Graph embedding in vector spaces by node attribute statistics

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

Graph-based representations are of broad use and applicability in pattern recognition. They exhibit, however, a major drawback with regards to the processing tools that are available in their domain. Graph embedding into vector spaces is a growing field among the structural pattern recognition community which aims at providing a feature vector representation for every graph, and thus enables classical statistical learning machinery to be used on graph-based input patterns. In this work, we propose a novel embedding methodology for graphs with continuous node attributes and unattributed edges. The approach presented in this paper is based on statistics of the node labels and the edges between them, based on their similarity to a set of representatives. We specifically deal with an important issue of this methodology, namely, the selection of a suitable set of representatives. In an experimental evaluation, we empirically show the advantages of this novel approach in the context of different classification problems using several databases of graphs.

Keywords: Structural Pattern Recognition, Graph Embedding, Data Clustering, Graph Classification

1. Introduction

Many pattern recognition problems are solved by means of feature vector representations of data. Biometric person identification, optical character recognition or industrial quality control are prominent examples that make use of vectorial representations [1]. Formally, representing patterns as points in a vector space, problems from the real world are converted into the analysis of a population of vectors. The components of these vectors are numerical features of the patterns under study that are problem-dependent. Obviously, their choice is of crucial importance.

Feature vectors in pattern recognition are widely used since the comparison of two patterns reduces to simple operations between points in a vector space. The Euclidean distance, for instance, is a concept that provides us with a proper similarity measure between patterns. Moreover, the fact that vector spaces are flexible and manageable mathematical objects has led the scientific community during the past decades to devote efforts into developing data analysis algorithms for patterns represented by vectors. Dimensionality reduction techniques such as Principal Component Analysis or Linear Discriminant Analysis are notable examples [1]. Classification methods such as Artificial Neural Networks or Support Vector Machine are other examples of successful algorithms that are based on feature vectors [2, 3].

On the other hand, there are a couple of drawbacks of feature vector representations that should be noted. In some applications, the nature of the patterns to be studied is such that there is an intrinsic need for representing not only features but also structural relations between the features. Also, feature vector representations ask for all patterns to be represented with a fixed number of features regardless of their complexity. These two problems can be addressed by the use of graph-based representations instead of feature vectors. Graphs are able to express relations between parts of the underlying objects, and they are not restricted to a predefined size of the representation. A good review of applications of graph-based representations in pattern recognition can be found in [4].

Regardless of their representative power, graph representations exhibit also some weaknesses when compared to feature vectors. Most seriously, graphs are complex mathematical objects that, because of their complexity lack operations needed to implement pattern recognition algorithms. Examples are the sum or the product of two graphs. This lack has an important consequence that there exist just a few algorithms to process and analyse graph instances of patterns. Indeed, these algorithms are restricted to ones that operate on distances exclusively. An example is the k Nearest Neighbour classifier (kNN) family, in which the only necessary tool is a similarity measure between graphs. Thus, we encounter a situation where we have a good representational framework but we lack a repository of algorithms allowing us a proper analysis and processing of the given objects. In the past years, signif-

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

icant effort has been put into bridging the gap between the structural representation of objects and the repository of algorithms for feature vector representations [5]. Two prominent lines of research can be found in the literature. These are graph kernels and graph embeddings in vector spaces.

Graph kernels is a natural way of making kernel machines applicable to graph-based representations. A kernel machine is any learning algorithm that only depends on scalar products between input patterns. This fact, in conjunction with the kernel trick (an implicit feature space where the kernel function is the standard dot product), allows one to apply kernel machines to graphs as long as a kernel function for graph instances is available. Various graph kernels have been proposed in the literature [6]. Diffusion kernels are based on the idea of building a kernel matrix for graph instances by exponentiating a similarity matrix between graphs [7, 8]. Convolution kernels build similarity measures between graphs by similarity measures between smaller parts of the same graphs, which in general are easy to infer [9, 10]. Finally, Random Walk kernels infer a similarity measure between two graph instances by calculating the number of common random walks they share [11, 12, 13]. An important and notable work that provides a unified framework for various graph kernels is [14].

A more general approach, which is not restricted to kernel machines, is the embedding of graphs into vector spaces. In order to provide access to all vector-based data processing algorithms, graph embedding methods associate a feature vector to every graph. 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 [15, 16] 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. Spectral-based embedding is another important approach. For instance, in [17], the authors extract features from a graph based on the eigen-decomposition of the adjacency and Laplacian matrices. Graph clustering and visualization is then performed by dimensionality reduction techniques on the extracted eigen-features. Another interesting spectral approach is the one proposed in [18]. The authors show how the spectral matrix of the Laplacian of a graph can be used to construct symmetric polynomials, the coefficients of which can be employed as graph features. A last spectral-based approach is [19]. The polynomial coefficients of the Ihara zeta function of a given graph are used for describing its structure and topology. Such features can be regarded as the spectrum of the Perron-Frobenius operator. Finally, based on the dissimilarity representation studied in [20, 21], the authors of [22] propose to classify and cluster graphs using a vectorial representation whose components are features expressing the distances to a set of predefined prototype graphs.

All of these approaches have their individual strengths and weaknesses. For instance, the methods based on finding subgraphs are capable of adding specific domain knowledge to the representation since they can look for relevant substructures to the problem to be solved. However, finding these substructures in a given graph can be computationally challenging. Spectral methods provide solid theoretical insight into the meaning of the extracted features, but they remain restricted to graphs without node labels. Moreover, they are sensitive to structural errors such as missing or spurious node and edges. The dissimilarity embedding can handle arbitrary graphs and add domain specific knowledge since the distance in use is the graph edit distance. On the other hand, graph edit distance is costly to compute and thus is the embedding method.

The contributions of this work are twofold. In the first place, we present a novel graph embedding methodology for graphs with continuous node attributes based on different statistics on the node labels and the edge relations between them. The fact that node attributes are continuous demands for a first step in which these attributes are discretized and a set of representative nodes are located in the attribute's space. Given node representatives, the nodes are evaluated for similarity to the representatives and thus, only statistics on the representatives are taken into account. The second contribution of the present paper addresses an inherent issue of the proposed embedding methodology. As the set of representatives for the node attributes is of crucial importance, we propose and experimentally evaluate different clustering algorithms to perform the selection of these representatives. The proposed approach to graph embedding is conceptually simple and computationally faster than the embedding methods discussed before. As our experimental results suggest, the new embedding procedure provides vector representations of graphs that can compete with state of the art methods for graph classification.

A preliminary version of this work appeared in [23]. The current paper puts the proposed methodology into a more general context and relates it to other graph embedding algorithms. Also, the experimental evaluation has been enlarged by testing on more databases of graphs. This work furthermore extends the approach proposed in [24] from the case of discrete to continuous labels and thus makes it applicable to a much wider spectrum of applications.

The rest of the paper is organized as follows. The notation and background concepts related to graphs and graph matching are introduced in the next section. In Section 3 we propose our novel methodology of embedding a graph into a vectorial space by counting frequencies of node attributes and their respective edge relations. In Section 4, we discuss the different algorithms that are used in this work to select a set of node attribute representatives. Then, in Section 5, we present the results of an experimental evaluation of the proposed methodology in terms of classification rates for different datasets of graphs using different classifiers. Finally, Section 6, concludes the article by highlighting the main points of this work and discussing open questions and possible future research.

2. Concepts and terminology

In this section, we introduce our basic notation and give a summary of the main concepts related to graphs and graph matching.

Definition 1 (Graph). A graph g is a four-tuple $g = (V, E, \mu, \nu)$, where V is a non-empty set of nodes, $E \subseteq V \times V$ is the set of edges, $\mu : V \longrightarrow L_V$ is the node labelling function and $\nu : E \longrightarrow L_E$ is the edge labelling function. L_V and L_E are the corresponding sets of labels for the nodes and edges, respectively.

