Graph Classification Based on Dissimilarity Space Embedding

Horst Bunke and Kaspar Riesen

Institute of Computer Science and Applied Mathematics, University of Bern,
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{bunke,riesen}@iam.unibe.ch

Abstract. Recently, an emerging trend of representing objects by graphs can be observed. In fact, graphs offer a powerful alternative to feature vectors in pattern recognition, machine learning, and related fields. However, the domain of graphs contains very little mathematical structure, and consequently, there is only a limited amount of classification algorithms available. In this paper we survey recent work on graph embedding using dissimilarity representations. Once a population of graphs has been mapped to a vector space by means of this embedding procedure, all classification methods developed in statistical pattern recognition become directly available. In an experimental evaluation we show that the proposed methodology of first embedding graphs in vector spaces and then applying a statistical classifier has significant potential to outperform classifiers that directly operate in the graph domain. Additionally, the proposed framework can be considered a contribution towards unifying the domains of structural and statistical pattern recognition.

1 Introduction

The field of pattern recognition can be divided into two sub-fields, namely the statistical and the structural approach. In statistical pattern recognition, patterns are represented by feature vectors $(x_1, \ldots, x_n) \in \mathbb{R}^n$. The recognition process is based on the assumption that patterns of the same class are located in a compact region of $\mathbb{R}^n$. In recent years a huge amount of methods for the classification of patterns represented by feature vectors have been proposed, such as Bayes classifier, neural network, support vector machine, and many more [1]. Object representations given in terms of feature vectors have a number of useful properties. For example, object similarity, or distance, can easily be computed by means of Euclidean distance. Yet graph-based representations, which are used in the field of structural pattern recognition, have a number of advantages over feature vectors. Graphs are much more powerful and flexible than vectors, as feature vectors provide no direct possibility to describe structural relations in the patterns under consideration. Furthermore, while the size of a graph can be adjusted to the size and complexity of the underlying pattern, vectors are constrained to a predefined length, which has to be preserved for all patterns encountered in a particular application. On the other hand, a major drawback of graph representations is their lack of suitable methods for classification. This is mainly due to the fact that some of the basic operations needed in classification are not available for graphs.

A promising direction to overcome the lack of algorithmic tools for graph classification is graph embedding. Basically, an embedding of graphs into a vector space establishes access to the rich repository of algorithmic tools developed in statistical pattern recognition. The present paper considers a new class of graph embedding procedures which are based on dissimilarity representation and graph edit distance computation. Originally the idea was proposed in [2] in order to map feature vectors into dissimilarity spaces. This idea was first generalized to string based object representation [3] and later to the domain of graphs [4,5,6,7,8,9]. Graphs from a given problem domain are mapped to feature spaces by computing the distance to some predefined prototype graphs, or sets of prototype graphs. The resulting distances can be used as a vectorial representation of the considered graph.

Note that our graph embedding approach can be applied to both directed and undirected graphs, as well as to graphs without and with labels on their nodes and/or edges. In case there are labels on the nodes and/or edges, these labels can be of any nature (discrete symbols, the set of integer or real numbers, or whole attribute vectors). Even hypergraphs can be embedded with the method described in this paper [10]. Hence, the proposed embedding approach is more general than other graph embedding techniques where sometimes restrictions on the type of underlying graph are imposed (e.g. [11,12,13]).

2 Dissimilarity Space Embeddings of Graphs

In [2] the authors claim that the concept of proximity is more fundamental than that of a feature or a class. Furthermore, it is pointed out that in the case of structural data (like graphs) the extraction of numerical features may be difficult or even intractable, while proximity can directly be derived from the data using an adequate dissimilarity model.

