Graph embedding using constant shift embedding — **Source link**

Salim Jouili, Salvatore Tabbone

Institutions: French Institute for Research in Computer Science and Automation

Related papers:

- IAM Graph Database Repository for Graph Based Pattern Recognition and Machine Learning

- Approximate graph edit distance computation by means of bipartite graph matching

- Pattern vectors from algebraic graph theory

- Graph embedding in vector spaces by node attribute statistics

- Thirty years of graph matching in pattern recognition

Share this paper:

Graph Embedding Using Constant Shift Embedding

Salim Jouili, Salvatore Tabbone

Graph Embedding Using Constant Shift Embedding[*]

Salim Jouili and Salvatore Tabbone

LORIA - INRIA Nancy Grand Est UMR 7503 - University of Nancy 2
BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France
`{salim.jouili,tabbone}@loria.fr`

Abstract. In the literature, although structural representations (e.g. graph) are more powerful than feature vectors in terms of representational abilities, many robust and efficient methods for classification (unsupervised and supervised) have been developed for feature vector representations. In this paper, we propose a graph embedding technique based on the constant shift embedding which transforms a graph to a real vector. This technique gives the abilities to perform the graph classification tasks by procedures based on feature vectors. Through a set of experiments we show that the proposed technique outperforms the classification in the original graph domain and the other graph embedding techniques.

Key words: Structural pattern recognition, graph embedding, graph classification

1   Introduction

In pattern recognition, object representations can be broadly divided into statistical and structural methods [1]. In the former, the object is represented by a feature vector, and in the latter, a data structure (e.g. graphs or trees) is used to describe the components and their relationships into the object. In the literature, many robust and efficient methods for classification (unsupervised and supervised), retrieval and other related tasks are henceforth available [5]. Most of these approaches are often limited to work with a statistical representation. Indeed, the use of feature vectors has numerous helpful properties. That is, since recognition is to be carried out based on a number of measurements of the objects, and each feature vector can be regarded as a point in an n-dimensional space, the measurement operations can be performed in simple way such that the Euclidean distance as the distance between objects. When a numerical feature vector is used to represent the object, all structural information is discarded. However a structural representation (e.g. graph) is more powerful than feature

---

vector in terms of representational abilities. The graph structure provides a flexible representation such that there is no fixed dimensionality for objects (unlike vectors), and provides an efficient representation such that an object is modeled by its components and the existing relations between them. In the last decades, many structural approaches have been proposed [3]. Nevertheless, dealing with graphs suffers, on the one hand from the high complexity of the graph matching problem which is a problem of computing distances between graphs, and on the other hand from the robustness to structural noise which is a problem related to the capability to cope with structural variation. These drawbacks have brought about a lack of suitable methods for classification. However, unlike for structural-based representation, a lot of robust classification approaches have been developed for the feature vector representation such as neural network, support vector machines, k-nearest neighbors, Gaussian mixture model, Gaussian, naive Bayes, decision tree and RBF classifiers [12]. In contrast, as remarked by Riesen et al. [2] the classification of graphs is limited to the use of the nearest-neighbor classifiers using one graph similarity measure. On that account, the community of pattern recognition speaks about a gap between structural and statistical approaches [1, 16].

Recently, a few works have been carried out concerning the bridging of this gap between structural and statistical approaches. Their aim is to delineate a mapping between graphs and real vectors, this task is called graph embedding. The embedding techniques were originally introduced for statistical approaches with the objective of constructing low dimensional feature-space embeddings of high-dimensional data sets [6, 9, 17, 22]. In the context of graph embedding, different procedures have been proposed in the literature. De Mauro et al. [4] propose a new method, based on recurrent neural network, to project graphs into vector space. Moreover, Hancock et al. [7, 27, 14, 21] use spectral theory to convert graphs into vectors by means of spectral decomposition into eigenvalues and eigenvectors of the adjacency (or Laplacian) matrix of a graph. Besides, a new category of graph embedding techniques was introduced by Bunke et al. [2, 20, 19], their method is based on the selection of some prototypes and the computation of the graph edit distance between the graph and the set of prototypes. This method was originally developed for the embedding of feature vectors in a dissimilarity space [16, 17]. This technique was also used to project string representations into vector spaces [25].