An edge $e \in E$ in a graph g is usually represented by its source and its target nodes, $e = (u, v)$. A graph is called undirected when for all edge $(u, v) \in E$ there also exists the edge $(v, u) \in E$. A path of length n is a sequence of $n + 1$ nodes $(v_1, \ldots, v_{n+1})$, where $v_j \in V$, $\forall\, j \in \{1, \ldots, n + 1\}$ and $(v_i, v_{i+1}) \in E$, $\forall\, i \in \{1, \ldots, n\}$. The sets of labels can be of any type, thus allowing for different families of graph representations. For instance, they can be finite alphabets, sets of numbers, or even sets of vectors. A special labelling function is the one that maps every node (or edge) to the same label, called the null label ϵ. In this situation the graph is said to be node unattributed (or edge unattributed). When one of the labelling functions is of this special type, we can omit the respective labelling function in the definition of the graph. For instance, an edge unattributed graph is defined by $g = (V, E, \mu)$. In this paper, we will work with undirected and edge unattributed graphs whose node labelling sets are always $L_V = \mathbb{R}^d$, for some $d \geq 1$.

Graph matching is the process by which one is capable to tell how similar or dissimilar two graphs are. Graph matching techniques can be split into two main categories, namely, exact and inexact graph matching. Exact graph matching aims at deciding whether there is a one-to-one map from the nodes of one graph to the nodes of the other graph respecting the topology and the labelling characteristics of the involved graphs. Such a map is called a graph isomorphism and, if there exist a graph isomorphism, the two corresponding graphs are called isomorphic. In the context of exact graph matching, similarity measures can be defined by means of maximum common subgraph and minimum common supergraph [25, 26, 27]. However, such approaches are often not applicable to pattern recognition problems since the extraction of graphs from patterns is usually a noisy procedure that leads to errors and distortions. Thus, more general algorithms are needed.

The second category of graph matching algorithms consists of error-tolerant, or inexact, methods. Because of its broad applicability, and because it will serve us as a reference system to compare our methodology with, we mention here graph edit distance as the main paradigm of error-tolerant graph matching [28, 29, 30].

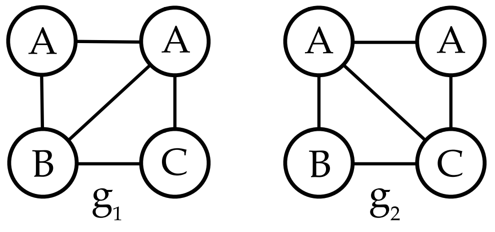


Figure 1: Two non-isomorphic graphs with the same vectorial representation counting node label appearances.

Briefly, the similarity of two graphs by means of the edit distance is based on the idea that there should not be a high distortion between two similar graphs, this is, the more similar two graphs are, the less distortion is need to transform one into the other. A set of edit operations with their implicit costs are defined in terms of substitutions, deletions and insertions of nodes and edges. An edit path is a set of operations that convert one graph into the other. The edit distance of two graphs is thus defined as the cost of the cheapest path between the two graphs. This definition applies to arbitrary graphs and, by means of the cost function, one can incorporate domain specific knowledge into the graph comparison process.

3. Graph Embedding by node representatives

In this section we give a formal description of our novel graph embedding procedure. We define the embedding of a graph into a vector space in terms of single and binary relations between node representatives.

3.1. Basic approach

Given a graph $g = (V, E, \mu)$, a simple vectorial representation of g is, for instance, the one that takes, as each component, the number of times a specific node label appears in the graph, *i.e.*,

$$\mathbf{x}_g = (\#(l_1, g), \#(l_2, g), \ldots, \#(l_n, g))\,, \tag{1}$$

where $\#(l_i, g)$ refers to the frequency that l_i happens to be the label of any node of graph g. For example, both graphs g_1 and g_2 in Fig. 1 have two labels A, one B and one C. Their respective vectorial representation in this form would be $\mathbf{x}_{g_1} = \mathbf{x}_{g_2} = (2, 1, 1)$.

In the simple example of Fig. 1, on gets the same vectorial representation from both graphs, although the graphs differ in their edge structure. Thus, more components should be added to the vectors in order to make this representation more discriminative. In the context of bioinformatics, in reference [24] this vectorial representation is enriched by considering not only the labelling frequencies but also the frequencies of the structural links between any two different nodes according to their corresponding attributes. More precisely, the vector representation (1) is enriched by $\mathcal{O}(n^2)$ components of the form

$$\#(l_i \leftrightarrow l_j, g), \tag{2}$$

counting how many edges between every pair of node labels occur in a given graph. With this information at hand, the vectors $\mathbf{x}_{g_1}$ and $\mathbf{x}_{g_2}$ will no longer be equal because the features $\#(A \leftrightarrow B, g)$ and $\#(A \leftrightarrow C, g)$ are, in fact, different. In the following order,

$$\begin{aligned}\mathbf{x}_g = (&\#(A,g), \#(B,g), \#(C,g),\\ &\#(A \leftrightarrow A, g), \#(A \leftrightarrow B, g), \#(A \leftrightarrow C, g),\\ &\#(B \leftrightarrow B, g), \#(B \leftrightarrow C, g), \#(C \leftrightarrow C, g)), \quad (3)\end{aligned}$$

the vectors $\mathbf{x}_{g_1}$ and $\mathbf{x}_{g_2}$ become

$$\begin{aligned}\mathbf{x}_{g_1} &= (2,1,1,1,2,1,0,1,0),\\ \mathbf{x}_{g_2} &= (2,1,1,1,1,2,0,1,0).\end{aligned}$$

Obviously, these two vectors are now a more proper representation of the graphs in Fig. 1.

With respect to the complexity of the construction of such a vector, it is worth mentioning the fact that all features that are being considered can be obtained by just visiting the nodes of the graph and their respective entries in the adjacency matrix. Other embedding methodologies look for paths or cycles in the graphs with the same label sequence [15, 16]. These procedures and others based on finding substructures in the graphs require for more expensive operations. The embedding based on (1) and (2) is a particular case of them, in which we look for paths of length 0 (labels of the nodes) and paths of length 1 (edges between nodes with specific labels).

Another key issue to notice at this point is the nature of the graphs to which the method according to (1) and (2) is applicable. There is no restriction on the nature of the graphs from which we can extract such a vectorial representation. This is, there is no specification about the set of node labels. Nevertheless, it seems clear that counting frequencies of node labels in the graph is only feasible if such labels are discrete. If they were continuous, the proposed procedure would lead to highly sparse vectors of possibly infinite dimensions (depending on whether we consider all possible labelling values or not). The work described in this paper is mainly concerned with the application of this methodology to graphs whose nodes have continuous labels.

3.2. Transition to the continuous case: node attribute representatives

From now on, we consider the case where the set of node labels is $L_V = \mathbb{R}^d$. We propose a methodology that is based on the idea that the more similar two node labels are the more likely they should be considered the same. In other words, the nodes of the graphs are assigned to some *representatives* of the node labels in terms of Euclidean similarities, and then only statistics of these representatives are counted as features in the vectorial representation of the graph. As representatives of a set we chose those points -not necessarily in the set- that best represent the whole set, just as the classical cluster analysis techniques do.

