Feature selection on node statistics based embedding of graphs

Jaume Gibert[a,*], Ernest Valveny[a], Horst Bunke[b]

[a]*Computer Vision Center, Universitat Autònoma de Barcelona, Edifici O, Campus UAB, 08193 Bellaterra (Spain)*
[b]*Institute for Computer Science and Applied Mathematics, University of Bern, Neubrückstrasse 10, CH-3012 Bern (Switzerland)*

Abstract

Representing a graph with a feature vector is a common way of making statistical machine learning algorithms applicable to the domain of graphs. Such a transition from graphs to vectors is known as graph embedding. A key issue in graph embedding is to select a proper set of features in order to make the vectorial representation of graphs as strong and discriminative as possible. In this article, we propose features that are constructed out of frequencies of node label representatives. We first build a large set of features and then select the most discriminative ones according to different ranking criteria and feature transformation algorithms. On different classification tasks, we experimentally show that only a small significant subset of these features is needed to achieve the same classification rates as competing to state-of-the-art methods.

Keywords: Structural Pattern Recognition, Graph Embedding, Feature Ranking, PCA, Graph Classification

1. Introduction

Graph representations have gained quite some popularity in the past years. They offer a strong paradigm in terms of representational power mainly thanks to their ability to encode relations among the elements of a given pattern. Graphs have been extensively used in bioinformatics [1, 2, 3], computer network analysis [4, 5], web content mining [6, 7, 8], image analysis [9, 10] and in many other subfields of computer science. For an extensive review on graph-based representations for pattern recognition we refer to [11].

Classically, graph matching and graph processing algorithms rely on finding common structures between instances of graphs. These methods are usually computationally very costly. This is due to the lack of a natural ordering in the graph nodes and to the fact that, because of noise and distortions, different graph instances of the same object might have a different number of nodes and edges, and different labels. We thus encounter a situation where a strong representational paradigm rests on a complicated algorithmic basis.

*Corresponding author. Tel: +34.93.581.4090
Email address: `jgibert@cvc.uab.es` (Jaume Gibert)

Therefore, it is generally not straightforward to define data processing and machine learning algorithms that are directly applicable to the domain of graphs.

Modern approaches try to avoid the high computational complexity arising from graph representations and to provide more algorithmic tools for their processing. One of the main and most promising directions to overcome the lack of efficient processing algorithms is graph embedding into vector spaces. The main idea is to associate a feature vector to each graph so that it enables the access to any learning machine that has been originally developed for statistical feature vectors. Of course, the choice of the features that are used for constructing such a vectorial representation of graphs is of crucial importance.

Various examples of graph embedding can be found in the literature. A first family of algorithms can be found in the context of chemo-informatics. The authors of [12, 13] assign to every molecule (represented as a graph) a feature vector whose components are frequencies of appearance of specific knowledge-dependent substructures in the graph. Another family of embedding methods is based on spectral properties of graphs. In [14, 15, 16, 17] the authors extract different features from the eigen-decomposition of matrices regarding the topology of graphs. Another line of investigation, based on the dissimilarity representation studied in [18, 19], is proposed by the authors of [20]. They classify and cluster graphs using a vectorial representation whose components are features expressing the distances to a set of predefined prototype graphs. Finally, other works embed every node of a graph into a feature point so that the problem of graph matching is translated into that of point set alignment [21, 22].

In previous works [23, 24] we have proposed another embedding methodology based on statistics on the node attributes of the graphs. In particular, we initially select a set of representative elements of the node attributes in the set of graphs by using clustering methods. Each node in a graph can be described by one or more of these representatives. Thus, we accumulate the amount of importance of each representative in the graph. Moreover, we can also make use of the edge information and translate the graph topology into relations between these representatives. This approach has empirically demonstrated its efficiency and good performance on several classification scenarios. Nevertheless, the features we construct are based on a set of elements that we build without prior knowledge and this may lead to noisy or redundant features and to high dimensional and sparse vectorial representations of graphs. The aim of this work is to apply feature selection algorithms to this vectorial representation of graphs so that we can discover the relevant features in order to avoid having too high dimensional vectors while keeping the recognition levels comparable to other graph classification approaches.

A preliminary version of this work appeared in [25]. In the current paper we have extended the embedding methodology so that features that are extracted from graphs can describe distortions better than the original version. This is done by fuzzifying the assignment from nodes to representative elements and from edges to relations between these elements. Moreover, the vectorial representations that we extract from graphs have been put under a wider perspective of feature selection algorithms, in the sense that several approaches of different nature have been employed. Also, the experimental part has been extended by using a number of datasets considerably larger and by performing a more exhaustive comparison of our results with reference systems.

The rest of the article is organized as follows. In the next section we formally present the embedding methodology. In Section 3, the application of different feature selection algorithms to the embedding is properly described. In Section 4, the experimental part is presented in detail and, finally, Section 5 finishes the article by drawing the conclusions of this work.

2. Graph embedding by node representatives

In this section we give a formal description of the graph embedding procedure that is used in this work. We define the embedding of a graph into a vector space in terms of unary and binary relations between node representatives. In particular, the node labels of all graphs are clustered using Fuzzy kMeans (see Section 2.2) obtaining a set of representatives that model the labels' distribution. By computing statistics of how much each of these representatives is present in each graph (both in terms of nodes and edges occurrences) we can provide a vector representation of graphs.

2.1. Definition

We now formally describe the complete embedding methodology. A graph g is a four-tuple $g = (V, E, L_V, L_E)$, where V is a non-empty set of nodes, $E \subseteq V \times V$ is the set of edges and L_V and L_E are the corresponding labelling sets. Suppose we are given a set of N graphs $\mathcal{G} = \{g_1, \ldots, g_N\}$. For all $i \in \{1, \ldots, N\}$, the set of nodes attributes is $L_{V_i} = \mathbb{R}^d$ and edges remain unattributed (other situations are not considered in this work). Let $\mathcal{P} \subset \mathbb{R}^d$ be the set of all node labels in all the graphs of $\mathcal{G}$. Furthermore, let $\mathcal{W} = \{w_1, \ldots, w_n\}$ be a set of n representatives of all vectors in $\mathcal{P}$. Elements in $\mathcal{W}$ do not necessarily belong to $\mathcal{P}$. For each node in a graph, we seek how much is this node being represented by each of the elements in the set of representatives. Formally, we define the function

$$\begin{aligned} \lambda_s : V &\longrightarrow S_n^+ \subset \mathbb{R}^n \\ v &\longmapsto \lambda_s(v) = (p_1(v), \ldots, p_n(v)), \end{aligned} \tag{1}$$

where $p_i(v) = P(v \in w_i)$ is the probability of the node v being represented by the representative w_i and S_n^+ is the positive orthant of the L_1-hypersphere in $\mathbb{R}^n$. In other words, we put these degrees of belongingness under a probability framework by requiring that $p_i(v) \geq 0$ and $\sum_{i=1}^n p_i(v) = 1$.

We can now compute statistics on the influence of each representative in each graph. This is, we count how much weight each element in the set of representatives receives from all the nodes in the graph. Then, we define unary features for our vectorial representation of graphs based on this amount of weight. Formally,

$$U_i = \#(w_i, g) = \sum_{v \in V} p_i(v). \tag{2}$$

We can also extract features from edges in the graphs. Assume we have an edge $(u, v) \in E$ and we have available the corresponding assignment representations of the source and the target nodes:

$$\lambda_s(u) = (p_1(u), \dots, p_n(u)),$$
$$\lambda_s(v) = (p_1(v), \dots, p_n(v)).$$

From these two vectors of probabilities we want to find out how much the edge (u, v) is contributing to the relation between every pair of representatives. We consider all possibilities of nodes connecting every two representatives, and thus, an edge $(u, v) \in E$ will contribute to all relations between any two representatives. In particular, we define

$$\begin{aligned} B_{ij} &= \#(w_i \leftrightarrow w_j, g) \\ &= \sum_{(u,v) \in E} p_i(u)p_j(v) + p_j(u)p_i(v). \end{aligned} \tag{3}$$

The intuition behind (3) is based on walks of length 1 on the graph. The edge $(u, v) \in E$ of a graph g will contribute to the relation $w_i \leftrightarrow w_j$ the probability of representing a path between w_i and w_j, that can be obtained from the probability of assigning u and v to w_i and w_j, respectively, this is $p_i(u)p_j(v)$. Then, since we work with undirected graphs, we should also consider the path back and aggregate the probability of travelling from w_j to w_i, *i.e.*, $p_j(u)p_i(v)$.