Assume we have a labeled set of sample graphs, $\mathcal{G} = \{g_1, \ldots, g_N\}$ and a graph dissimilarity measure $d(g_i, g_j)$. Note that $\mathcal{G}$ can be any kind of graph set. For the sake of convenience, however, in the context of the present work we define $\mathcal{G}$ to be the set of training graphs $\mathcal{T}$. After having selected a set of prototype graphs $\mathcal{P} = \{p_1, \ldots, p_n\} \subseteq \mathcal{T}$, we compute the dissimilarity of a given input graph g to each prototype $p \in \mathcal{P}$. Note that g can be an element of $\mathcal{T}$ or any other graph. This leads to n dissimilarities, $d_1 = d(g, p_1), \ldots, d_n = d(g, p_n)$, which can be arranged in an n-dimensional vector $(d_1, \ldots, d_n)$. In this way we can transform any graph from the training as well as any other graph set (for instance a validation or a test set of a classification problem), into a vector of real numbers.

Definition 1 (Graph Embedding). *Let $\mathcal{G}$ be a finite or infinite set of graphs and $\mathcal{P} = \{p_1, \ldots, p_n\} \subseteq \mathcal{G}$ be a set of prototypes. Then, the mapping $\varphi_n^{\mathcal{P}} : \mathcal{G} \rightarrow \mathbb{R}^n$ is defined as the function*

$$\varphi_n^P(g) \mapsto (d(g,p_1), \ldots, d(g,p_n)),$$

where $d(g,p_i)$ is some graph dissimilarity measure between graph g and the i-th prototype.

Though conceptually $d(g,p_i)$ can be any kind of dissimilarity measure, the embedding procedure proposed in this paper makes use of graph edit distance. The key idea of graph edit distance is to define the dissimilarity, or distance, of graphs by the minimum amount of distortion that is needed to transform one graph into another [14]. Compared to other approaches, graph edit distance is very flexible since it can handle arbitrary graphs and any type of node and edge labels. Furthermore, by defining costs for edit operations, the concept of edit distance can be tailored to specific applications.

Optimal algorithms for computing the edit distance of graphs are typically based on combinatorial search procedures that explore the space of all possible mappings of the nodes and edges of the first graph to the nodes and edges of the second graph [14]. A major drawback of those procedures is their computational complexity, which is exponential in the number of nodes of the involved graphs. However, a number of efficient suboptimal methods for graph edit distance computation have been proposed (e.g. [15] with cubic time complexity). Consequently, given n predefined prototypes the embedding of one particular graph is established by means of n distance computations with polynomial time.

Regarding the general embedding procedure established above, a strong relationship to graph kernel methods can be observed [16]. In fact, based on the graph embedding $\varphi_n^{\mathcal{P}}$ established above, one can define a valid graph kernel κ by applying valid kernel functions defined for vectors to two graph maps in the resulting vector space (e.g. Scalar Product, Radial Basis Function, Sigmoid Function, etc.) [17].

3 Survey of Work on Dissimilarity Based Graph Embedding

In this section we provide an overview of our recent work on graph embeddings using dissimilarity representation, i.e. we recapitulate several papers addressing different aspects of graph embedding and its applications [4,5,6,7,8,9].

3.1 Prototype Selection

The embedding method described in the previous section crucially depends on the prototypes. Therefore, an important problem to be solved is an appropriate choice of the prototype set $\mathcal{P} = \{p_1, \ldots, p_n\}$. A good selection of n prototypes seems to be crucial to succeed with the classification algorithm in the feature vector space. The prototypes should avoid redundancies in terms of selection of similar graphs, and prototypes should include as much information as possible.

In the following six different prototype selection strategies are discussed. For a more thorough review of the prototype selection methods we refer to [4], where these strategies are applied to several graph data sets.

Center. The Centers prototype selector selects prototypes situated in the center of the training set $\mathcal{T}$. This is achieved by iteratively taking the set median graph out of the set $\mathcal{T}$. The set median graph is the graph whose sum of distances to all other graphs in the set is minimal.

