Non-linear Transformations of Vector Space Embedded Graphs

Kaspar Riesen and Horst Bunke

Institute of Computer Science and Applied Mathematics, University of Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland

Abstract. In pattern recognition and related areas an emerging trend of representing objects by graphs can be observed. As a matter of fact, graph based representation offers a powerful and flexible alternative to the widely used feature vectors. However, the space of graphs contains almost no mathematical structure, and consequently, there is a lack of suitable algorithms for graph classification, clustering, and analysis. Recently, a general approach to transforming graphs into n-dimensional real vectors has been proposed. In the present paper we use this method, which can also be regarded as a novel graph kernel, and investigate the application of kernel principal component analysis (kPCA) on the resulting vector space embedded graphs. To this end we consider the common task of object classification, and show that kPCA in conjunction with the novel graph kernel outperforms different reference systems on several graph data sets of diverse nature.

1 Introduction

The first step in any system in pattern recognition, machine learning, data mining, and related fields consists of the representation of objects by adequate data structures. In statistical approaches the data structure is given by n-dimensional vectors $\mathbf{x} \in \mathbb{R}^n$, where each of the n dimensions represents the value of a specific feature. In recent years a huge amount of algorithms for classification, clustering, and analysis of objects given in terms of feature vectors have been developed [1–3].

Yet, the use of feature vectors implicates two limitations. First, as vectors describe a predefined set of features, all vectors of a set have to preserve the same length regardless of the size or complexity of the corresponding objects. Furthermore, there is no direct possibility to describe binary relationships among different parts of an object. It is well known that both constraints can be overcome by graph based representation [4]. That is, graphs allow us to adapt their size to the complexity of the underlying object and they offer a convenient possibility to describe relationships among different parts of an object.

The major drawback of graphs, however, is that they offer little mathematical structure, i.e. most of the basic mathematical operations are not available or not defined in a standardized way for graphs. Nevertheless, since the concept of kernel machines has been extended from vectors to symbolic data structure, and in particular to graphs [5], this drawback can be overcome. The key idea of such kernel machines is that rather than

Riesen K. and Bunke H. (2008).
Non-linear Transformations of Vector Space Embedded Graphs.
In *Proceedings of the 8th International Workshop on Pattern Recognition in Information Systems*, pages 173-183

defining handcrafted mathematical operations in the original graph domain, the graphs are mapped into a vector space where all those operations are readily available. Obviously, by means of graph kernels one can benefit from both the high representational power and flexibility of graphs and the wide range of pattern recognition algorithms for feature vectors.

In the present paper we use the dissimilarity space embedding graph kernel [6]. In order to map individual graphs into a vector space, this graph kernel employs graph edit distance. Consequently, it can be applied to any kind of graphs (directed or undirected, labeled or unlabeled). If we allow labels on the nodes or edges, these labels can be symbolic, numerical or whole attribute vectors. The basic idea of this approach is to transform a given graph g into a n'-dimensional[1] vector $\mathbf{x} = (x_1, \ldots, x_{n'})$, where each component x_i ($1 \leq i \leq n'$) is equal to the graph edit distance of g to the i-th graph of a predefined set of prototype graphs $\mathcal{P}$.

In a first attempt to define an appropriate set $\mathcal{P}$, prototype selection methods have been applied to a training set [6]. Next, a more general approach without heuristic prototype selection methods has been proposed [7]. In this approach the whole training set $\mathcal{T}$ is used as prototype set $\mathcal{P}$, i.e. $\mathcal{P} = \mathcal{T}$. Two classical approaches of linear transformations, i.e. Principal Component Analysis (PCA) and Multiple Discriminant Analysis (MDA) [1], have been applied to vector space embedded graphs subsequently.

In the present paper we build upon this idea, but in the extended fashion of non-linear transformation by means of Kernel PCA. That is, we transform graphs into vectors by edit distance computation utilizing a whole training set $\mathcal{T}$ and eventually apply kernel PCA to the resulting vectorial descriptions of the graphs. With several experimental results we show that this approach outperforms both a reference system applied in the original graph domain and a reference system in the embedded vector space in conjunction with linear PCA.

2 Dissimilarity Space Embedding of Graphs

Similarly to the graph kernels described in [8] 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. A standard set of distortion operations is given by *insertions*, *deletions*, and *substitutions* of nodes and edges.