Now that these features are defined (Eqs. (2) and (3)), we can formalize the embedding of a graph into a vector space.

Definition 1 (Graph Embedding). Given a set of node representatives $\mathcal{W} = \{w_1, \dots, w_n\}$, we define the embedding of a graph g into a vector space as the vector

$$\varphi_{\mathcal{W}}(g) = (U_1, \dots, U_n, B_{11}, \dots, B_{ij}, \dots, B_{nn}), \tag{4}$$

where $1 \leq i \leq j \leq n$, $U_i = \#(w_i, g)$ and $B_{ij} = \#(w_i \leftrightarrow w_j, g)$.

Note that computing these features has a very low cost since only simple operations between node labels have to be performed.

2.2. Selection of representatives: Fuzzy kMeans

The embedding methodology described in the previous sections depends on a set of representative elements of the node labels. In a previous work [26], we have investigated the dependence of the embedding on the way this set of representatives is selected. Such a selection is not the focus of this article and we just use the Fuzzy kMeans algorithm [27] to construct the set of representatives and to assign probability values. The reason is that this is the method that experimentally provided more stable results.

The main idea of Fuzzy kMeans is to assign to a point $x \in \mathcal{P}$ a degree of belongingness to each cluster center in $\mathcal{W}$, which is inversely proportional to the distance between x and the cluster center. This leads to

$$p_i(x) = \alpha \cdot \left(\frac{1}{\parallel x - w_i \parallel_2} \right)^s, \tag{5}$$

where α is a constant assuring that $\sum_{i=1}^{n} p_i(x) = 1$ and s is a parameter that controls the amount of *fuzzyness* the user is giving to the assignment. The larger is s, the more weight is given to points close to the centres. In our experiments we use $s = 2$.

2.3. Dimensionality, sparsity and feature correlation

The steps that define the embedding provide us with a representation of graphs that might suffer from some problems. First, since the selection of representatives is an unsupervised task, we do not have any control on these elements. We might be selecting irrelevant points in the set of representatives for the task of graph representation. Moreover, the number of edge based features is quadratic in the size of the representative set, leading to high dimensional vectors for large sets of representatives. This may weaken efficient operations between the vectorial representations of graphs.

Another consequence of the quadratic number of edge based features could be some sparsity in the vectorial representations. The selection of representatives might come up with some elements that are barely represented in the graphs under consideration, and also, to representatives the relations between which are not present in the vectors. These situations would lead to too many zero-valued features that would impoverish the final representation. As a final concern, we could also wonder how much correlation is there between the node-based and edge-based features extracted from a specific element in the representative set. Correlation between features is not desired and we ought to tackle this scenario.

3. Feature selection

Feature selection algorithms try to select a proper subset of features such that the performance of a certain learning algorithm is improved [28]. Some of these algorithms are based on searching the most relevant features. Search strategies can be split into *forward* selection and *backward* elimination. The former starts with an empty set and iteratively adds important features, while the later keeps eliminating useless features from the set of all features. Also *floating* search strategies have been proposed that allow to variably add relevant and remove useless features [29].

In this work, we will use two different kind of methods. The first group are those methods that assign a weight to every feature in its original form. We select three algorithms from the literature that are well established and that have proved their good performance on different scenarios. The first one is based on the ability to discriminate among classes in terms of relative distances between feature values, the second one on entropy measurements and the third one is based on the SVM classifier. The second category of feature selection methods is formed by those methods that initially transform the original features and then rank the resulting features by means of some measurements coming from the transformation itself. In particular we use variance-based methods such as PCA and Kernel PCA.

3.1. Ranking methods

Ranking methods are based on a ranking map that gives to every feature at hand a certain value that is eventually used to rank it with respect to the others. Based on different ranking strategies we have different ranking methods.

3.1.1. Relief

The Relief algorithm is a classical ranking method that is based on the ability of features to discriminate between different classes [30]. For every instance of a given feature, the closest value among elements of the same class (*Near Hit*) and the closest value among elements of other classes (*Near Miss*) are found. Then a weight is given to every feature in terms of the distances of every sample to the Near Hit and the Near Miss. This is, given the set $\mathcal{S}$ of m samples of feature i, we compute the rank value as

$$\omega_i = \frac{1}{m} \sum_{x \in \mathcal{S}} |x - Z_x^-| - |x - Z_x^+|, \tag{6}$$

where Z_x^+ and Z_x^- are the near hit and the near miss of the sample x, respectively. It is clear that a good feature should give low values to the distances between each sample and its near hit and high values to the distances to the near miss. Thus, a good feature should have a high ranking value ω_i.

3.1.2. Mutual Information

Mutual information is a measure of dependency between random variables. Let X and Y be two random variables. Their mutual information $I(X, Y)$ is defined by

$$I(X,Y) = \int_Y \int_X p(x,y) \log\left(\frac{p(x,y)}{p(x)p(y)}\right) dxdy, \tag{7}$$

where $p(x, y)$, $p(x)$ and $p(y)$ are the joint and the marginal probability density functions, respectively. By using mutual information, one is capable to find those features with largest relevance with respect to the existing classes.

Our data needs for a discretization of the feature values so that integrals can be reduced to sums. To discretize features, we make use of the multi-interval discretization of continuous-valued attributes algorithm described in Ref. [31]. Once discretized, if we consider $X^{(i)}$ the set of discrete values that the feature $\mathbf{x}_i$ can take, the mutual information between $\mathbf{x}_i$ and the set of class labels Ω reduces to

$$I(\mathbf{x}_i, \Omega) = \sum_{\omega \in \Omega} \sum_{x_i \in X^{(i)}} p(x_i, \omega) \log \left(\frac{p(x_i, \omega)}{p(x_i)p(\omega)} \right), \tag{8}$$

where both the joint and marginal density functions can be estimated by counting the instances in the training set. Finally, features are ranked based on their mutual information in a forward selection fashion.

3.1.3. SVM based ranking

The last ranking method we will use in our experimental evaluation was originally proposed in [32] and is based on the support vector machine classifier (SVM). An SVM classifier seeks for a hyperplane $f(x) = \langle w, x \rangle + b$, where $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$, that best separates the involved classes. The components of the vector w can be used as feature rankings since they weight how much each of the components (features) influences the final decision boundary. The idea is thus to consider those features with high values in the vector w as relevant features.

3.2. PCA-based methods

The other category of feature selection methods considered in this paper do not rank the original features but, instead, a transformation of those.

3.2.1. Principal Component Analysis

Given a set of N feature vectors $x_1, \dots, x_N \in \mathbb{R}^n$, principal component analysis (PCA) finds a linear transformation of the data $y_i = Ax_i \in \mathbb{R}^m$ so that linear correlation among the new features is reduced and the new $m \leq n$ features capture most of the variance. Such transformation is obtained by an orthogonal mapping where each column of the matrix A is an eigenvector of the covariance matrix of the centered original data. These eigenvectors $v_1, \dots, v_n$ are called principal components and are ordered from greater to smaller variance. By taking $m \leq n$ principal components, the dimensions are reduced and most of the variance is being kept.

