Discriminative Prototype Selection Methods for Graph Embedding

Ehsan Zare Borzeshi[a,*], Massimo Piccardi[a,*], Kaspar Riesen[b], Horst Bunke[c]

[a] *School of Computing and Communications, Faculty of Engineering and Information Technology, University of Technology, Sydney (UTS), Ultimo, 2007 NSW, Australia*
[b] *Institute for Informations Systems, University of Applied Sciences and Arts Northwestern, Riggenbachstrasse 16, 4600 Olten , Switzerland*
[c] *Institute of Computer Science and Applied Mathematics, University of Bern, Neübrckstrasse 10, CH-3012 Bern, Switzerland*

Abstract

Graphs possess a strong representational power for many types of patterns. However, a main limitation in their use for pattern analysis derives from their difficult mathematical treatment. One way of circumventing this problem is that of transforming the graphs into a vector space by means of graph embedding. Such an embedding can be conveniently obtained by using a set of "prototype" graphs and a dissimilarity measure. However, when we apply this approach to a set of class-labelled graphs, it is challenging to select prototypes capturing both the salient structure within each class as well as inter-class separation. In this paper, we introduce a novel framework for selecting a set of prototypes from a labelled graph set taking their discriminative power into account. Experimental results showed that such a discriminative prototype selection framework can achieve superior results in classification compared to other well-established prototype selection ap-

*Corresponding author. Tel:+61 2 9514 7942. Fax: +61 2 9514 4535.
Email addresses: Ehsan.ZareBorzeshi@uts.edu.au (Ehsan Zare Borzeshi), Massimo.Piccardi@uts.edu.au (Massimo Piccardi), kaspar.riesen@fhnw.ch (Kaspar Riesen), Bunke@iam.unibe.ch (Horst Bunke)

proaches.

Keywords: Graph embedding, Discriminative prototype selection, Graph classification, Dissimilarity representation

---

1. Introduction

Although the vast majority of pattern recognition algorithms rely on vectorial data representations, more and more effort is now rendered in various research fields on graph based representations [1]. Unlike the vectorial representation which ignores the dependencies between observations, graphs preserve these dependencies and relations. To phrase it more generally, the main merits of a graph-based representation are: i) the number of nodes and edges in the graph is not fixed a priori; rather, it adjusts to the complexity of the target object, and ii) graphs are capable of encapsulating the object's structure not merely by storing the object's features, but by also explicitly modeling the relations amongst such features (beyond simple co-statistics).

Leveraging on these appealing properties, many approaches have used graphs, for instance, for human action recognition [2, 3], bioinformatics and chemoinformatics [4, 5, 6], web content analysis and data mining [7, 8, 9], classifying images from various fields [10, 11, 12], symbol and character recognition [13, 14, 15] and computer network analysis [16, 17].

However, object representations given in terms of graphs suffer from a number of severe drawbacks when compared to feature vectors. One major limitation is the significantly increased complexity of many algorithms. For instance, the comparison of two feature vectors for identity can be accomplished in linear time with respect to the length of the two vectors. For the equivalent operation on graphs,

i.e. testing two graphs for isomorphism, only exponential algorithms are known to date. Another major drawback of graph-based representations is that even basic mathematical operations such as sums and products cannot be performed on graphs, making them unsuited for conventional pattern recognition approaches. As a consequence of these general limitations, the lack of algorithmic tools for graph-based pattern recognition appears obvious.

One way of circumventing this problem is *graph embedding* in a real vector space. By this approach, we can benefit from the wide range of statistical pattern recognition methods while retaining the universality of graphs for pattern representation. Various approaches have been proposed in the literature to embed graphs in a vector space. In [11], for instance, features derived from the eigendecomposition of graphs are exploited. Another approach uses an "edit distance" to compute the matrix of distances between any two graphs in a set and then use it to embed the graphs into a vector space by means of multidimensional scaling [18]. In [19], the authors turn to the spectral decomposition of the Laplacian matrix. They show how the elements of the spectral matrix for the Laplacian can be used to construct symmetric polynomials. In order to encode a graph as a vector, the coefficient of these polynomials are used as graph features. Another approach for graph embedding has been proposed in [20]. The authors use the relationship between the LaplaceBeltrami operator and the graph Laplacian to embed a graph onto a Riemannian manifold.

The present paper follows another approach of graph embedding where a graph is embedded into a point in the vector space by means of a template set and a dissimilarity measure. This approach is primarily based on the idea proposed in [21, 22], where a dissimilarity representation over vectors was first introduced,

and then extended in [23] to map strings onto vector spaces and finally generalized to graphs in [24, 25]. The key idea of this graph embedding approach is to convert a graph into an n-dimensional feature vector by way of a set of "prototype" graphs P and a dissimilarity measure such that the feature vector consists of the dissimilarity between the graph and each prototype. Intuitively, the prototype set should be distributed over the graph domain in a uniform way. However, this is difficult to ensure in principle since uniformity over a graph domain is a vague concept.

Let us assume to be given a training set, $\mathbf{C}$, of class-labelled graphs from N different classes, $C_1, ..., C_N$. Various approaches have been proposed to date for selecting informative prototypes from $\mathbf{C}$. In [25], all available elements from the training set are used as prototypes $P = \mathbf{C}$, and then feature extraction algorithms, e.g. *principal component analysis* (PCA) [26], are applied to the embedded graphs in the vector space. Although by this approach the authors bypass the difficult problem of selecting adequate prototypes, it is obvious that it may prove computationally too expensive for large datasets and that its run-time cost on a new graph may be too high. To overcome this limitation, in [27], the authors proposed different heuristic approaches based on the distances between the graphs in $\mathbf{C}$. The authors distinguish between *unlabelled* and *labelled* selection. The unlabelled selection is executed over the whole training set at once ignoring the class labels, while the labelled selection selects prototypes separately for each of the N classes, $C_1, ..., C_N$. In general, labelled approaches have reported higher classification accuracy than unlabelled methods.

Labelled prototype selection can be likened to the training of class likelihoods in generative classifiers, where each likelihood is estimated based on only the

samples from that class. Conversely, discriminative classifiers, choose parameters based on the information from multiple classes at once, maximizing objective functions such as the class margin, Fisher discriminants, mutual information and others, and often proving more accurate than their generative counterparts. Inspired by discriminative approaches, in this paper we propose various *discriminative prototype selection* methods where the prototype set is chosen by weighing intra-class compactness and inter-class separation and demonstrate their ability to outperform previous methods. We have also recently become aware of another proposal for discriminative prototype selection from Raveaux *et al*. [28]. In [28], the authors propose to conduct the search for prototypes in the exponential space of possible selections by way of a genetic algorithm. While this is certainly an attractive strategy, we believe that the tradeoffs we propose between intra-class compactness and inter-class separation offer a more immediate interpretive framework.

The remainder of this paper is organized as follows: in the next section we define basic concepts and introduce our notation. The proposed approaches are then described in Section 3. In Section 4, we present an experimental evaluation of the proposed methods on a diverse range of publicly available datasets. Finally, we give concluding remarks and a discussion of possible extensions in Section 5.

2. Basic concepts and notations

This section briefly conveys the key terminology and some concepts used in this paper.