Given two graphs, the source graph g_1 and the target graph g_2, the idea of graph edit distance is to delete some nodes and edges from g_1, relabel (substitute) some of the remaining nodes and edges, and insert some nodes and edges in g_2, such that g_1 is finally transformed into g_2. A sequence of edit operations $e_1, \ldots, e_k$ that transform g_1 into g_2 is called an *edit path* between g_1 and g_2. Obviously, for every pair of graphs (g_1, g_2), there exist a number of different edit paths transforming g_1 into g_2. Let $\Upsilon(g_1, g_2)$ denote the set of all such edit paths. To find the most suitable edit path out of $\Upsilon(g_1, g_2)$, one introduces a cost for each edit operation, measuring the strength of the corresponding operation. The idea of such cost functions is to define whether or not an edit operation

[1] We use n' instead of n for the sake of consistency with the remainder of this paper.

represents a strong modification of the graph. Hence, between two similar graphs, there should exist an inexpensive edit path, representing low cost operations, while for different graphs an edit path with high costs is needed. Consequently, the *edit distance* of two graphs is defined by the minimum cost edit path between two graphs.

Definition 1 (Graph Edit Distance) *Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ be the source graph and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ be the target graph. The graph edit distance between g_1 and g_2 is defined by*

$$d(g_1, g_2) = \min_{(e_1,\ldots,e_k)\in\Upsilon(g_1,g_2)} \sum_{i=1}^{k} c(e_i),$$

where $\Upsilon(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2, and c denotes the edit cost function measuring the strength $c(e_i)$ of edit operation e_i.

The edit distance of graphs can be computed, for example, by a tree search algorithm [9] or by faster, suboptimal methods which have been proposed recently [10].

2.1 Basic Embedding Approach

Different approaches to graph embedding have been proposed in the literature [11–13]. In [11], for instance, an embedding based on algebraic graph theory and spectral matrix decomposition is proposed. Applying an error-tolerant string matching algorithm to the eigensystem of graphs to infer distances of graphs is proposed in [12]. These distances are then used to embed the graphs into a vector space by multidimensional scaling. In [13] features derived from the eigendecompostion of graphs are studied. In fact, such feature extraction defines an embedding of graphs into vector spaces, too.

Recently, it has been proposed to embed graphs in vector spaces by means of edit distance and prototypes [6]. The idea underlying this method was first developed for the embedding of real vectors in a dissimilarity space [14]. In this method, after having selected a set $\mathcal{P} = \{p_1, \ldots, p_{n'}\}$ of $n' \leq n$ prototypes from a training set $\mathcal{T} = \{g_1, \ldots, g_n\}$, the dissimilarity of a graph g to each prototype $p \in \mathcal{P}$ is computed. This leads to n' dissimilarities, $d_1 = d(g, p_1), \ldots, d_{n'} = d(g, p_{n'})$, which can be interpreted as an n'-dimensional vector $(d_1, \ldots, d_{n'})$. In this way we can transform any graph from the training set, as well as any other graph from a validation or testing set, into a vector of real numbers.

Definition 2 (Graph Embedding) *Given are a graph space $\mathcal{G}$ and a training set of graphs $\mathcal{T} = \{g_1, \ldots, g_n\} \subseteq \mathcal{G}$. If $\mathcal{P} = \{p_1, \ldots, p_{n'}\} \subseteq T$ is a set of prototypes, the mapping $t^{\mathcal{P}}_{n'} : \mathcal{G} \to \mathbb{R}^{n'}$ is defined as a function*

$$t^{\mathcal{P}}_{n'}(g) \mapsto (d(g, p_1), \ldots, d(g, p_{n'}))$$

where $d(g, p_i)$ is the graph edit distance between the graph $g \in \mathcal{G}$ and the i-th prototype. □

One crucial question in this approach is how to find a subset $\mathcal{P}$ of prototypes that lead to a good performance of the classifier in the feature space. As a matter of fact,

both the individual prototypes selected from $\mathcal{T}$ and their number have a critical impact on the classifier's performance. In [6, 14] different prototype selection algorithms are discussed. It turns out that none of them is globally best, i.e. the quality of a prototype selector depends on the underlying data set.