3.2.2. Kernel PCA

Kernel principal component analysis (kPCA) is a non-linear generalization of PCA by means of the kernel trick [33]. kPCA finds linear behaviors of the data in the implicit space of the kernel function, which in general correspond to non-linear properties of the input patterns. The projection of $\phi(x)$ onto a (non-linear) principal component u_p of the input feature space is given by

$$u_p \cdot \phi(x) = \sum_{j=1}^{N} \beta_j^p \kappa(x_j, x) \tag{9}$$

where $\beta^p = (\beta_1^p, \ldots, \beta_N^p) \in \mathbb{R}^N$ is the p-th leading eigenvector of the kernel matrix $K = (\kappa(x_s, x_t))_{1 \leq s,t, \leq N}$. The final transformation is given by $y_i = (u_1 \cdot \phi(x_i), \ldots, u_n \cdot \phi(x_i))$. Exactly as in PCA, by keeping $m \leq n$ principal components one captures most of the variance in $\mathcal{H}$.

Besides standard PCA, in the experimental part of this work (Section 4), we have used two other well-known and used kernel functions, namely, the radial basis function -or Gaussian kernel- and the χ^2 kernel:

$$\kappa_{\text{rbf}}(x, y) = \exp(-\gamma \cdot \parallel x - y \parallel^2),\ \gamma > 0 \tag{10}$$

$$\kappa_{\chi^2}(x, y) = \exp(-\gamma \cdot d_{\chi^2}(x, y)),\ \gamma > 0 \tag{11}$$

where $\parallel \cdot \parallel$ stands for the L_2-norm and $d_{\chi^2}(\cdot, \cdot)$ is the χ^2 distance, a commonly used tool for histogram-based feature vectors [34] and defined by

$$d_{\chi^2}(x, y) = \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)}. \tag{12}$$

4. Experimental evaluation

4.1. Databases of graphs

In this work, we have considered both synthetic and real datasets of graphs. All datasets are publicly available from the IAM graph database repository [35].

The first three datasets of graphs are the *Letter Databases*, which represent synthetically distorted letter drawings. Starting from a manually constructed prototype of every of the 15 Roman alphabet letters that consist of straight lines only, different degrees of distortion are applied: low, medium and high. Each ending point of a line is represented by a node of the graph and labelled with its (x, y) coordinates. Unlabelled edges represent the existing lines in the letters by linking the corresponding nodes.

The next set of graphs is the *Digits Database.* This data set is representing handwritten digits [36]. The digits were originally acquired by recording the pen position at constant steps of time. The sequence of (x, y) coordinates constitute the set of nodes of the graphs (and their corresponding labels), while consecutive nodes are linked by an undirected edge.

The *Fingerprint Database* is the next database of graphs. It consists of graphs that are obtained from a subset of the NIST-4 fingerprint image database [37] by means of particular image processing operations. Ending point and bifurcations of the skeleton of the processed images constitute the (x, y)-attributed nodes of the graphs, plus some nodes that are inserted between these points. All points connected through a ridge in the image skeleton are connected with an unlabelled edge.

The sixth graph dataset is the *GREC Database* [38], which represents architectural and electronic symbols under different levels of noise. Depending on the level of noise, different morphological operations are applied to the symbols until lines of one pixel width are obtained. Intersections and corners of such lines constitute the set of nodes, which are labelled with their position on the 2-dimensional plane.

Table 1: Characteristics of the different datasets. Size of the training (tr), validation (va) and test (te) sets, the number of classes (#Cls), the average number of nodes and edges (An/Ae) and the maximum number of nodes and edges (Mn/Me).

Dataset	Size tr, va, te	#Cls	An/Ae	Mn/Me
Letter low	750, 750, 750	15	4.7/3.1	8/6
Letter medium	750, 750, 750	15	4.7/3.2	9/7
Letter high	750, 750, 750	15	4.7/4.5	9/9
Digits	1000, 500, 2000	10	8.9/7.9	17/16
Fingerprints	500, 300, 2000	4	5.4/4.4	26/25
GREC	286, 286, 528	22	11.5/12.2	25/30
COIL	2400, 500, 1000	100	21.5/54.2	77/222

Finally, the *COIL database* is a subset of the COIL-100 database [39]. The original set of images is representing 100 different objects by taking samples of these objects at 5 degrees intervals of rotation. The set of graphs we use in this work is restricted to images at every 15 degrees of rotation only. Graphs are extracted by considering salient points in the images using the Harris corner detection algorithm [40], labelling these points with their corresponding coordinates on the 2D plane, and linking points using a Delaunay triangulation.

Some of these datasets include edge labels which were not considered in the experiments. Each of the datasets is split into a training set, a validation set and a test set. In Table 1, the size of the resulting subsets and other relevant information concerning the datasets is provided.

4.2. Ranking methods

The choice of the representative elements is of crucial importance because the semantics of the representation will depend on them. Nevertheless, we have no presumptive manner to select them beforehand and thus we assume this step as one to be validated. In particular, for each dataset, we have built sets of representative elements of different sizes, starting from 5 elements up to 100, in steps of 5, leading to 20 different vectorial representations for each dataset of graphs.

For each of these representations, we have to select the subset of features that best solves a specific task. In particular, we are interested in solving a classification problem. Once all the features in one of these representations are ranked, we can construct a structure of nested features: from the most important one, we iteratively add the rest of them in decreasing order of importance, obtaining several subsets of features.

These nested features are the candidates for the optimal subset of features that we seek for each representation. If we try out all these subset candidates for all the different vectorial representations that we have created, the computational complexity of the validation stage increases dramatically. To avoid this situation, we do not use all candidates but instead we use just some of them. In particular, we use those subsets that correspond to the most relevant feature (first subset) and that of all features (last subset), plus the subsets containing $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{3}{8}$, $\frac{1}{2}$, $\frac{5}{8}$, and $\frac{3}{4}$ of the most relevant features.

For the Relief and Mutual Information cases, we compute all the nested feature subsets since ranking each feature is not a complex task. For the SVM ranking case, we proceed differently. Instead of ranking all features and then building up the nested structure, we directly build such a structure in its reduced version. We initially train an SVM classifier with all features and then remove all of them except for $\frac{3}{4}$ of the most relevant features. With the remaining ones, we proceed analogously and, after training, remove all except for $\frac{5}{8}$ of the most relevant features. This procedure is repeated until, finally, we end up with the most relevant feature.

In Fig. 1, we show classification results on the validation sets of the Fingerprints and the Digits datasets. These are two representative examples of the general behavior observed. In particular, we use a k-Nearest Neighbour (kNN) classifier together with the χ^2 distance (see Eq. (12)). We plot, for each ranking method, different curves that correspond to different choices of the size of the representative set (10, 30, 50 and 75 representatives). And we plot these curves in two different ways: in relative and in absolute terms regarding the size of the feature subset that is being used for classification. Moreover, on each relative curve and using a Z-test of statistical significance with a confidence level of $\alpha = 0.05$, we draw a dot at the best configuration, where by *best* we understand the one based on the least number of features from all those configurations that are statistically at the same level of significance than the maximum accuracy rate obtained.

The main behavior we can observe in Fig. 1 is the fact that, in general, those configurations that are constructed with larger vocabularies need a -relatively- smaller number of components in order to reach the proper subset of features. This can be seen on the relative curves since the dots corresponding to the larger sets of representatives can be found before those of the smaller ones. In the absolute curves this behaviour can also be noticed by the fact that curves tend to flatten rapidly when the representative set becomes larger. This situation also suggests that large sets of representatives introduce noisy and redundant features to a higher degree than smaller sets. It is clear that most of the elements in the set of representatives will tend to be not edge-linked in the graphs as the size of the representative set increases. However, as it happens in the Fingerprint dataset, a larger vocabulary might obtain better results than a smaller one.

In Table 2 we show a deeper analysis into the actual features that the ranking methods are considering as relevant. In particular, we put attention on whether the ranking methods keep the U_i features and on how much these features influence the final subsets in the nested structures. For each choice of the size of the representative set, we show the configuration that has obtained the best classification results of a kNN classifier with the χ^2 distance on the validation set.