Border. The Border prototype selector selects prototypes situated at the border of the training set $\mathcal{T}$, i.e. it iteratively takes the set marginal graph out of the set $\mathcal{T}$. The set marginal graph is the graph whose sum of distances to all other graphs in this set is maximal.

Random. A random selection of n prototypes from $\mathcal{T}$ is performed.

Spanning. The first prototype selected is the set median graph. Each additional prototype selected by the Spanning prototype selector is the graph the furthest away from the already selected prototype graphs.

k-Centers. The k-Centers prototype selector tries to adapt to the graph distribution of set $\mathcal{T}$ and selects graphs that are in the center of densely populated areas. First a k-means clustering procedure is applied to set $\mathcal{T}$. The number of clusters to be found is equal to the number of prototypes to be selected. Once the clusters have been established, the set median of each cluster is selected as a prototype.

Targetsphere. The Targetsphere prototype selector first selects a graph g_c situated in the center of $\mathcal{T}$. Next the graph $g_f \in \mathcal{T}$ whose distance to g_c is maximum is located. The remaining prototypes are uniformly distributed from g_c to g_f. That is, the interval $[0, d_{max}]$ ($d_{max} = d(g_c, g_f)$) is divided into $m-1$ equidistant subintervals of width $w = \frac{d_{max}}{m-1}$. The $m-2$ graphs for which the corresponding distances to the center graph g_c are located nearest to the interval borders in terms of edit distance are selected as prototypes.

In [4] it has been our intention to improve the accuracy achieved by nearest-neighbor classifiers in the graph domain by the use of classifiers operating on graph maps, i.e. vectorial representations. We applied support vector machine (SVM) as a popular method from statistical pattern recognition and showed that this approach outperforms the nearest-neighbor classifiers in the graph domain. In the experiments reported in [4] it turns out that the quality of the prototype selectors introduced above depends on the application and the underlying data. Therefore, the question which of the prototype selectors is globally best cannot definitely be answered. Nevertheless, there is a clear tendency that prototype selectors that distribute the graphs more or less uniformely over the whole graph set (Spanning, k-Centers, Targetsphere) lead to higher recognition rates.

3.2 Prototype Reduction

The prototype selection strategies introduced above use some heuristics based on the underlying dissimilarities in the original graph domain. Another idea to select the members of set $\mathcal{P}$ is prototype reduction methods in conjunction with nearest neighbor classifiers [18]. These reduction schemes determine a subset $\mathcal{P} \subseteq \mathcal{T}$ such that the elements in $\mathcal{T}$ (or at least a considerable part of them) are still correctly classified using a nearest neighbor classifier.

We use *selective* prototype selectors where the number of prototypes is *uncontrollable* [19]. These two constraints are motivated through the following considerations. First, the fact that we are dealing with graphs makes the creation of new prototypes quite difficult. Therefore, we restrict set $\mathcal{P}$ to include only graphs from $\mathcal{T}$. Secondly, we want to bypass the time consuming validation of the dimensionality of the resulting embedding space by means of the target classifier. Hence, we leave the determination of the number of prototypes to the prototype selection algorithm.

Condensing (Cond). The idea of condensing a training set $\mathcal{T}$ is to iteratively select graphs $g_i \in \mathcal{T}$ as prototypes until all graphs from $\mathcal{T}$ are correctly classified using the respective prototypes. As a disadvantage, this procedure depends on the order in which the graphs are processed.

Modified Condensing (mCond). Modified condensing overcomes the limitation of order dependency. In this scheme we start with a basic set of prototypes containing the set center graph (*centroid*) of each class. The centroid of $\mathcal{G}$ is the graph for which the maximum distance to all other graphs in $\mathcal{G}$ is minimum. The graphs from $\mathcal{T}$ are classified by means of this initial set of prototypes. Using only the misclassified graphs, the class centroids are computed and subsequently added to the existing set of prototypes. This procedure is repeated until all graphs from $\mathcal{T}$ are correctly classified.