In this paper, we propose a new graph embedding technique by means of constant shift embedding. Originally, this idea was proposed in order to pairwise data into Euclidean vector spaces [22]. It was used to embed protein sequences into an n-dimensional feature vector space. In the current work, we generalize this method to the domain of graphs. Here, the key issue is to convert general dissimilarity data into metric data. The constant shift embedding increases all dissimilarities by an equal amount to produce a set of Euclidean distances. This set of distances can be realized as the pairwise distances among a set of points in an Euclidean space. In our method, we generalize this method by means of graph similarity measure [11, 10]. The experimental results have shown that the pro-

posed graph embedding technique improves the accuracy achieved in the graph domain by the nearest neighbor classifier. Furthermore, the results achieved by our method outperforms some alternative graph embedding techniques.

2 Graph similarity measure by means of node signatures

Before introducing the graph embedding approach, let us recall how the dissimilarity in the domain of graphs can be computed. Similarity or (dissimilarity) between two graphs is almost always refered as a graph matching problem. Graph matching is the process of finding a correspondence between nodes and edges of two graphs that satisfies some constraints ensuring that similar substructures in one graph are mapped to similar substructures in the other. Many approaches have been proposed to solve the graph matching problem. In this paper we use a recent technique proposed by Jouili et al. in [10, 11]. This approach is based on node signatures notion. In order to construct a signature for a node in an attributed graph, all available information into the graph and related to this node is used. The collection of these informations should be refined into an adequate structure which can provides distances between different node signatures. In this perspective, the node signature is defined as a set composed by four subsets which represent the node attribute, the node degree and the attributes of its adjacent edges and the degrees of the nodes on which these edges are connected. Given a graph G=(V,E,α,β), the node signature of $n_i \in V$ is defined as follows:

$$\gamma(n_i) = \left\{ \alpha_i, \ \theta(n_i), \ \{\theta(n_j)\}_{\forall ij \in E}, \ \{\beta_{ij}\}_{\forall ij \in E} \right\}$$

where

- α_i the attribute of the node n_i.
- $\theta(n_i)$ the degree of n_i.
- $\{\theta(n_j)\}_{\forall ij \in E}$ the degrees set of the nodes adjacent to n_i.
- $\{\beta_{ij}\}_{\forall ij \in E}$ the attributes set of the incident edges to n_i.

Then, to compute a distance between node signatures, the *Heterogeneous Euclidean Overlap Metric* (HEOM) is used. The HEOM uses the *overlap* metric for symbolic attributes and the normalized Euclidean distance for numeric attributes. Next the similarities between the graphs is computed: Firstly, a definition of the distance between two sets of node signatures is given. Subsequently, a matching distance between two graphs is defined based on the node signatures sets. Let S_γ be a collection of local descriptions, the set of node signatures S_γ of a graph g=(V,E,α,β) is defined as :

$$S_\gamma(\ g\ ) = \left\{ \gamma(n_i) \mid \forall n_i \in V \right\}$$

Let A=(V$_a$,E$_a$) and B=(V$_b$,E$_b$) be two graphs. And assume that $\phi : S_\gamma(A) \rightarrow S_\gamma(B)$ is a function. The distance d between A and B is given by φ which is the

distance between $S_\gamma(A)$ and $S_\gamma(B)$

$$d(A, B) = \varphi(S_\gamma(A), S_\gamma(B)) = \min_\phi \sum_{\gamma(n_i) \in S_\gamma(A)} d_{nd}(\gamma(n_i), \phi(\gamma(n_i)))$$

The calculation of the function $\varphi(S_\gamma(A), S_\gamma(B))$ is equivalent to solve an assignment problem, which is one of the fundamental combinatorial optimization problems. It consists of finding a maximum weight matching in a weighted bipartite graph. This assignment problem can be solved by the Hungarian method. The permutation matrix P, obtained by applying the Hungarian method to the cost matrix, defines the optimum matching between two given graphs.

3   Graph embedding via constant shift embedding

Embedding graph corresponds to finding points in a vector space, such that their mutual distance is as close as possible to the initial dissimilarity matrix with respect to some cost function. Embedding yields points in a vector space, thus making the graph available to numerous machine learning techniques which require vectorial input.