2.1. Graph

Different definitions for a graph can be found in the literature based on the considered applications. The following one provides a versatile definition of graph which is sufficiently flexible for a large variety of tasks.

Definition 1. A graph, g, is a four-tuple $g = (V, E, \alpha, \beta)$, where

- V is the finite set of vertices (or nodes),
- $E \subseteq (V \times V)$ is the set of edges,
- $\alpha : V \to L_V$ is the vertex labeling function, and
- $\beta : E \to L_E$ is the edge labeling function.

L_V and L_E are finite or infinite label sets of vertices and edges, respectively.

The labeling functions (α and β) in this definition are unconstrained, thus they can easily handle arbitrarily structured graphs. For instance, the vertices and edges of graph g can get labels from the set of integers $L = \{1, 2, ...\}$, the vector space $L = \mathbb{R}^n$, or a set of symbolic labels $L = \{\rho, \sigma, \kappa, ...\}$. Given that the vertices and/or the edges are labelled, these graphs are referred to as *attributed graphs*.

2.2. Graph Edit Distance

With a graph-based object representation, the concept of similarity in pattern recognition turns into that of graph (dis)similarity. Evaluating the dissimilarity of a pair of graphs is commonly referred to as *graph matching* (for an extensive review of graph matching techniques and application, the reader is referred to [1]). Graph matching measures the dissimilarity of arbitrarily structured and arbitrarily

labelled graphs and it is flexible thanks to its ability to cope with any kind of structural errors.

One of the most widely used methods for graph matching is the graph edit distance (GED) [29]. The main idea of the graph edit distance is that of finding the dissimilarity of two graphs by the minimum amount of distortion required to transform one graph into the other. The underlying distortion model is composed of six types of edit operations: insertion, deletion and substitution for both nodes and edges. A sequence of edit operations $(e_1, ..., e_K)$ that transforms g_1 into g_2 is called an edit path from g_1 to g_2. Figure 1 shows an example of an edit path between g_1 and g_2 consisting, in step order, of three edge deletions, one node deletion, one node insertion, two edge insertions, and two node substitutions.

Figure 1: An example edit path between g_1 and g_2 (node labels are represented by different shades of grays)

Based on the above definition, every graph can be transformed into another by applying a sequence of edit operations or edit path. Clearly, for every pair of graphs, there exists an infinite number of different edit paths. Thus, to select the best edit path between each pair of graphs, an edit cost function is introduced to assign a cost to each edit operation. Then, given a set of edit paths and an edit cost function, the dissimilarity of a pair of graphs is defined as the minimum-cost edit path in the set.

Definition 2. Let $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$ be a pair of

graphs in a set. The graph edit distance of such graphs is defined as:

$$d(g_1, g_2) = \min_{(e_1,\ldots,e_K)\in E(g_1,g_2)} \sum_{k=1}^{K} C(e_k) \quad (1)$$

where $E(g_1, g_2)$ denotes the set of edit paths between the two graphs, C denotes the edit cost function and e_k denotes the individual edit operation.

As it turns out, given a set of edit costs (which can be assigned heuristically or learned from a set of sample graphs [30, 31, 32]), the dissimilarity between each pair of arbitrarily structured and arbitrarily labelled graphs can be measured by means of the graph edit distance. Furthermore, a certain degree of robustness against various graph distortions can be expected.

Among various methods [29, 33, 34, 35], the bipartite approach proposed in [33] was chosen to compute the graph edit distance in this paper. This method is based on a fast bipartite optimization procedure mapping local substructures of one graph to local substructures of another graph. The main advantage of this method is that it is much less computationally demanding than other approaches which are based on combinatorial search procedures (e.g. [29]).

2.3. Graph Embedding Via Dissimilarity Measures

An embeddings performs an injective transformation of high-dimensional vectors or non-vectorial objects onto a vector space of suitable dimensionality. Graph embedding is a specific transformation for graphs. Its motivation is that of trying to take advantage of the rich space of statistical pattern recognition techniques, yet retaining the spatial representational power of graphs. This approach should not be confused with graph *node* embedding where each node of a graph is individually transformed into a point in vector space while attempting to preserve the nodes' mutual distances (e.g., [11]).

Definition 3. Let $G = \{g_1, g_2, ..., g_m\}$ be a set of graphs, $P = \{p_1, p_2, ..., p_n\}$ be a set of prototypes with $n < m$ (see details in subsection 3), and d be a dissimilarity measure. For graph embedding, dissimilarity measure d_{ji} of graph $g_j \in G$ to prototype $p_i \in P$ is computed. Then, an n-dimensional real vector $(d_{j1}, ..., d_{jn})$ is obtained by computing the n dissimilarities, $d_{ji} = d(g_j, p_i), \forall i$. Formally, the mapping $t^P : G \to \mathbb{R}^n$ is defined as the following function:

$$t^P(g) \to (d(g, p_1), ..., d(g, p_n)) \tag{2}$$

where $d(g, p_i)$ is a dissimilarity measure between graph g and prototype p_i.

Based on the above definition, this embedding approach can transform any graph g from an arbitrary graph set (e.g. a training, a validation or a test set of a classification problem) into a vector of real numbers. Moreover, this approach can flexibly use any dissimilarity measure on graphs, including an edit distance. A comprehensive review of the graph embedding by means of dissimilarity measures can be found in [21, 24].

3. Prototype selection

Selecting informative *prototypes* from the underlying graph domain plays a vital role in graph embedding [36]. In order to obtain a meaningful as well as class-discriminative vector representation in the embedding space, a set of selected prototypes $P = \{p_1, p_2, ..., p_n\}$ should be adequately distributed over the whole graph domain, at the same time avoiding redundancies in terms of selection of similar graphs [21, 24, 37].

Let us assume that the graphs in the graph domain can be classified into N different classes, $c_1, ..., c_N$. Given a labelled training set, $\mathbf{C}$, we note as $C_1, ..., C_N$

the N subsets spanned by the classes, such that $\mathbf{C} = \bigcup_{n=1}^{N} C_n$. We then categorise the prototype selection methods into labelled and unlabelled approaches. In the former, the selection is performed individually for each class, while the latter determines all prototypes from the whole training set, $\mathbf{C}$, ignoring the class label information. As shown in [27], labelled selection approaches tend to deliver higher classification accuracy than corresponding unlabelled approaches. Yet, labelled selection methods choose the class' prototypes based solely on the graphs in the class. This is in a way similar to generative classifiers, where a class' model is learned based on only the samples from that class. Conversely, discriminative classifiers use information from samples from multiple classes jointly and have been in the spotlight thanks to their reported accuracy (e.g. [38], [39]). Based on a similar rationale, we propose *discriminative* approaches for the selection of prototypes which maximize a function of the inter- and intra-class distances. In this way, we elicit prototype selection strategies imposing that the selected prototypes for the class be well-distributed within the class, yet being discriminative with respect to the graphs in the remaining classes.

3.1. Learning discriminative prototypes

The ultimate goal of prototype selection for classification is to identify the most discriminative graphs in the training set , $\mathbf{C}$. In this paper, similar to [40], each of the selection algorithms in section 3.2 is learned in two different ways. If the prototypes are chosen for a class to discriminate well against all other classes, they form a *one-vs-all* prototype set. Similarly, if the selection strategy tries to maximize this discrimination between the selected class and the closest class, it obtains a *one-vs-nearest* prototype set.