We now formally describe the complete embedding methodology. Suppose we are given a set of N graphs $\mathcal{G} = \{g_1, \dots, g_N\}$. Each of these graphs has nodes with continuous attributes. This is, for all $i \in \{1, \dots, N\}$, the set of nodes attributes is $L_{V_i} = \mathbb{R}^d$. Let $\mathcal{P} \subset \mathbb{R}^d$ be the set of all node labels in all the graphs of $\mathcal{G}$. Furthermore, let $\mathcal{W} = \{w_1, \dots, w_n\}$ be a set of n representatives of all vectors in $\mathcal{P}$. In Section 4 we describe different approaches for defining such sets of vectors. Elements in $\mathcal{W}$ do not necessarily belong to $\mathcal{P}$. The *node to representative* map is a function assigning every node of a given graph to the vector in the representative set which is closer to the label of the node. Formally, for every graph, we have

$$\begin{aligned}\lambda_h : V &\longrightarrow \mathcal{W}\\ v &\longmapsto \lambda_h(v) = \operatorname*{argmin}_{w_i \in \mathcal{W}} \parallel \mu(v) - w_i \parallel_2 \qquad (4)\end{aligned}$$

where $\parallel \cdot \parallel_2$ stands for the Euclidean metric.

Using this function, we can redefine the concept of appearance of a specific label by checking how many nodes have been assigned to this specific label. This is, given a graph $g = (V, E, \mu)$ and a representative set $\mathcal{W}$, the frequency of a representative $w_i \in \mathcal{W}$ is

$$U_i = \#(w_i, g) = |\{v \in V \mid w_i = \lambda_h(v)\}|. \qquad (5)$$

Similarly, the frequency of a specific relation between two representatives is defined as

$$\begin{aligned}B_{ij} &= \#(w_i \leftrightarrow w_j, g)\\ &= |\{(u,v) \in E \mid w_i = \lambda_h(u) \wedge w_j = \lambda_h(v)\}|. \qquad (6)\end{aligned}$$

It is important to notice the symmetry of the features $\#(w_i \leftrightarrow w_j, g)$ as long as the involved graph g is undirected. Because of this symmetry, we will consider each such feature once instead of both $\#(w_i \leftrightarrow w_j, g)$ and $\#(w_j \leftrightarrow w_i, g)$.

Definition 2 (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

$$\varphi_{\mathcal{W}}(g) = (U_1, \dots, U_n, B_{11}, \dots, B_{ij}, \dots, B_{nn}), \qquad (7)$$

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

Let us illustrate this definition with an example. In Fig. 2, all the nodes of the four depicted graphs have attribute values that are close to either one of the following points: $\{(0,0), (0,1), (1,0), (1,1)\}$. These four points are a natural set of representatives for the whole set of node labels of the four different graphs. By using this set of representatives, we would have four different node labels and, for example, the node of the leftmost graph with label $(0, 0.8)$, will count as an appearance of its closest representative $(0, 1)$,

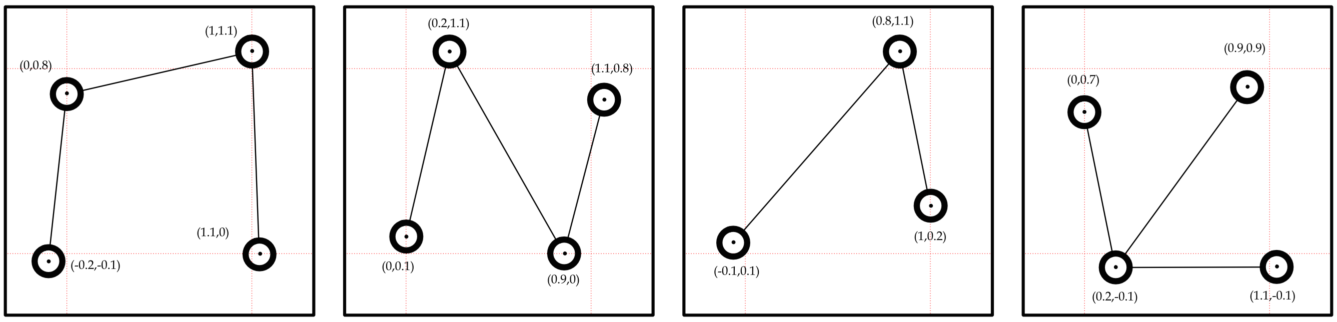


Figure 2: Four graphs whose node attributes from $\mathbb{R}^2$ are all around one of the four representatives $\{(0,0),(0,1),(1,0),(1,1)\}$.

thus incrementing the corresponding U_i feature by one. Regarding the relations between labels, the B_{ij} features, the edge in the second graph connecting the nodes with labels $(0.2, 1.1)$ and $(0.9, 0)$, would count as an appearance of the relation between the representatives $(0, 1)$ and $(1, 0)$, which are the corresponding nearest representatives to the node attributes.

Given a dataset of graphs, the graph embedding methodology we propose strongly depends on the set of representatives of the nodes. We will experimentally show the effect of choosing such set of representatives by different approaches and also of different size.

3.3. The fuzzy version

In noisy situations, it might be the case that a node label is between two representatives, and there is no clear rule telling us to which representative the node should be assigned. Also, a node label could be far away from all representatives such that no element of $\mathcal{W}$ is actually a proper representative of the node. These are typical situations where a *soft* rather than a *hard* assignment can be beneficial. *Fuzzy* clustering assigns to every node a certain degree of belongingness to every cluster, rather than saying that a node is assigned to just one representative [1, 31]. In this work, we also address this situation and propose a fuzzy version of the graph vectorization procedure.

While Definition 2 is still our basic embedding approach, the features U_k and U_{ij} will be redefined. Eq. (4) is no longer usable for the assignment of nodes to representatives and every node will be given a certain degree of assignment to every representative. Formally, we define the function

$$\begin{aligned} \lambda_s : V &\longrightarrow S_n^+ \subset \mathbb{R}^n \\ v &\longmapsto \lambda_s(v) = (p_1(v), \ldots, p_n(v)), \end{aligned} \tag{8}$$

where $p_i(v) = P(v \in w_i)$ is the probability of the node v being assigned to 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 in order to ease the methodology, while requiring that $p_i(v) \geq 0$ and $\sum_{i=1}^n p_i(v) = 1$.

In this situation, the appearance frequency of a certain representative, Eq. (5), has to be reformulated in term of the probabilities of belongingness for all nodes in the graph. This leads to

$$U_i = \#(w_i, g) = \sum_{v \in V} p_i(v). \tag{9}$$

When there is an edge between two nodes in the graph, $(u, v) \in E$, the fact that nodes are assigned to representatives according to Eq. (8), makes it unclear how one should compute the features U_{ij}. Assume we have available the corresponding fuzzy assignment representations of the source and the target nodes:

$$\begin{aligned} \lambda_s(u) &= (p_1(u), \ldots, p_n(u)), \\ \lambda_s(v) &= (p_1(v), \ldots, p_n(v)). \end{aligned}$$

From these two vectors of probabilities we need to find out how much the edge (u, v) is contributing to the relation between two representative points. We handle this situation using two different approaches.

1) The first one is the naive and conservative approach in which the edge only contributes to the relations of the representatives with maximum probability. This is, if w_t is the representative to which the node u has maximum probability of belongingness and w_s the corresponding one for the node v, then the edge (u, v) will only count as a relevant relation between the representatives w_t and w_s. Formally,

$$B_{ij} = \#(w_i \leftrightarrow w_j, g) = \sum_{(u,v) \in E} \delta_{ij}(u, v), \tag{10}$$

where

$$\delta_{ij}(u, v) = \begin{cases} 1, & \text{if } w_i = \underset{w_k \in \mathcal{W}}{\text{argmax}}\ p_k(u) \\ & \quad \text{and } w_j = \underset{w_k \in \mathcal{W}}{\text{argmax}}\ p_k(v), \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

This procedure for constructing the B_{ij} features will be referred to the *max* assignment, since we only take into account the edge relation from the most (maximum) probable representative of one node to the most (maximum) probable representative of the other one.