Editing (Edit). The basic idea of editing a training set $\mathcal{T}$ is to delete outliers from $\mathcal{T}$. For this purpose, we classify each graph g_i from $\mathcal{T}$ with a 3-NN classifier. If g_i is misclassified we assume that this particular graph is an outlier and therefore should not be included in the prototype set.

Reducing (Red). The idea of reducing is built up on condensing. First, the training set $\mathcal{T}$ is condensed to a prototype set $\mathcal{P}$. Next, each prototype p_i is iteratively removed from $\mathcal{P}$. The training graphs are then classified using the reduced prototype set $\mathcal{P} \setminus \{p_i\}$. If all graphs are classified correctly with this reduced prototype set, the respective prototype is useless and can therefore be omitted. Otherwise, the prototype is necessary and therefore kept in $\mathcal{P}$.

Merging (Merg). The basic idea of merging a training set is to define two graph sets $\mathcal{P}$ and $\mathcal{Q}$, where initially $\mathcal{P}$ is empty and $\mathcal{Q}$ contains all training graphs from $\mathcal{T}$. First, an arbitrary graph from $\mathcal{Q}$ is selected as prototype, i.e. moved from $\mathcal{Q}$ to

$\mathcal{P}$. Next, we consider the two closest graphs p and q from $\mathcal{P}$ and $\mathcal{Q}$, respectively. If the class of p is not the same as that of q, q is moved from $\mathcal{Q}$ to $\mathcal{P}$. Otherwise, p and q are merged to $p^* \in \mathcal{Q}$, where p^* minimizes the sum of distances to p and q. The accuracy of the NN classifier using $\mathcal{P} \cup \{q\}$ is then compared with the accuracy when $\mathcal{P} \setminus \{p\} \cup \{p^*\}$ is used as prototype set. Whenever the former outperforms the latter, q is moved from $\mathcal{Q}$ to $\mathcal{P}$. Otherwise, p and q are removed from $\mathcal{P}$ and $\mathcal{Q}$, respectively, and p^* is moved from $\mathcal{Q}$ to $\mathcal{P}$. This procedure is repeated until no graphs are left in $\mathcal{Q}$.

Selecting (Sel). Another algorithm for reducing the training set $\mathcal{T}$ is based on the idea of related neighbors. We define $g_j \in \mathcal{T}$ as a related neighbor to $g_i \in \mathcal{T}$ if g_i and g_j are out of the same class, and g_j is nearer to g_i than any other sample $g_k \in \mathcal{T}$ from another class. The selection of the prototypes is now stated as finding a small number of graphs such that each of these graph has at least one related neighbor. In the present paper a greedy algorithm is employed seeking for a small number of prototypes.

For a more detailed discussion on these prototype reduction schemes in conjunction with graph embeddings we refer to [9]. Furthermore, in Section 4.3, an experimental evaluation of this approach is given.

3.3 Dimensionality Reduction

In Section 3.1 and 3.2 a number of prototype selection strategies have been introduced. In the current section we describe an alternative approach where we use all available elements from the training set as prototypes, i.e. $\mathcal{P} = \mathcal{T}$ and subsequently apply dimensionality reduction methods. This process is more principled and allows us to completely avoid the problem of finding the optimal prototype selection strategy. For dimensionality reduction, we make use of Principal Component Analysis (PCA), Fisher's Linear Discriminant Analysis (LDA) [1], and kernel PCA [20].

Principal Component Analysis (PCA). PCA [1] is a linear transformation. It seeks the projection which best represents the data. PCA is an unsupervised method which does not take any class label information into consideration. We first normalize the data by shifting the mean to the origin of the coordinate system and making the variance of each feature equal to one. Then we calculate the covariance matrix of the normalized data and determine the eigenvectors $\mathbf{e}_i$ and the eigenvalues λ_i of the covariance matrix. The eigenvectors are ordered according to decreasing magnitude of the corresponding eigenvalues, i.e. $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N \geq 0$. The data is then represented in a new coordinate system defined by the eigenvectors. For reducing the dimensionality of the transformed data we retain only the $n < N$ eigenvectors with the n highest eigenvalues.