Definition 4. Let $C_n = \{g_{n1}, .., g_{ni}, .., g_{n|C_n|}\}$ and $C_m = \{g_{m1}, .., g_{mj}, .., g_{m|C_m|}\}$ be the subsets of **C** for classes c_n and c_m, respectively. We adopt the following definition for the class distance between c_n and c_m:

$$d_{class}(c_n, c_m) = \frac{\sum_{i=1}^{|C_n|} \sum_{j=1}^{|C_m|} d(g_{ni}, g_{mj})}{|C_n||C_m|} \tag{3}$$

where $d(u, v)$ is the distance between graphs u, v and $|C_n|$ and $|C_m|$ are the total number of graphs in C_n and C_m, respectively. Alternative definitions are also possible.

Definition 5. Based on Definition 4, the nearest class $c_{n_{near}}$ to class c_n is the class which has the minimum class distance to c_n, formally defined as:

$$c_{n_{near}} = \underset{\overline{n}=1...N, \overline{n} \neq n}{\operatorname{argmin}} d_{class}(c_n, c_{\overline{n}}) \tag{4}$$

3.2. Discriminative prototype selection algorithms

As stated in section 2.3, an appropriate choice of the prototype set, P, plays a critical role in graph embedding as it impacts the classification accuracy. The six deterministic algorithms used to select the discriminative prototypes in this paper are described below. In the selection of these discriminative prototypes, different objective functions are proposed which not only provide high intra-class compactness, but also consider inter-class separation. The part influencing the intra-class compactness is weighted by a weight, W_c, and the part controlling the inter-class separation is weighted by W_s where $\{W_c, W_s\} \in [0, 1]$ and $W_c + W_s = 1$. Each of these algorithms is an extension of an existing labeled algorithm. All these objective functions allow selecting an arbitrary number, K, of prototypes from each class.

3.2.1. Discriminative Center Prototype Selection

In *discriminative center prototype selection* (d-cps), a prototype set, $P_n = P_{n(1:K)} = \{p_{n1}, ..., p_{nk}, ..., p_{nK}\}$, is generated from each C_n subset, with each p_{nk} prototype simultaneously located near the "center" of the graphs from C_n, and away from the graphs of the remaining classes, $\overline{C_n}$. Prototypes are selected incrementally, with each prototype p_{nk} determined as a graph $g_{nj} \in C_n$ which is not already selected as prototype and such that the difference between the sum of distances between g_{nj} and all other graphs in its class, excluding the already selected prototypes, and the sum of distances between g_{nj} and all other graphs in $\overline{C_n}$ is minimal:

$$p_{nk} = \underset{g_{nj} \in C_n, g_{nj} \notin P_{n(1:k-1)}}{\operatorname{argmin}} \left[W_c \cdot \sum_{\substack{g_{ni} \in C_n, i \neq j, \\ g_{ni} \notin P_{n(1:k-1)}}} d(g_{nj}, g_{ni}) - W_s \cdot \sum_{g_{\overline{n}i} \in \overline{C_n}} d(g_{nj}, g_{\overline{n}i}) \right] \tag{5}$$

This objective function promotes class discrimination. However, it may suffer from redundancy as it tends to select multiple prototypes from the center of the class. Moreover, it should be noted that because the number of graphs in C_n is usually much lower than that in $\overline{C_n}$, the objective function in (5) usually takes negative values. However, this has no impact on the minimisation.

3.2.2. Discriminative Border Prototype Selection

The idea of *discriminative border prototype selection* (d-bps) is to choose the prototype set, P_n, such that each p_{nk} prototype be situated near the border of its class. The rationale for this selection is that of having prototypes which are simultaneously mutually spread apart and distant from the graphs in the other classes.

Prototypes are selected incrementally, with each prototype p_{nk} determined as a graph $g_{nj} \in C_n$ which is not already selected as prototypes and such that the total sum of the sum of distances between g_{nj} and all other graphs in C_n, excluding the already selected prototype, and $\overline{C_n}$ is maximal:

$$p_{nk} = \underset{g_{nj} \in C_n, g_{nj} \notin P_{n(1:k-1)}}{\operatorname{argmax}} \left[W_c \cdot \sum_{\substack{g_{ni} \in C_n, i \neq j, \\ g_{ni} \notin P_{n(1:k-1)}}} d(g_{nj}, g_{ni}) + W_s \cdot \sum_{g_{\overline{n}i} \in \overline{C_n}} d(g_{nj}, g_{\overline{n}i}) \right] \tag{6}$$

In contrast to the previous prototype selector, where many prototypes could be structurally similar, this selection procedure prevents redundancy. However, it lacks prototypes from the inner region of the class and this may lead to poorly discriminative embedded vectors for graphs located in such regions.

3.2.3. Discriminative Repelling Prototype Selection

In order to overcome the inherent limitations of both previous approaches, *discriminative repelling prototype selection* (d-rps) chooses the set of prototypes of each C_n subset based on the following procedure: the first prototype, p_{n1}, is selected by means of d-cps. To select any additional prototype, $p_{nk}, k = 2, ...K$, we pick a graph g_{nj} from the class' graphs not already selected as prototypes so as to minimize the following equation:

$$
\begin{aligned}
p_{nk} = & \underset{g_{nj} \in C_n, g_{nj} \notin P_{n(1:k-1)}}{\operatorname{argmin}} \\
& [\, W_c \cdot \sum_{\substack{g_{ni} \in C_n, i \neq j, \\ g_{ni} \notin P_{n(1:k-1)}}} d(g_{nj}, g_{ni}) \; - \\
& W_s \cdot (\sum_{g_{\overline{n}i} \in \overline{C_n}} d(g_{nj}, g_{\overline{n}i}) \cdot \sum_{g_{ni} \in P_{n(1:k-1)}} d(g_{nj}, g_{ni}))\,]
\end{aligned}
\tag{7}
$$

This objective function is similar to that in (5), but encourages p_{nk} to be also distant from all previously selected prototypes, $P_{n(1:k-1)} = \{p_{n1}, ..., p_{n(k-1)}\}$ ("repelling" component). This favors mutual separation amongst the class' prototypes and their more uniform distribution within the class.

3.2.4. Discriminative Spanning Prototype Selection

Along a similar rationale, *discriminative spanning prototype selection* (d-sps) selects each prototype with the following iterative procedure: the first prototype, p_{n1}, is selected by d-cps. Each additional prototype, $p_{nk}, k = 2, ...K$, is the graph in C_n that preserves the following conditions: be the farthest graph from the already selected prototypes, $P_{n(1:k-1)}$, as well as all graphs in the other classes, $\overline{C_n}$:

$$
\begin{aligned}
p_{nk} = & \underset{g_{nj} \in C_n, g_{nj} \notin P_{n(1:k-1)}}{\operatorname{argmax}} \\
& [W_c \cdot \sum_{g_{ni} \in P_{n(1:k-1)}} d(g_{nj}, g_{ni}) + W_s \cdot \sum_{g_{\overline{n}i} \in \overline{C_n}} d(g_{nj}, g_{\overline{n}i})]
\end{aligned}
\tag{8}
$$

Compared to (7), this objective function ignores the compactness term and composes the other two terms in an additive rather than multiplicative scale.