In a recent paper it is proposed to use all available elements from the training set of prototypes, i.e. $\mathcal{P} = \mathcal{T}$ and subsequently apply dimensionality reduction methods. This process is much more principled than the previous approaches and completely avoids the difficult problem of heuristic prototype selection. For dimensionality reduction Principal Component Analysis (PCA) and Fisher's Linear Discriminant Analysis (LDA) [1, 15] are used. In the present paper we use the same idea but with an extension of PCA to non-linear distributions. That is, instead of directly applying a PCA to the vectors resulting from mapping $t^{\mathcal{P}}_{n'}$ with $n' = n$, the vectors are implicitly mapped into a higher-dimensional feature space by means of a kernel function. In this higher-dimensional feature space PCA is then applied to the vector maps. This procedure is commonly referred to as kernel PCA [16].

3 PCA and Kernel PCA

In this section we first review linear transformations by means of PCA and then describe a non-linear extension by means of kernel PCA.

3.1 PCA

Principal Component Analysis (PCA) [3] is a linear transformation which basically seeks the projection that best represents the data. PCA finds a new space whose basis vectors correspond to the maximum variance directions in the original space. PCA falls in the category of unsupervised transformation methods, i.e. PCA does not take any class label information into consideration. Let us assume that m objects are given in terms of n-dimensional column vectors $\mathbf{x} \in \mathbb{R}^n$. We first normalize the data by shifting the sample mean $\mu = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i$ to the origin of the coordinate system, i.e. we center the data. Next we compute the covariance matrix $\mathbf{C}$ of the centered data which is defined as

$$\mathbf{C} = \frac{1}{m}\sum_{i=1}^{m} \mathbf{x}_i \mathbf{x'}_i$$

The covariance matrix $\mathbf{C}$ is symmetric and, therefore, an orthogonal basis can be defined by finding the eigenvalues λ_i and the corresponding eigenvectors $\mathbf{e}_i$ of $m\mathbf{C}$. To this end the following eigenstructure decomposition has to be solved: $m\mathbf{C}\mathbf{e}_i = \lambda_i \mathbf{e}_i$. The eigenvectors $\mathbf{W} = (\mathbf{e}_1, \ldots, \mathbf{e}_n)$ are also called *principal components* and they build an orthogonal basis. Consequently, the matrix $\mathbf{W}$ represents a linear transformation that maps the original data points $\mathbf{x} \in \mathbb{R}^n$ to new data points $\mathbf{y} \in \mathbb{R}^n$ where

$$\mathbf{y} = \mathbf{W}'\mathbf{x}$$

That is, the data is projected into the space spanned by the eigenvectors. The eigenvalues λ_i represent the variance of the data points in the direction of the corresponding

eigenvector $\mathbf{e}_i$. Hence, the eigenvectors $\mathbf{e}_i$ can be ordered according to decreasing magnitude of their corresponding eigenvalues. Consequently, the first principal component points in the direction of the highest variance and, therefore, includes the most information about the data. The second principal component is perpendicular to the first principal component and points in the direction of the second highest variance and so on. In order to project the n-dimensional data $\mathbf{x} \in \mathbb{R}^n$ into a subspace of lower dimensionality, we retain only the $n' < n$ eigenvectors $\mathbf{W}_{n'} = (\mathbf{e}_1, \ldots, \mathbf{e}_{n'})$ with the highest n' eigenvalues. Formally, the mapping of $\mathbf{x} \in \mathbb{R}^n$ to $\tilde{\mathbf{y}} \in \mathbb{R}^{n'}$ is defined by

$$\tilde{\mathbf{y}} = \mathbf{W}_{n'}^{'}\mathbf{x}$$

Note that the larger the resulting dimensionality n' is defined, the greater is the fraction of the captured variance.

3.2 Kernel PCA

Kernel machines constitutes a very powerful class of algorithms in the field of pattern recognition. The idea of kernel machines is to map a vector by means of a kernel function into a higher-dimensional vector space. This procedure offers a very elegant way to construct non-linear extensions of linear algorithms in pattern recognition.

