assignment problem by assigning the nodes of graph G_1 to the nodes of graph G_2, such that the overall edit costs are minimal. One of the most commonly used methods for solving the assignment problem is Munkres' algorithm.[33] Despite the assignment of the nodes is with the minimal cost, the edit path found by this method is only suboptimal. The reason is that edge operations are always implied by node operations. Munkres' algorithm finds the minimum cost of the node assignment, without considering edge edit operations. The costs of these implied edge operations are added at the end of the



(a) (b)

Fig. 1. Examples of 2 graphs (a) g_1; (b) g_2.

computation. Hence, the node assignment and the implied edge assignments found by the algorithm is not necessary corresponding to the optimal graph edit path.

Figures 1(a) and 1(b) are two example of graphs g_1 and g_2. In order to compute the distance between g_1 and g_2, these two graphs are decomposed node by node into local structures as those in the right and left side of Fig. 2. Each graph is represented by a set of nodes with their connecting edges. Then the problem becomes finding the best assignment between these two sets of nodes with their connecting edges.

2.4. *Graph kernels*

A common approach to define kernels on graphs is to decompose the graphs into sets of substructures and count the number of matching substructures, i.e. the number of substructures in the intersection of the two sets. This is also called the intersection kernel.[26] Let X_1 and X_2 be respectively the sets of substructures of graphs G_1 and G_2. The intersection kernel is then defined as:

$$K_{\cap}(G_1, G_2) = |X_1 \cap X_2|. \tag{1}$$

This is equivalent to[26]:

$$K(G_1, G_2) = \sum_{x_i \in X_1, x_j \in X_2} k_{\delta}(x_i, x_j). \tag{2}$$

Different substructures have been used to define such kernels, such as walks,[22,29] subtrees[39] or cycles.[26] For instance, the random walk kernel counts the number of matching walks in two graphs. The match k_{walk} is originally defined to reflect an exact match between node labels. In Ref. 5, this definition is extended to:

$$k_{\text{walk}}((v_1, v_2, \ldots, v_l), (w_1, w_2, \ldots, w_l)) = k_{\text{step}}((v_i, v_{i+1}), (w_i, w_{i+1})), \tag{3}$$

where

$$k_{\text{step}}((v_i, v_j), (w_i, w_j)) = k_{\text{node}}(v_i, w_i) * k_{\text{node}}(v_j, w_j) * k_{\text{edge}}((v_i, v_j), (w_i, w_j)) \tag{4}$$

and where k_{node} and k_{edge} are kernels on node and edge labels (for instance, Radial Basis Function kernels).

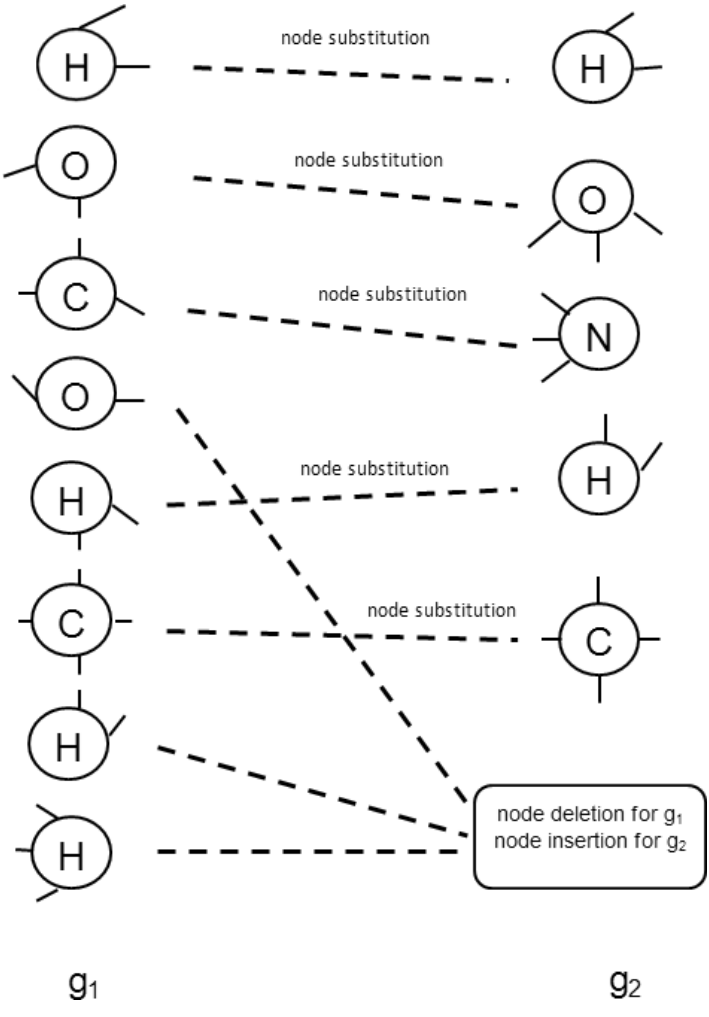


Fig. 2. Example of the suboptimal graph edit distance approach.

For random walks, Eq. (2) cannot be used directly because it is not possible to enumerate all the walks in a graph, as going back and forth between nodes ("tottering") is permitted. However, an approach to find the $K(G_1, G_2)$ without explicitly doing the calculation is proposed in Ref. 22. Unfortunately, this approach requires the inversion of a $|V_1| * |V_2| \times |V_1| * |V_2|$ matrix, which leads to large time and memory requirements, especially for large graphs.

Next to being computationally expensive, the random walk and other similar kernels often have limited expressiveness, as pointed or walks which totter inflate the similarity value of two graphs, so that even two (globally) very different graphs can have a high similarity. Using paths instead of walks can help the tottering problem, however, the computation of this kernel is NP-hard.