Let $G = \{g_1, ..., g_n\}$ be a set of graph and $d: G \times G \rightarrow \mathbb{R}$ a graph distance function between pairs of its elements and let $D = D_{ij} = d(g_i, g_j) \in \mathbb{R}^{n \times n}$ be an $n \times n$ dissimilarity matrix. Here, the aim is to yield n vectors x_i in a p-dimensional vector space such that the distance between x_i and x_j is close to the dissimilarity D_{ij} which is the distance between g_i and g_j.

3.1   Dissimilarity Matrix: from Graph domain to Euclidean space via Constant shift embedding

Before stating the main method, the notion of *centralized matrix* is reminded [22]. Let P be an $n \times n$ matrix and let I_n be the $n \times n$ identity matrix and $e_n = (1, ..., 1)^\mathsf{T}$. Let $Q_n = I_n - \frac{1}{n} e_n e_n^\mathsf{T}$. Q_n is the projection matrix onto the orthogonal complement of e_n. The *centralized matrix* P^c is given by:

$$P^c = Q_n P Q_n$$

Roth et al. [22] observed that the transition from the original matrix D (in our case: the graph domain) to a matrix $\tilde{D}$ which derives from a squared Euclidean distance, can be achieved by a off-diagonal shift operation without influencing the distribution of the initial data. Hence, $\tilde{D}$ is given by:

$$\tilde{D} = D + d_0(e_n e_n^\mathsf{T} - I_n), \text{ or similarly } (\tilde{D}_{ij} = D_{ij} + d_0, \forall i \neq j)$$

Where d_0 is constant. In addition, since D is a symmetric and zero-diagonal matrix, it can be decomposed [13] by means of a matrix S:

$$D_{ij} = S_{ii} + S_{jj} - 2S_{ij}$$

Obviously, S is not uniquely determined by D. All matrices $S + \alpha e_n e_n^\intercal$ yield the same D, $\forall \alpha \in \mathbb{R}$. However, it is proven that the centralized version of S is uniquely defined by the given matrix D (Lemma 1 in [22]):

$$S^c = -\frac{1}{2}D^c, \text{ with } D^c = Q_n D Q_n$$

From the following important theorem [26], we remark the particularity of the interesting matrix S^c.

Theorem 1. D is squared Euclidean distance matrix if and only if S^c is positive and semidefinite.

In other words, the pairwise similarities given by D can be embedded into an Euclidean vector space if and only if the associated matrix S^c is positive and semidefinite. As far as graph domain, S^c will be indefinite. We use the constant shift embedding [22] to cope this problem. Indeed, by shifting the diagonal elements of S^c it can be transformed into a positive semidefinite matrix $\tilde{S}$ (see Lemma 2. in [22]):

$$\tilde{S} = S^c - \lambda_n(S^c)I_n$$

where $\lambda_n(S^c)$ is the minimal eigenvalue of the matrix S^c. The diagonal shift of the matrix S^c transforms the dissimilarity matrix D in a matrix representing squared Euclidean distances. The resulting embedding of D is defined by:

$$\tilde{D}_{ij} = \tilde{S}_{ii} + \tilde{S}_{jj} - 2\tilde{S}_{ij} \Longleftrightarrow \tilde{D} = D - 2\lambda_n(S^c)(e_n e_n^\intercal - I_n)$$

Since every positive and semidefinite matrix can be thought as representing a dot product matrix, there exists a matrix X for which $\tilde{S}^c = XX^\intercal$. The rows of X are the resulting embedding vectors x_i, so each graph g_i has been embedded in a Euclidean space and is represented by x_i. Then, it can be concluded that the matrix $\tilde{D}$ contains the squared Euclidean distances between these vectors x_i. In the next section, we discuss the extension of this method to the graph embedding.

