Kernel k-Means Clustering Applied to Vector Space Embeddings of Graphs

Kaspar Riesen and Horst Bunke

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

Abstract. In the present paper a novel approach to clustering objects given in terms of graphs is introduced. The proposed method is based on an embedding procedure that maps graphs to an n-dimensional real vector space. The basic idea is to view the edit distance of an input graph g to a number of prototype graphs as a vectorial description of g. Based on the embedded graphs, kernel k-means clustering is applied. In several experiments conducted on different graph data sets we demonstrate the robustness and flexibility of our novel graph clustering approach and compare it with a standard clustering procedure directly applied in the domain of graphs.

1 Introduction

Clustering, a common task in pattern recognition, data mining, machine learning, and related fields, refers to the process of dividing a set of given objects into homogeneous groups. Whereas a large amount of clustering algorithms based on pattern representations in terms of feature vectors have been proposed in the literature (see [1] for a survey), there are only few works where symbolic data structures, and in particular graphs, are used [2]. This is rather surprising since the use of feature vectors implicates two severe limitations. First, as vectors describe a predefined set of features, all vectors in one particular application have to preserve the same length regardless of the size or complexity of the corresponding objects. Furthermore, there is no direct possibility to describe relationships among different parts of an object. However, both constraints can be overcome by graph based object representation [3], as graphs allow us to adapt their size to the complexity of the underlying objects and they also offer a convenient possibility to describe relationships among different parts of an object.

The lack of graph clustering algorithms arises from the fact that there is little mathematical structure in the domain of graphs. For example, computing the sum, the weighted sum, or the product of a pair of entities (which are elementary operations, required in many clustering algorithms) is not possible or not defined in a standardized way in the domain of graphs. However, graph kernels, a relatively novel class of algorithms for pattern recognition, offer an elegant solution to overcome this drawback of graph based representation [4]. Originally, kernel methods have been developed for transforming a given feature space into

L. Prevost, S. Marinai, and F. Schwenker (Eds.): ANNPR 2008, LNAI 5064, pp. 24–35, 2008.

another one of higher dimensionality without computing the transformation explicitly for each individual feature vector. Recently, however, as a fundamental extension, the existence of kernels for symbolic data structures, especially for graphs, has been shown [5].

In the present paper we address the problem of graph clustering by means of kernel k-means clustering. The underlying graph kernel functions are based on a dissimilarity space embedding procedure that has been introduced recently. With a comparison based on four different validation indices we empirically confirm that our novel procedure results in better clusterings when compared to results achieved with an approach directly applied in the domain of graphs.

2 Dissimilarity Space Embedding Graph Kernel

In recent years, kernel methods have become one of the most rapidly emerging sub-fields in intelligent information processing [4]. The fundamental observation in kernel theory is that, given a valid kernel function $\kappa : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$, there exists a (possibly infinite dimensional) feature space $\mathcal{F}$ endowed with an inner product $\langle ., . \rangle : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ and a mapping $\phi : \mathbb{R}^n \to \mathcal{F}$ such that $\kappa(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$, for all $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^n$. That is, instead of mapping patterns from $\mathbb{R}^n$ to the feature space $\mathcal{F}$ and computing their scalar product in $\mathcal{F}$, one can simply evaluate the value of the kernel function κ in the original space $\mathbb{R}^n$. This procedure is commonly referred to as *kernel trick*.

What makes kernel theory very interesting is the fact that many algorithms can be kernelized, i.e. reformulated such that only pairwise scalar products rather than explicit vectors are needed[1]. Obviously, by replacing the scalar product by a valid kernel function it is possible to run kernelizable algorithms in a higher dimensional feature vector space $\mathcal{F}$.

Recently, kernel theory has been generalized to the domain of graphs [5]. That is, by means of suitable kernel functions, graphs can be implicitly mapped to vector spaces. Consequently, the whole theory of kernel machines, which has been developed for feature vectors originally, become applicable to graphs. Hence by means of kernel functions one can benefit from both the high representational power of graphs and the rich repository of algorithms available in vector spaces.