Therefore in Ref. 4 a kernel based on only shortest paths, which can be computed in polynomial time, is proposed. In this kernel, a graph G_1 is first transformed into a shortest path graph S_1, such that if there is a path of length l between two nodes in G_1, there is an edge with label l between these nodes in. Because any walk $(v_1, v_2, \ldots, v_{l+1})$ in G_1 is reduced to a shortest path representation (v_1, v_{l+1}, l) in S_1, the kernel can be computed as the random walk kernel on walks of length 1.

$$K_{\text{shortpath}}(G_1, G_2) = \sum_{(v_i,v_j)\in E_1,(w_i,w_j)\in E_2} k_{\text{path}}((v_i, v_j), (w_i, w_j)), \tag{5}$$

where the kernel on two paths is defined as:

$$k_{\text{path}}((v_i, v_j), (w_i, w_j)) = k_{\text{node}}(v_i, w_i) * k_{\text{node}}(v_j, w_j) * k_{\text{edge}}((v_i, v_j), (w_i, w_j)). \tag{6}$$

3. A Toy Example

In this section, we use a toy example to illustrate the limitations of using only features or only structures.

Consider the example in Fig. 3. Looking only at the features, there are two bags of instances {0,1,0} and {0,0,1} which have a distance of 0 with many common MIL measurements. On the other hand, the structure of G_1 and G_2 is also identical. Simply averaging between distances on the feature representations and on the structure representations will not be able to highlight the difference between the graphs, therefore an approach that takes both types of information into account is necessary.

4. Modified Procedures

In this section, we modify the graph edit distance in Sec. 2.3 and the shortest path kernel in Sec. 2.4 to include an extra parameter for controlling the impact of structure and feature representations.

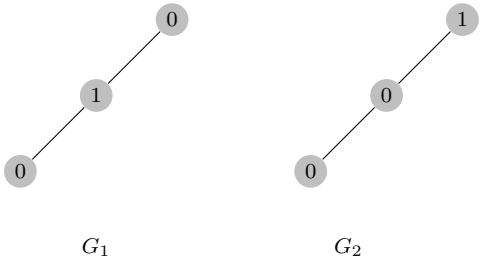


Fig. 3. Toy example.

4.1. *Modified graph edit distance for bridging feature and structure representations*

Here we extend the idea proposed in Ref. 40 which is introduced in Sec. 2.3 by separating the cost matrix C into two parts: C_F for features and C_S for structure.

The cost matrix C becomes the weighted sum of these two matrices C_F and C_S in the form of $C = \alpha C_S + (1-\alpha)C_F$, where α is an user-defined parameter. It is more general than the original graph edit distance method in the sense that the feature and structure representations are scaled by the parameter α which decides how much feature and structure representations should be used in the distance measurement.

The cost matrices for structure C_S and feature C_F are by definition quadratic and are defined as

$$C_S = \left(\begin{array}{cccc|cccc} s_{1,1} & s_{1,2} & \cdots & s_{1,m} & s_{1,\phi} & \infty & \cdots & \infty \\ s_{2,1} & s_{2,2} & \cdots & s_{2,m} & \infty & s_{2,\phi} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ s_{n,1} & s_{n,2} & \cdots & s_{n,m} & \infty & \cdots & \infty & s_{n,\phi} \\ \hline s_{\phi,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & s_{\phi,2} & \ddots & \vdots & 0 & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \infty & \cdots & s_{\phi,m} & 0 & \cdots & 0 & 0 \end{array}\right)_{n+m,n+m} \tag{7}$$

and

$$C_F = \left(\begin{array}{cccc|cccc} f_{1,1} & f_{1,2} & \cdots & f_{1,m} & f_{1,\phi} & \infty & \cdots & \infty \\ f_{2,1} & f_{2,2} & \cdots & f_{2,m} & \infty & f_{2,\phi} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\ f_{n,1} & f_{n,2} & \cdots & f_{n,m} & \infty & \cdots & \infty & f_{n,\phi} \\ \hline f_{\phi,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\ \infty & f_{\phi,2} & \ddots & \vdots & 0 & 0 & \cdots & \vdots \\ \vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\ \infty & \infty & \cdots & f_{\phi,m} & 0 & \cdots & 0 & 0 \end{array}\right)_{n+m,n+m}, \tag{8}$$

where $s_{i,j}$ and $f_{i,j}$ denote the feature and structure costs of a node substitution, $s_{i,\phi}$ and $f_{i,\phi}$ denote the feature and structure costs of a node deletion ($v_i \to \phi$), and $s_{\phi,j}$ and $f_{\phi,j}$ denote the feature and structure costs of a node insertion ($\phi \to v_j$).

To keep the simplicity of the approach, $f_{i,j}$ is the feature difference of node i and j, that is, the distance of two feature vectors from node i and j. If the graphs are with pure structure which means there are no feature vectors on the nodes, $f_{i,j}$ is set to 0. $f_{i,\phi}$ and $f_{\phi,j}$ are the distance of the feature vector to the origin.

Similarly, $s_{i,j}$ is the difference of numbers of edges between node i and j. If the graphs are with pure features which means there are no edges on the nodes, $s_{i,j}$ is set to 0. $s_{i,\phi}$ and $s_{\phi,j}$ are the number of edges of node i and node j, respectively.

The left upper corner of both cost matrices C_F and C_S represent the costs of all possible node substitutions, the diagonal of the right upper corner the costs of all possible node deletions, and the diagonal of the bottom left corner the costs of all possible node insertions. Note that each node can be deleted or inserted at most once. Therefore any nondiagonal element of the right-upper and left-lower part is set to ∞. The bottom right corner of the cost matrices is set to zero since substitutions of the form $(\phi \to \phi)$ should not cause any cost.

By restricting the scaling parameter α between 0 and 1, the new cost matrix C can be computed as $\alpha C_S + (1-\alpha)C_F$. Based on the new cost matrix C, Munkres' algorithm can be executed to find the best assignment between the nodes of two graphs. Consequently, each node of graph G_1 is either uniquely assigned to a node of G_2, or to the deletion node ϕ. Vice versa, each node of graph G_2 is either uniquely assigned to a node of G_1, or to the insertion node ϕ. The ϕ nodes in G_1 and G_2 corresponding to rows $n+1,\ldots,n+m$ and columns $m+1,\ldots,m+n$ that are not used cancel each other out without any costs.