3.2   From graphs to vectors

In this section, an algorithm for constructing embedded vectors is described. This algorithm is inspired from the Principal Component Analysis (PCA) [23]. A pseudo-code description of the algorithm is given in Algorithm 1. From a given dissimilarity matrix D of a set of graphs $G=\{g_1, ..., g_n\}$, the algorithm returns a set of embedded vectors $X=\{x_1, \cdots, x_n\}$ such that x_i embed the graph g_i. Firstly, the squared Euclidean distances matrix $\tilde{D}$ is computed by means of the constant shift embedding (line 1). Next, the matrix $\tilde{S}^c$ is computed, and as stated above it can be calculated as $\tilde{S}^c = -\frac{1}{2}\tilde{D}^c$ (line 2). Since $\tilde{S}^c$ is positive and semidefinite matrix, thus, $\tilde{S}^c=XX^\intercal$. The rows of X are the resulting embedding vectors x_i and they will be recovered by means of an eigendecomposition (line 3). Let note that, due to the centralization, it exists at least one the eigenvalue $\lambda_i=0$, hence, $p \leq n-1$ (line 3-4). Finally by computing the $n \times p$ map matrix:

$X_p = V_p(\Lambda_p)^{1/2}$, the embedded vectors are given by the rows of $X = X_p$, in p-dimensional space.

However, in PCA it is known that small eigenvalues contain the noise. Therefore, the dimensionality p can reduced by choosing $t \leq p$ in line 4 of the algorithm. Consequently, a $n \times t$ map matrix $X_t = V_t(\Lambda_t)^{1/2}$ will be computed instead of X_p, where V_t is the column-matrix of the selected eigenvectors (the first t column vectors of V) and Λ_t the diagonal matrix of the corresponding eigenvectors (the top $t \times t$ sub-matrix of Λ). One can ask how to find the optimal t that yields the better performance of a classifier in the vector space. Indeed, the dimensionality t has a pronounced influence on the performance of the classifier. In this paper, the optimal t is chosen empirically. That means, the optimal t is the one which provides the better classification accuracy from 2 to p.

Algorithm 1 Construction of the embedded vectors

Require: Dissimilarity matrix D of the a set of graphs $G = \{g_1, ..., g_n\}$
Ensure: set of vectors $X = \{x_1, \cdots, x_n\}$ where x_i embed g_i
1: Compute the squared Euclidean distances matrix $\tilde{D}$
2: $\qquad\qquad\qquad\qquad\qquad\qquad \triangleright \tilde{D} = D - 2\lambda_n(S^c)(e_n e_n^\intercal - I_n)$
3: $\qquad\qquad\qquad\qquad\qquad\qquad \triangleright S^c = -\dfrac{1}{2}D^c$, with $D^c = Q_n D Q_n$
4: $\qquad\qquad \triangleright Q_n = I_n - \dfrac{1}{n}e_n e_n^\intercal$, with I_n the $n \times n$ identity matrix and $e_n = (1,...,1)^\intercal$.
5:
6: Compute $\tilde{S}^c = -\dfrac{1}{2}\tilde{D}^c$, where $\tilde{D}^c = Q\tilde{D}Q$.
7: Eigendecomposition of $\tilde{S}^c$, $\tilde{S}^c = V\Lambda V^\intercal$
 $\quad$ – $\Lambda = \text{diag}(\lambda_1, ...\lambda_n)$ is the diagonal matrix of the eigenvalues
 $\quad$ – $V = \{v_1, ...v_n\}$ is the orthonormal matrix of corresponding eigenvectors v_i.
 $\qquad\qquad\qquad\qquad \triangleright$ Note that, $\lambda_1 \geq ... \lambda_p \geq \lambda_{p+1} = 0 = ... = \lambda_n$
8: Compute the $n \times p$ map matrix: $X_p = V_p(\Lambda_p)^{1/2}$, with $V_p = \{v_1, ...v_p\}$ and $\Lambda_p = \text{diag}(\lambda_1, ...\lambda_p)$
9: **Output:** The rows of X_p contain the embedded vectors in p-dimensional space.

---

4 Experiments

4.1 Experimental setup

To perform the evaluation of the proposed algorithm, we used four data sets :

 – **GREC**: The GREC data set [18] consists of graphs representing symbols from architectural and electronic drawings. Here, the ending points (i.e. corners, intersections and circles) are represented by nodes which are connected by undirected edges and labeled as lines or arcs. The graph subset used in our experiments has 814 graphs, 24 classes and 37 graphs per class.