Several statistics are shown. First, the size of the set of representatives (*rss*) which is, actually, the number of U_i features before feature selection. The resulting dimensionality of the vectors after mapping the graphs under the described embedding methodology, this is, the original number of features (*onf*), all U_i and B_{ij} features. The next column of the table (*onbf*, original node-based features) tells which is the

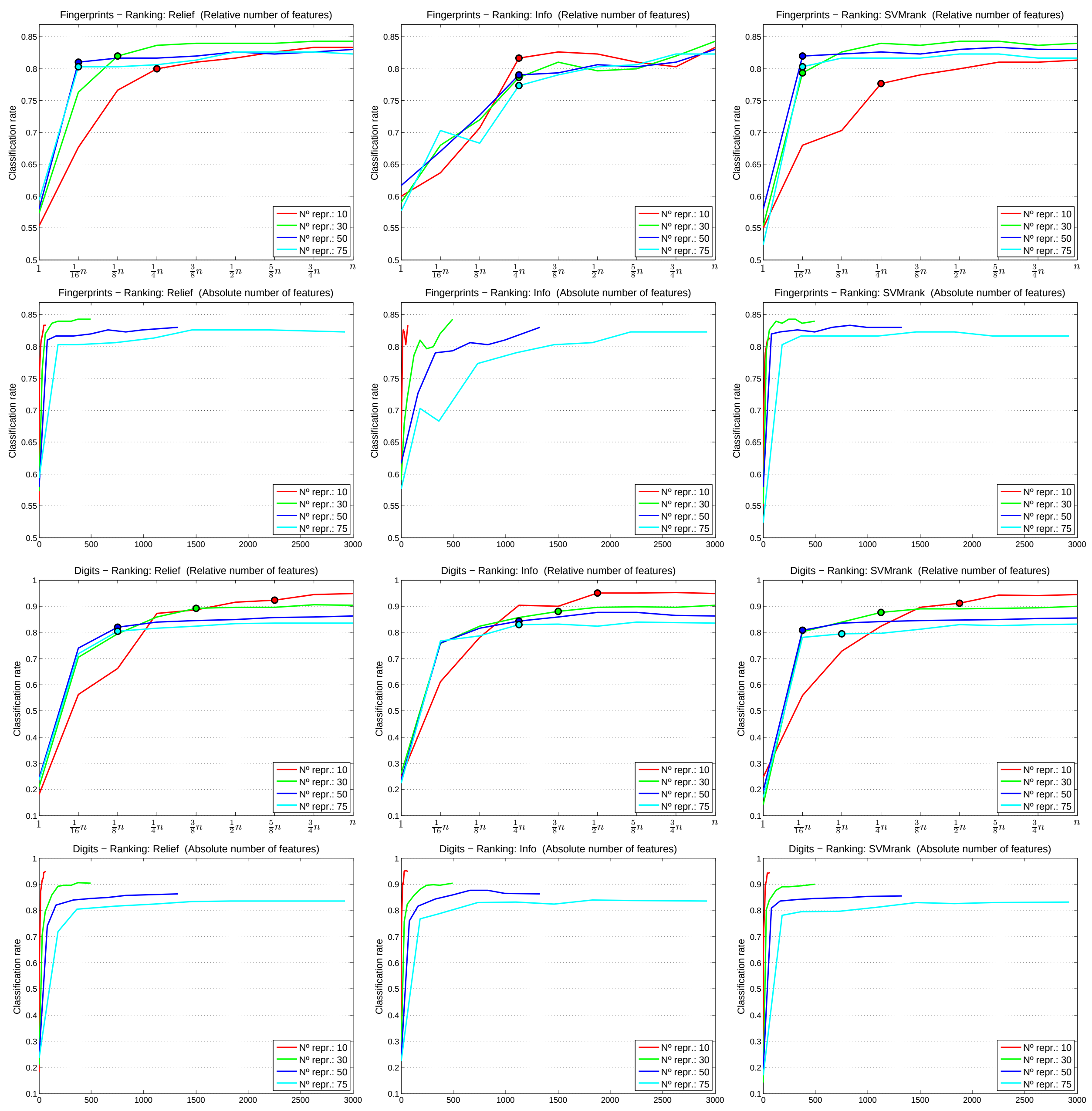

Figure 1: Validation results for some configurations on the Fingerprint and Digits databases. Accuracy rates of a kNN classifier in conjunction with a χ^2 distance on the validation set. First and third rows show the behavior when keeping a relative number of components. Dots on the curves show the best configuration. Second and fourth rows plot the same curves in absolute terms of the selected features.

percentage of the node-based features over all features in the original representation (first column over the second column).

We display the number of significant features (nsf), this is, the size of the optimal subset of features that

Table 2: Feature statistics for the different ranking methods under different embedding configurations.

				Ranking method											
Dataset	Original configuration			Info				Relief				SVMrank			
	rss	*nof*	*onbf*	*nsf*	*snbf*	*nbfk*	*cr*	*nsf*	*snbf*	*nbfk*	*cr*	*nsf*	*snbf*	*nbfk*	*cr*
Letter LOW	10	65	15.4	24	29.2	70.0	99.2	65	15.4	100.0	99.7	24	33.3	80.0	99.3
	20	230	8.7	86	14.0	60.0	99.3	57	10.5	30.0	99.3	143	11.9	85.0	99.5
	30	495	6.1	123	8.1	33.3	99.2	61	6.6	13.3	99.2	61	44.3	90.0	99.9
	50	1325	3.8	496	5.0	50.0	99.3	496	7.3	72.0	99.7	165	27.9	92.0	99.6
	75	2925	2.6	731	4.8	46.7	99.3	365	7.9	38.7	98.9	182	34.6	84.0	98.9
	100	5150	1.9	1287	3.9	50.0	98.7	321	6.2	20.0	98.8	321	26.5	85.0	98.9
Letter MED	10	65	15.4	48	12.5	60.0	83.5	65	15.4	100.0	84.4	48	16.7	80.0	80.1
	20	230	8.7	86	7.0	30.0	66.5	143	7.7	55.0	63.2	143	13.3	95.0	63.2
	30	495	6.1	309	7.4	76.7	59.3	309	5.2	53.3	60.5	247	11.7	96.7	60.4
	50	1325	3.8	82	11.0	18.0	47.7	331	2.7	18.0	50.1	662	7.4	98.0	48.7
	75	2925	2.6	182	7.7	18.7	44.8	182	2.2	5.3	42.5	1096	6.7	97.3	43.3
	100	5150	1.9	321	7.5	24.0	39.6	321	3.1	10.0	44.0	1931	5.0	96.0	38.9
Letter HIGH	10	65	15.4	32	21.9	70.0	76.4	40	17.5	70.0	77.6	24	29.2	70.0	75.5
	20	230	8.7	86	10.5	45.0	69.5	86	14.0	60.0	65.2	57	28.1	80.0	67.2
	30	495	6.1	123	12.2	50.0	61.2	123	5.7	23.3	60.1	61	39.3	80.0	60.7
	50	1325	3.8	165	6.1	20.0	53.1	82	3.7	6.0	49.9	82	46.3	76.0	51.7
	75	2925	2.6	182	7.1	17.3	49.6	182	1.6	4.0	50.8	182	34.1	82.7	50.3
	100	5150	1.9	321	5.9	19.0	48.8	321	1.9	6.0	42.4	321	26.8	86.0	43.2
Digits	10	65	15.4	32	25.0	80.0	95.0	40	22.5	90.0	92.4	32	21.9	70.0	91.2
	20	230	8.7	115	11.3	65.0	88.4	86	18.6	80.0	88.2	57	29.8	85.0	87.4
	30	495	6.1	185	9.2	56.7	88.0	185	13.0	80.0	89.2	123	18.7	76.7	87.6
	50	1325	3.8	331	7.3	48.0	84.4	165	12.7	42.0	82.0	82	48.8	80.0	80.8
	75	2925	2.6	731	5.7	56.0	83.0	365	7.9	38.7	80.4	365	19.2	93.3	79.4
	100	5150	1.9	1287	4.3	55.0	82.0	1287	5.9	76.0	79.8	643	14.9	96.0	76.8
Fingerprints	10	65	15.4	16	31.3	50.0	81.7	16	43.8	70.0	80.0	16	31.3	50.0	77.7
	20	230	8.7	86	11.6	50.0	81.0	28	42.9	60.0	81.3	57	29.8	85.0	82.0
	30	495	6.1	123	10.6	43.3	78.7	61	34.4	70.0	82.0	30	60.0	60.0	79.3
	50	1325	3.8	331	7.3	48.0	79.0	82	28.0	46.0	81.0	82	46.3	76.0	82.0
	75	2925	2.6	731	4.4	42.7	77.3	182	24.7	60.0	80.3	182	35.2	85.3	80.3
	100	5150	1.9	643	5.1	33.0	77.3	321	27.4	88.0	79.3	321	27.7	89.0	79.7
GREC	10	65	15.4	8	12.5	10.0	94.8	16	31.3	50.0	96.9	16	43.8	70.0	97.9
	20	230	8.7	28	14.3	20.0	95.8	57	24.6	70.0	96.9	28	57.1	80.0	96.5
	30	495	6.1	30	10.0	10.0	94.4	30	26.7	26.7	93.7	30	73.3	73.3	96.9
	50	1325	3.8	165	7.3	24.0	97.2	82	22.0	36.0	95.8	82	50.0	82.0	95.8
	75	2925	2.6	182	9.3	22.7	94.1	182	19.8	48.0	96.2	182	36.3	88.0	95.5
	100	5150	1.9	643	4.7	30.0	95.5	321	14.3	46.0	96.9	321	28.0	90.0	96.2
COIL	10	65	15.4	40	17.5	70.0	91.0	24	20.8	50.0	89.4	40	17.5	70.0	90.6
	20	230	8.7	172	9.9	85.0	96.8	86	18.6	80.0	97.0	57	26.3	75.0	96.2
	30	495	6.1	309	8.1	83.3	98.6	123	13.0	53.3	98.0	123	18.7	76.7	98.4
	50	1325	3.8	496	7.3	72.0	98.4	331	7.6	50.0	98.8	165	21.2	70.0	98.0
	75	2925	2.6	1096	5.1	74.7	98.2	365	6.0	29.3	97.8	365	17.0	82.7	98.2
	100	5150	1.9	1931	3.8	73.0	98.4	1287	3.0	39.0	98.6	321	24.0	77.0	98.2