After finding the optimal node assignment between a pair of graphs, the implied edge costs need to be added to the cost computed with the Munkres' algorithm to derive the real graph edit cost.

4.2. *Modified graph kernel*

We use the shortest path kernel from Ref. 4 as the starting point. The kernel on two paths is defined as in Eq. (6). For simplicity of notation, let $k_{\text{path}} = k_{\text{start}} * k_{\text{end}} * k_{\text{edge}}$. These kernels may themselves also be a product of different kernels. In the original problem domain of protein prediction, multiple types of labels for nodes and edges are present. The kernel on nodes is then a product of kernels on the different labels of these nodes. We restrict ourselves to a case where nodes and edges only have a numeric label, and use a simple choice for both k_{node} and k_{edge}, namely the "bridge kernel". This kernel converts a distance between two vectors to a similarity value between 0 and a predefined threshold c, as follows:

$$k(x_i, x_j) = \max(0, c - |x_i, x_j|). \tag{9}$$

To be able to vary the amount of influence that the features and structure have on the shortest path kernel, we want to introduce a parameter $\alpha \in [0,1]$ in such a way that for $\alpha = 1$, k_{path} is only influenced by the node kernels, and for $\alpha = 0$ it is only influenced by the edge kernel. Because in the original formulation, k_{path} is a product

and not a sum, we cannot apply α in a straightforward, weighted averaging way. A possible way to let α influence the product is:

$$k_{\text{path}} = (k_{\text{start}} * k_{\text{end}})^{\alpha} * (k_{\text{edge}})^{1-\alpha}. \tag{10}$$

Note that this approach does not guarantee that the kernel is positive definite, however, the information contained in such kernels may still be of interest.[37]

5. Experiments

In this section, we compare the performance of the modified procedures with the weighted average of structure and feature representations using the 1-nearest neighbor classifier[14] in the original space (1nnc), the 1-nearest neighbor in the dissimilarity space (1nndc) and support vector machines (svc)[48] in the dissimilarity space.[8,35,36,41–43] All the classifiers are built with the PRTOOLS.[19]

Dissimilarity space has became a common way for representing graphs lately. First, a subset of graphs is selected as the prototype set with R objects. Next, all the dissimilarities between all graphs and the prototype set are computed. Now each graph is represented by its R dissimilarities. This representation is called the dissimilarity representation. The features of the graphs defined in this way are the distances to the other graphs but not intrinsic features that, for instance, encode the linkages between nodes within a graph.

Two real-world datasets, i.e. COIL-100 dataset[34] and Mutagenicity,[44] are used in the experiments. The COIL-100 dataset consists of images of 100 different objects. Images of each objects are taken at intervals of 5 degrees, resulting in 72 images per object or 72 images per class. However, in our experiments, we do not use all available images. Only 10 classes (10, 13, 18, 40, 44, 49, 51, 61, 65, 66) are used. These 10 classes of images are selected because they have the lowest one-against-all performance, i.e. are relatively difficult to classify compared to the other classes. Figure 4 shows an example image of each object from these 10 classes.

Three different datasets, i.e. coil-segment, coil-harris and coil-sift, of graphs are then generated from these 720 images. This is done as follows:

The coil-segment dataset is obtained by first converting the color images to grayscale,[a] applying a mean-shift algorithm to get homogeneous graylevel segments, and finally removing image segments that were smaller than 50 pixels. A structure graph was obtained by connecting the segments in the graph that also have some neighboring pixels in the image domain. Overall, each image has around 3−10 segments.

The coil-harris dataset is directly obtained from the coil-del dataset in the IAM graph database.[44] To derive graphs from images, the Harris corner detection algorithm[23] is used to extract corner features from images. Based on these corner points,

[a]The original color features are very informative, such that the classification performance was already so high that the structure could not add any additional information.

(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Fig. 4. An example image of class (a) 10; (b) 13; (c) 18; (d) 40; (e) 44; (f) 49; (g) 51; (h) 61; (i) 65; and (j) 66, respectively.

a Delaunay triangulation is applied. The result of the triangulation is then converted into a graph by representing lines by undirected edges and ending points of lines by nodes. Each node is labeled with a two dimensional attribute giving its position, while edges are unlabeled.

The coil-sift dataset detects the sift keypoints[31] instead of Harris points. Sift points are corner points with 128 dimensional descriptors, which are used as the feature vectors for graph nodes. The coil-sift uses the same procedure for transforming images into graphs as the coil-harris.

The difference among the coil-segment, coil-harris and coil-sift datasets is that the coil-segment has much simpler structure compared to coil-harris and coil-sift because usually only few segments can be extracted from an image while many corner points or keypoints can be detected. The coil-sift dataset has more complex features (128 in total) while coil-harris and coil-segment only have 2 and 4 dimensions, respectively.

Another real-world dataset Mutagenicity[44] is also used in the experiments. Mutagenicity refers to a property of chemical compounds. The molecules are converted into graphs in a straightforward manner by representing atoms as nodes and the covalent bonds as edges. Nodes are labeled with the corresponding chemical symbol. The average number of nodes of a graph is 30.3 ± 20.1, and the average number of edges is 30.7 ± 16.8. The Mutagenicity dataset is divided into two classes, i.e. mutagen and nonmutagen. There are 4337 objects in total (2401 mutagen and 1936 nonmutagen). However, we only take a subset of this dataset in the experiments by using only every 5th object, resulting in a subset of 867 objects.

In the experiments, 20% of images are randomly taken as the training dataset, and the rest 80% are taken as the testing dataset. All the experiments are repeated 100 times and the average classification error is recorded. The standard deviations of the results are very small due to 100 repetitions and are therefore ignored.