2) The second approach is going to keep considering all possible assignments from one node to the other, and thus, and edge $(u, v) \in E$ will contribute to all relations between any two representatives. In particular, we call this approach the *all* assignment method since all probabilities are taken into account. In particular, we define

$$\begin{aligned} B_{ij} &= \#(w_i \leftrightarrow w_j, g) \\ &= \sum_{(u,v)\in E} p_i(u)p_j(v) + p_j(u)p_i(v). \end{aligned} \tag{12}$$

The intuition behind (12) 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 amount of probability of travelling from the part of u assigned to w_i to the part of v assigned to w_j, 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 the part of v which is assigned to w_i to the part of u assigned to w_j, *i.e.*, $p_j(u)p_i(v)$.

3.4. Discussion

As already stated above, the main idea behind this methodology is that similar nodes will count for the same feature with similar weights. We assume there is an intrinsic topological model for every category in a given database of graphs and by this approach we believe that we can extrapolate such model, via undoing possible deformations on the different instances of a certain class.

This way, it seems reasonable that the methodology is going to perform well on graphs whose continuous attributes of the nodes describe positions on the plane. Indeed, as we will see later (Section 5), the results confirm this hypothesis since we work on graphs of this kind and the results are quite satisfactory. Moreover, this point is reinforced by the fact that in those cases where we have severe deformations, the proposed methodology does not perform properly, and this is because such high degree of distortion makes it impossible to discover the intrinsic model for each class.

Nevertheless, we believe that this methodology is going to perform well with graphs with higher dimensional node attributes for the same reasons, but this is out of the scope of this work. Also, as proven in other works [24, 32], the methodology is discriminative enough for graph whose nodes are labelled with discrete attributes.

4. Selection of representatives

Selecting a proper set of representatives for the set of all graphs in a given dataset is a crucial issue in the proposed embedding methodology. This section focuses on describing four different approaches of classical cluster analysis in order to select representatives. Two of the considered methods are hard methods and two are soft (fuzzy) ones. For the rest of this section, let $\mathcal{P} \subset \mathbb{R}^d$ be a set of m labels from which we want to extract a set $\mathcal{W}$ of n representatives.

4.1. Spanning prototypes

The first representative (hard) selector considered in this work is an approach that tries to find points as much uniformly distributed as possible within the whole range of points at hand [21]. The algorithm starts by selecting the point that minimizes the sum of distances to all the other points (called the median vector) and then, it keeps adding points to the representative set by iteratively selecting the point which is furthest away -the one maximizing the minimum distance- to the already selected set of representatives. The algorithm stops when a predefined number of points is obtained.

The algorithm is capable to find a set of representative vectors $\mathcal{W}$ that *span* the whole range of points in $\mathcal{P}$ as uniformly as possible.

4.2. kMeans algorithm

The second hard selector for the set of representatives is the very well-known kMeans clustering algorithm [1]. Briefly, this algorithm starts by initializing the set of cluster centres $\mathcal{W}$ to some random points and then assign each point in $\mathcal{P}$ to its closest centre. The set $\mathcal{W}$ is updated as the mean of all points assigned to the same cluster. These steps run iteratively until there are no changes in the set $\mathcal{W}$.

A common problem one encounters using this algorithm is the uncertainty in the results due to the random initialization of the set $\mathcal{W}$. A possible solution is to repeat the experiment a certain number of times and finally average the results. In our case, the kMeans algorithm is always deterministically initialized with a set of points of fixed size using the spanning prototypes described in the previous section.

4.3. Fuzzy kMeans

The next approach for the selection of representatives is the Fuzzy kMeans algorithm [1]. Its main idea 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

$$p_i(x) = \alpha \cdot \left(\frac{1}{\| x - w_i \|_2} \right)^s, \tag{13}$$

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 s is the more weight is given to points close to the centres. In our experiments we use $s = 2$.

The algorithm is basically the same as kMeans, although the assignment from points to clusters is made by means of

Eq. (13), and the update of the clusters at each iteration is done by a weighted mean of all points (weighted by their degree of belongingness to each specific cluster). To avoid uncertainty, we also initialize the algorithm using the spanning prototypes.

4.4. Mixture of Gaussians

As the last representative set selector, we make use of an important probabilistic framework in which all the data are assumed to be generated by a model in which several probability distributions are involved. In particular, we make use of a Gaussian Mixture Model (GMM) or Mixtures of Gaussians [1]. Technically, the set of points in $\mathcal{P}$ is generated by

$$f(x) = \sum_{i=1}^{n} \pi_i \mathcal{N}(x \mid \mu_i, \Sigma_i), \qquad (14)$$

which is a linear mixture of n Gaussian densities $\mathcal{N}(\cdot \mid \mu_k, \Sigma_k)$, where μ_k is the mean vector and Σ_k the covariance matrix. The parameters π_k are usually called mixing coefficients and can be understood as probability weights for the mixture components. The estimation of the involved parameters (mixing coefficients, means and covariances) is usually done by maximization of the log-likelihood of the mixture with respect to the parameters. In our experiments, this estimation has been carried out by means of the Expectation-Maximization (EM) algorithm [33, 34].

Note that in this scenario the set of representatives $\mathcal{W}$ is no longer a set of points but a set of probability (Gaussian) densities. We can thus assign the degree of membership of a point x to a certain representative $w_i = \mathcal{N}(\cdot \mid \mu_i, \Sigma_i)$ by the probability

$$p_i(x) = \beta \cdot \mathcal{N}(x \mid \mu_i, \Sigma_i), \qquad (15)$$

where β will be again a constant making $\sum_{i=1}^{n} p_i(x) = 1$.

For the initialization of the parameters in the EM algorithm we use the kMeans results as described above. For a Gaussian $\mathcal{N}(\cdot \mid \mu_i, \Sigma_i)$, its mean μ_i is initialized as the i-th kMeans cluster centres, the mixing coefficient π_i is the amount of point mass assigned to this cluster and the covariance Σ_i is the covariance matrix of the data that were assigned to the i-th kMeans cluster.

4.5. Summary

We here briefly summarize the different configurations of the presented graph embedding methodology. In the hard case of the representative set construction we have two configurations:

- Spanning prototypes (hard assignment)
- kMeans (hard assignment)

while for the *soft* methodologies we have two ways of constructing the representative sets -and thus to assign nodes of the graphs-, and also two ways of constructing the U_{ij} features. This leads to another four embedding configurations:

- Fuzzy kMeans + Soft *max* edge assignment
- Fuzzy kMeans + Soft *all* edge assignment
- Mixture of Gaussians + Soft *max* edge assignment
- Mixture of Gaussians + Soft *all* edge assignment

In the experimental part of this work (see Section 5), we will test these six embedding configurations on different datasets to get more insight into the strength of the proposed methodology.

5. Experimental evaluation

In this section we present different experiments that will illustrate the strengths of the proposed embedding methodology and provide insight into the different configurations summarized in Section 4.5. We will evaluate our vectorial representation of graphs in terms of accuracy rates achieved in the context of different classification tasks on different sets of graphs. The next subsections describe the databases that have been used, the reference systems we compare our methodology with, the validation of the proposed embedding configurations and, finally, the results.

5.1. 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 [36].

The first three datasets of graphs are the *Letter Databases*, which represent synthetic 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 fourth graph dataset is the *GREC Database* [37], 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.

The next set of graphs is the *Digits database.* This data set is representing handwritten digits [38]. The digits were originally acquired by recording the pen position at constant steps of time. The sequence of (x, y) coordinates

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
GREC	286, 286, 528	22	11.5/12.2	25/30
Digits	1000, 500, 2000	10	8.9/7.9	17/16
Fingerprints	500, 300, 2000	4	5.4/4.4	26/25
COIL	2400, 500, 1000	100	21.5/54.2	77/222

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 [39] 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.

Finally, the *COIL database* is a subset of the COIL-100 database [40]. 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 [41], labelling these points with their corresponding coordinates on the 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 results subsets and other relevant information concerning the datasets is provided.