3.2.5. Discriminative Targetsphere Prototype Selection

In *discriminative targetsphere prototype selection* (d-tps), the first and second prototypes, $\{p_{n1}$,$p_{n2}\}$, for C_n are selected by means of d-cps and d-bps, respectively. These two prototypes represent the center and farthest boundary of the class. Then, the distance between these two prototypes, $d_{max} = d(p_{n1}, p_{n2})$, is computed and each other prototype, $p_{nk}, k = 3, ...K$, is selected as the graph closest to a distance of $(k-2)d_{max}/(K-1)$ from p_{n1} and furthest away from the graphs in the other classes, $\overline{C_n}$. This procedure is called "targetsphere" as it is reminiscent of the evenly-spaced divisions of a shooting target circle:

$$p_{nk} = \underset{g_{nj} \in C_n, g_{nj} \notin P_{n(1:k-1)}}{\operatorname{argmin}} [W_c \cdot \left| d(g_{nj}, p_{n1}) - (k-2) \cdot \frac{d_{max}}{(K-1)} \right| - W_s \cdot \sum_{g_{\overline{n}i} \in \overline{C_n}} d(g_{nj}, g_{\overline{n}i})] \tag{9}$$

3.2.6. Discriminative k-Center Prototype Selection

They key idea of *discriminative k-center prototype selection* (d-$k$cps) is to select the prototypes of each class by a procedure similar to k-medoids clustering, at the same time maintaining separation from the graphs in the remaining classes, $\overline{C_n}$ [41]. The six steps of this method are:

1. Select an initial set of K prototypes, $P_{n_{initial}} = \{p_{n1}, .., p_{nk}, .., p_{nK}\}$, by means of any of the previous prototype selectors.
2. Construct K sets, with each set containing one of the initial prototypes: $S_1 = \{p_{n1}\}, .., S_k = \{p_{nk}\}.., S_K = \{p_{nK}\}$.
3. For each other graph $g \in C_n$, $g \notin P_{n_{initial}}$, find its nearest prototype in terms of a distance between elements and add g to the corresponding set.

As a result of this, we attain a partition on C_n with K disjoint subsets and $C_n = \bigcup_{k=1}^{K} S_k$.

4. For each set S_k, find its center graph c_k by means of d-cps. This retains the discriminative aspect of the selection.
5. For each set S_k, if its center c_k is not equal to prototype p_{nk}, replace p_{nk} by c_k.
6. If any replacement has occurred, return to step 2; otherwise, select the centers of the K disjoint sets, $\{c_1, ..., c_K\}$, as the set of prototypes, P_n.

4. Experiments

This section provides the experimental evaluation of the proposed methods and shows the benefits of using discriminative prototype selection approaches compared to existing methods. To this aim, several classification tasks are carried out over a wide number and variety of datasets including letters, digits, drawings, fingerprints, HTML webpages, molecular compounds, and proteins.

4.1. Datasets

For extensive testing of the proposed approaches, we have chosen a total of 10 different graph datasets from the publicly available IAM graph database repository [42]. These datasets are significantly varied in nature and differ in terms of number of classes, number of graphs per classes and typical size of graphs (Table 1).

The first three datasets are the *Letter* datasets, which represent distorted letter drawings. Starting from a manually constructed prototype of every of the 15 Roman alphabet letters that consist of straight lines only, different degrees of distortion are applied: low, medium and high. Each ending point of a line is represented

by a node of the graph and labelled with its (x,y) coordinates. The edges represent the existing lines in the letters linking the corresponding nodes. The *Digit* dataset contains graphs representing different handwritten digits [43]. The digits were originally acquired by recording the pen position at constant steps of time. The sequences of (x,y) coordinates compose the set of nodes of the graphs and their corresponding labels, while the edges are given by the links between consecutive nodes.

The *GREC* dataset consists of graphs representing symbols from 22 classes of architectural and electronic drawing under different levels of noise [44]. Depending on the level of noise, different morphological operations are applied to the symbols until lines of one pixel width are obtained. Intersections and corners of such lines constitute the set of nodes, which are labelled with a two-dimensional attribute encoding their position.

The next set of graphs is the *Fingerprint* dataset. It consists of graphs that are obtained from a subset of the NIST-4 fingerprint image database [45] by means of particular image processing operations. Ending point and bifurcations of the skeleton of the processed images constitute the (x,y) attributed nodes of the graphs, plus some nodes that are inserted between these points. All points connected through a ridge in the image skeleton are connected with an unlabelled edge. Unlike the previous datasets, this dataset is not balanced in the number of samples per class.

The *AIDS* dataset represents molecules [46]. Each molecule is converted into a graph in a straightforward manner by representing its atoms as nodes and the covalent bonds as edges. Nodes are labeled with the number of the corresponding chemical symbol and edges by the valence of the linkage. While this dataset is

not visual and does not suffer from acquisition noise, it has a larger average graph size and is not balanced across classes.

Similar to the AIDS dataset, the *Mutagenicity* dataset follows the same approach of converting molecular compounds into attributed graphs [47]. However, the average graph size is even larger, and typical recognition rates are the lowest amongst all the 10 datasets. An equally challenging dataset is the *Protein* dataset, consisting of graphs representing proteins [5]. The graphs are constructed from the Protein Data Bank [48] and labeled with their corresponding enzyme class from the BRENDA enzyme database [49]. The proteins are converted into graphs by representing the structure, the sequence, and chemical properties of a protein by nodes and edges.

Finally, the *Webpage* dataset contains graphs representing various web documents from [7]. Each node represents a word in the web document and is attributed with the word itself and its frequency in the document. A directed edge connects a pair of consecutive words in the documents and is labeled by the corresponding word section label. This dataset has the largest average graph size (over a hundred nodes and a hundred edges per graph) amongst the selected datasets.

For the sake of accuracy evaluation, each of these datasets has been divided into three disjoint subsets which have been used for training, validation and testing. A summary of the datasets is reported in Table 1 showing the subsets' size, number of classes, average and maximum number of nodes and edges, and the balanced/unbalanced attribute.

4.2. Experimental setup and results

The aim of the experimental evaluation described in this section is to empirically verify the power and applicability of the feature vectors extracted by

Dataset	Size(tr, va, te)	# Cls	$\varnothing\,\lvert V\rvert$	$\varnothing\,\lvert E\rvert$	max$\lvert V\rvert$	max$\lvert E\rvert$	Balanced
Letter low	750, 750, 750	15	4.7	3.1	8	6	Y
Letter medium	750, 750, 750	15	4.7	3.2	9	7	Y
Letter high	750, 750, 750	15	4.7	4.5	9	9	Y
Digit	1000, 500, 2000	10	8.9	7.9	17	16	Y
GREC	836, 836, 1628	22	11.5	12.2	25	30	Y
Fingerprint	500, 300, 2000	4	5.4	4.4	26	24	N
AIDS	250, 250, 1500	2	15.7	16.2	95	103	N
Mutagenicity	500, 500, 1000	2	30.3	30.8	417	112	Y
Protein	200, 200, 200	6	32.6	62.1	126	149	Y
Webpage	780, 780, 780	20	186.1	104.6	834	596	N

Table 1: Summary of graph data set characteristics, e.g. the size of the training (tr), the validation (va) and the test set (te), the number of classes (# Cls), the average and max number of nodes and edges ($\varnothing\,\lvert V\rvert$, max$\lvert V\rvert$, $\varnothing\,\lvert E\rvert$, max$\lvert E\rvert$), and whether the graphs are uniformly distributed over the classes or not (balanced).

the discriminative prototype selection approaches compared to those obtained by other methods, e.g. [27]. For the sake of comparison, the following settings are identically applied in all experiments.