Definition 1 (Graph Kernel). *Let $\mathcal{G}$ be a finite or infinite set of graphs, $g_1, g_2 \in \mathcal{G}$, and $\varphi : \mathcal{G} \to \mathbb{R}^n$ a function with $n \in \mathbb{N}$. A graph kernel function is a mapping $\kappa : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ such that $\kappa(g_1, g_2) = \langle \varphi(g_1), \varphi(g_2) \rangle$.* □

According to this definition a graph kernel function takes two graphs g_1 and g_2 as arguments and returns a real number that is equal to the result achieved by first mapping the two graphs by a function φ to a vector space $\mathbb{R}^n$ and then computing the scalar product $\langle \varphi(g_1), \varphi(g_2) \rangle$ in $\mathbb{R}^n$. The kernel function $\kappa(g_1, g_2)$ provides us with a shortcut (kernel trick) that eliminates the need for computing $\varphi(.)$ explicitly.

[1] Such algorithms together with a kernel function κ are commonly termed *kernel machines*.

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. In order to find the most suitable edit path out of all possible edit paths, 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 represents a strong modification of the graph. Consequently, the *edit distance* of two graphs is defined by the minimum cost edit path between two graphs. The edit distance of graphs can be computed, for example, by a tree search algorithm [6] or by faster, suboptimal methods which have been proposed recently (e.g. [7]).

The idea underlying our graph embedding method was originally developed for the problem of embedding sets of feature vectors in a dissimilarity space [8]. In this paper we use the extension of this method to the domain of graphs proposed in [9] for the problem of classification and apply it to clustering. Assume we have available a set of graphs $\mathcal{G} = \{g_1, \ldots, g_N\}$ and a graph dissimilarity measure $d(g_i, g_j)$ (in our case graph edit distance). After having selected a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of $n \leq N$ prototypes from $\mathcal{G}$, we compute the dissimilarity of a given graph $g \in \mathcal{G}$ to each prototype $p \in \mathcal{P}$. 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 $\mathcal{G}$ into a vector of real numbers.

Definition 2 (Graph Embedding). *If $\mathcal{G} = \{g_1, \ldots, g_N\}$ is a set of graphs and $P = \{p_1, \ldots, p_n\} \subseteq \mathcal{G}$ is a set of prototype graphs, the mapping $\varphi_n^{\mathcal{P}} : \mathcal{G} \to \mathbb{R}^n$ is defined as the function*

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

where $d(g, p_i)$ is the graph edit distance between graph g and the i-th prototype.

Regarding the graph embedding procedure, the importance of the prototype set $\mathcal{P} = \{p_1, \ldots, p_n\} \subseteq \mathcal{G}$ is obvious. Recently, different prototype selectors have been proposed in the literature which we adopt in this work [8,9]. Note that $\mathcal{P}$ can be an arbitrary set of graphs in principle. However, for the sake of convenience we always use subsets of $\mathcal{G}$ which are obtained by means of prototype selection methods.

Since the computation of graph edit distance is exponential in the number of nodes for general graphs, the complexity of this graph embedding is exponential as well. However, one can use efficient approximation algorithms for graph edit

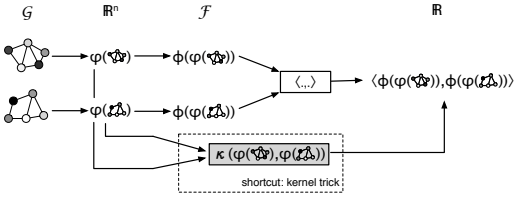


Fig. 1. Graph kernel trick illustrated

distance (e.g. [7] 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.

Clearly, the graph embedding procedure described above provides a foundation for a novel class of graph kernels. Based on the mapping $\varphi_n^{\mathcal{P}}$, one can define a valid graph kernel κ by computing the standard scalar product of two graph maps in the resulting vector space

$$\kappa(g_i, g_j) = \langle \varphi_n^{\mathcal{P}}(g_i), \varphi_n^{\mathcal{P}}(g_j) \rangle$$

Note that, in contrast to some other kernel methods, the approach proposed in this paper results in an explicit embedding of the considered graphs in a vector space. Hence, not only scalar products, but individual graph maps are available in the target space. We observe that this approach is more powerful than conventional graph kernels as not only kernel machines, but also other non-kernelizable algorithms can be applied to the resulting vector representation. Furthermore, based on the resulting graph maps standard kernel functions for feature vectors in $\mathbb{R}^n$ can be applied, mapping the vector space embedded graphs implicitly into a higher dimensional feature space $\mathcal{F}$. An example is the RBF kernel.

$$\kappa_{RBF}(g_i, g_j) = exp\left(-\gamma ||\varphi_n^P(g_i) - \varphi_n^P(g_j)||^2\right) \text{ , with } \gamma > 0.$$

Obviously, in every kernel machine the scalar product can be replaced by $\kappa(g_i, g_j)$ such that these algorithms can be applied to objects originally given in terms of graphs. In Fig. 1 this procedure is schematically illustrated.