5.2. Reference systems

As already discussed before, there is an important lack of processing tools in the graph domain. For graph-based pattern classification we are mainly restricted to use a k Nearest Neighbour classifier based on the edit distance between graphs (any other distance measure could be used as well). This classifier will be our reference system in the graph domain. As for the edit distance computation, we use the suboptimal approximation described in [42].

The second reference system is a classifier on another embedding space. In particular, we use the embedding methodology proposed in [22]. 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

$$\phi_n^{\mathcal{P}}(g) = (d(g, p_1), \ldots, d(g, p_n)), \tag{16}$$

where $d(g, p_i)$ is the edit distance between the graph g and the prototype p_i. We will not explain here how the set of prototypes has been selected nor how many prototypes were used, but will only report the best results of this approach using a Support Vector Machine on the set of vectors defined by Eq. (16).

5.3. The effect of the representative set

As we said, we test our proposed methodology under different classification scenarios. In the first place, we want to validate both the number of elements in the set of representatives and the different configurations of the embedding. We construct all vectors from the different configurations using a number of representatives in a predefined range. In particular, we vary the size of the set of representatives from 5 to 100 in steps of 5. This assures that a rather large interval of essentially different representations of the graphs is explored.

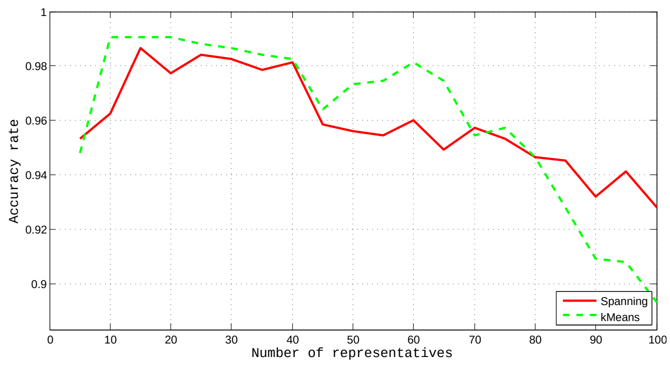
After the vectors are constructed, we classify them using a kNN classifier. We pick a simple classifier like kNN for two reasons: the first one is that it really does not need a complex parameter tuning step (besides the value of the number of neighbours, which is validated on the validation set), and the second one is that this classifier already gives a good and reliable measure of how the vectors are class-wise distributed in the input space. Together with the kNN classifier, different distance measures have been tried, such as the L_1, L_2 or χ^2 metrics. Due to its broad applicability to histogram-based feature vectors [43] and to the recognition rates obtained, we here report the results using the χ^2 distance, which is defined for two vectors $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ in $\mathbb{R}^n$ by

$$d_{\chi^2}(x, y) = \frac{1}{2} \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{(x_i + y_i)}. \tag{17}$$

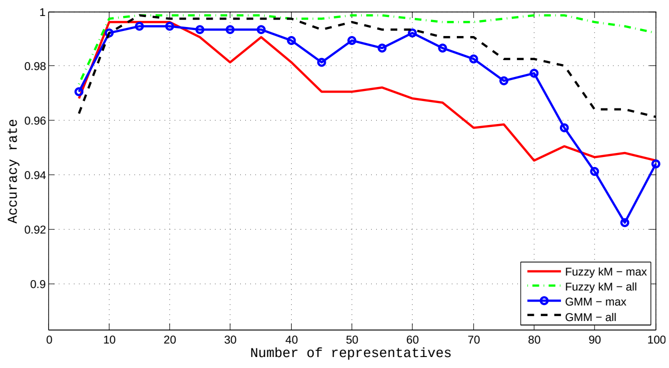
Figures 3 and 4 show the results for all the datasets on the validation set. We have distinguished the hard cases from the soft ones using separated figures.

Let us first discuss the case of the Letter databases. There is a clear difference in the results between the low distortion case and the medium and high ones. The more distorted the graphs are, the lower the results are obtained. We discuss these cases separately.

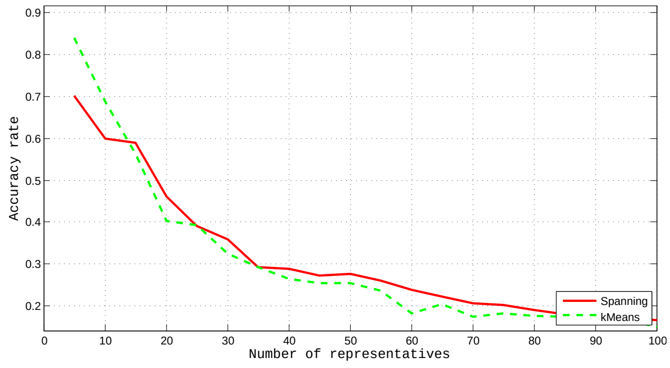
With respect to the low distorted database, we can see that the kMeans configuration adapts more properly to the node distribution than the spanning prototypes method, and thus the accuracy rates are higher (Fig. 3(a)). Nevertheless, the recognition rate tends to decrease as the size of the representative set is increasing, which is explained by the fact that the more representatives are considered, the more sparse the resulting vectors become, making the classifier not able any more to distinguish among the different