The results of the modified graph edit distance are presented in Figs. 5–8, the results of the modified shortest path kernel are presented in Figs. 9–12. In each

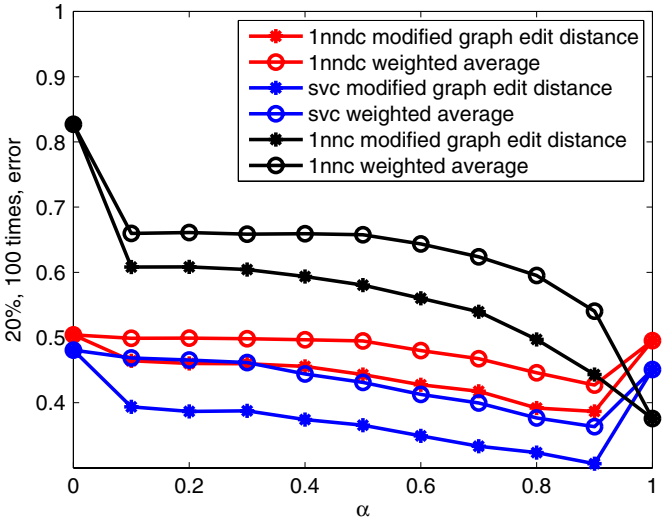


Fig. 5. Results of coil-segment for modified graph edit distance (color online).

figure, a comparison is made between the modified method (represented by solid lines with star symbols) and the weighted average of pure structure and pure feature information (represented by solid lines with circles). On the x-axis is the parameter α, when $\alpha = 0$, only structure information is used and when $\alpha = 1$, only feature information is used. Therefore, at $\alpha = 0$, the graph edit distance and shortest path kernel are applied on unlabeled graphs, producing a dissimilarity matrix D_0, whereas at $\alpha = 1$, the procedures are applied to two sets of feature vectors, resulting in

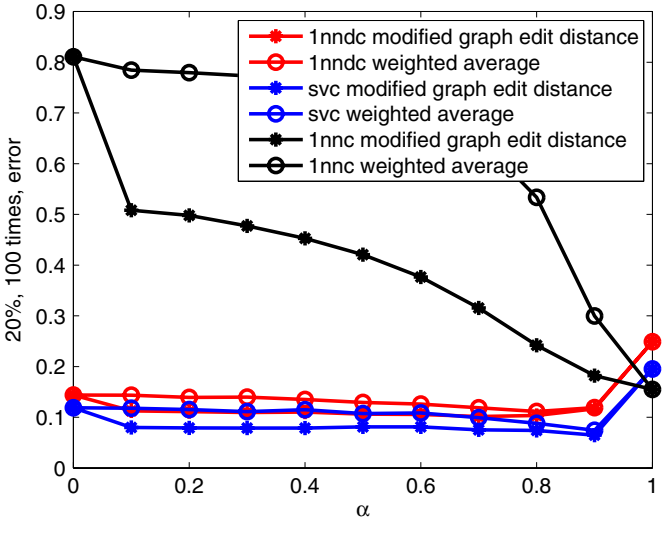


Fig. 6. Results of coil-harris for modified graph edit distance (color online).

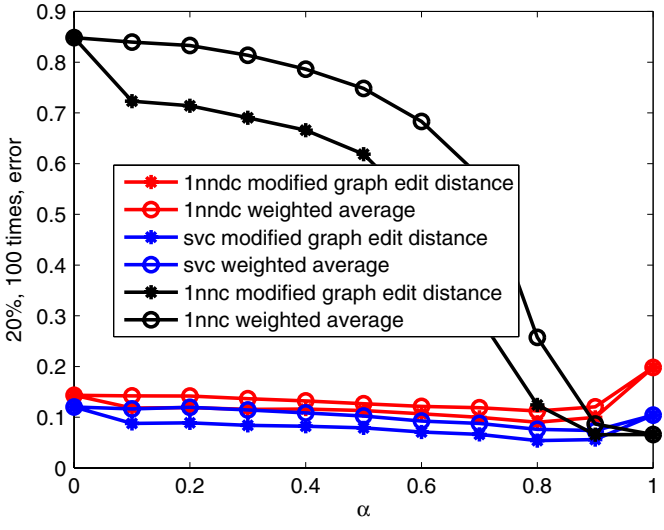


Fig. 7. Results of coil-sift for modified graph edit distance (color online).

dissimilarity matrix D_1. The weighted average matrix $D_W(\alpha)$ is then computed as follows:

$$D_W(\alpha) = \alpha D_1 + (1 - \alpha) D_0. \tag{11}$$

On the y-axis of the figures is the average classification error of three different classifiers: the nearest neighbor in the original space (black lines, also denoted by 1nnc), the nearest neighbor in the dissimilarity space (red lines, also denoted by

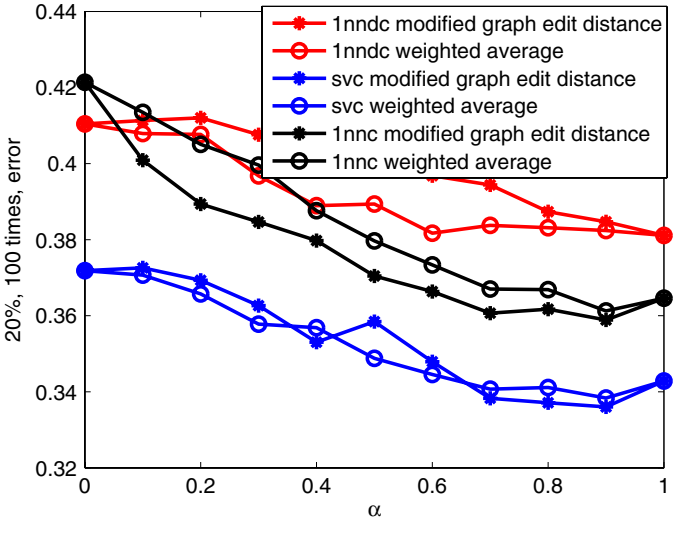


Fig. 8. Results of Mutagenicity for modified graph edit distance (color online).