For graph embedding, the graph edit distance is computed by means of the suboptimal algorithms introduced in [33]. This approach shows superior performance in time and accuracy compared to other suboptimal algorithms. The classification task of the vector space embedded graphs is carried out by employing the support vector machine, or SVM for short [50]. Although any other statistical classifier could be used for this purpose, SVM enjoys a theoretical characterization as well as a remarkable empirical performance [51]. In our experiments, we make use of an SVM with the Radial Basis Function (RBF) kernel [52]. In [24], this kernel was reported more accurate than *linear* and *polynomial* kernels for classifying graphs embedded in a vector space. The number of prototypes per

class, k, and the SVM parameters are tuned using a training and a validation set, and the accuracy on the test set is then measured "blindly" by using the parameters selected on the validation set, without any further tuning. All our experiments were performed on a personal computer with an Intel(R) Core(TM)2 Duo CPU (E8500, 3.16GHz) and 4GB RAM using Matlab R2009b. As software, we have used the LIBSVM toolbox for Matlab [53].

4.2.1. Comparison between the discriminative and labeled approaches

In order to assess the individual contribution of the two learning approaches described in section 3.1, we have first conducted experiments with feature vectors extracted with different prototype selection methods, learned with one-vs-all and one-vs-nearest approaches (Table 2). All results for each dataset are then compared and the best accuracy per dataset is displayed in bold face. According to Table 2, 16 out of the top 27 prototype selectors were obtained with the one-vs-all approach rather than the one-vs-nearest. In most cases, the differences are very limited.

In the literature, graph embedding by means of prototype selection has reported higher classification accuracy than alternative methods such as K-NN classification directly in the graph domain and SVM classification over similarity kernels [54, 25]. Moreover, labeled prototype selectors have reported higher classification accuracy compared to unlabeled approaches [27]. Thus, we limit our comparison to the proposed discriminative approaches, existing labeled prototype selectors and, as a term of reference/baseline approach, using all the graphs in the training set as prototypes (Table 4).

We first evaluate the discriminative selection approaches (d-cps, d-bps, d-sps, d-tps and d-kcps) in comparison with corresponding labeled prototype selectors

	One-Vs-All						One-Vs-Nearest					
Dataset	d-cps	d-bps	d-drps	d-sps	d-tps	d-kcps	d-cps	d-bps	d-rps	d-sps	d-tps	d-kcps
Letter low	99.5	99.5	99.5	99.5	99.5	99.5	99.4	**99.6**	99.4	99.5	99.5	99.4
Letter medium	94.4	**95.6**	94.0	95.4	95.4	95.4	95.4	95.4	95.2	95.2	95.4	95.0
Letter high	92.2	92.8	93.0	**93.4**	93.0	92.8	92.6	92.3	91.8	92.7	92.3	92.8
Digit	98.6	98.6	98.6	**98.7**	98.5	98.6	98.5	98.6	98.4	**98.7**	98.6	98.6
GREC	92.0	92.0	92.0	**92.5**	92.0	92.2	91.9	92.1	92.1	92.1	92.2	92.2
Fingerprint	81.2	80.6	81.1	**81.6**	80.9	81.4	81.2	81.0	**81.6**	**81.6**	80.8	81.5
AIDS	98.0	98.0	98.0	**98.2**	**98.2**	98.1	98.0	98.0	98.0	**98.2**	**98.2**	98.1
Mutagenicity	71.1	71.1	69.9	**71.5**	71.1	70.6	71.1	71.1	69.9	**71.5**	71.1	70.6
Protein	**75.0**	72.0	72.0	73.0	**75.0**	61.0	**75.0**	72.0	72.0	73.0	**75.0**	62.0
Webpage	**82.4**	**82.4**	**82.4**	**82.4**	**82.4**	**82.4**	82.3	82.3	82.3	82.3	**82.4**	**82.4**

Table 2: Classification accuracy (%) of SVM-RBF applied to graphs embedded using different learning approaches (One-Vs-All and One-Vs-Nearest). The best result per dataset is displayed in bold face.

(l-cps, l-bps, l-sps, l-tps and l-kcps) [27, 54]. These labeled prototype selectors are defined as 3.2.1-6 with weights $W_c = 1$ and $W_s = 0$. In other word, each of the labeled approaches is equivalent to the corresponding discriminative approach without the inter-class term in its objective function. Table 3 shows that the discriminative selection strategy has increased the classification accuracies in 42 out of 50 cases over all datasets.

Next, we report the full accuracy over the various selectors and datasets in Table 4 (best values are highlighted in bold face). We observe that there is only one dataset (AIDS) where the classification accuracy with the best labeled approach is as high as that of the best discriminative approach. For all other datasets, using a discriminative approach significantly outperforms all labeled methods. Moreover, narrowing our comparison to the discriminative prototype selectors alone, we note that d-sps generally outperforms the other methods and achieves the best accuracy

Dataset	cps	bps	sps	tps	kcps
Letter low	+0.4	+0.4	0.0	+0.1	+7.3
Letter medium	+0.6	+0.8	+1.0	+0.8	+1.2
Letter high	+0.4	+0.4	+0.8	+0.8	+0.8
Digit	+0.3	+0.1	+0.1	+0.2	+0.1
GREC	+0.8	+0.4	+0.1	+0.8	-0.2
Fingerprint	+0.2	+0.6	+0.8	+1.2	+1.4
AIDS	+0.9	-0.1	0.0	+0.2	0.0
Mutagenicity	+5.1	-0.1	+1.9	+2.8	+0.5
Protein	+4.5	-0.5	0.0	+3.5	+1.5
Webpage	+0.1	+0.1	+0.1	+0.1	+0.1

Table 3: Increment of classification accuracy (%) with discriminative prototype selectors

		Labeled Prototype Selectors					Discriminative Prototype Selectors					
Dataset	All	l-cps	l-bps	l-sps	l-tps	l-kcps	d-cps	d-bps	d-rps	d-sps	d-tps	d-kcps
Letter low	99.4	99.1	99.2	99.5	99.4	99.2	99.5	**99.6**	99.5	99.5	99.5	99.5
Letter medium	95.0	94.8	94.8	94.4	94.6	94.2	95.4	**95.6**	95.2	95.4	95.4	95.4
Letter high	92.3	92.2	92.4	92.6	92.2	92.0	92.6	92.8	93.0	**93.4**	93.0	92.8
Digit	98.4	98.3	98.5	98.6	98.4	98.5	98.6	98.6	98.6	**98.7**	98.6	98.6
GREC	92.1	91.2	91.7	92.4	91.4	92.4	92.0	92.1	92.1	**92.5**	92.2	92.2
Fingerprint	81.0	81.0	80.4	80.8	79.7	80.1	81.2	81.0	**81.6**	**81.6**	80.9	81.5
AIDS	**98.2**	97.1	98.1	**98.2**	98.0	98.1	98.0	98.0	98.0	**98.2**	**98.2**	98.1
Mutagenicity	67.6	66.0	71.2	69.6	68.3	70.1	71.1	71.1	69.9	**71.5**	71.1	70.6
Protein	73.0	70.5	72.5	73.0	71.5	60.5	**75.0**	72.0	72.0	73.0	**75.0**	62.0
Webpage	82.3	82.3	82.3	82.3	82.3	82.3	**82.4**	**82.4**	**82.4**	**82.4**	**82.4**	**82.4**

Table 4: Classification accuracy (%) of SVM-RBF applied to graphs embedded using all the graphs in the training set (All), the labeled prototype selectors and the discriminative prototype selectors. The best result per dataset is displayed in bold face.

in 8 out of 10 datasets.