3 Kernel k-Means Clustering

The k-means algorithm [10] is one of the most popular clustering algorithms in pattern recognition and related areas. Let us assume that N objects $O = \{\mathbf{o}_1, \ldots, \mathbf{o}_N\}$ are given. Starting with an initial set of k cluster centers, i.e. a set of $k < N$ objects $M_k = \{\mathbf{o}_{(1)}, \ldots, \mathbf{o}_{(k)}\} \subset O$, we assign each of the N objects to the closest cluster centers. Based on this clustering, the cluster centers are recomputed. The two preceding steps, i.e. the assignment of objects to the nearest cluster center and the recomputation of the centers, are repeated until

a predefined termination criterion is met (e.g. no reassignment of objects from one cluster to another has taken place during the last iteration).

The initialization of k-means is commonly done with a random selection of k objects. However, in the present paper a deterministic procedure is applied. The set of initial cluster centers M_k is constructed by iteratively retrieving the median of set O minus the objects already selected. The median of set O is the object $\mathbf{o} \in O$ that minimizes the sum of distances to all other objects in O, i.e. $median = argmin_{\mathbf{o}_1 \in O} \sum_{\mathbf{o}_2 \in O} d(\mathbf{o}_1, \mathbf{o}_2)$. Obviously, this procedure initializes k-means with objects situated in, or near, the center of the set O.

K-means clustering makes use of the squared error criterion as an objective function. Formally, the k-means algorithms finds k clusters $C_1, \ldots, C_k$ such that the objective function

$$f\left(\{C_j\}_{j=1}^k\right) = \sum_{j=1}^{k} \sum_{\mathbf{o}_i \in C_j} d(\mathbf{o}_i, \mathbf{m}_j)$$

is minimized. In this formula, the j-th cluster is denoted by C_j, a clustering by $\{C_j\}_{j=1}^k$, d is an appropriate distance function, and $\mathbf{m}_j$ refers to the mean of cluster C_j.

Note that the objects $\mathbf{o}_i$ can either be graphs or vectors. If the objects are given in terms of graphs, the distance function d is given by the graph edit distance and the mean $\mathbf{m}_j$ of the j-th cluster is defined as the set median graph ($\mathbf{m}_j = argmin_{g_1 \in C_j} \sum_{g_2 \in C_j} d(g_1, g_2)$). In the remainder of the present paper we denote k-means applied to graphs as k-medians. If the objects are given in terms of feature vectors[2], the dissimilarity function d is defined as the Euclidean distance, and $\mathbf{m}_j$ is the mean vector of C_j (i.e. $\mathbf{m}_j = \frac{1}{|C_j|} \sum_{\mathbf{x}_i \in C_j} \mathbf{x}_i$).

A well known drawback of k-means clustering is that the individual clusters C_j need to be spherical in order to achieve satisfactory results. (This drawback directly follows from the minimization of the squared error.) Now let us assume that a function ϕ is given, mapping n-dimensional vectors $\mathbf{x}_i$ into a higher dimensional feature space $\mathcal{F}$. Applying k-means clustering in the resulting feature space $\mathcal{F}$, i.e. finding spherical clusters C_j in $\mathcal{F}$, corresponds to finding (possibly) non-spherical clusters in the original vector space $\mathbb{R}^n$. Hence, this clustering procedure is much more powerful than the conventional k-means algorithm.

The objective function for k-means clustering in the higher dimensional feature space $\mathcal{F}$ can be written as the minimization of

$$f\left(\{C_j\}_{j=1}^k\right) = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} ||(\phi(\mathbf{x}_i) - \mathbf{m}_j|| \text{ , where } \mathbf{m}_j = \frac{\sum_{\mathbf{x}_i \in C_j} \phi(\mathbf{x}_i)}{|C_j|}$$

In fact, it turns out that the k-means algorithm can be written as a kernel machine, i.e. can be reformulated in terms of pairwise scalar products only.