Algorithm 2 Construction of the embedded vectors

Require: Dissimilarity matrix D of the a set of graphs $G=\{g_1,...,g_n\}$
Ensure: set of vectors $X=\{x_1,\cdots,x_n\}$ where x_i embed g_i
 1: Compute the squared Euclidean distances matrix $\tilde{D}$
 2: Compute $\tilde{S}^c = -\dfrac{1}{2}\,\tilde{D}^c$, where $\tilde{D}^c = Q\tilde{D}Q$.
 3: Eigendecomposition of $\tilde{S}^c$, $\tilde{S}^c = V\Lambda V^{\mathsf{T}}$
    - $\Lambda=\mathrm{diag}(\lambda_1,...\lambda_n)$ is the diagonal matrix of the eigenvalues
    - $V=\{v_1,...v_n\}$ is the orthonormal matrix of corresponding eigenvectors v_i.
                    ▷ Note that, $\lambda_1 \geq ... \lambda_p \geq \lambda_{p+1} = 0 = ... = \lambda_n$
 4: Compute the $n \times p$ map matrix: $X_p=V_p(\Lambda_p)^{1/2}$, with $V_p=\{v_1,...v_p\}$ and $\Lambda_p=\mathrm{diag}(\lambda_1,...\lambda_p)$
 5: **Output**: The rows of X_p contain the embedded vectors in p-dimensional space.

- **Letter**: The Letter data set [18] consists of graphs representing distorted Letter drawings. This data set contains 15 capital letters (A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z), arbitrarily strong distortions are applied to each letter to obtain large sample sets of graphs. Here, the ending points are represented by nodes which are connected by undirected edges. The graph subset used in our experiments has 1500 graphs, 15 classes and 100 graphs per class.
- **COIL**: The COIL data set [15] consists of graphs representing different views of 3D objects in which two consecutive images in the same class represent the same object rotated by 5°. The images are converted into graphs by feature points extraction using the Harris interest points [8] and Delaunay triangulation. Each node is labeled with a two-dimensional attribute giving its position, while edges are unlabeled. The graph subset used in our experiments has 2400 graphs, 100 classes and 24 graphs per class.
- **Mutagenicity**: The Mutagenicity data set [18] consists of graphs representing molecular compounds, the nodes represent the atoms labeled with the corresponding chemical symbol and edges by valence of linkage. The graph subset used in our experiments has 1500 graphs, 2 classes and 750 graphs per class.

The experiments consist in applying our algorithm for each dataset. Our intention is to show that the proposed graph embedding technique is indeed able to yield embedded vectors that can improve classification results achieved with the original graph representation. We begin by computing the dissimilarities matrix of each data set by means of the graph matching introduced in [10, 11] and briefly reviewed in Section 2. Then, since the classification in the graph domain can be performed by only the k-NN classifiers, hence, it is used as reference system in the graph domain. Whereas in vector space the k-NN and the SVM[1] classifiers [24] are used to compare the embedding quality of the vectors resulting from our algorithm and the vectors resulting from the graph embedding approach

---

[1] We used the C-SVM classifier with linear kernel function available in weka software
   http://www.cs.waikato.ac.nz/ml/weka/

recently proposed by Bunke et al. [2, 19, 20]. This method was originally developed for the embedding of feature vectors in a dissimilarity space [16, 17] and is based on the selection of some prototypes and the computation of the graph edit distance between the graph and the set of prototypes. That is, let assume that $G=\{g_1, ..., g_n\}$ is a set of graphs and d: $G \times G \rightarrow \mathbb{R}$ is some graph dissimilarity measure. Let $PS=\{ps_1, ..., ps_p\}$ be a set of $p \leq n$ selected prototypes from G ($PS \subseteq G$). Now each graph $g \in G$ can be embedded into p-dimensional vector $(d_1, ..., d_p)$, where $d_1 = d(g, ps_1), ..., d_p = d(g, ps_p)$. As remarked in [20], the results of this method depends on the choice of p appropriate prototypes. In this paper, four different prototype selector are used, namely, the k-centers prototype selector (KCPS), the spanning prototype selector (SPS), the target-sphere prototype selector (TPS) and the random prototype selector (RandPS). We refer the reader to [20] for a definition of these selectors. A second key issue of this method is the number p of prototypes that must be selected to obtain the better classification performance. To overcome this problem, we define the optimal p by the same procedure that is applied to determine the optimal t of our algorithm (cf. section 3.2).