Studying the optimal number of prototypes, i.e. the dimensionality of the embedding vector space, is also of interest. For the results reported in Tables 4 and 2, we have set an equal number of prototypes for each class with the balanced datasets and proportionally equal with the unbalanced datasets in all experiments. Then, we have followed the usual training-validation-test protocol to identify the optimal number of prototypes. In addition, Figure 2 reports the classification accuracy on the test set as a function of the number of selected prototypes per class. The discriminative approaches almost invariably achieve better accuracy than their labeled counterparts at a parity of number of prototypes, or the same accuracy with fewer. Thus, from a computational point of view, the proposed selection strategy is also preferable to a labeled strategy as it requires a smaller number of prototypes to deliver equivalent accuracy. This translates into fewer graph edit distances to be computed for transforming each graph, with shorter training and run-time computational times.

Considering the aforementioned definitions and explanations, the labeled approaches are the same as the discriminative ones but with $W_c = 1$ and $W_s = 0$. Thus, studying the optimal value of weights as well as their influence on the classification accuracy is also of interest. In our experiments, a grid search is used to optimize the weights. Based on the definition $W_c + W_s = 1$, thus W_s is the only free parameter, making a grid search easily feasible (the values explored range between 0.01 and 1 in 0.01 step). Table 5 shows the W_s value in correspondence with the accuracies reported in Table 2. Furthermore, Figure 3 presents the classification accuracy on the test set as a function of the value of W_s for two exemplary cases. Figure 3.a shows a desirable case where the cross-validation accuracy is highly insensitive to the tuning of the W_s parameter. Figure 3.b shows

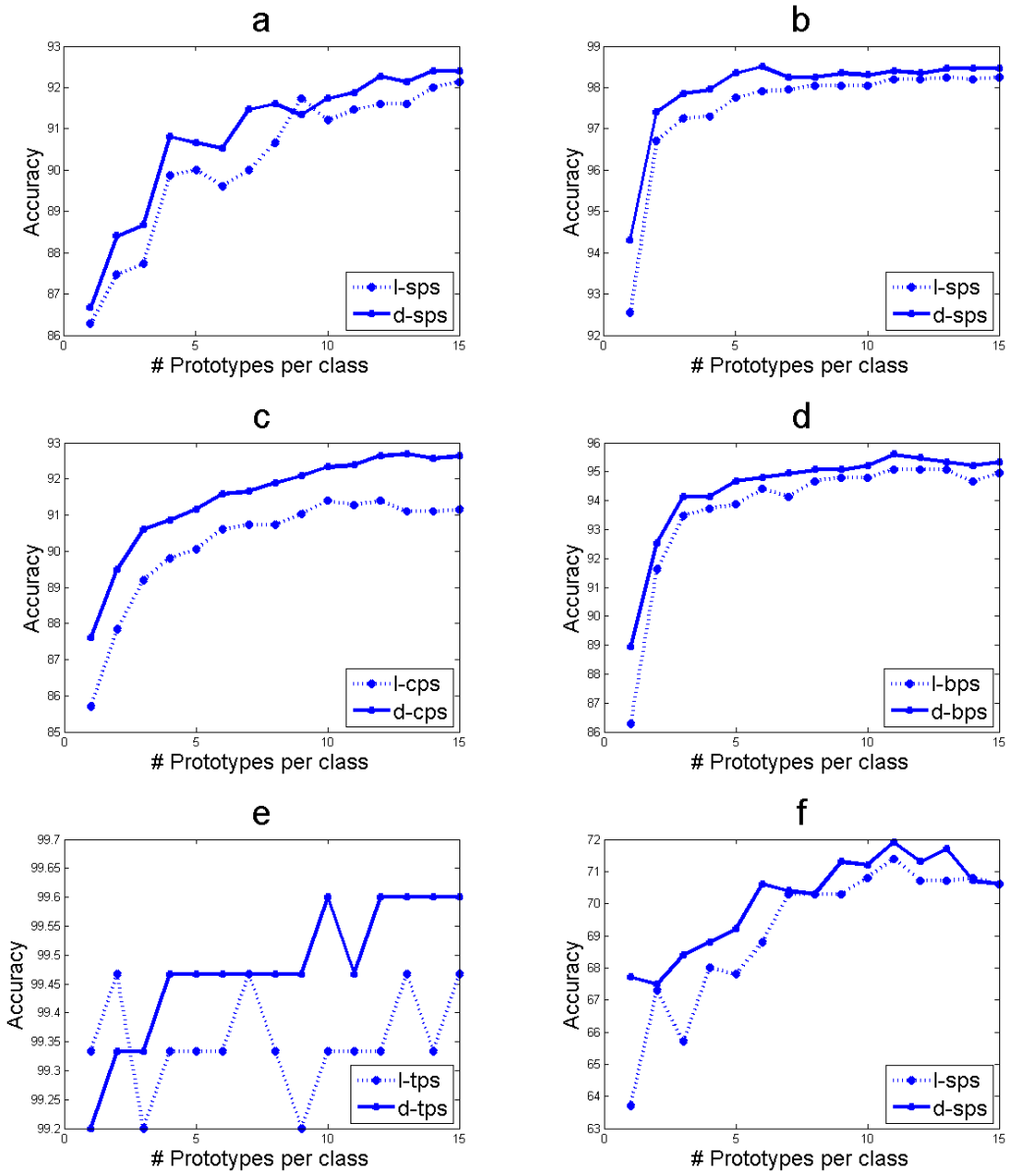

Figure 2: Accuracy with various prototype selection approaches and datasets as a function of the number of prototypes per class. (a) Letter High, l-sps vs d-sps; (b) Digit, l-sps vs d-sps; (c) Grec, l-cps vs d-cps; (d) Letter Medium, l-bps vs d-bps; (e) Letter Low, l-tps vs d-tps; (f) Mutagenecity, l-sps vs d-sps.

instead a case where this empirical dependence is stronger. However, there still is an interval of values over which the cross-validation accuracy is unaffected.