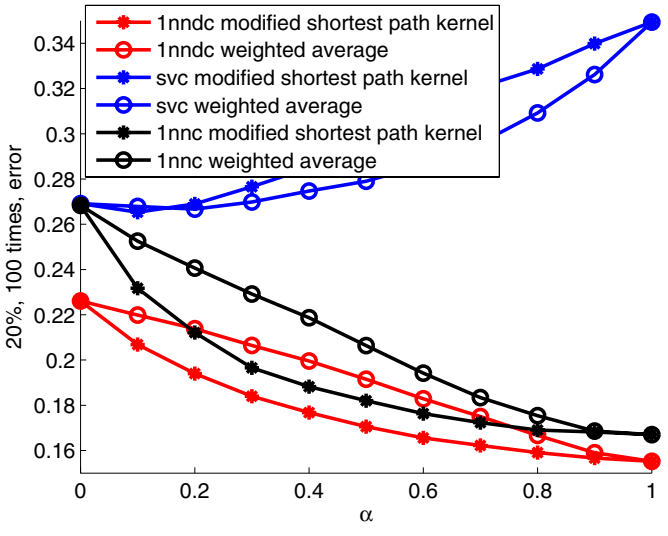


Fig. 9. Results of coil-segment for modified shortest path kernel (color online).

1nndc) and the support vector classifier (blue lines, also denoted by svc). Note that the graph edit distance procedures can only produce pairwise distances for graphs, and therefore only the nearest neighbor classifiers can be adopted in the original space. But in the dissimilarity space, the distances of one object to the other objects are taken as the feature values and therefore one can adopt any classical pattern recognition techniques.

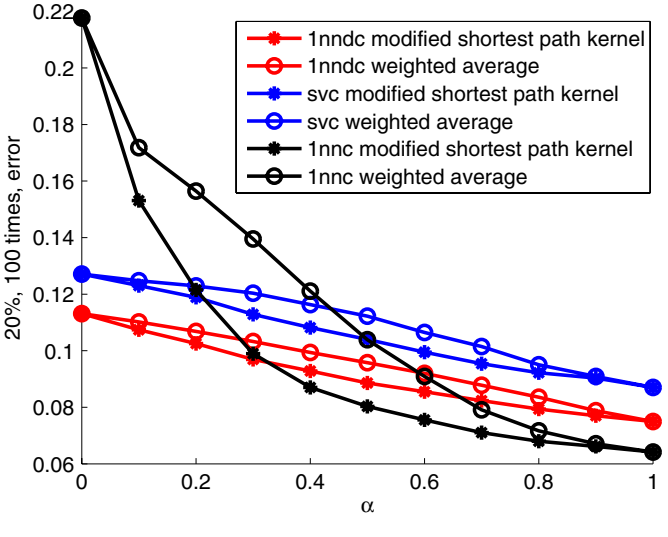


Fig. 10. Results of coil-harris for modified shortest path kernel (color online).

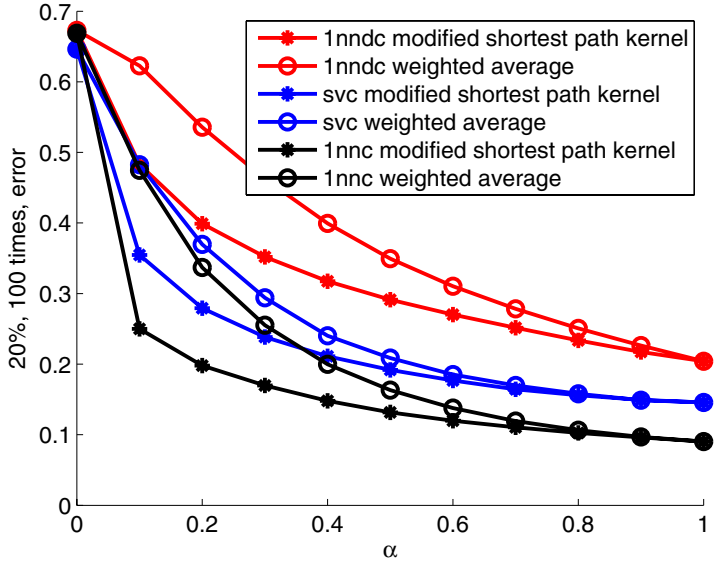


Fig. 11. Results of coil-sift for modified shortest path kernel (color online).

5.1. *Modified graph edit distance*

The coil-segment dataset is with simple structure and simple features, and therefore its performance of using only features or structures are both not good as in Fig. 5. However, if we combine these two sources of information, significant improvements

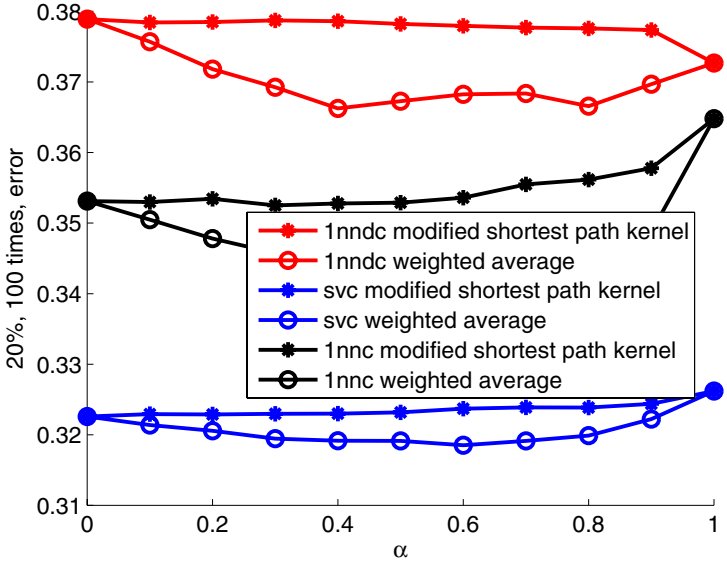


Fig. 12. Results of Mutagenicity for modified shortest path kernel (color online).

can be observed both with weighted average and modified graph edit distance in the dissimilarity space, especially when $\alpha = 0.9$. This suggests that not only the combination of feature and structure information can be helpful, but also changing the impacts of these two information could make a lot of difference, and it is necessary to balance these two information to obtain a better result. This phenomena can also be observed in Figs. 6 and 7, even though the structure information is doing very good already in these two datasets, and therefore the improvement of combining the two information is less significant as with the coil-segment dataset.