Definition 3 *(Kernel Function) Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ be feature vectors and $\psi : \mathbb{R}^n \rightarrow \mathcal{F}$ a (possibly nonlinear) function where $n \in \mathbb{N}$ and $\mathcal{F}$ is a (possibly infinite dimensional) feature space. A kernel function is a mapping $\kappa : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ such that $\kappa(\mathbf{x}, \mathbf{y}) = \langle \psi(\mathbf{x}), \psi(\mathbf{y}) \rangle$.* □

According to the definition above, the result of a kernel function $k(\mathbf{x}, \mathbf{y})$, applied to two feature vectors **x** and **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. The fundamental observation that makes kernel theory so interesting in the field of pattern recognition is that many of the clustering and classification algorithms can be *kernelized*, i.e. they can be formulated entirely in terms of dot products. Consequently, instead of mapping patterns from the original pattern domain to a feature space and computing the dot product there, one can simply evaluate the value of the kernel function κ in the original pattern space. This procedure is commonly referred to as the *kernel trick* [17, 18] because of its property of determining the dot product in a higher-dimensional feature space immediately without performing the transformation explicitly.

As a matter of fact, PCA can be reformulated in terms of dot products only, i.e. PCA is kernelizable [16]. In order to see this, let us first assume that we apply the mapping $\psi : \mathbb{R}^n \rightarrow \mathcal{F}$ explicitly to our data. For further processing we assume that the data is centered in the kernel feature space[2]. We compute the covariance matrix $\mathbf{C}$ in the feature space $\mathcal{F}$

$$\mathbf{C} = \frac{1}{m} \sum_{i=1}^{m} \psi(\mathbf{x}_i)\psi(\mathbf{x}_i)'$$

[2] We will come back to this point later in this section.

Similarly to PCA the principal components of this feature space are extracted by means of the eigenstructure decomposition

$$\lambda \mathbf{w} = m\mathbf{C}\mathbf{w} \tag{1}$$

Since

$$m\mathbf{C}\mathbf{w} = \sum_{i=1}^{m} (\psi(\mathbf{x}_i)'\mathbf{w})\psi(\mathbf{x}_i)$$

all solutions $\mathbf{w}$ must lie in the span of $\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_m)$. This observation is crucial since it allows us to rewrite $\mathbf{w}$ as a linear combination of the vectors $\psi(\mathbf{x}_i)$ with coefficients $\alpha_i (i = 1, \ldots, m)$

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \psi(\mathbf{x}_i) \tag{2}$$

Furthermore Equation 1 can be rewritten as

$$\lambda \sum_{i=1}^{m} \alpha_i \psi(\mathbf{x}_i) = \sum_{i,j=1}^{m} \alpha_i \psi(\mathbf{x}_j)(\psi(\mathbf{x}_j)'\psi(\mathbf{x}_i))$$

Obviously this is equivalent to m equations ($k = 1, \ldots, m$)

$$\lambda \sum_{i=1}^{m} \alpha_i (\psi(\mathbf{x}_i)'\psi(\mathbf{x}_k)) = \sum_{i,j=1}^{m} \alpha_i (\psi(\mathbf{x}_j)'\psi(\mathbf{x}_k))(\psi(\mathbf{x}_j)'\psi(\mathbf{x}_i)) \tag{3}$$

The tremendous benefit of Equation 3 is that it is entirely formulated in terms of dot products. Hence, we can define a kernel matrix $\mathbf{K}$ by $\mathbf{K}_{i,j} = \psi(\mathbf{x}_i)'\psi(\mathbf{x}_j)$ and replace the dot products in Equation 3 by the kernel function. Formally, $\lambda \mathbf{K}\boldsymbol{\alpha} = \mathbf{K}^2\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m)'$. $\mathbf{K}$ is by definition symmetric and has a set of eigenvectors that span the whole space. Consequently, all solutions α can be obtained by the eigendecomposition of $\mathbf{K}$: $\lambda\boldsymbol{\alpha} = \mathbf{K}\boldsymbol{\alpha}$.

Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_m$ denote the eigenvalues and $\boldsymbol{\alpha^1}, \ldots \boldsymbol{\alpha^m}$ the corresponding eigenvectors of $\mathbf{K}$. The principal components, i.e. eigenvectors $\mathbf{w}$ in the feature space $\mathcal{F}$, need to be normalized to have unit length. This can be achieved by means of the kernel matrix $\mathbf{K}$.

$$||\mathbf{w}||^2 = \left(\sum_{i=1}^{m} \alpha_i \psi(\mathbf{x}_i)\right)' \left(\sum_{j=1}^{m} \alpha_j \psi(\mathbf{x}_j)\right) = \boldsymbol{\alpha'}\mathbf{K}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha'}\boldsymbol{\alpha}$$