[2] Note that the vectors $\mathbf{x}_i$ in the present paper are embedded graphs by means of the embeddding function $\varphi_n^{\mathcal{P}}(.)$

Formally, the squared Euclidean distance $||(\phi(\mathbf{x}) - \mathbf{m}||^2$ between vector $\mathbf{x}$ and the mean $\mathbf{m}$ of cluster C can be rewritten as

$$||\phi(\mathbf{x}) - \mathbf{m}||^2 = \langle\phi(\mathbf{x}), \phi(\mathbf{x})\rangle + \frac{1}{n^2}\sum_{\mathbf{x}_i \in C}\sum_{\mathbf{x}_j \in C}\langle\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle - \frac{2}{n}\sum_{\mathbf{x}_i \in C}\langle\phi(\mathbf{x}), \phi(\mathbf{x}_i)\rangle$$

Obviously, now we can replace the scalar products $\langle ., . \rangle$ with a valid vector kernel function κ to represent the scalar product in an implicit feature space $\mathcal{F}$ without explicitly computing the transformation $\phi : \mathbb{R}^n \to \mathcal{F}$. That is, one can apply k-means in an implicitly existing feature space $\mathcal{F}$. The resulting procedure is commonly referred to as kernel k-means clustering.

4 Clustering Validation

In order to compare the novel kernel k-means clustering algorithm proposed in this paper with the conventional k-medians algorithm in the graph domain, we use four different validation indices, viz. Dunn [11], C [12], Rand [13], and the Bipartite index. Whereas the two former indices (Dunn and C) do not need any ground truth information, the latter ones (Rand and Bipartite) are defined with respect to the class memberships of the underlying objects. Note that all indices can be applied to both graphs and vectors.

4.1 Dunn Index

We define the distance between two clusters C and C' as $d(C, C') = min\{d(\mathbf{o}_i, \mathbf{o}_j) \mid \mathbf{o}_i \in C, \mathbf{o}_j \in C'\}$. The diameter of a cluster C is given by $\varnothing(C) = max\{d(\mathbf{o}_i, \mathbf{o}_j) \mid \mathbf{o}_i, \mathbf{o}_j \in C\}$ and accordingly the maximum diameter of all k clusters is defined by $\varnothing_{max} = max\{\varnothing(C_i)|1 \leqslant i \leqslant k\}$. Dunn index measures the ratio of the minimum distance of two different clusters and the maximum diameter of a cluster. Formally,

$$\text{Dunn} = min\left\{\frac{d(C_i, C_j)}{\varnothing_{max}}|1 \leqslant i < j \leqslant K\right\}$$

Dunn is considered to be positively-correlated such that higher values indicate higher clustering quality.

4.2 C Index

One defines

$$c(\mathbf{o}_i, \mathbf{o}_j) = \begin{cases} 1 & \text{if } \mathbf{o}_i \text{ and } \mathbf{o}_j \text{ belong to the same cluster} \\ 0 & \text{else} \end{cases}$$

Furthermore, Γ is defined by the sum of all distances of objects belonging to the same cluster, and the number of pairs of objects in the same cluster is denoted by a. Formally,

$$\Gamma = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d(\mathbf{o}_i, \mathbf{o}_j)c(\mathbf{o}_i, \mathbf{o}_j) \qquad a = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} c(\mathbf{o}_i, \mathbf{o}_j)$$

With $min(max)$ we denote the sum of the a smallest (largest) distances $d(\mathbf{o}_i, \mathbf{o}_j)$ where $\mathbf{o}_i \neq \mathbf{o}_j$. The C index is then defined as

$$\mathrm{C} = \frac{\Gamma - min}{max - min}$$

Obviously, the numerator of the C index measures how many pairs of objects of the a nearest neighboring pairs belong to the same cluster. The denominator is a scale factor ensuring that $0 \leq \mathrm{C} \leq 1$. The smaller the C index value is, the more frequently do pairs with a small distance belong to the same cluster, i.e. the higher is the clustering quality.

4.3 Rand Index

For computing the RAND index we regard all pairs of objects $(\mathbf{o}_i, \mathbf{o}_j)$ with $\mathbf{o}_i \neq \mathbf{o}_j$. We denote the number of pairs $(\mathbf{o}_i, \mathbf{o}_j)$ belonging to the same class and to the same cluster with N_{11}, whereas N_{00} denotes the number of pairs that neither belong to the same class nor to the same cluster. The number of pairs belonging to the same class but not to the same cluster is denoted by N_{10}, and conversely N_{01} represents the number of pairs belonging to different classes but to the same cluster. The RAND index is defined by

$$\mathrm{RAND} = \frac{N_{11} + N_{00}}{N_{11} + N_{00} + N_{01} + N_{10}}$$

RAND index measures the consistency of a given clustering, and therefore higher values indicate better clusterings.