Fisher's Linear Discriminant Analysis (LDA). LDA [1] is a linear transformation as well. In contrast with PCA, LDA takes class label information into account. In its original form, LDA can be applied to two-class problems only.

However, we make use of a generalization, called Multiple Discriminant Analysis (MDA), which can cope with more than two classes. In MDA, we are seeking the projection of the data which best separates the classes from each other. For all further details, we refer to [1]. Note that under this transformation the maximal dimensionality of the transformed feature space is $c-1$, where c is the number of classes.

Kernel PCA. The result of a kernel function $k(\mathbf{x}, \mathbf{y})$, applied to two feature vectors $\mathbf{x}$ and $\mathbf{y}$, is equal to the result that one obtains by mapping those vectors to a possibly infinite dimensional feature space $\mathcal{F}$ and computing their dot product subsequently [17]. This procedure offers a very elegant way to construct nonlinear extensions of linear algorithms in pattern recognition. The fundamental observation that makes kernel theory so interesting in the field of pattern recognition is that many of the algorithms (e.g. PCA [20]) can be *kernelized*, i.e. they can be formulated entirely in terms of dot products. Consequently, instead of applying PCA in the original vector space, the linear transformation is applied in an implicit existing feature space $\mathcal{F}$ by substituting the dot product by an appropriate kernel function κ. This procedure is commonly referred to as the *kernel trick* [17].

Experimental results and a more detailed description of these dimensionality reduction strategies applied to vector space embedded graphs can be found in [6,7]. The main finding in [6] is that the performance of a k-nearest neighbor classifier in the graph domain, used as a reference system, can be outperformed with statistical significance using this embedding approach. In case of classification problems with many classes, the MDA based system is preferable, while for a small number of classes the PCA based system is the method of choice. Moreover, in [7] we observe that the approach with kernelized PCA outperforms the former approach with linear PCA on most of the data sets.

3.4 Relationship to Lipschitz Embeddings

Our graph embedding approach can also be regarded as a special instance of a Lipschitz embedding [21,22] where a coordinate space is defined such that each axis corresponds to a reference set of objects. Formally, rather than a single prototype set $\mathcal{P} = \{p_1, \ldots, p_n\}$ we define a set $\mathcal{S} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ that consists of n prototype sets. The n subsets $\mathcal{P}_i \subset \mathcal{T}$ define the *reference sets* of the Lipschitz embedding. The extended graph edit distance function between graphs and reference sets is defined as $d(g, \mathcal{P}_i) = \min_{p \in \mathcal{P}_i}\{d(g,p)\}$. The Lipschitz embedding with respect to $\mathcal{S} = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ is then defined as

$$\varphi_n^{\mathcal{S}}(g) = (d(g, \mathcal{P}_1), \ldots, d(g, \mathcal{P}_n)) \quad .$$

Obviously, the range of function $\varphi_n^{\mathcal{S}}$ is a vector space where each dimension corresponds to a subset $\mathcal{P}_i \subset \mathcal{T}$ and the coordinate values of the embedded graph g are the distances from g to the nearest element in $\mathcal{P}_i$.

In [8] two different methods to define our reference sets are applied, viz. random selection and a more advanced technique based on k-Centers prototype selection. In the random method we randomly select n subsets $\mathcal{S} = \{\mathcal{P}_1, \dots, \mathcal{P}_n\}$ each of size m. After drawing a graph g from $\mathcal{T}$, g is put back such that the same graph can have multiple occurences in $\mathcal{P}_i$, and can occur in different sets $\mathcal{P}_i$ and $\mathcal{P}_j$ of $\mathcal{S}$ as well.