Hence, we normalize the coefficients $\boldsymbol{\alpha}$ by $1 = \lambda\boldsymbol{\alpha'}\boldsymbol{\alpha}$. In order to compute a projection of $\psi(\mathbf{x}) \in \mathcal{F}$ onto a subspace spanned by the first n' eigenvectors $\mathbf{W}_{n'}$ the following equation is used

$$\begin{aligned} \tilde{\mathbf{y}}_k &= \mathbf{W'}_{n'}\psi(\mathbf{x}) \\ &= \left(\sum_{i=1}^{m} \alpha_i^k \psi(\mathbf{x}_i)'\psi(\mathbf{x})\right)_{k=1}^{n'} = \left(\sum_{i=1}^{m} \alpha_i^k \kappa(\mathbf{x}_i, \mathbf{x})\right)_{k=1}^{n'} \end{aligned}$$

That is, the mapping can be computed by means of the kernel matrix and the normalized coefficients $\boldsymbol{\alpha^k}$ only.

So far we assumed that the data is centered. Usually, this is not fulfilled, of course. In an explicit fashion one would center the data by

$$\hat{\psi}(\mathbf{x}) = \psi(\mathbf{x}) - \frac{1}{m}\sum_{i=1}^{m}\psi(\mathbf{x}_i)$$

However, it turns out that the centering of the data has not to be done explicitly. That is, the kernel matrix $\mathbf{K}$ can be replaced by $\hat{\mathbf{K}}$ which is defined by

$$\begin{aligned}\hat{\mathbf{K}}_{i.j} &= \hat{\psi}(\mathbf{x})'\hat{\psi}(\mathbf{y}) \\ &= \frac{1}{m}\sum_{i=1}^{m}\psi(\mathbf{x})'\psi(\mathbf{x}_i) - \frac{1}{m}\sum_{i=1}^{m}\psi(\mathbf{y})'\psi(\mathbf{x}_i) + \frac{1}{m^2}\sum_{i,j=1}^{m}\psi(\mathbf{x_i})'\psi(\mathbf{x}_j) \\ &= \frac{1}{m}\sum_{i=1}^{m}\kappa(\mathbf{x},\mathbf{x}_i) - \frac{1}{m}\sum_{i=1}^{m}\kappa(\mathbf{y},\mathbf{x}_i) + \frac{1}{m^2}\sum_{i,j=1}^{m}\kappa(\mathbf{x_i},\mathbf{x}_j)\end{aligned}$$

4 Experimental Results

The experiments in the present paper consider the task of object classification on six different graph datasets. We use three reference systems to compare our algorithm with. First, a nearest neighbor classifier is applied directly in the domain of graphs. Note that as of today – up to a few exceptions, e.g. in [19] – this is the only classifier directly applicable to general graphs. The second reference system is a support vector machine (SVM) applied to the untransformed embedding vectors. The third reference system is an SVM with RBF-Kernel applied to PCA reduced vectors. The RBF-kernel SVM used in this paper has parameters C and γ. C corresponds to the weighting factor for misclassification penalty and $\gamma > 0$ is used in our kernel function $\kappa(\mathbf{u},\mathbf{v}) = exp(-\gamma||\mathbf{u}-\mathbf{v}||^2)$. Note that both the number of dimensions n' retained by PCA and the SVM parameters (C,γ) are optimized on an independent validation set.

Our new approach also makes use of the vector space embedded graphs. However, we apply an RBF-kernel PCA to the resulting vectors instead of a linear transformation. Then an SVM with linear kernel $\kappa(\mathbf{u},\mathbf{v}) = ||\mathbf{u}-\mathbf{v}||$ is applied to the non-linearly transformed and reduced data for the purpose of classification. Hence, besides the number of dimensions n' retained by kernel PCA only the SVM parameter C has to be optimized.

4.1 Datasets

For our experimental evaluation, six data sets with quite different characteristics are used. The datasets vary with respect to graph size, edge density, type of labels for the nodes and edges, and meaning of the underlying objects. Lacking space we give a short description of the data only. For a more thorough description we refer to [6] where the same datasets are used. Note that all of the datasets are divided into three disjoint subsets, i.e. a training set, a validation set, and a test set.