4.4 Bipartite Index

In order to compute the BIPARTITE index, we first define the confusion matrix $\mathbf{M}$. Assume a clustering with k clusters $(C_1, \ldots, C_k)$, and a set of l classes $(\Omega_1, \ldots, \Omega_l)$ are given. Note that $\cup_{i=1}^{k} C_i = \cup_{i=1}^{l} \Omega_i$, i.e. the elements underlying the clustering and the classes are identical. Moreover, we define $k = l$, i.e. the number of clusters is equal to the number of classes[3]. The $k \times k$ confusion matrix is defined by

$$\mathbf{M} = \begin{bmatrix} m_{1,1} & \cdots & m_{1,k} \\ \vdots & \ddots & \vdots \\ m_{k,1} & \cdots & m_{k,k} \end{bmatrix}$$

where $m_{i,j}$ represents the number of elements from class Ω_j occuring in cluster C_i. The problem to be solved with this confusion matrix is to find an optimal assignment of the k clusters to the k classes. Such an optimal assignment maximizes the sum of the corresponding cluster-class values $m_{i,j}$. Formally, one has to find a permuation p of the integers $1, 2, \ldots, k$ maximizing $\sum_{i=1}^{k} m_{ip_i}$. Let p be

[3] For the sake of convenience we use k to denote both the number of clusters and the number of classes from now on.

the optimal permuation. The BIPARTITE index (BP index for short) is defined as

$$\mathrm{BP} = \frac{\sum_{i=1}^{k} m_{ip_i}}{N}$$

Note that BP gives us the maximum possible classification accuracy of the given clustering. The computation of the BP index can be efficiently accomplished by means of Munkres' algorithm [14].

Note that other validation indices could be also used. However, we feel that a validation based on the four indices proposed covers the different aspects of cluster quality evaluation quite well and we leave an analysis involving additional indices to future work.

5 Experimental Results

The intention of the experimental evaluation is to empirically investigate whether kernel k-means based on the proposed graph embedding outperforms the standard k-medians clustering algorithm in the original graph domain. For our novel approach the most widely used RBF kernel $\kappa_{RBF}(g_i, g_j)$ as defined in Section 2 is used. Class information is available for all of our graph data sets. Therefore the number of clusters k is defined for each data set as the number of classes in the underlying graph set.

5.1 Databases

For our experimental evaluation, six data sets with quite different characteristics are used. The data sets 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 [9] where the same data sets are used for the task of classification.

The first database used in the experiments consists of graphs representing distorted letter drawings out of 15 classes (Letter). Next we apply the proposed method to the problem of image clustering, i.e. we use graphs representing images out of two categories (*cups, cars*) from the COIL-100 database [15] (COIL). The third data set is given by graphs representing fingerprint images of the NIST-4 database [16] out of the four classes *arch*, *left*, *right*, and *whorl* (Fingerprint). The fourth set is given by the Enzyme data set. The graphs are constructed from the Protein Data Bank [17] and labeled with their corresponding enzyme class labels (*EC 1,. . ., EC 6*) (Enzymes). The fifth graph set is constructed from the AIDS Antiviral Screen Database of Active Compounds [18]. Graphs from this database belong to two classes (*active, inactive*), and represent molecules with activity against HIV or not (AIDS). The last data set consists of graphs representing webpages [19] that originate from 20 different categories (*Business*, *Health*, *Politics*, . . .) (Webgraphs).

Each of our graph sets is divided into two disjoint subsets, viz. validation and test set. The validation set is used to determine those meta parameters of

the clustering algorithm which cannot be directly inferred from the specific application. For k-medians clustering in the original graph domain only the cost function for graph edit distance has to be validated. For our novel approach, however, there are three additional parameters to tune, viz. the prototype selection method (PS), the number of prototypes n (dimensionality of the vector space $\mathbb{R}^n$), and the parameter γ in the RBF kernel. For each of the four validation indices, these three meta parameters are optimized individually on the validation set. Thereafter the best performing parameter combination is applied to the independent test set.