4.2 Results

Data-set	Graph domain	Vector space				
			Bunke with			
		Our Embedding	KCPS	SPS	TPS	RandPS
GREC	98.11%	**99.50**%	97.17%	96.19%	97.78%	97.54%
Letter	79.33%	91.33%	92.4%	**92.66**%	92.46%	92.4%
COIL	50.30%	**52.35**%	48.64%	49.1%	48.6%	48.64%
Mutagenicity	63.70%	63.8%	60.93%	63.73%	64.13%	**64.73**%

Table 1. K-NN classifier results

The first experiment consists of applying the k-NN classifier in the graph domain and in the embedded vector space. In Table 1 the classification accuracy rates for k-NN classifier are given for all data sets. One can remark that the results achieved in the vector space improve the results achieved in the graph domain for all data sets. However, this improvement is not statistically significant since it do not exceed $\tilde{2}\%$ for almost all data sets, except for the Letter data set. In the embedded vector space, the performances of our method and the Bunke's methods are quite similar. This first experiment is essentially to show that the embedding techniques are accurate since using the same classifier we obtain rates better then the graph domain. However, in the vector space we are not limited to a simple nearest neighbor classifier. So, in Table 2 the classification accuracy rates for the k-NN classifier in the graph domain (in Table 1) and the SVM classifier in the embedded vector space are given for all data sets. We can remark that our embedding methods clearly improves the accuracy achieved in

the graph domain. Regarding the GREC data set, the best classification accuracy is achieved by our method and the Bunke's method with the KCPS selector. The results improved the k-NN classifier in the graph domain by 1.76%. This improvement has no statistical signification. This is due to the fact that the classification in the graph domain of the GREC data set provides already a very good accuracy 98.11%. For the Letter data set, all the embedding methods clearly improve the accuracy achieved in the graph domain. Regarding the COIL data set, it can be remarked that the accuracy achieved in the graph domain is quite low (50.30%). This accuracy is highly improved by the SVM classifiers in the different vector spaces (by 25.43% using our embedding technique). Finally, the results concerning the Mutagenicity data set show that all the variants of the Bunke's embedding fail to improve the accuracy achieved in the graph domain and provide the worst results. Whereas our graph embedding using constant shift embedding results improve the results of the k-NN classifier achieved in the original graph representation (by 7.5%). Therefore, our method provides more significant embedded vector for the graphs in the Mutagenicity data set.

Data-set	Graph domain	Vector space			
			Bunke with		
		Our Embedding	KCPS	SPS	TPS	RandPS
GREC	-	**99.87**%	**99.87**%	99.50% 99.75% 99.75%
Letter	-	**94.87**%	91.60%	91.40% 91.53% 91.53%
COIL	-	**75.73**%	73.29%	73,10% 73.25% 73.23%
Mutagenicity	-	**71.20**%	57.53%	57.31% 57.48% 57.45%

Table 2. SVM Classifier results

To summarize, with the graph embedding algorithm proposed in this paper and the SVM classifiers, the classification accuracy rates improve the results achieved in the original graph domain for all data sets used in the experiments. This agrees with our intention to improve classification results achieved in graph domain by means of graph embedding technique. Furthermore, the comparison with the four variants of the Bunke's graph embedding has shown that the proposed technique outperforms these alternatives for almost the data sets.

5 Conclusions

In the context of graph-based representation for pattern recognition, there is a lack of suitable methods for classification. However almost the huge part of robust classification approaches has been developed for feature vector representations. Indeed, the classification of graphs is limited to the use of the nearest-neighbor classifiers using one graph similarity measure. In this paper, we proposed a new graph embedding technique based on the constant shift embedding which was originally developed for vector spaces. The constant shift embedding increases all dissimilarities by an equal amount to produce a set of Euclidean distances.