From Figs. 5−7, we can also see that the modified graph edit distance is better than weighted average in all cases, and therefore the dependency between the feature and structure information should also be taken into account when computing the graph distances. Also, the classifiers in the dissimilarity spaces perform much better than classifiers in the original space, and svc is better than 1nnc. The dissimilarity space therefore not only allows different classifiers to be applied on structure data, but also in general gives much better results than the original space. One very intriguing results from these figures is the significant improvement of the dissimilarity space over the original space when $\alpha = 0$ in the case of considering pure structure. For the coil-harris and coil-sift datasets, the improvement is even around 70%. This suggests that there are intrinsic features within the graph structure and these features can only be utilized by the classifiers in the dissimilarity space, but not in the original space.

For the Mutagenicity dataset, the results are given in Fig. 8. The structures of graphs are naturally given by the molecules, and the structure is more sparse (the number of edges is much fewer) than the ones of the coil-100 dataset. One of the reasons why there is no significant improvement of modified graph edit distance over weighted average on the performance could be that the sparse structures contains less intrinsic features. The sparse structures of graphs probably also introduce less dependency between structures and features, and therefore modified graph edit distance is not always better than weighted average. Nevertheless, we can still observe that the extra parameter α helps for finding better results. Also, svc is much better than 1nnc classifiers in both original space and dissimilarity space.

5.2. *Modified graph kernel*

The results of the modified shortest path kernel compared to weighted averaging are presented in Figs. 9−12 for the coil-segment, coil-harris, coil-sift and mutagenicity datasets, respectively.

Before running the classifiers on the matrices produced by the modified shortest path kernel, we normalized the kernel matrix such that the diagonal is equal to 1 (by $K(x, y) = \sqrt{(K(x, x) * K(y, y))}$) and converted the kernel matrix to a dissimilarity matrix using a procedure in DisTools.[18] This was done in order to be able to compare 1nnc (which needs distances) to 1nndc and svc (which could also work with a similarity matrix directly).

For the coil-segment, coil-harris and coil-sift datasets, the best results are achieved when only feature information is used. This is somewhat surprising, because it is expected that combining both information sources would be beneficial. A possible explanation could lie in the thresholds used for the bridge kernel on node values. The main point of the threshold is to convert a distance to a similarity value greater or equal to 0, so in principle it should be sufficient to set the threshold to the maximum possible distance between nodes or edges, this way preserving as much information as is contained in the distances. However, in our preliminary experiments, we noticed that choosing a lower threshold, thus setting the larger distances to 0, often had a positive effect on performance. We did not investigate which thresholds are best for a particular dataset, but attempted to choose reasonable default values. For the coil-segment and coil-harris datasets, the original node distances were very large, but after normalizing the features to zero mean and unit variance, a threshold of 2 was reasonable. This threshold was also used for the coil-sift dataset. It might be the case that due to these thresholds, the performances at $\alpha = 1$ are very good, which masks the fact that it is beneficial to use structure information.

Similarly to the results of the edit distance, we see that in general, at different values of α the kernel performs better than the weighted average. This is only not the case for svc on the coil-segment dataset, where svc performs much worse than the nearest neighbor classifiers. Apart from this result, the three classifiers seem to have comparable performance. This is very different from the edit distance results, where the classifiers in the dissimilarity space are clearly outperforming 1nnc.

For the mutagenecity dataset, we see quite a different picture than in the coil datasets. As each of the classifiers performs similarly using only feature or only structure information, we see that combining the two is advantageous in this case. However, for all three clasifiers the weighted averaging outperforms the shortest path kernel. This suggests that the way α is introduced in the modified kernel is not able to effectively combine the feature and structure similarities. This has an intuitive explanation. If we examine the inter-node distances in the dataset, we will notice that this distance is either 0 or 1, because the node features are discrete (presence or absence of chemical symbol). Therefore, applying α as we proposed does not change the node similarity of the path. As a result, the matrices produced by the kernel are very similar at different values of α, which is clearly reflected by the results. Weighted averaging does achieve different matrices, and therefore much more variable perfomances can be seen. A different usage of α in the modified kernel might therefore be advantageous.

6. Discussion and Conclusion

Combining different sources of information is expected to give better performance than single information sources. However, how to combine the information sources to achieve the most performance improvement is not always trivial. In this work, we

propose the modified graph edit distance and the modified shortest path graph kernel to adopt an extra parameter for controlling the impact of feature and structure representations on computing graph distances. From the experimental results, we can see improvements when the impact of the feature and structure representations is properly scaled.

Another intriguing observation is that classifiers in the dissimilarity space can outperform the nearest neighbor in original space if the graph dataset has intrinsic dimensions within the structure. Even though the measurement of the intrinsic dimensions is still not clear, it is still worth further investigation on the performances of dissimilarity space with respect to the intrinsic dimensions of datasets. Another point which might be worth investigating is the way the parameter which affects feature and structure representations is applied to the shortest path kernel or graph kernels in general.

Acknowledgment

We acknowledge financial support from the FET programme within EU FP7, under the SIMBAD project (contract 213250).