5.2 Results and Discussion

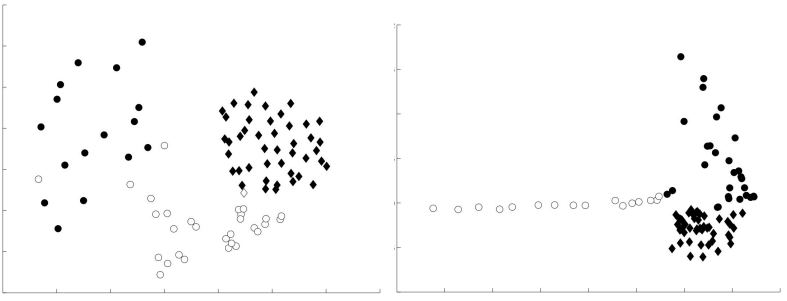
In Table 1 the clustering validation indices for both the k-medians clustering in the original graph domain (GD) and kernel k-means in the embedding vector space (VS) are given for all test data sets. Regarding the DUNN index we observe that the clustering based on the vector space embedded graphs outperforms the clustering in the original graph domain in three out of six cases, i.e. the clusterings in the vector space are not necessarily better than the clusterings in the original graph domain. This finding can be explained by the fact that DUNN's index is very instable in presence of outliers since only two distances are considered. Regarding the other indices the superiority of our novel approach compared to the reference system becomes obvious. Kernel k-means outperforms the k-medians clustering under C, RAND, and BP on all data sets. That is, with the novel procedure pairs with small distances are more frequently in the same cluster, and the clusterings in the embedding space are more accurate and consistent according to the ground truth.

At first glance the meta parameters to be tuned in our novel approach seem to be a kind of drawback. However, since the k-means algorithm is able to find spherical clusters only, these meta parameters establish a powerful possibility to optimize the underlying vector space embedding with respect to a specific validation index. In Fig. 2 clusterings on the COIL database are illustrated as an example. Note that in these illustrations the different colors (black and white) refer to the cluster assignment, while the different shapes (circle and diamond) reflect the class membership of the respective points.

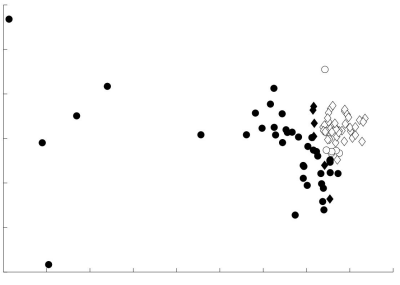
In Fig. 2 (a) the original graphs and the clustering found by the reference system (k-medians) are projected onto the plane by multidimensional scaling

Table 1. Clustering results on the data sets in the graph domain (GD) and the vector space (VS). Bold numbers indicate superior performance over the other system.

	Dunn		C		Rand		BP	
Data Set	GD	VS	GD	VS	GD	VS	GD	VS
Letter	0.016	**0.157**	0.419	**0.026**	87.30	**90.76**	22.90	**46.27**
COIL	0.132	**0.142**	0.377	**0.053**	69.01	**76.63**	81.11	**86.67**
Fingerprint	**0.209**	0.057	0.094	**0.017**	32.20	**77.66**	45.20	**68.40**
Enzymes	0.027	**0.036**	0.591	**0.021**	49.23	**72.37**	22.00	**27.00**
AIDS	**0.054**	0.044	0.015	**0.000**	83.06	**83.17**	90.67	**90.73**
Webgraphs	**0.064**	0.042	0.111	**0.016**	88.51	**90.08**	53.72	**53.97**



(a) Clustering in the graph domain. (b) Clustering in the vector space (Dunn and C optimized)



(c) Clustering in the vector space (Rand and BP optimized)

Fig. 2. Different clusterings of the same data. (Different shapes refer to different classes while different colors refer to different clusters.)

(MDS) [20]. In Fig. 2 (b) and (c) the same graphs processed by different vector space embeddings and kernel k-means clustering in the embedding space are displayed. Fig. 2 (b) shows the vector space embedded graphs and the clustering which is optimized with respect to the validation indices DUNN and C[4]. In this case the underlying vector space is optimized such that the resulting clusters are more compact and better separable than the clusters achieved in the original graph domain (Fig. 2 (a)). In other words, the two clusters of white and black elements can be much better approximated by non-overlapping ellipses than in Fig. 2 (a). Note that for both indices DUNN and C the class membership is not taken into account but only the size and shape of the clusters. In Fig. 2 (c) the embedding and clustering are optimized with respect to RAND and BP. In this case the embedding and clustering are optimized such that spherical clusters are able to seperate the data with a high degree of consistency and accuracy according to ground truth. This can also be seen in Fig. 2 (c) where the shape and color, i.e. class- and cluster membership, correspond to each other more often than in Fig. 2 (a).