rss: representative set size. *onf*: original number of features. *onbf*: original node-based features (%).
nsf: number of significant features. *snbf*: significant node-based features (%). *nbfk*: node-based features kept (%).
cr: classification rate on the validation set (%).

leads to the best classification performance given a set of representatives. From these sets, we are interested in the proportion of features that originally come from U_i, namely, the significant node-based features ($snbf$) and also the proportion of node-based features that are kept in the final optimal subset ($nbfk$), this is, the actual number of U_i features that the ranking algorithm has selected. We finally show the classification rate (cr) of each specific configuration in %.

A first observation we make is how much all feature ranking methods reduce the original number of features. By comparing the *nof* and *nsf* columns, regardless of the database and the ranking methodology we work with, we see that the number of features is, in general, drastically reduced, resulting in a situation in which further learning algorithms are computationally more feasible than when using all the original features.

A last interesting observation is the fact that node-based features are more present in the reduced version -this is, in the optimal subsets of features- than in the original sets (see *onbf* versus *snbf*). Indeed, it only happens in a very few number of cases that the percentage of features coming from node probabilities in the original vector representations is higher than in the reduced versions. This is indicating the importance of these U_i features. Nevertheless, B_{ij} features do introduce important additional information in the embedded representation as long as several of these features are kept in the reduced versions. We also observe that the SVM ranking methodology tends to keep a higher proportion of the features than the other methods.

4.3. PCA-based methods

For the PCA-based methods we have also built the same vectorial representations based on the 20 different sets of representatives. We have, however, adopted another validation strategy. Although we also find a ranking on the transformed features, we make use of the variance that each component is preserving and we threshold these values, keeping a certain amount of them as relevant.

In particular, we initially apply the PCA and Kernel PCA transformations and keep all features. In Fig. 2 we show the variance for different sets of representatives in the GREC dataset. We depict the fraction of variance curves for PCA, and for different values of the γ parameter in the Kernel PCA approach with the two mentioned kernel functions.

We clearly see how PCA is capable of easily keeping most of the variance with just a small number of features. It is much faster than any of the kernel PCA approaches in all cases. The same behavior is observed in all the other datasets we work with. In any case, this does not necessarily mean that PCA reduced features outperform the kernel PCA ones since the performance will depend on the transformation rather than the precise number of features that the method is keeping. It is also worth noticing that higher values of the γ parameter for the kernel functions will produce transformations that maintain the same rate of variance with less dimensions than lower values of it. Nevertheless, this is again not synonymous to the fact that these higher values will produce better transformations of features with regard to the classification performances.

The optimal subset of the transformed features is obtained by different cut-off points that we do on the variance values that each component is preserving. Particularly, we make cuts on the fraction of variance at the following points: 0.9, 0.925, 0.95, 0.975, 0.99 and 0.999. Each of these cut-off points determines a particular number of features that are being considered as potential candidates for the optimal subset of

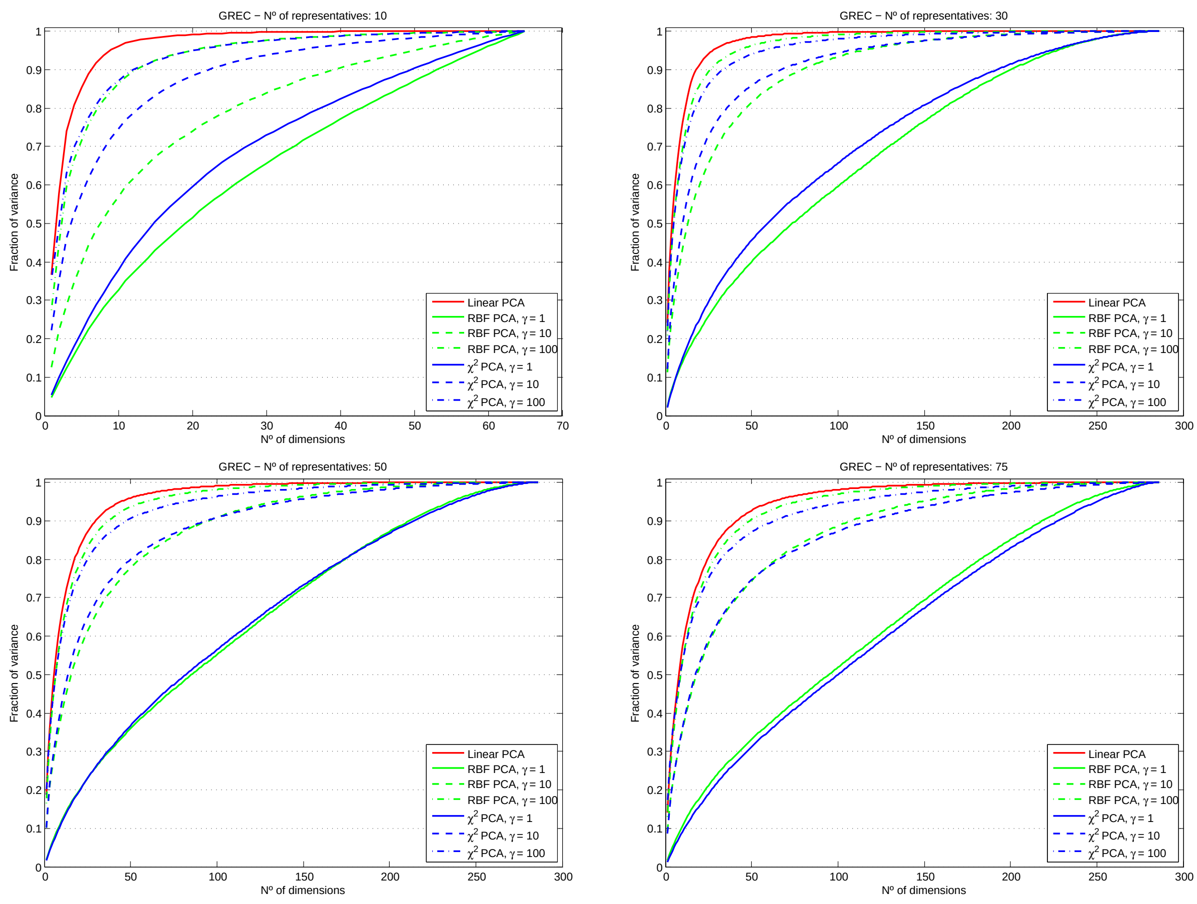

Figure 2: Fractions of variance for different choices of the representative set on the GREC dataset.

features in the final representation.