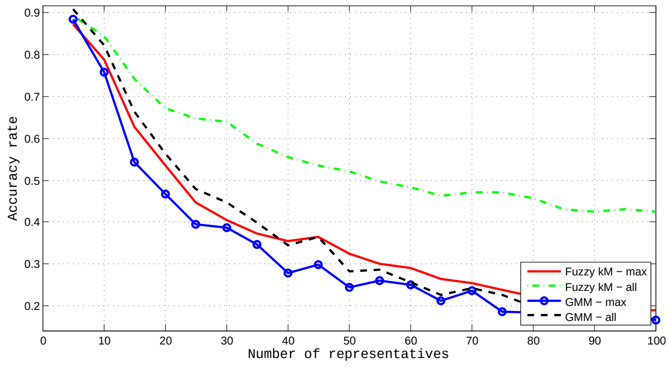
(a) Letter LOW; hard configurations



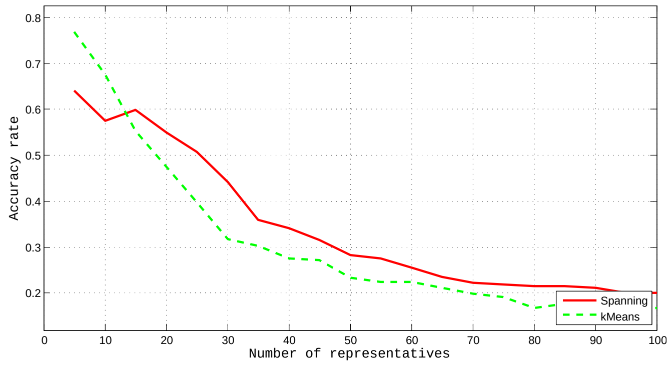
(b) Letter LOW; soft configurations



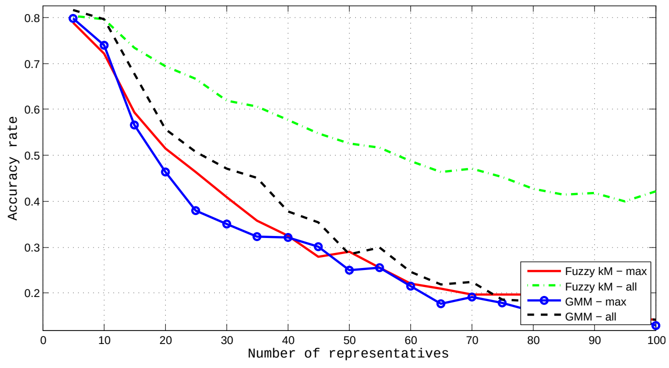
(c) Letter MEDIUM; hard configurations



(d) Letter MEDIUM; soft configurations



(e) Letter HIGH; hard configurations



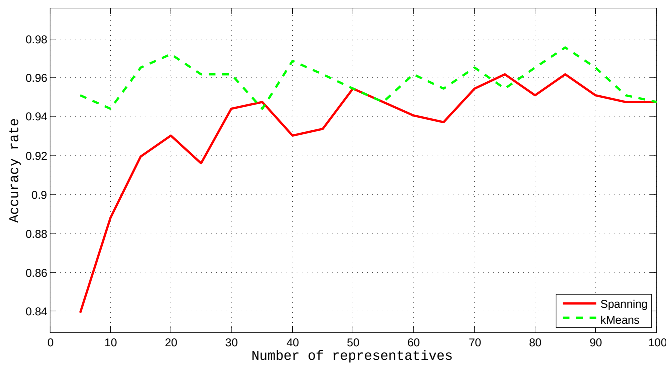
(f) Letter HIGH; soft configurations

Figure 3: Validation results for the different databases of letters. Accuracy rates of a kNN classifier in conjunction with a χ^2 distance on the validation set. The horizontal axis shows different choices of the size of the representative set. The hard configurations of the proposed embedding are depicted in the figures on the left. Soft configurations are shown on the right column of figures.

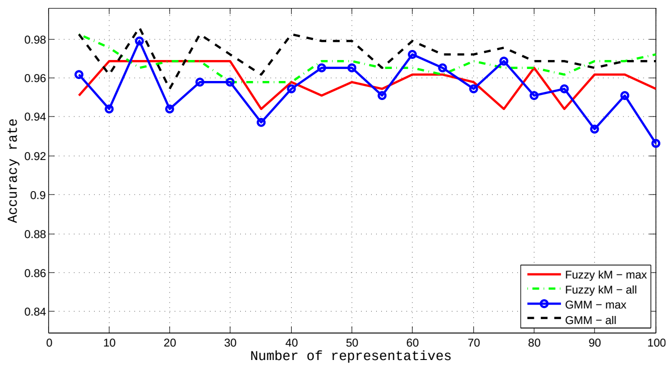
classes. The soft versions (Fig. 3(b)) perform generally better than the hard ones and the fuzzification makes the results more stable as the size of the representative set increases. It is also interesting to note that the *all* configurations are more capable than the *max* ones.

In cases with medium and high distortion (Figs. 3(c)-3(f)), the input graphs are so badly distorted that there is no actual way to properly reflect the discriminative features of every class in the vectorial representation. Since the nodes of the graph are not in their expected location, the representatives they are assigned to are not the ones they should be, which results in highly distorted vectorial features. This fact even leads to accuracy rates as low as 20% for large representative sets, while for small ones the results are still acceptable but definitively not good. This behaviour is evident for the hard and the soft versions of our embedding procedure.

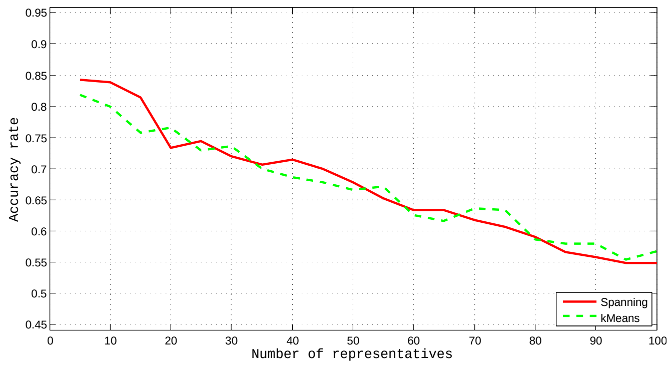
Regarding the GREC dataset, the results of the kMeans selector are again better than the spanning prototypes ones, which makes sense since kMeans adapts in a more accurate manner to the inherent clusters of the label space. Here again, the soft versions obtain better results than the hard configurations, supporting the idea of a better adaptation to the possible deformations in the represented objects. Concerning the differences between the soft assignments, both the fuzzy kMeans and the mixture of Gaus-



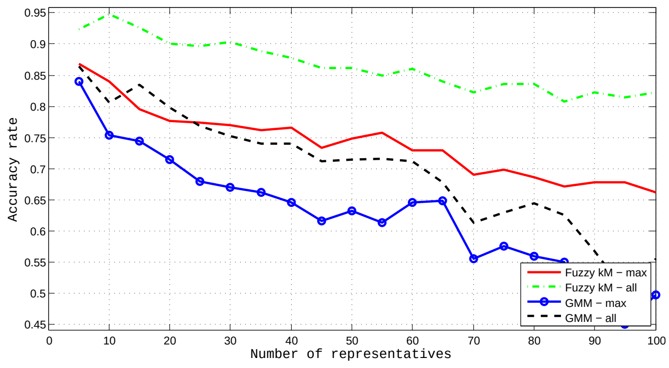
(a) GREC; hard configurations



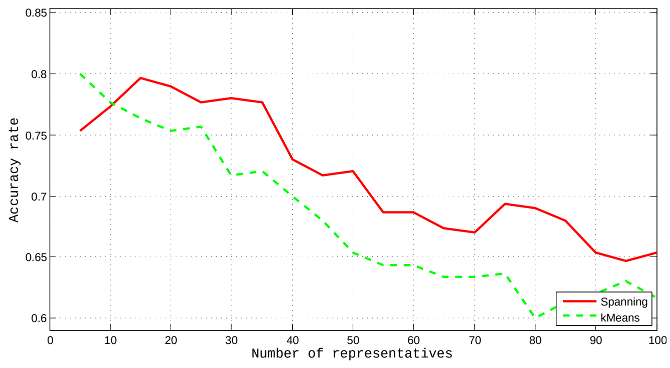
(b) GREC; soft configurations



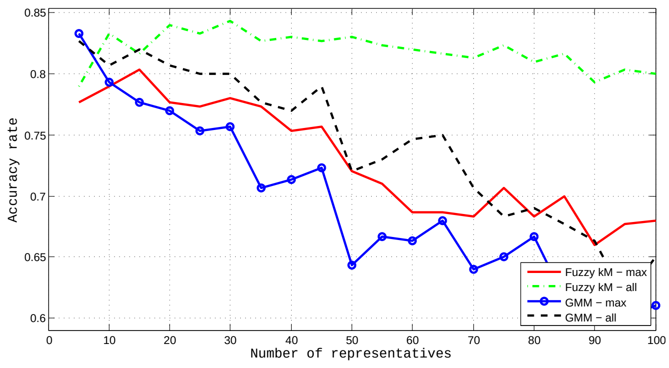
(c) Digits; hard configurations



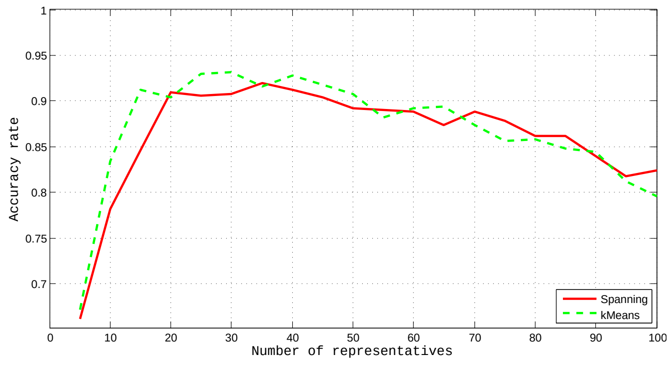
(d) Digits; soft configurations



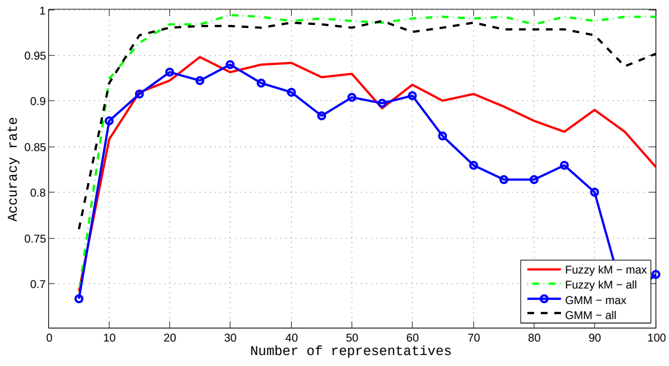
(e) Fingerprints; hard configurations



(f) Fingerprints; soft configurations



(g) COIL; hard configurations



(h) COIL; soft configurations

Figure 4: Validation results for the GREC, Digits, Fingerprints and COIL databases. Accuracy rates of a kNN classifier in conjunction with a χ^2 distance on the validation set. The horizontal axis shows different choices of the size of the representative set. The hard configurations of the proposed embedding are depicted in the figures on the left. Soft configurations are shown on the right column of figures.