The first dataset used in the experiments consists of graphs representing distorted letter drawings out of 15 classes (LETTER DATABASE). The next dataset consists of graphs representing distorted symbols from architecture, electronics, and other technical fields out of 32 different classes [20] (GREC DATABASE). Next we apply the proposed method to the problem of image classification, i.e. we use graphs representing images out of the four different classes *city*, *countryside*, *snowy*, and *people* (IMAGE). The fourth dataset is given by graphs representing fingerprint images of the NIST-4 database [21] out of the four classes *arch*, *left*, *right*, and *whorl* (FINGERPRINT DATABASE). The fifth graph set is constructed from the AIDS Antiviral Screen Database of Active Compounds [22]. Graphs from this database belong to two classes (*active*, *inactive*), and represent molecules with activity against HIV or not (AIDS DATABASE). The last dataset consists of graphs representing webpages [23] that origin from 20 different categories (*Business*, *Health*, *Politics*, ...) (WEB DATABASE).

4.2 Results and Discussion

In Table 1 the classification accuracies on the test sets of all three reference systems (k-nearest neighbor classifier in the graph domain, SVM applied to unreduced data, SVM based on the PCA reduced data) and our novel approach (SVM based on kernel PCA reduced data) are given.

First of all we observe that the SVM based on the unreduced embedding data, SVM (all), outperforms the first reference system in five out of six cases. PCA-SVM and kPCA-SVM outperform the first reference system even on all datasets. All performance improvements are statistically significant. Obviously, the dissimilarity space embedding graph kernel in conjunction with SVM is a powerful and robust methodology for graph classification.

Comparing the SVM applied on unreduced data with the SVM based on PCA reduced data, we observe that the latter outperforms the former on five out of six datasets (twice with statistical significance). That is, the reduction of the data by means of PCA leads to a further increase in classification accuracy. The SVM in conjunction with the kPCA reduced data also improves five out of six results compared to the second reference system (twice with statistical significance). Note that the percentage of the dimensions retained by kernel PCA is indicated in brackets.

Comparing the classification results achieved with SVM on PCA and kernel PCA reduced data, we observe that in four out of six cases the accuracy is improved and only once it deteriorates by the non-linear transformation. Hence, there is a clear tendency that the new approach with kernel PCA is favorable.

From some higher level of abstraction the last observation reflects the fact that both systems (PCA-SVM and kPCA-SVM) are quite similar. In Fig. 1 both approaches are shown schematically. The first system (solid arrows) transforms the graph domain $\mathcal{G}$ into a vector space $\mathbb{R}^n$. Next linear transformation (PCA) for mapping the data into an n'-dimensional vector space is used. Finally, an RBF-kernel support vector machine is applied which implicitly maps the graphs into a (possibly infinite) feature space $\mathcal{F}_2$ and labels the objects according to a label space Ω. The second system (dashed arrows), however, uses non-linear transformation of the data (kPCA via $\mathcal{F}_1$ to $\mathbb{R}^{n'}$) in conjunction with linear support vector machine. So both systems consist of a linear and

Table 1. Classification accuracy in the original graph domain and in the embedded vector space.

	Reference Systems			Proposed System
Database	k-NN (graph)	SVM (all)	PCA-SVM	kPCA-SVM
LETTER	91.1	92.3 ①	92.7 ①	93.2 (6%) ①
GREC	86.0	91.3 ①	92.9 ①	95.0 (25%) ①②③
IMAGE	59.5	75.2 ①	75.3 ①	76.0 (28%) ①
FINGERPRINT	80.6	80.4	83.1 ①②	83.3 (36%) ①②
AIDS	97.1	98.3 ①	98.2 ①	98.2 (72%) ①
WEBGRAPHS	80.4	82.3 ①	84.0 ①②	83.2 (15%) ①

Z-test for testing statistical significance ($\alpha = 0.05$):

①/②/③ Statistically significant improvement over the first/second/third reference system.

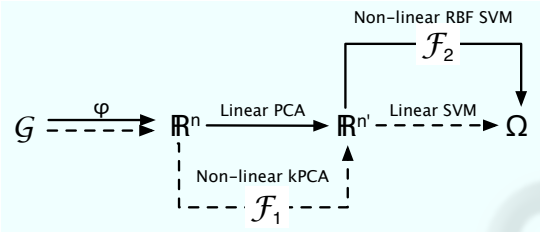

Fig. 1. PCA and Kernel PCA based classification.

non-linear part, and the main difference is the order in which these components are applied.