This set of distances can be realized as the pairwise distances among a set of points in an Euclidean space. Here, the main idea was to generalize this technique in the domain of graphs. In the experiments, the application of the SVM classifiers on the resulting embedded vectors has shown a significantly statistically improvement.

References

1. H. Bunke, S. Günter, and X. Jiang. Towards bridging the gap between statistical and structural pattern recognition: Two new concepts in graph matching. In *ICAPR, LNCS 2013*, pages 1–11, 2001.
2. H. Bunke and K. Riesen. Graph classification based on dissimilarity space embedding. In *IAPR International Workshop, SSPR & SPR 2008, LNCS 5342*, pages 996–1007, 2008.
3. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *IJPRAI*, 18(3):265–298, 2004.
4. C. de Mauro, M. Diligenti, M. Gori, and M. Maggini. Similarity learning for graph-based image representations. *PRL*, 24(8):1115 – 1122, 2003.
5. J. Marques de sa. *Pattern Recognition : Concepts, Methods and Applications*. Springer-Verlag, 2001.
6. I. S. Dhillon, D. S. Modha, and W. S. Spangler. Class visualization of high-dimensional data with applications. *Computational Statistics & Data Analysis*, 41(1):59–90, 2002.
7. D. Emms, R. C. Wilson, and E. R. Hancock. Graph embedding using quantum commute times. In *IAPR International Workshop GbRPR, LNCS 4538*.
8. C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.
9. T. Iwata, K. Saito, N. Ueda, S. Stromsten, T. L. Griffiths, and J. B. Tenenbaum. Parametric embedding for class visualization. In *NIPS*, 2004.
10. S. Jouili, I. Mili, and S. Tabbone. Attributed graph matching using local descriptions. In J. Blanc-Talon, W. Philips, D. C. Popescu, and P. Scheunders, editors, *ACIVS*, volume 5807 of *LNCS*, pages 89–99. Springer, 2009.
11. S. Jouili and S. Tabbone. Graph matching based on node signatures. In A. Torsello, F. Escolano, and L. Brun, editors, *GbRPR*, volume 5534 of *LNCS*, pages 154–163. Springer, 2009.
12. S. B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica*, 31(3):249–268, 2007.
13. J. Laub and K. R. Müller. Feature discovery in non-metric pairwise data. *Journal of Machine Learning Research*, 5:801–818, 2004.
14. B. Luo, R. C. Wilson, and E. R. Hancock. Spectral embedding of graphs. *Pattern Recognition*, 36(10):2213–2230, 2003.
15. S.A. Nene, S.K. Nayar, and H. Murase. Columbia object image library (coil-100). Technical report, Department of Computer Science, Columbia University, 1996.
16. E. Pekalska and Robert P. W. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications*. World Scientific Publishing, 2005.
17. E. Pekalska, Robert P. W. Duin, and Pavel Paclík. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, 39(2):189–208, 2006.
18. K. Riesen and H. Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *IAPR International Workshop, SSPR & SPR 2008, LNCS 5342*, pages 287–297, 2008.

19. K. Riesen and H. Bunke. Graph classification based on vector space embedding. *IJPRAI*, 23(6):1053–1081, 2009.
20. K. Riesen, M. Neuhaus, and H. Bunke. Graph embedding in vector spaces by means of prototype selection. In *IAPR International Workshop GbRPR, LNCS 4538*.
21. A. Robles-Kelly and E. R. Hancock. A riemannian approach to graph embedding. *Pattern Recognition*, 40(3):1042–1056, 2007.
22. V. Roth, J. Laub, M. Kawanabe, and J.M. Buhmann. Optimal cluster preserving embedding of nonmetric proximity data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(12):1540–1551, Dec. 2003.
23. B. Schölkopf, A. Smola, and K. R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.
24. J. Shawe-Taylor and N. Cristianini. *Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.
25. B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, and R. P. W. Duin. Transforming strings to vector spaces using prototype selection. In *IAPR International Workshop, SSPR & SPR 2006, LNCS 4109*, pages 287–296, 2006.
26. W. S. Torgerson. *Theory and Methods of Scaling*. Johm Wiley and Sons, 1958.
27. A. Torsello and E. R. Hancock. Graph embedding using tree edit-union. *Pattern Recognition*, 40(5):1393–1405, 2007.