References

1. S. Andrews, T. Hofmann and I. Tsochantaridis, Multiple instance learning with generalized support vector machines, in *Proc. AAAI National Conf. Artificial Intelligence* (Alberta, Canada, 2002), pp. 943–944.
2. S. Andrews, I. Tsochantaridis and T. Hofmann, Support vector machines for multiple-instance learning, in *Proc. Conf. Advances in Neural Information Processing Systems* (2002), pp. 561–568.
3. M. Basu, H. Bunke and A. D. Bimbo (eds.), Syntactic and structural pattern recognition, Special Issue on *IEEE Trans. Pattern Anal. Mach. Intell.* **7**(27) (2005).
4. K. Borgwardt and H. Kriegel, Shortest-path kernels on graphs, in *Proc. of ICDM'05*, Houston, USA (IEEE Computer Society, 2005), pp. 74–81.
5. K. Borgwardt, C. Ong, S. Schönauer, S. Vishwanathan, A. Smola and H. Kriegel, Protein function prediction via graph kernels, *Bioinformatics* **21** (2005) i47–i56.
6. H. Bunke and G. Allermann, Inexact graph matching for structural pattern recognition, *Pattern Recogn. Lett.* **1** (1983) 245–253.
7. H. Bunke and T. Caelli (eds.), Graph matching in pattern recognition and machine vision, Special Issue on *Int. J. Pattern Recogn. Artif. Intell.* **18**(3) (Springer, 2004).
8. H. Bunke and K. Riesen, Graph classification based on dissimilarity space embedding, in *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, Vol. 5342 (Springer, 2008), pp. 996–1007.
9. H. Bunke and K. Riesen, Recent advances in graph-based pattern recognition with applications in document analysis, *Pattern Recogn.* **44**(5) (2011) 1057–1067.
10. H. Bunke and A. Sanfeliu, *Syntactic and Structural Pattern Recognition: Theory and Applications* (World Scientific, Singapore, 1990).
11. T. Caelli and S. Kosinov, An eigenspace projection clustering method for inexact graph matching, *IEEE Trans. Pattern Anal. Mach. Intell.* **26** (2004) 515–519.

12. Y. Chen, J. Bi and J. Wang, Miles: Multiple-instance learning via embedded instance selection, *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(12) (2006) 1931−1947.
13. F. Chung, *Spectral Graph Theory*, CBMS Regional Conference Series in Mathematics, Vol. 92 (AMS, USA, 1997).
14. T. Cover and P. Hart, Nearest neighbor pattern classification, *IEEE Trans. Inf. Theory* **13**(1) (1967) 21−27.
15. P. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach.* (Prentice-Hall, London, 1982).
16. T. Dietterich, R. Lathrop and T. Lozano-Pérez, Solving the multiple instance problem with axis-parallel rectangles, *Artif. Intell.* **89**(1−2) (1997) 31−41.
17. R. Duda, P. Hart and D. Stork, *Pattern Classification* (John Wiley and Sons, New York, 2001).
18. R. Duin and E. Pekalska, Dis-Tools a matlab toolbox for pattern recognition (2009), http://prtools.org.
19. R. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D. Tax and S. Verzakov, PR-Tools4.1, a matlab toolbox for pattern recognition (2007), http://prtools.org.
20. T. Gartner, Predictive graph mining with kernel methods, in *Advanced Methods for Knowledge Discovery from Complex Data* (Springer, 2005), pp. 95−121.
21. T. Gärtner, P. Flach, A. Kowwalczyk and A. Smola, Multi-instance kernels, in *Proc. 19th Int. Conf. Machine Learning*, eds. C. Sammut and A. Hoffmann (Morgan Kaufmann, 2002), pp. 179−186.
22. T. Gartner, P. Flach and S. Wrobel, On graph kernels: Hardness results and efficient alternatives, in *Proc. of COLT/Kernel'03*, Washington DC, USA (Springer-Verlag, 2003), pp. 129−143.
23. C. Harris and M. Stephens, A combined corner and edge detector, in *Proc. 4th Alvey Vision Conf.* (1988), pp. 147−151.
24. P. Hart, N. Nilson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybern.* **4**(2) (1968) 100−107.
25. T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning* (Springer-Verlag, New York, 2001).
26. T. Horváth, T. Gärtner and S. Wrobel, Cyclic pattern kernels for predictive graph mining, in *Proc. of KDD'04* (ACM, 2004), pp. 158−167.
27. A. Jain, R. Duin and J. Mao, Statistical pattern recognition: A review, *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1) (2000) 4−7.
28. A. Kandel, H. Bunke and M. Last, *Applied Graph Theory in Computer Vision and Pattern Recognition, Studies in Computational Intelligence*, Vol. 52 (Springer, 2007).
29. H. Kashima, K. Tsuda and A. Inokuchi, Marginalized kernels between labeled graphs, in *Proc. ICML'03*, Vol. 20 (2003), p. 321.
30. W. Lee and R. Duin, An inexact graph comparison approach in joint eigenspace, in *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, Vol. 5342 (2008), pp. 35−44.
31. D. G. Lowe, Distinctive image features from scale-invariant keypoints, *Int. J. Comput. Vis.* **60**(2) (2004) 91−110.
32. O. Maron and T. Lozano-Pérez, A framework for multiple-instance learning, in *Proc. Conf. Advances in Neural Information Processing Systems* (1998), pp. 570−576.
33. J. Munkres, Algorithm for the assignment and transportation problems, *J. Soc. Ind. Appl. Math.* **5** (1957) 32−38.
34. S. Nene, S. Nayar and H. Murase, Columbia object image liberary: Coil-100, Tech. rep., Department of Computer Science, Columbia University, New York (1996).