	One-Vs-All						One-Vs-Nearest					
Dataset	d-cps	d-bps	d-rps	d-sps	d-tps	d-kcps	d-cps	d-bps	d-rps	d-sps	d-tps	d-kcps
Letter low	0.11	0.01	0.10	0.53	0.01	0.06	0.54	**0.22**	0.37	0.01	0.04	0.03
Letter medium	0.10	**0.60**	0.01	0.93	0.05	0.05	0.28	0.31	0.05	0.38	0.16	0.09
Letter high	0.13	0.14	0.01	**0.79**	0.01	0.12	0.39	0.03	0.18	0.01	0.01	0.31
Digit	0.24	0.01	0.06	**0.04**	0.03	0.56	0.81	0.04	0.03	**0.08**	0.03	0.13
GREC	0.24	0.06	0.01	**0.09**	0.03	0.14	0.02	0.17	0.29	0.18	0.08	0.81
Fingerprint	0.02	0.27	0.15	**0.20**	0.02	0.15	0.01	0.22	**0.05**	**0.20**	0.03	0.95
AIDS	0.46	0.46	0.01	**0.42**	**0.01**	0.03	0.46	0.46	0.01	**0.42**	**0.01**	0.03
Mutagenicity	0.82	0.19	0.30	**0.15**	0.01	0.04	0.82	0.19	0.30	**0.16**	0.01	0.04
Protein	**0.04**	0.01	0.05	0.01	**0.01**	0.01	**0.24**	0.01	0.14	0.01	**0.04**	0.03
Webpage	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	**0.01**	0.01	0.01	0.01	0.01	**0.01**	**0.01**

Table 5: The W_s value of the best classification accuracy (%) reported in Table 2. The W_s value which returns the best result per dataset is displayed in bold face.

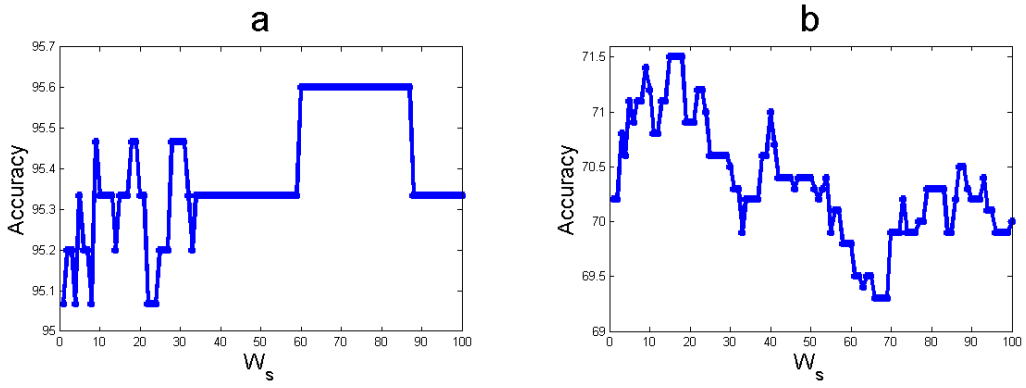

Figure 3: Accuracy with various prototype selection approaches and datasets as a function of the value of W_s (The reported W_s value is multiplied by 100). (a) Letter Medium, d-bps; (b) Mutagenicity, d-sps;.

4.3. Discussion

Many common data types can be seen as special cases of graphs. For example, from an algorithmic perspective both strings and trees are simple instances of graphs. A string is a graph in which each node represents one character, and consecutive characters are connected by an edge. A tree is a graph in which any two nodes are connected by exactly one path. Obviously, also a feature vector $\mathbf{x} \in \mathbb{R}^n$ can be represented as a graph, whereas the contrary, i.e. finding a vectorial description for graphs, is highly non-trivial. In other words, dissimilarity embedding can be applied to any objects which allow a distance, but they are most urgent in the domain of graphs as there are no classifiers directly available, other than nearest neighbors and those based on graph kernels. Therefore, in the case of graphs, the availability of an embedding method is of crucial importance. This motivates and justifies the search for well performing embedding methods and shows that the selection of prototypes is very important for dissimilarity embedding of graphs. Hence, we have proposed novel, discriminative approaches for selecting prototypes from a class-labeled collection of graphs and achieved superior results in classification compared to other well-established prototype selection approaches. Although our present focus is on graphs, it would be very interesting to investigate whether or not these methods are beneficial with other data structures.

5. Conclusion

In this paper, we have presented novel, discriminative approaches for selecting prototypes from a class-labeled collection of graphs. The proposed approaches select prototypes based on a trade off between intra-class compactness, intra-class uniform spread and inter-class separation. Experiments were carried out over a

range of datasets as diverse as letters, digits, drawings, fingerprints, antiviral compounds, mutagenicity, proteins and web pages. From the experimental results, it is possible to draw the following conclusions:

- the proposed discriminative prototype selectors have increased the classification accuracy over the corresponding labeled prototype selector in 42 out of 50 cases, with increases comprised between 0.1% and 5.1% (Table 3);
- the best discriminative prototype selector has outperformed the best compared selector in all cases except one in which they scored equal accuracy, with increases comprised between 0.1% and 2.0% over the range of datasets (Table 4);
- training in a one-vs-all manner has achieved higher accuracy than one-vs-nearest training in the majority of cases (Table 2);
- the accuracy for the proposed discriminative approaches has proved almost invariably the highest for any tested number of prototypes per class (1 to 15) (Figure 2).

The overall conclusion brought forward by this paper is that prototype selection operated in a class-discriminative manner is an ideal approach for selecting effective prototypes for the ensuing classification task. Application is possible with any type of graphs including spatial, structural, temporal, spatio-temporal and others and therefore suits a wide range of classification tasks.

Acknowledgments

The authors wish to thank the Australian Research Council and its industry partners that have partially supported this work under the Linkage Project funding

scheme - grant LP 0990135 “Airport of Future“.

References

[1] D. Conte, P. Foggia, C. Sansone, M. Vento, Thirty years of graph matching in pattern recognition, International journal of pattern recognition and artificial intelligence 18 (3) 265–298.

[2] E. Borzeshi, R. Xu, M. Piccardi, Automatic human action recognition in videos by graph embedding, Springer, 2011, pp. 19–28.

[3] W. Brendel, S. Todorovic, Learning spatiotemporal graphs of human activities, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 778–785.

[4] P. Mahé, N. Ueda, T. Akutsu, J. Perret, J. Vert, Graph kernels for molecular structure-activity relationship analysis with support vector machines, Journal of Chemical Information and Modeling 45 (4) (2005) 939–951.

[5] K. Borgwardt, Graph kernels, Ph.D. thesis, Ludwig-Maximilians-Universität München (2007).

[6] L. Ralaivola, S. Swamidass, H. Saigo, P. Baldi, Graph kernels for chemical informatics, Neural Networks 18 (8) (2005) 1093–1110.

[7] A. Schenker, H. Bunke, M. Last, A. Kandel, Graph-theoretic techniques for web content mining, Vol. 62, World Scientific Pub Co Inc, 2005.

[8] A. Schenker, M. Last, H. Bunke, A. Kandel, Classification of web documents using graph matching, International Journal of Pattern Recognition and Artificial Intelligence 18 (3) (2004) 475–496.

[9] D. Cook, L. Holder, Mining graph data, Wiley-Blackwell, 2007.

[10] Z. Harchaoui, F. Bach, Image classification with segmentation graph kernels, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, Ieee, 2007, pp. 1–8.