For the second method k-Centers prototype selection is applied on $\mathcal{T}$. This results in n disjoint subsets $\mathcal{P}_i \subset \mathcal{T}$ of different size. Next, we iteratively remove the set marginal graphs of $\mathcal{P}_i$ until m graphs remain in $\mathcal{P}_i$. Note that no graphs are removed whenever the size of a set $\mathcal{P}_i$ is smaller than, or equal to, m. Note that in contrast with the random selection this method leads to disjoint subsets.

From the experimental evaluation conducted in [8] one can draw the following conclusions. A definition of the reference sets based on a more elaborated method than random selection is preferable. Classifiers using the Lipschitz embedded graphs outperform classification systems using the original graph edit distances. Finally, the generalization of the Lipschitz embedding to the case where the subsets are not necessarily singletons ($m \geq 1$) is clearly beneficial.

3.5 Multiple Classifier System

Recently, the field of multiple classifier systems has become a very active area of research. The fundamental observation that motivates the combination of classifiers is that the sets of patterns misclassified by different classifiers do not necessarily heavily overlap. Hence, errors of a single classifier can be compensated by the other classifiers of an ensemble [23]. In the case of statistical patterns, that is, patterns represented by feature vectors, a large number of methods for the creation and combination of classifiers have been developed over the past years [23,24,25].

Regarding the graph embedding procedure proposed in this paper, if we repeat the process of random prototype selection a number of times, we can derive different graph subsets that can be used to map a given population of graphs to various vector spaces. That is, we obtain n different vector sets all representing the same graph set. For each vector set an individual classifier is trained and thus one gets an ensemble of classifiers. Consequently, a number of methods become available for combining the results of the individual ensemble members. In [5] this method was tested on a number of graph data sets with different characteristics, comming from various application domains. From the results of our experiments, one can conclude that the classification accuracy can be enhanced by most ensemble methods on almost all data sets.

4 Experimental Results

Lacking space we present the results for one particular graph embedding setting only, namely the graph embedding in conjunction with the prototype reduction schemes introduced in Section 3.2. For further results achieved with the novel graph embedding methodology, we refer to [4,5,6,7,8,9].

4.1 Experimental Setup

The classifier applied in the graph domain is a k-nearest-neighbor classifier. Note that this specific classifier is basically the only classifier that is directly applicable in the graph domain. The classifier applied to the vector space embedded graphs $\varphi_n^{\mathcal{P}}(g)$ is the support vector machine (SVM) with RBF Kernel. Hence, the kernel values are given by

$$\kappa_{RBF}(g_i, g_j) = exp\left(-\gamma||\varphi_n^{\mathcal{P}}(g_i) - \varphi_n^{\mathcal{P}}(g_j)||^2\right)$$

where $\gamma > 0$.

In each of our experiments we make use of three disjoint graph sets, viz. a *validation set*, a *test set* and a *training set*. The validation set is used to determine optimal parameter values for classification. They consist of parameter k for the nearest neighbor classifier and the different parameters for the SVM, viz. γ (RBF) and C (weighting of the maximization of the margin and the minimization of the error). The parameter combination that results in the lowest classification error on the validation set is finally applied to the independent test set.

The pattern classification tasks considered in this paper involves a total of five different graph data sets. Because of lack of space, we can give only a short description of the data. For a more thorough description we refer to [4] where the data sets are discussed in greater detail. Note that each of our graph sets is divided into three disjoint subsets, viz. a training, a validation, and a test set.

The first database used in the experiments consists of graphs representing distorted letter drawings out of 15 classes (Letter). The second data set is given by graphs representing fingerprint images of the NIST-4 database [26] out of the four classes *arch*, *left*, *right*, and *whorl* (Fingerprint). The third graph set is constructed from the AIDS Antiviral Screen Database of Active Compounds [27] (AIDS). Graphs from this data set represent molecules out of two classes (*active*, *inactive*). The last data set consists of graphs representing webpages [28] that originate from 20 different categories (*Business*, *Health*, *Politics*, ...) (Webgraph).