Again, the accuracy of a classifier can be regarded as a function of both the size of the representative set and the amount of variance that is being kept. In Table 3 we show some statistics of different configurations for all datasets. Using a representative set of a certain size (*rss* in the table), with its respective number of original features (*onf*), we apply all the cut-off points mentioned above. For each of them, we apply a kNN classifier together with the Euclidean distance (features are transformed and are no longer histogram-based). For each of these representations, we report the best classification rate (*cr*), the cut-off point on the fraction of variance that has produced this performance (*fov*) and the corresponding number of features in the reduced version of the embedded graphs (*ndrv*).

A first and important comment we should make here is that almost all configurations that we have considered (the ones we show and the ones we do not show) already reach the best performance by using the lowest threshold that we have considered for the variance cut-off points. This means that all further cut-off points define sets of features that do not actually improve the performance of this lowest cut, and

Table 3: Feature statistics for the different PCA-based methods under different embedding configurations.

			Reduction method										
Dataset	Original configuration		Linear PCA			*rbf* PCA				χ^2 PCA			
	rss	*onf*	*fov*	*ndrv*	*cr*	*fov*	*ndrv*	γ	*cr*	*fov*	*ndrv*	γ	*cr*
Letter LOW	10	65	0.9	12	97.87	0.9	28	2.00	98.80	0.9	27	2.00	99.47
	20	230	0.9	19	96.80	0.9	112	2.00	99.60	0.9	125	2.00	99.87
	30	495	0.9	30	93.33	0.9	310	1.00	98.80	0.9	192	4.00	99.87
	50	1325	0.9	54	91.73	0.9	532	0.50	95.47	0.9	414	2.00	99.60
	75	2925	0.9	82	90.00	0.9	575	0.50	92.40	0.9	380	4.00	99.47
	100	5150	0.9	108	87.87	0.9	589	0.50	91.20	0.975	702	0.50	98.80
Letter MED	10	65	0.9	21	52.27	0.9	52	0.50	58.27	0.9	43	2.00	74.00
	20	230	0.9	48	38.93	0.9	88	10.00	39.73	0.9	168	2.00	60.93
	30	495	0.9	71	37.07	0.9	157	10.00	37.33	0.9	423	1.00	65.47
	50	1325	0.9	113	31.73	0.9	224	10.00	37.20	0.9	622	1.00	65.47
	75	2925	0.9	132	32.40	0.9	246	10.00	35.73	0.925	659	1.00	66.67
	100	5150	0.9	157	30.00	0.9	229	16.00	36.67	0.9	598	2.00	65.47
Letter HIGH	10	65	0.9	26	56.40	0.9	44	2.00	60.40	0.9	49	1.00	74.67
	20	230	0.9	58	51.47	0.9	99	10.00	55.47	0.9	174	2.00	71.87
	30	495	0.9	87	50.27	0.9	173	10.00	52.67	0.9	394	2.00	70.80
	50	1325	0.9	135	43.33	0.9	263	8.00	49.33	0.9	472	5.00	72.80
	75	2925	0.9	167	42.13	0.9	271	10.00	45.33	0.9	458	8.00	72.40
	100	5150	0.9	192	38.53	0.9	258	16.00	44.40	0.9	524	5.00	72.80
Digits	10	65	0.9	11	83.40	0.9	44	4.00	86.00	0.9	50	1.00	89.00
	20	230	0.9	26	81.20	0.9	143	5.00	83.60	0.9	177	2.00	89.00
	30	495	0.9	39	76.20	0.9	153	16.00	78.80	0.9	349	4.00	87.60
	50	1325	0.9	69	74.20	0.9	441	8.00	76.20	0.999	997	1.00	91.00
	75	2925	0.9	106	69.80	0.9	503	8.00	72.60	0.9	864	2.00	89.20
	100	5150	0.9	140	68.60	0.9	491	10.00	72.00	0.975	965	2.00	90.00
Fingerprints	10	65	0.9	7	80.00	0.9	8	500.00	80.33	0.9	10	100.00	82.67
	20	230	0.9	13	79.67	0.9	17	100.00	79.67	0.9	70	8.00	78.33
	30	495	0.9	20	77.00	0.9	133	5.00	78.67	0.9	66	16.00	81.33
	50	1325	0.9	34	80.67	0.9	35	500.00	80.67	0.9	56	64.00	81.00
	75	2925	0.9	52	80.67	0.9	58	100.00	82.67	0.9	71	100.00	80.33
	100	5150	0.9	66	80.67	0.9	87	32.00	81.33	0.9	113	32.00	80.33
GREC	10	65	0.9	6	84.27	0.9	32	16.00	92.31	0.9	43	2.00	95.10
	20	230	0.9	12	87.06	0.9	100	5.00	93.36	0.9	82	4.00	97.20
	30	495	0.9	17	86.36	0.9	128	4.00	90.91	0.9	154	2.00	96.50
	50	1325	0.9	29	90.56	0.9	144	4.00	94.06	0.9	184	2.00	96.85
	75	2925	0.9	41	93.71	0.9	53	64.00	95.10	0.9	154	5.00	97.20
	100	5150	0.9	56	91.61	0.9	72	50.00	92.31	0.9	186	4.00	97.90
COIL	10	65	0.9	7	52.60	0.9	25	500.00	60.80	0.9	37	16.00	75.20
	20	230	0.9	15	63.80	0.9	24	1000.00	66.00	0.9	180	4.00	80.00
	30	495	0.9	25	65.20	0.9	35	1000.00	67.40	0.9	395	5.00	85.20
	50	1325	0.9	50	65.60	0.9	74	1000.00	65.60	0.9	1126	5.00	87.80
	75	2925	0.9	112	72.00	0.95	2115	2.00	79.80	0.975	2245	2.00	97.40
	100	5150	0.9	191	73.40	0.95	2117	2.00	77.60	0.99	2312	2.00	96.60

rss: representative set size. *onf*: original number of features. *fov*: fraction of variance.
ndrv: number of dimensions in the reduced version. γ: weighting parameter in kernel PCA methods.
cr: classification rate on the validation set (%).

thus, most of the redundancy is removed from the vectorial representations. It also suggests the use of lower cut-off points. Yet, we have experimentally seen that these lower points lead to too few features and too low classification rates.

Related to this finding is the fact that the final number of dimensions is drastically reduced with respect to the original ones. This fact is even more prominent when compared to the size of the optimal subsets that were obtained using the ranking methods. Thus, PCA-based methods reduce to a higher degree the dimensionality of the embedded representations of graphs than the ranking methodologies.