Apparently, one could use a combination of kernel PCA based transformation and non-linear SVM classification. With this setting, however, poor classification results are achieved on all datasets. We ascribe this performance degradation to overfitting, i.e. the complexity of the model becomes too high, such that the generalization property is crucially compromised.

5 Conclusions

The present paper considers the task of classification where graphs are used as representation formalism for the underlying objects. Graphs provide us with a versatile alternative compared to feature vectors, but they suffer from the lack of mathematical operations in the domain of graphs. However, graph kernels, a relatively novel approach in pattern recognition and related fields, offer an elegant solution to overcome this major drawback of graphs. In the work of this paper we make use of the dissimilarity space embedding graph kernel which maps the graphs explicitly into an n-dimensional vector space where each dimension represents the graph edit distance to a predefined prototype graph. In contrast with previous papers we use the whole set of training patterns as prototypes and apply kernelized principal component analysis (kPCA) on the resulting vectorial description in order to reduce the dimensionality. The main finding of the

experimental evaluation is that kPCA reduction leads to an improvement of the classification accuracy compared to all other systems in general. In particular we observe that the novel approach with kernelized PCA outperforms the former approach with linear PCA in four out of six cases.

References

1. Duda, R., Hart, P., Stork, D.: Pattern Classification. 2nd edn. Wiley-Interscience (2000)
2. Jain, A., Dubes, R.: Algorithms For Clustering Data. Prentice-Hall, Englewood Cliffs, NJ (1988)
3. Jolliffe, I.: Principal Component Analysis. Springer (1986)
4. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. Journal of Pattern Recognition and Artificial Intelligence **18**(3) (2004) 265–298
5. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explorations **5**(1) (2003) 49–58
6. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In Escolano, F., Vento, M., eds.: Proc. 6th Int. Workshop on Graph Based Representations in Pattern Recognition. LNCS 4538 (2007) 383–393
7. Riesen, K., Kilchherr, V., Bunke, H.: Reducing the dimensionality of vector space embeddings of graphs. In Perner, P., ed.: Proc. 5th Int. Conf. on Machine Learning and Data Mining. LNAI 4571, Springer (2007) 563–573
8. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific (2007)
9. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters **1** (1983) 245–253
10. Riesen, K., Neuhaus, M., Bunke, H.: Bipartite graph matching for computing the edit distance of graphs. In Escolano, F., Vento, M., eds.: Proc. 6th Int. Workshop on Graph Based Representations in Pattern Recognition. LNCS 4538 (2007) 1–12
11. Wilson, R., Hancock, E., Luo, B.: Pattern vectors from algebraic graph theory. IEEE Trans. on Pattern Analysis ans Machine Intelligence **27**(7) (2005) 1112–1124
12. Wilson, R., Hancock, E.: Levenshtein distance for graph spectral features. In: Proc. 17th Int. Conf. on Pattern Recognition. Volume 2. (2004) 489–492
13. Luo, B., Wilson, R., Hancock, E.: Spectral embedding of graphs. Pattern Recognition **36**(10) (2003) 2213–2223
14. Duin, R., Pekalska, E.: The Dissimilarity Representations for Pattern Recognition: Foundations and Applications. World Scientific (2005)
15. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press (1996)
16. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10** (1998) 1299–1319
17. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
18. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press (2002)
19. Bianchini, M., Gori, M., Sarti, L., Scarselli, F.: Recursive processing of cyclic graphs. IEEE Transactions on Neural Networks **17**(1) (2006) 10–18
20. Dosch, P., Valveny, E.: Report on the second symbol recognition contest. In Wenyin, L., Lladós, J., eds.: Graphics Recognition. Ten years review and future perspectives. Proc. 6th Int. Workshop on Graphics Recognition (GREC'05). LNCS 3926, Springer (2005) 381–397

21. Watson, C., Wilson, C.: NIST Special Database 4, Fingerprint Database. National Institute of Standards and Technology. (1992)
22. DTP, D.T.P.: AIDS antiviral screen (2004) http://dtp.nci.nih.gov/docs/aids/aids_data.html.
23. Schenker, A., Bunke, H., Last, M., Kandel, A.: Graph-Theoretic Techniques for Web Content Mining. World Scientific (2005)