4.2 Reference Systems

Two reference systems are used to compare the performance of the proposed graph embedding procedure with. Similarly to our novel approach, both reference systems make use of an SVM. The first reference system uses a similarity kernel directly derived from the edit distances [29] (referred to as GED). That is, for this reference method no explicit graph embedding is conducted but the dissimilarites are merely turned into kernel values $\kappa(g_i, g_j) = -d(g_i, g_j)^2$. The second reference system interprets all distances to the whole training set $\mathcal{T}$ as a vectorial description, i.e. the graphs are explicitly embedded but no attempts are made to reduce the number of prototypes (referred to as All). For this reference system an RBF kernel is applied to the vector space embedded graphs, too.

4.3 Results and Discussion

In Table 1 the classification results of all reference systems and the proposed approach using all prototype reduction schemes are given. Comparing the results of our novel approach with the results achieved by the first reference system (GED), we observe the following. On the Webpage data the first reference method outperforms all other systems. On this particular data set, the transformation of the edit distances into kernel values seems to be the best choice. However, on the three remaining data sets it is beneficial to use the embedding approach rather than the direct transformation. On the Letter data the SVM based on the merged prototypes performes better than the first reference system and on the AIDS and Fingerprint data even all reduction schemes lead to better classification accuracies than the similarity kernel. Note that 12 out of 13 improvements, but only 5 out of 11 deteriorations, compared to the first reference system are statistically significant.

Regarding the results achieved by the second reference system (All), we observe that our approach using prototype reduction outperforms this reference system on all data sets (at least with one of the proposed reduction schemes). Hence, besides the speed-up in computation, it is beneficial for achieving a higher recognition rate to use prototype reduction for embedding rather than using the whole training set as prototypes.

Comparing the prototype reduction schemes against each other, one can conclude that the merging approach performs generally best. On three out of four data sets this reduction scheme leads to the overall best classification result (including the reference methods). On the other hand, condensing and modified condensing lead on three data sets to the lowest recognition rate among the prototype reduction approaches.

Table 1. Experimental Results

	Ref. System		Proposed Method					
Data Set	GED	All	Cond	mCond	Edit	Red	Merg	Sel
Letter	92.27	91.73	91.73	91.47	92.00	92.00	92.53	92.00
AIDS	93.60	97.20	97.13	97.53	97.20	97.20	98.27	97.20
Fingerprint	79.35	82.10	81.60	81.75	81.70	82.10	82.80	82.30
Webpage	84.62	82.44	82.44	83.21	76.92	82.95	81.15	81.03

5 Conclusions

Although graphs have a higher representational power than feature vectors, there is a lack of methods for pattern classification using graph representations. By contrast, a large number of methods for classification have been proposed for object representations given in terms of feature vectors. The present paper introduces a novel graph embedding procedure in order to bridge the gap between structural and statistical pattern recognition. These graph embeddings make explicit use of graph edit distance and can therefore deal with various kinds of graphs (labelled, unlabelled, directed, undirected, etc.). The basic idea of the embedding procedure is to describe a graph by means of n dissimilarities to a predefined prototype set or to sets of prototypes. That is, a graph g is mapped explicitly to the n-dimensional real space $\mathbb{R}^n$ by arranging the distances of g to all of the n prototypes, or n prototype sets, as a vector. We show that the embedding process can be controlled by different prototype selectors or by well-known dimensionality reduction algorithms. Furthermore, the proposed graph embedding process lends itself to a method for the automatic generation of classifier ensembles. From the results of our experiments presented in the present paper as well as in other contributions, one can conclude that the classification accuracy can be statistically significantly enhanced by all embedding methods compared to reference systems directly operating in the graph domain.

Acknowledgements

This work has been supported by the Swiss National Science Foundation (Project 200021-113198/1).