On the other hand, we encounter that such a reduction is not necessarily related the performance of the considered classifier. In general, when comparing both tables, we observe that the ranking methods usually outperform methods based on PCA or Kernel PCA. In other words, transforming the features does not seem to make the final configuration stronger. We should of course check whether this is a problem of the general methodology used or just the fact that other kernel functions should be applied together with other distance measures in the kNN algorithm. However, we understand that such a deeper study is out of scope with regard to the original objectives of this work and we leave it for future work.

In any case, the results of Table 3 suggest again that there is no clear *a priori* way to define which is the number of elements in the set of representatives and that this step should always be validated since it depends on the dataset under study.

4.4. Results

We first embed graphs into vector spaces and then select features of such vectorial representations. We may classify these final representations of graphs using any learning algorithm that is available for vectorial representations of data. For the results on the test sets of all databases, we have used both the kNN rule and an SVM classifier [41]. For the ranking method configurations a χ^2 distance is used for the kNN and a χ^2 kernel is used for the SVM. In the case of the transformed configurations, we use the Euclidean distance and a linear kernel for the kNN and SVM, respectively.

We want to compare this methodology with other graph classification methods proposed in the literature. Therefore, we need to pick reference systems. In order to make the comparison as much independent as possible on the classification algorithms, we use the very same classifiers: kNN and SVM. In this case though, the kNN classifier is based on the edit distance of graphs (as described in [42]), and the SVM classifier is trained on another embedding space. In particular, we use the embedding methodology proposed in [20]. A graph is represented as a vector the components of which are edit distances to a predefined set of prototypes. Formally, given $\mathcal{P} = \{p_1, \ldots, p_n\}$ a set of graph prototypes, the dissimilarity embedding of a graph g is defined as

$$\phi_n^{\mathcal{P}}(g) = (d(g, p_1), \ldots, d(g, p_n)), \tag{13}$$

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

We seek for the best configuration possible on the validation set for each dataset. In case of the kernel PCA algorithms we also have the γ parameter that is selected using the same criteria. Again, we assume that higher accuracies do not mean better performance as long as they keep in the same level of statistical

Table 4: kNN results on the test set. The best result for each dataset is shown bold face.

	Reference System	Feature Ranking			PCA-based methods		
Dataset	kNN - Graph Edit Distance	Relief	Info	SVMrank	Linear PCA	*rbf* PCA	χ^2 PCA
Letter LOW	99.3	99.1	99.5	**100.0**	96.7	98.9	98.9
Letter MED	94.4	**88.4**	85.9	86.4	72.1 ✗	76.5 ✗	80.5 ✗
Letter HIGH	89.1	80.8	**80.9**	70.1 ✗	67.9 ✗	69.9 ✗	69.2 ✗
Digits	97.4	89.5 ✗	**92.5**	89.8 ✗	82.3 ✗	86.3 ✗	71.5 ✗
Fingerprints	79.1	77.6	77.7	78.8	79.5	**80.5**	77.3
GREC	95.5	96.4	**98.7**	95.5	93.6	94.9	96.6
COIL	93.3	97.0	**97.6**	96.8	71.1 ✗	79.6 ✗	96.3

✗ Statistically significant deterioration over the reference system (Z-test using $\alpha = 0.05$).

Table 5: SVM results on the test set. The best result for each dataset is shown bold face.

	Reference System	Feature Ranking			PCA-based methods		
Dataset	SVM - Dissimilarity embedding	Relief	Info	SVMrank	Linear PCA	*rbf* PCA	χ^2 PCA
Letter LOW	99.3	99.6	**99.9**	99.7	98.0	99.3	99.7
Letter MED	94.9	**92.8**	88.8	90.5	85.5 ✗	80.9 ✗	85.5 ✗
Letter HIGH	92.9	87.7	**88.4**	82.1 ✗	81.2 ✗	78.4 ✗	78.5 ✗
Digits	98.7	94.8	**96.0**	94.1	87.3 ✗	90.1 ✗	67.5 ✗
Fingerprints	83.1	79.1	80.1	81.5	**80.3**	79.6	79.9
GREC	95.1	97.0	97.9	96.8	95.5	96.0	**98.3**
COIL	96.8	97.0	97.2	**97.4**	86.5 ✗	59.7 ✗	82.0 ✗

✗ Statistically significant deterioration over the reference system (Z-test using $\alpha = 0.05$).

significance. The accuracy rate for defining the best configuration is obtained by the kNN classifier on the validation set as defined in the previous sections.

In Tables 4 and 5, we show the results of the reference systems and the proposed feature selection strategies on the embedded graphs for the described datasets. With regard to the kNN results, we observe how the Mutual Information ranking method is the only one of the six different strategies that is capable to obtain results at the same level of statistical significance than the reference system. Besides, it obtains the best result in four of the seven datasets. Nevertheless, every feature selection method is good for at least three datasets, even reaching a perfect performance for the SVM-based ranking method in the lower distortion level of the Letters dataset.

We observe an interesting fact for the Medium and High distorted versions of the Letters datasets and the Digits dataset. It turns out that none of the PCA-based methods is capable to reach a level similar to the reference system. This is related to the nature of the embedding methodology which is gathering information of the distribution of nodes of the graphs. These three cases are those where nodes are distorted to the highest degree from all cases we consider. Thus, this methodology seems to be not applicable for these specific cases.

With respect to the SVM results, we see how we can solve two of the three scenarios where the ranking methods would not work using a kNN classifier, but we confront the same situation regarding the distorted cases of graph datasets. This correlation between the classifiers is reinforcing the idea that the features we propose might not be a good option for these cases. Anyhow, we already said that there is still some work regarding possible improvements in this direction, since more kernel functions and distance measures could be applied.

As a final conclusion, for each of the datasets that we have considered in this work, we have a configuration of the proposed embedding that is at the same level of statistical significance than the reference systems. In this line, we want to emphasize the fact that the features we extract from graphs are computationally much more efficient than those of the reference systems. Our method requires only a linear number of Euclidean distance computations with respect to the number of nodes in the graphs, while edit distance computation is exponential in this number. Also, by applying feature selection algorithms, we have been able -as seen in the validation stage of this experimental work- to remove most of the features that we initially extract, so we finally obtain a representation of graphs with a few number of features, leading to situations in where learning algorithms are easily applicable.

5. Conclusions

Embedding a set of graphs into a vector space is a way of making statistical machine learning algorithms applicable to the domain of graphs. Classical graph matching approaches suffer from high computational cost and thus embedding methodologies have gained broad interest among the community. Nevertheless, the features that constitute these vectorial representations are of decisive importance and attention should be put on their construction.

In this paper, we have proposed a way of embedding a set of graphs into vector spaces by means of statistics of the appearances of a set of representative elements of the node labels in the graphs. Such a set is constructed initially and then each node can be described as a probability vector regarding how much each of these elements is representing it. Then, using such assignments of nodes to representatives, one can compute how much each representative is being reflected in each graph. Also, the edge information of the graphs can be described in terms of these representative elements.

The way of constructing these representations is such that we do not have an *a priori* intuition of the amount of information that the representatives are actually providing. To discover this, we apply different feature selection algorithms such as ranking methods and PCA-based methods. The first kind of these approaches ranks the set of all features and then keeps only an optimal subset of them, while the second initially computes a transformation of the features and then ranks the resulting ones in terms of the amount of variance that they are retaining, such that a subset of them can be chosen.

The experimental part of the work on different and diverse datasets of graphs has shown that most of the features that are constructed can be discarded for the purpose of graph classification, suggesting that some redundancy and noise is being given to the vectorial representation of graphs under the described embedding methodology. Particularly, ranking algorithms have been shown to be more stable in terms of classification rates than PCA-based methods although the number of features in the final representations is usually higher.

In comparison to the reference methods, we are able to achieve the same classification rates while the features we extract are computationally much less costly. Moreover, by applying the feature selection algorithms, we finally consider just a small proportion of the originally features, and thus, the eventual learning algorithms that are used require less computational resources.