[11] B. Luo, R. C Wilson, E. Hancock, Spectral embedding of graphs, Pattern recognition 36 (10) (2003) 2213–2230.

[12] R. Ambauen, S. Fischer, H. Bunke, Graph edit distance with node splitting and merging, and its application to diatom identification, Graph Based Representations in Pattern Recognition (2003) 259–264.

[13] J. Llados, G. Sanchez, Graph matching versus graph parsing in graphics recognition- a combined approach, International Journal of Pattern Recognition and Artificial Intelligence 18 (3) (2004) 455–473.

[14] J. Rocha, T. Pavlidis, A shape analysis model with applications to a character recognition system, Pattern Analysis and Machine Intelligence, IEEE Transactions on 16 (4) (1994) 393–404.

[15] P. Suganthan, H. Yan, Recognition of handprinted chinese characters by constrained graph matching, Image and vision computing 16 (3) (1998) 191–201.

[16] H. Bunke, A graph-theoretic approach to enterprise network dynamics, Vol. 24, Birkhauser, 2007.

[17] P. Dickinson, H. Bunke, A. Dadej, M. Kraetzl, Matching graphs with unique node labels, Pattern Analysis & Applications 7 (3) (2004) 243–254.

[18] R. Wilson, E. Hancock, Levenshtein distance for graph spectral features, in: Proc. 17th Int. Conf. on Pattern Recognition, Vol. 2, 2004, pp. 489–492.

[19] R. Wilson, E. Hancock, B. Luo, Pattern vectors from algebraic graph theory, Pattern Analysis and Machine Intelligence, IEEE Transactions on 27 (7) (2005) 1112–1124.

[20] A. Robles-Kelly, E. Hancock, A riemannian approach to graph embedding, Pattern Recognition 40 (3) (2007) 1042–1056.

[21] E. Pekalska, R. Duin, The dissimilarity representation for pattern recognition: foundations and applications, Vol. 64, World Scientific Pub Co Inc, 2005.

[22] E. Pekalska, R. Duin, P. Paclík, Prototype selection for dissimilarity-based classifiers, Pattern Recognition 39 (2) (2006) 189–208.

[23] B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, R. Duin, Transforming strings to vector spaces using prototype selection, Structural, Syntactic, and Statistical Pattern Recognition (2006) 287–296.

[24] K. Riesen, M. Neuhaus, H. Bunke, Graph embedding in vector spaces by means of prototype selection, in: Proceedings of the 6th IAPR-TC-15 international conference on Graph-based representations in pattern recognition, Springer-Verlag, 2007, pp. 383–393.

[25] H. Bunke, K. Riesen, Improving vector space embedding of graphs through feature selection algorithms, Pattern Recognition 44 (9) (2011) 1928–1940.

[26] I. Jolliffe, MyiLibrary, Principal component analysis, Vol. 2, Wiley Online Library, 2002.

[27] K. Riesen, H. Bunke, Graph classification based on vector space embedding, International Journal of Pattern Recognitionand Artificial Intelligence 23 (6) (2009) 1053.

[28] R. Raveaux, S. Adam, P. Hroux, E. Trupin, Learning graph prototypes for shape recognition, Computer Vision and Image Understanding 115 (7) (2011) 905–918.

[29] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognition Letters 1 (4) (1983) 245–253.

[30] M. Neuhaus, H. Bunke, Automatic learning of cost functions for graph edit distance, Information Sciences 177 (1) (2007) 239–247.

[31] M. Neuhaus, H. Bunke, Self-organizing maps for learning the edit costs in graph matching, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 35 (3) (2005) 503–514.

[32] T. Caetano, J. McAuley, L. Cheng, Q. Le, A. Smola, Learning graph matching, Pattern Analysis and Machine Intelligence, IEEE Transactions on 31 (6) (2009) 1048 –1058.

[33] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image and Vision Computing 27 (7) (2009) 950–959.

[34] M. Neuhaus, K. Riesen, H. Bunke, Fast suboptimal algorithms for the computation of graph edit distance, Structural, Syntactic, and Statistical Pattern Recognition (2006) 163–172.

[35] S. Sorlin, C. Solnon, Reactive tabu search for measuring graph similarity, Graph-Based Representations in Pattern Recognition (2005) 133–133.

[36] Y. Wang, I. Tetko, M. Hall, E. Frank, A. Facius, K. Mayer, H. Mewes, Gene selection from microarray data for cancer classificationa machine learning approach, Computational Biology and Chemistry 29 (1) (2005) 37–46.

[37] G. Hjaltason, H. Samet, Properties of embedding methods for similarity searching in metric spaces, Pattern Analysis and Machine Intelligence, IEEE Transactions on 25 (5) (2003) 530–549.

[38] J. Lafferty, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, Morgan Kaufmann, 2001, pp. 282–289.

[39] R. Gilad-Bachrach, A. Navot, N. Tishby, Margin based feature selection-theory and algorithms, in: Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 43.

[40] W. Brendel, S. Todorovic, Activities as time series of human postures, Computer Vision–ECCV 2010 (2010) 721–734.

[41] L. Kaufman, P. Rousseeuw, et al., Finding groups in data: an introduction to cluster analysis, Vol. 39, Wiley Online Library, 1990.

[42] K. Riesen, H. Bunke, Iam graph database repository for graph based pat-

tern recognition and machine learning, Structural, Syntactic, and Statistical Pattern Recognition (2008) 287–297.

[43] A. Frank, A. Asuncion, UCI machine learning repository (2010).
URL http://archive.ics.uci.edu/ml

[44] P. Dosch, E. Valveny, Report on the second symbol recognition contest, Graphics Recognition. Ten Years Review and Future Perspectives (2006) 381–397.

[45] C. Watson, C. Wilson, Nist special database 4, Fingerprint Database, National Institute of Standards and Technology 17.

[46] D. DTP, Aids antiviral screen (2004).
URL http://dtp.nci.nih.gov/docs/aids/aids-data.html

[47] J. Kazius, R. McGuire, R. Bursi, Derivation and validation of toxicophores for mutagenicity prediction, Journal of Medicinal Chemistry 48 (1) (2005) 312–320.

[48] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, P. Bourne, The protein data bank, Nucleic acids research 28 (1) (2000) 235–242.

[49] I. Schomburg, A. Chang, C. Ebeling, M. Gremse, C. Heldt, G. Huhn, D. Schomburg, Brenda, the enzyme database: updates and major new developments, Nucleic acids research 32 (suppl 1) (2004) D431–D433.

[50] V. Vapnik, An overview of statistical learning theory, Neural Networks, IEEE Transactions on 10 (5) (1999) 988 –999.

[51] A. Fischer, K. Riesen, H. Bunke, An experimental study of graph classification using prototype selection, in: Pattern Recognition, 2008. ICPR 2008. 19th International Conference on, IEEE, 2008, pp. 1–4.

[52] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge Univ Pr, 2004.

[53] C.-C. Chang, C.-J. Lin, LIBSVM: A library for support vector machines, ACM Transactions on Intelligent Systems and Technology 2 (2011) 27:1–27:27.

[54] H. Bunke, K. Riesen, Towards the unification of structural and statistical pattern recognition, Pattern Recognition Letters 33 (7) (2012) 811–825.