35. E. Pekalska and R. Duin, Dissimilarity representations allow for building good classifiers, *Pattern Recogn. Lett.* **23**(8) (2002) 943−956.
36. E. Pekalska and R. P. W. Duin, *The Dissimilarity Representation for Pattern Recognition. Fundations and Applications* (World Scientific, Singapore, 2005).
37. E. Pekalska, P. Paclik and R. Duin, A generalized kernel approach to dissimilarity-based classification, *J. Mach. Learn. Res.* **2** (2002) 175−211.
38. H. Qiu and E. Hancock, Clustering and embedding using commute times, *IEEE Trans. Patt. Anal. Mach. Intell.* **29** (2007) 1873−1890.
39. J. Ramon and T. Gärtner, Expressivity versus efficiency of graph kernels, *First International Workshop on Mining Graphs, Trees and Sequences*, Citeseer (2003), pp. 65−74.
40. K. Riesen and H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, *Image Vis. Comput.* **27**(7) (2009) 950−959.
41. K. Riesen and H. Bunke, Cluster ensembles based on vector space embeddings of graphs, in *MCS* (2009), pp. 211−221.
42. K. Riesen and H. Bunke, Feature ranking algorithms for improving classification of vector space embedded graphs, in *CAIP* (2009), pp. 377−384.
43. K. Riesen and H. Bunke, Graph classification based on vector space embedding, *Int. J. Patt. Recogn. Artif. Intell.* **23**(6) (2009) 1053−1081.
44. K. Riesen and H. Bunke, IAM graph database repository for graph based pattern recognition and machine learning, *Structural, Syntactic, and Statistical Pattern Recognition* (Springer, 2010), pp. 287−297.
45. K. Riesen, M. Neuhaus and H. Bunke, Bipartite graph matching for computing the edit distance of graphs, in *GbRPR* (2007), pp. 1−12.
46. K. Riesen, S. Fankhauser, H. Bunke and P. J. Dickinson, Efficient suboptimal graph isomorphism, in *GbRPR* (2009), pp. 124−133.
47. Q. Tao, S. Scott, N. Vinodchandran, T. Osugi and B. Mueller, Kernels for generalized multiple-instance learning, *IEEE Trans. Patt. Anal. Mach. Intell.* **30**(12) (2008) 2084−2098.
48. V. Vapnik, *The Nature of Statistical Learning Theory* (Springer-Verlag, New York, 1995).
49. P. Viola, J. Platt and C. Zhang, Multiple instance boosting for object detection, in *Advances in Neural Inf. Proc. Systems* (*NIPS 05*) (2005), pp. 1419−1426.
50. J. Wang and J. Zucker, Solving the multiple-instance problem: A lazy learning approach, in *Proc. Thirteenth Int. Conf. Machine Learning* (2000), pp. 1119−1126.
51. R. Wilson, B. Luo and E. Hancock, Pattern vectors from algebraic graph theory, *IEEE Trans. Patt. Anal. Mach. Intell.* **27** (2005) 1112−1124.

Wan-Jui Lee received her B.S. degree in Information and Computer Education from the Normal Taiwan University in 2000, and her Ph.D. from the Department of Electrical Engineering, National Sun Yat-Sen University, Taiwan, in 2008. She was a guest researcher from 2006 to 2007 in the Information and Communication Theory Group, Delft University of Technology, the Netherlands. From 2007 to 2011, she worked as a postdoctoral researcher in the Pattern Recognition Laboratory, Delft University of Technology. Her main research interests include pattern recognition, data mining and machine learning.

Veronika Cheplygina received her M.Sc. degree in Media and Knowledge Engineering from the Delft University of Technology, the Netherlands in 2010. She is currently working towards her Ph.D. at the Pattern Recognition Laboratory at the Delft University of Technology. Her research interests include multiple instance learning, dissimilarity representation and combining classifiers.

David M. J. Tax studied Physics at the University of Nijmegen, the Netherlands in 1996, and received his Masters degree with the thesis "Learning of structure by Many-take-all Neural Networks". After that he received his Ph.D. from the Delft University of Technology, the Netherlands in the Pattern Recognition group, under the supervision of Dr. Robert P.W. Duin. In 2001 he was promoted with the thesis 'One-class Classification'. After working for two years as a Marie Curie Fellow in the Intelligent Data Analysis group in Berlin, he is currently an Assistant Professor in the Pattern Recognition Laboratory at the Delft University of Technology. His main research interest is in the learning and development of detection algorithms and (one-class) classifiers that optimize alternative performance criteria like ordering criteria using the Area under the ROC curve or a Precision-Recall graph. Furthermore, the problems concerning the representation of data, multiple instance learning, simple and elegant classifiers and the fair evaluation of methods have focus.

Marco Loog received an M.Sc. degree in Mathematics from Utrecht University, the Netherlands, and in 2004 a Ph.D. from the Image Sciences Institute for the development and improvement of contextual statistical pattern recognition methods and their use in the processing and analysis of images. After this joyful event, he moved to Copenhagen, Denmark, where he became an assistant and, eventually, Associate Professor after which he worked as a research scientist at Nordic Bioscience. In 2008, after several splendid years in Denmark, Marco moved to Delft University of Technology, the Netherlands, where he now works as an Assistant Professor in the Pattern Recognition Laboratory. He is currently vice-chair of the Technical Committee 1 of the IAPR. Marco's ever-evolving research interests nowadays include multiscale image analysis, semi-supervised and multiple instance learning, saliency, computational perception, the dissimilarity approach, and black math.

Robert P. W. Duin received in 1978, his Ph.D. in Applied Physics from the Delft University of Technology, the Netherlands, for a thesis on statistical pattern recognition. He is currently with the Faculty of Electrical Engineering, Mathematics and Computer Science of the same university. From 1980−1990, he studied and developed hardware architectures and software configurations for interactive image analysis. After that he became involved with pattern recognition by neural networks. For many years he studied the design, evaluation, and application of algorithms that learn from examples, including neural network classifiers, support vector machines, classifier combining strategies, and one-class classifiers. From 2000 he started to investigate alternative object representations for classification and he thereby became interested in dissimilarity-based pattern recognition, trainable similarities. Currently, the significance of non-Euclidean representations for pattern recognition is his main research interest. Dr. Duin is an associate editor of Pattern Recognition Letters and a formar associate editor of the IEEE Transactions on Pattern Analysis and Machine Intelligence. He is a Fellow of the International Association for Pattern Recognition (IAPR). In August 2006 he was the recipient of the Pierre Devijver Award for his contributions to statistical pattern recognition.