There are still some issues that will attract our attention in the future. On the one hand, the embedding methodology could be improved in the direction of allowing more general graphs, for example graphs having edge attributes. So far, the proposed methodology just cannot consider these cases. On the other hand, another potential topic of feature research is to study whether there is a correlation between the performance of the proposed methodology and those sets of representatives that provide the best clustering situation of the node labels. Apart from that, the distance measures and the kernel functions both in the kernel PCA and in the SVM algorithm have been selected intuitively. A proper investigation on which are the optimal distance measures and kernel functions for these specific vectorial representations of graphs and how much correlated is the choice of them with the underlying learning machine would give us a more insight into the features that we are proposing.

References

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

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

[3] K. Borgwardt, *Graph Kernels*, Ph.D.Thesis, Ludwig-Maximilians-University Munich, 2007.

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

[5] H. Bunke, P. Dickinson, M. Kraetzl, W. Wallis, *A graph-theoretic approach to enterprise network dynamics*, Progress in Computer Science and Applied Logic (PCS), vol. 24, 2007, Birkhäuser.

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

[7] D. Cook, L. Holder (Eds.), *Mining Graph Data*, Wiley-Interscience, 2007.

[8] A. Schenker, H. Bunke, M. Last, A. Kandel, *Graph-Theoretic Techniques for Web Content Mining*, World Scientific, 2005.

[9] Z. Harchaoui, F. Bach, *Image classification with segmentation graph kernels*, in: IEEE Conference on Computer Vision and Pattern Recognition 2007, pp.1-8.

[10] R. Ambauen, S. Fischer, H. Bunke, *Graph edit distance with node splitting and merging and its application to diatom identification*, in:E. Hancock, M. Vento (Eds.), Proceedings of the 4th International Workshop on Graph Based Representations in Pattern Recognition, Lecture Notes in Computer Sciences, vol. 2726, Springer 2003, pp.95-106.

[11] D. Conte, P. Foggia, C. Sansone, M. Vento, *Thirty years of graph matching in pattern recognition.* International Journal of Pattern Recognition and Artificial Intelligence, 18 (3), pp. 265-298 (2004).

[12] S. Kramer, L. De Raedt, *Feature construction with version spaces for biochemical application*, in: Proceedings of the 18th International Conference on Machine Learning, 2001, pp. 258-265.

[13] A. Inokuchi, T. Washio, H. Motoda, *An Apriori-based algorithm for mining frequent substructures from graph data*, in: Proceedings of the 4th PKDD, 2000, pp. 13-32.

[14] B. Luo, R.C. Wilson, E.R. Hancock, *Spectral embedding of graphs.* Pattern Recognition, 36 (10), pp. 2213-2230 (2003).

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

[16] P. Ren, R. Wilson, E. Hancock, *Graph Characterization via Ihara Coefficients*, IEEE Transactions on Neural Networks 22 (2) (2011), pp. 233-245.

[17] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, S.W. Zucker, *Indexing hierarchical structures using graph spectra*, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (7) (2005), pp. 1-16.

[18] E. Pekalska, R. Duin, *The Dissimilarity Representation for Pattern Recognition: Foundations and Applications*, World Scientific (2005).

[19] E. Pekalska, R. Duin, P. Paclick, *Prototype selection for dissimilarity-based classifiers*, Pattern Recognition 39 (2) (2006), pp. 189-208.

[20] K. Riesen, H. Bunke, *Graph Classification and Clustering Based on Vector Space Embedding.* World Scientific (2010).

[21] M.F. Demirci, A. Shokoufandeh, Y. Keselman, L. Bretzner, S. Dickinson *Object recognition as many-to-many feature matching*, International Journal of Computer Vision, 69 (2) (2006), pp. 203-222.

[22] M.F. Demirci,Y. Osmanlioglu, A. Shokoufandeh, S. Dickinson, *Efficient many-to-many feature matching under the l_1 norm*, Computer Vision and Image Understanding 115 (7) (2011), pp. 976-983.

[23] J. Gibert, E. Valveny, H. Bunke, *Graph of Words Embedding for Molecular Structure-Activity Relationship Analysis.* in: I. Bloch, R.M. Cesar, Jr. (Eds.), Proceedings of the 15th Iberoamerican Congress on Pattern Recognition, Lecture Notes in Computer Sciences, vol. 6419, Springer 2010, pp. 30-37.

[24] J. Gibert, E. Valveny, H. Bunke, *Vocabulary Selection for Graph of Words Embedding.* in: J. Vitrià, J.M. Sanches, M. Hernández (Eds.), Proceedings of the 5th Iberian Conference on Pattern Recognition and Image Analysis, Lecture Notes in Computer Sciences, vol. 6669, Springer 2011, pp. 216-223.

[25] J. Gibert, E. Valveny, H. Bunke, *Dimensionality Reduction for Graph of Words Embedding.* in: X. Jiang, M. Ferrer, A. Torsello (Eds.), Proceedings of the 8th International Workshop on Graph Based Representations in Pattern Recognition, Lecture Notes in Computer Sciences, vol. 6658, Springer 2011, pp. 22-31.

[26] J. Gibert, E. Valveny, H. Bunke, *Graph embedding in vector spaces by node attribute statistics.* Accepted for publication in Pattern Recognition.

[27] R. Duda, P. Hart, D. Stork, *Pattern Classification*, second ed., Wiley-Interscience, 2000.

[28] I. Gyon, S. Gunn, M. Nikravesh, et al. (Eds.), *Feature Extraction, Foundations and Applications.* Springer, Heidelberg (2006).

[29] P. Pudil, J. Novovicova, J. Kittler, *Floating search methods in feature-selection*, Pattern Recognition Letters 15(11) (1994) 1119-1125.

[30] K. Kira, L.A. Rendell, *The Feature Selection Problem: Traditional Methods and a New Algorithm*, in: Proceeding of the 9th International Conference on Artificial Intelligence, pp. 129-134.

[31] U. Fayyad, K. Irani, *Multi-interval discretization of continuous valued attributes for classification learning*, in:Proceedings of the 13th International Joint Conference on Artificial Intelligence, vol.2, Morgan Kaufmann, 1993, pp. 1022-1027.

[32] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, *Gene selection for cancer classification using support vector machines*, Machine Learning 46(1-3)(2002) 389-422.

[33] B. Schölkopf, A. Smola, K.R. Müller, *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, Neural Computa-

tion, 10 (1998), pp. 1299-1319.

[34] J. Zhang, M. Marszalek, S. Lazebnik, C. Schmid, *Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study*, International Journal of Computer Vision (2007), pp. 213-238.

[35] K. Riesen, H. Bunke, *IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning*, in: N. da Vitoria Lobo et al. (Eds.), Proceedings of the International Workshops on Structural, Syntactic and Statistical Pattern Recognition, Lecture Notes in Computer Science, vol. 5342, 2008, pp. 287-297.

[36] E. Alpaydin, F. Alimoglu, *Pen-based recognition of handwritten digits*, Department of Computer Engineering, Bogazici University (1998).

[37] C. Watson, C. Wilson, *NIST Special Database 4, Fingerprint Database*, National Institute of Standards and Technology (1992).

[38] P. Dosch, E. Valveny, *Report on the second symbol recognition contest*, in: W. Liu, J .Lladós (Eds.), Graphics Recognition. Ten Years Review and Future Perspectives. Proceedings of the Sixth International Workshop on Graphics Recognition, Lecture Notes in Computer Science, vol. 3926, Springer, 2005, pp. 381-397.

[39] S. Nene, S. Nayar, H. Murase, *Columbia Object Image Library: COIL-100*, Technical report, Department of Computer Science, Columbia University, New York (1996).

[40] C. Harris, M. Stephens, *A combined corner and edge detector*, in: Proceedings of the 4th Alvey Vision Conference, pp. 147-151 (1988).

[41] B. Schölkopf, A.J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press (2002).

[42] K. Riesen, H. Bunke, *Approximate graph edit distance computation by means of bipartite graph matching*, Image and Vision Computing 27 (4) (2009), pp. 950-959.