sians methods obtain better results for the *all* approach than for the *max* one.

The results for the Digits and the Fingerprint datasets obey similar behaviours and thus can be discussed together. In these two datasets of graphs a peculiar phenomenon happens. Here, the spanning prototypes usually perform equally or even better than the kMeans configurations (Figs. 4(c) and 4(e)). The reason for this fact is the inherent distribution of the nodes in the label space. These are uniformly distributed among their range just like a regular grid in the space, preventing kMeans from properly discovering good representatives. In any case, small sets of representatives perform better since global relations in the nodes of the involved graphs are more accurate to describe their shape (and thus their class). Among the soft versions (Figs. 4(d) and 4(f)), the fuzzy kMeans with the *all* edge assignment stands out with respect to the other soft versions, since this configuration is capable to perform in a stable manner along the increasing sets of representatives.

Finally, the COIL database shows no relevant differences between the two hard versions of the proposed embedding (Fig. 4(g)) but a performance worth mentioning in the case of the *all* edge assignment configurations. This behaviour is explained by the nature of the database itself. The Harris salient point detector is quite unstable and, moreover, the images in the COIL database are turning around. By fuzzifing the assignment of nodes (fuzzy kMeans and GMM) and the assignment of edges (*all*), the proposed methodology is able to adapt to the changes that two similar images may show in their respective graph representations.

5.4. Results

After having studied the influence of the set of representatives for each database by means of a kNN classifier, we now apply a more sophisticated classifier to our vectorial representation of graphs. As discussed in the introduction of this work, this is in fact one of the main goals that are wanted to be achieved, this is, apply a complex learning algorithm to graph-based input representations. For each database, we pick up the best performing representative set for the six different configurations of the proposed embedding in terms of the accuracy rate of the previously presented kNN classifier results. With these vectors at hand, we train a support vector machine and validate the meta-parameters using the corresponding validation set. Solving the tasks of learning the set of representatives and the meta-parameters of the SVM classifiers by two different tasks is clearly suboptimal. Nevertheless, we proceed in this way because the joint task would not be computationally feasible since one SVM classifier would have to be trained for every choice of the size of the representative set.

For the SVM learning, we have used a linear kernel and the associated C parameter and a radial basis function kernel with the C and γ parameters associated to it. Due to the fact that the validation of the number of elements in the set of representatives is done using a χ^2 distance (Eq. (17)), we also use a χ^2 kernel for the SVM training step. The χ^2 kernel is defined analogously to the radial basis function kernel as:

$$\kappa_{\chi^2}(x, y) = \exp(-\gamma \cdot d_{\chi^2}(x, y)), \tag{18}$$

for two vectors $x, y \in \mathbb{R}^n$ and $\gamma > 0$. Finally, with the best performing set of parameters on the validation set, we retrain the SVM model on the train set and apply it on the test set. The results are reported in Table 2.

As we can see in the table, the results of the proposed embedding methodology can compete with the two chosen reference systems in both the graph and the embedding domain. Regarding the reference system in the graph domain, there is, for all databases, at least one of the six proposed configurations that is tied or even statistically better than a kNN classifier using the Graph Edit Distance.

These results suggest that the proposed embedding methodology is able to keep the inter-class and intra-class distances among graphs, at least to the same degree as Graph Edit Distance does. Moreover, a tied result compared to the edit distance classifier is indeed a success since the computation of the edit distance is more costly than the methodology we propose in this work. As reported in [42], the edit distance approximation has a complexity of $\mathcal{O}(n^3)$, where n is the size of the involved graphs (number of nodes). In our case, the description of a graph by a feature vector can be obtained by visiting the n nodes and the m edges of a graph, leading to a complexity of $\mathcal{O}(n+m)$, plus the cost of comparing each node to the set of representatives.

The results that are statistically significantly lower than those obtained with the first reference system can all be explained by the nature of the graphs where these results are gotten from. In particular, we only get lower results than the first reference system in some (not all) configurations of the proposed embedding approach for the medium and highly distorted letters and the Digits databases. The graphs in these sets represent objects with an inherent highly distorted structure, making the assignment of nodes to representatives a confusing step for the proposed approach, and thus obtaining low classification rates because we are not able to learn the intrinsic category model. In situations like this and although its high computational complexity, the edit distance is still an effective measure since deformations are properly taken into account. Nevertheless, it is worth noting that, even in these cases, the fuzzy kMeans with the *all* assignment configuration performs equally (or better) than the reference system.

With respect to the second reference system, the results of the proposed embedding approach are again adequate. The soft versions of the embedding methodology together with the *all* assignment of edges give rise to accuracy rates that are comparable to that of the embedding reference system. Only in the case of the highly distorted letters

Table 2: SVM results on the test set.

	Reference systems		Embedding configurations					
Dataset	kNN - Graph Edit Distance	Dissimilarity Embedding	Spanning Hard	kMeans Hard	Fuzzy kMeans Soft *max*	Fuzzy kMeans Soft *all*	GMM Soft *max*	GMM Soft *all*
Letter low	99.3	99.3	99.0	99.2	99.6	99.8	99.4	99.7
Letter medium	94.4	94.9	75.4 ❶❷	88.4 ❶❷	88.1 ❶❷	92.8	90.0 ❶❷	93.0
Letter high	89.1	92.9	67.2 ❶❷	82.0 ❶❷	85.0 ❶❷	87.7 ❷	84.6 ❶❷	87.8 ❷
GREC	95.5	95.1	97.7 ①②	97.9 ①②	97.9 ①②	98.1 ①②	99.2 ①②	99.0 ①②
Digits	97.4	98.7	88.9 ❶❷	89.3 ❶❷	91.3 ❶❷	97.1 ❷	87.0 ❶❷	91.4 ❶❷
Fingerprint	79.1	83.1	79.7 ❷	81.5 ①	80.4 ❷	81.5 ①	82.0 ①	81.8 ①
COIL	93.3	96.8	92.1 ❷	93.1 ❷	92.9 ❷	97.3 ①	93.5 ❷	98.1 ①②

①/② Statistically significant improvement over the first/second reference system (Z-test using $\alpha = 0.05$).
❶/❷ Statistically significant deterioration over the first/second reference system (Z-test using $\alpha = 0.05$).

and the Digits dataset -which are also highly distorted in nature- the results are statistically worse. The explanation for this fact is the same as the one given for the first reference system, namely that the edit distance is much more capable to cope with high distortions in graphs than the proposed embedding approach. Moreover, the dissimilarity representation based on different prototypes helps with the performance on these difficult graphs.