References

1. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley-Interscience, Hoboken (2000)
2. Pekalska, E., Duin, R.: The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. World Scientific, Singapore (2005)
3. Spillmann, B., Neuhaus, M., Bunke, H., Pekalska, E., Duin, R.: Transforming strings to vector spaces using prototype selection. In: Yeung, D.-Y., Kwok, J.T., Fred, A., Roli, F., de Ridder, D. (eds.) SSPR 2006 and SPR 2006. LNCS, vol. 4109, pp. 287–296. Springer, Heidelberg (2006)
4. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In: Escolano, F., Vento, M. (eds.) GbRPR 2007. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)
5. Riesen, K., Bunke, H.: Classifier ensembles for vector space embedding of graphs. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 220–230. Springer, Heidelberg (2007)
6. Riesen, K., Bunke, H.: Reducing the dimensionality of dissimilarity space embedding graph kernels. Engineering Applications of Artificial Intelligence Engineering Applications of Artificial Intelligence (accepted, 2008)
7. Riesen, K., Bunke, H.: Non-linear transformations of vector space embedded graphs. In: 8th International Workshop on Pattern Recognition in Information Systems (accepted, 2008)
8. Riesen, K., Bunke, H.: On Lipschitz embeddings of graphs. In: 12th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (accepted, 2008)
9. Riesen, K., Bunke, H.: Dissimilarity based vector space embedding of graphs using prototype reduction schemes (submitted)
10. Bunke, H., Dickinson, P., Kraetzl, M.: Theoretical and algorithmic framework for hypergraph matching. In: Roli, F., Vitulano, S. (eds.) ICIAP 2005. LNCS, vol. 3617, pp. 463–470. Springer, Heidelberg (2005)

11. Luo, B., Wilson, R., Hancock, E.: Spectral embedding of graphs. Pattern Recognition 36(10), 2213–2223 (2003)
12. Wilson, R., Hancock, E., Luo, B.: Pattern vectors from algebraic graph theory. IEEE Trans. on Pattern Analysis ans Machine Intelligence 27(7), 1112–1124 (2005)
13. Robles-Kelly, A., Hancock, E.: A Riemannian approach to graph embedding. Pattern Recognition 40, 1024–1056 (2007)
14. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1, 245–253 (1983)
15. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image and Vision Computing (accepted, 2008)
16. Gärtner, T., Lloyd, J., Flach, P.: Kernels and distances for structured data. Machine Learning 57(3), 205–232 (2004)
17. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge (2004)
18. Bezdek, J., Kuncheva, L.: Nearest prototype classifier designs: An experimental study. Int. Journal of Intelligent Systems 16(12), 1445–1473 (2001)
19. Kim, S., Oommen, B.: A brief taxonomy and ranking of creative prototype reduction schemes. Pattern Analysis and Applications 6, 232–244 (2003)
20. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10, 1299–1319 (1998)
21. Bourgain, J.: On Lipschitz embedding of finite metric spaces in Hilbert spaces. Israel Journal of Mathematics 52(1-2), 46–52 (1985)
22. Hjaltason, G., Samet, H.: Properties of embedding methods for similarity searching in metric spaces. IEEE Trans. on Pattern Analysis ans Machine Intelligence 25(5), 530–549 (2003)
23. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. John Wiley, Chichester (2004)
24. Breiman, L.: Bagging predictors. Machine Learning 24, 123–140 (1996)
25. Freund, Y., Shapire, R.: A decision theoretic generalization of online learning and application to boosting. Journal of Computer and Systems Sciences 55, 119–139 (1997)
26. Watson, C., Wilson, C.: NIST Special Database 4, Fingerprint Database. National Institute of Standards and Technology (1992)
27. DTP, AIDS antiviral screen (2004), `http://dtp.nci.nih.gov/docs/aids/aids_data.html`
28. Schenker, A., Bunke, H., Last, M., Kandel, A.: Graph-Theoretic Techniques for Web Content Mining. World Scientific, Singapore (2005)
29. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific, Singapore (2007)