[4] Coincidentally, for both indices the best performing parameter values are the same.

Summarizing, the embedding process lends itself to a methodology for adjusting a given data distribution such that the clustering algorithm becomes able to achieve good results according to a specific validation criterion. This explains the findings reported in Table 1.

6 Conclusions

In the present paper we propose a procedure for embedding graphs in an n-dimensional vector space by means of prototype selection and graph edit distance. Based on this vector space embedding of graphs, a standard kernel function for feature vectors – the RBF kernel – is applied. This leads to an implicit embedding of the graph maps in a feature space $\mathcal{F}$. These implicit embeddings are then fed into a kernel k-means clustering algorithm. The power of our novel approach is threefold. First, it makes the k-means clustering algorithm available to the graph domain. Because of the lack of suitable procedures for computing the mean of a graph population, only k-medians algorithm has been traditionally applied to graphs. Secondly, by means of the embedding procedure we gain the possibility to apply kernel k-means clustering to data that are not spherically structured, as implicitly assumed by the k-means clustering algorithm. Thirdly, by means of the embedding parameters, the resulting vector space can be adjusted with respect to a specific validation index. The applicability and performance of our novel approach is tested on six different graph sets with four clustering validation indices. According to the Dunn index our novel approach outperforms the reference system in three out of six cases. The other indices C, Rand, and BP indicate that our novel approach outperforms the reference system even on all data sets.

In future work we will study additional clustering validation indices and performance measures, for example, cluster stability [21]. Moreover, other clustering algorithms applicable to vector space embedded graphs will be investigated.

Acknowledgements

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

References

1. Jain, A., Murty, M., Flynn, P.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
2. Englert, R., Glantz, R.: Towards the clustering of graphs. In: Kropatsch, W., Jolion, J. (eds.) Proc.2nd Int.Workshop on Graph Based Representations in Pattern Recognition, pp. 125–133 (2000)
3. 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), 265–298 (2004)

4. Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge (2002)
5. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explorations 5(1), 49–58 (2003)
6. Bunke, H., Allermann, G.: Inexact graph matching for structural pattern recognition. Pattern Recognition Letters 1, 245–253 (1983)
7. Riesen, K., Neuhaus, M., Bunke, H.: Bipartite graph matching for computing the edit distance of graphs. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 1–12. Springer, Heidelberg (2007)
8. Duin, R., Pekalska, E.: The Dissimilarity Representations for Pattern Recognition: Foundations and Applications. World Scientific, Singapore (2005)
9. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)
10. MacQueen, J.: Some methods for classification and analysis of multivariant observations. In: Proc. 5th. Berkeley Symp., vol. 1, pp. 281–297. University of California Press 1 (1966)
11. Dunn, J.: Well-separated clusters and optimal fuzzy partitions. Journal of Cybernetics 4, 95–104 (1974)
12. Hubert, L., Schultz, J.: Quadratic assignment as a general data analysis strategy. British Journal of Mathematical and Statistical Psychology 29, 190–241 (1976)
13. Rand, W.: Objective criteria for the evaluation of clustering methods. Journal of the American Statistical Association 66(336), 846–850 (1971)
14. Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the Society for Industrial and Applied Mathematics 5, 32–38 (1957)
15. Nene, S., Nayar, S., Murase, H.: Columbia Object Image Library: COIL-100. Technical report, Department of Computer Science, Columbia University, New York (1996)
16. Watson, C., Wilson, C.: NIST Special Database 4, Fingerprint Database. National Institute of Standards and Technology (1992)
17. Berman, H., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T., Weissig, H., Shidyalov, I., Bourne, P.: The protein data bank. Nucleic Acids Research 28, 235–242 (2000)
18. DTP, DTP.: AIDS antiviral screen (2004), `http://dtp.nci.nih.gov/docs/aids/aids_data.html`
19. Schenker, A., Bunke, H., Last, M., Kandel, A.: Graph-Theoretic Techniques for Web Content Mining. World Scientific, Singapore (2005)
20. Cox, T., Cox, M.: Multidimensional Scaling. Chapman and Hall, Boca Raton (1994)
21. Kuncheva, L., Vetrov, D.: Evaluation of stability of k-means cluster ensembles with respect to random initialization. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(11), 1798–1808 (2006)