In the other databases, we obtain a tied result (or a better one in the GREC and COIL cases) which should be considered as a success due to the computational complexity of both methods. The complexity of the reference system is governed by the computation of the edit distance between each graph and the set of prototype graphs. As already discussed before, graph edit distance is much more costly than the construction of our vectorial representation.

Despite the good results obtained for the soft versions of the proposed embedding, together with the *all* edge assignment, it should be noted that the other four representations are not generally capable to get comparable results to the second reference system. Only by fuzzifying the node (Fuzzy kMeans and GMM) and the edge assignments (*all* method, Eq. (12)), we are capable to compete with the chosen reference system. Nevertheless, these remarkable results, together with its low computational complexity, make the described embedding methodology of graphs an attractive choice.

6. Conclusions

Feature vectors for pattern recognition are of great importance due to their easy manipulation and the wealthy algorithmic repository available. It is known, though, that graph-based representations can overcome their limitations in terms of their representational power. However, the use of graphs as inputs to be processed is restricted to the kNN family of algorithms for data analysis. Graph embedding in vector spaces is a powerful technique to bridge the gap between the representational power of graphs and the rich set of algorithms that are available for feature vector representations of patterns.

In this work, we have proposed a novel technique for embedding a graph into a vector space. The proposed approach explicitly builds a vector of features for each graph counting the frequency of appearance of a specific set of representatives of the labels of the nodes and their respective relations in terms of appearing edges. The embedding lies on the idea that two nodes should count as the same representatives if they are close enough, and an edge in the graph should count as the relation between the closest representatives of the nodes that this edge in linking. It is crucial, thus, a proper selection of this set of representatives. In this paper, different representative selectors have been used and also different corresponding edge assignment methodologies have been proposed.

The major benefit of this methodology is the fact that, unlike most of the other graph embedding methodologies, it does not require a high computational cost. The extracted features can be obtained by just visiting the explicit representation of the graphs -node labels and adjacency matrix- and they do not depend upon searching substructures among the graphs, which dramatically increases the complexity of other techniques.

We have provided a wide experimental study of the proposed methodology. It reveals that the proposed algorithm can compete with state-of-the-art classification approaches, both in the graph domain and in another embedding space, even when other techniques make use of domain-specific knowledge. In particular, we have used several databases of graphs describing objects of different nature. Despite the simplicity of the feature we propose in this work, we have reported results that are statistically at the same levels as the those of the reference systems.

The question of which should be the representative set selector has no clear answer. The proper answer would be in terms of every specific database that is being analysed. Different databases have different distribution of node attributes and thus different selectors should be considered. In any case, those selectors that aim at correcting possible wrong node attributes due to a noisy extraction process of graphs obtain better results that those that do not take care of such situations.

There is still a lot of work ahead with respect to the proposed methodology. First of all, the embedding approach has been only evaluated under graphs whose node attributes are points on the plane. It would be interesting to check for its performance on graphs with higher dimensional labels. Another main concern of the authors is how to apply this methodology to edge attributed graphs. The way it has been proposed does not allow for the introduction of labelling information in the edges of the graphs, since only appearances between representatives are taken into account as long as there is an edge between two nodes close to those representatives. An important issue that should also be considered is the fact that different representative sets could be complementing ones to others, thus incorporating the proposed methodology into a multiple classifier system could be beneficial for classification purposes. Finally, in this work, only SVM classifiers have been considered from the wide set of tools that are available in the vector domain. Perhaps other classifiers are more capable to exploit the features we propose for embedding a graph into a vector space.

Acknowledgments

This work has been supported by ... Moreover, the authors would like to thank Dr. Kaspar Riesen for providing us with the databases that are used in this work and the corresponding implementations of the reference systems.

References

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

[2] C. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, 1995.

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

[4] 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).

[5] H. Bunke, K. Riesen, *Towards the unification of structural and statistical pattern recognition*, Patter Recognition Letters, accepted for publication.

[6] T. Gärtner, *Kernels for Structured Data*, World Scientific (2008).

[7] J. Lafferty, G. Lebanon, *Information diffusion kernels*, Advances in Neural Information Processing Systems, vol. 15, MIT Press, 2003, pp. 375-382.

[8] R. Kondor, J. Lafferty, *Diffusion kernels on graphs and other discrete input spaces*, in: Proceedings of 19th International Conference on Machine Learning, 2002, pp. 315-322.

[9] D. Haussler, *Convolution kernels on discrete structures*, Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

[10] C. Watkins, *Dynamic alignment kernels*, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (Eds.), Advances in Large Margin Classifiers, MIT Press, 2000, pp.39-50.

[11] H. Kashima, K. Tsuda, A. Inokuchi, *Marginalized kernels between labelled graphs*, in: Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 321-328.

[12] K. Borgwardt, C. Ong, S. Schönauer, S. Vishwanathan, A. Smola, H.P. Kriegel, *Protein function prediction via graph kernels*, Bioinformatics 21 (1) (2005), pp. 47-56.

[13] T. Gärtner, P. Flach, S. Wrobel, *On graph kernels: hardness results and efficient alternatives*, in: B. Schölkopf and M. Warmuth (Eds.), Proceedings of the 16th Annual Conference on Learning Theory, 2003, pp. 129-143.

[14] S. Vishwanathan, N.N. Schraudolph, R. Kondor, K. Borgwardt, *Graph Kernels*, Journal of Machine Learning Research, 11 (2010), pp. 1201-1242.

[15] 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.

[16] 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.

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

[18] 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.

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

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

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

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

[23] 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 Science, vol. 6669, Springer, 2011, p.216-223.

[24] 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 Science, vol 6419, Springer, 2010, pp. 30-37.

[25] H. Bunke, K. Shearer, *A graph distance metric based on the maximal common subgraph*, Pattern Recognition Letters 19 (3) (1998), pp. 255-259.

[26] M.L. Fernandez, G. Valiente, *A graph distance metric combining maximum common subgraph and minimum common supergraph*, Pattern Recognition Letters 22 (6-7) (2001), pp. 753-758.

[27] W.D. Wallis, P. Shoubridge, M. Kraetzl, D. Ray, *Graph distances using graph union*, Pattern Recognition Letters 22 (6) (2001), pp. 701-704.

[28] H. Bunke, G. Allermann, *Inexact graph matching for structural pattern recognition*, Pattern Recognition Letters 1 (1983), pp. 245-253.

[29] A. Sanfeliu, K.S. Fu, *A distance measure between attributed relational graphs for pattern recognition*, IEEE Transactions on Systems, Man, and Cybernetics (Part B), 13 (3) (1983), pp. 353-363.

[30] A. Robles-Kelly, E.R. Hancock, *Graph edit distance from spectral seriation*, IEEE Transactions on Pattern Analysis and Machine Intelligence 27 (3) (2005), pp. 365-378.

[31] J. van Gemert, J. Geusebroek, C. Veenman, A. Smeulders, *Kernel Codebooks for Scene Categorization*, in: D. Forsyth, P. Torr, A. Zisserman (Eds.), Proceedings of the 10th European Conference in Computer Vision, Lecture Notes in Computer Science, vol 5304, Springer, 2008, pp. 696-709.

[32] J. Gibert, E. Valveny, H. Bunke, *Embedding of Graphs with Discrete Attributes via Label Frequencies.* Submitted to IJPRAI.

[33] A.P. Dempster, N.M. Laird, D.B. Rubin, *Maximum likelihood for incomplete data via the EM algorithm*, Journal of the Royal Statistical Society, B 39 (1) (1977), pp. 1-38.

[34] G.J. McLachlan, T. Krishnan, *The EM algorithm and its extensions*, Wiley (1997).

[35] B. Schölkopf, A. Smola, K.R. Müller, *Nonlinear Component Analysis as a Kernel Eigenvalue Problem*, Neural Computation, 10 (1998), pp. 1299-1319.

[36] 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.

[37] 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.

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

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

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

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

[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.

[43] 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.