Prototype Selection for Graph Embedding Using Instance Selection

Magdiel Jiménez-Guarneros(✉), Jesús Ariel Carrasco-Ochoa, and José Fco. Martínez-Trinidad

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Luis Enrique Erro No. 1, Sta. María Tonantzintla, 72840 San Andrés Cholula, Puebla, Mexico
{magdiel.jg,ariel,fmartine}@inaoep.mx

Abstract. Currently, graph embedding has taken a great interest in the area of structural pattern recognition, especially techniques based on representation via dissimilarity. However, one of the main problems of this technique is the selection of a suitable set of prototype graphs that better describes the whole set of graphs. In this paper, we evaluate the use of an instance selection method based on clustering for graph embedding, which selects border prototypes and some non-border prototypes. An experimental evaluation shows that the selected method gets competitive accuracy and better runtimes than other state of the art methods.

Keywords: Structural pattern recognition · Graph embedding · Prototype selection · Graph classification

1 Introduction

Selecting a suitable representation for different types of objects is very important in pattern recognition [1]. There are two approaches that have been studied: the statistical and the structural. In the statistical approach [2], objects are represented as feature vectors, being its main advantage the wide collection of algorithms developed for its treatment. However, this type of representation involves constraints such as the use of a certain number of characteristics or the lack of usefulness in the representation of the components of an object and the relationships that exist between them. Moreover into the structural approach, especially with representations based on graphs [3], it is possible to overcome these major statistical approach's drawbacks. As a result, there is a growing interest in the use of this type of representation for different applications, such as: document analysis [1], image segmentation [4], medical diagnosis [5], face recognition [6], fingerprint classification [7], among others. Nevertheless, the main disadvantage of the graph-based representation is the high complexity of the algorithms for their treatment.

In the literature [8–12], graph embedding emerges as a solution to deal with the major problems of the structural approach, especially for graph-based representations. The key idea of this technique is transforming a graph into an

n-dimensional feature vector, making possible the use of those methods and techniques developed for the statistical approach [10]. Although different solutions for graph embedding have been proposed in the literature, this paper follows the approach initially proposed by Riesen and Bunke in [8–11], called graph embedding via dissimilarity. This technique transforms a input graph g into an n-dimensional feature vector by means of a set of prototype graphs P and a dissimilarity measure. This feature vector is built by computing the dissimilarities between g and the n prototypes in P. However, one of the major drawbacks of using this technique is the search for a suitable set of prototypes, since the number of prototypes in P determines the dimensionality of the feature vectors. A small number of prototypes is desirable because it will lead to lower computational cost.

In [8–12], several approaches for prototype selection for graph embedding are proposed. In works as [8,10] for example, the authors propose different prototype selection methods by means of two selection strategies, the first one [8] is carried out on the whole graph dataset regardless of the class each selected prototype belongs to; while in the second [10], the prototypes are selected separately by class. The second strategy provides better classification results than the first one. Moreover in [11], the authors propose extensions of the prototype selection methods proposed in [10]. The key idea of their proposal involves the discrimination between classes through a weight assigned to the intra-class compactness and the inter-class separation in a prototype selection. The methods presented in [11] show better classification results in contrast to the methods reported in [10]. On the other hand, the method proposed in [12] tries to optimize the dimensionality of the feature vectors; such as some reduction methods studied in [9]. The method proposed in [12] uses a genetic algorithm to perform prototype selection by means of two operations called compression and expansion. In this paper, we propose the use of an instances selection method based on clustering for prototype selection for graph embedding via dissimilarity.

The rest of the paper is organized as follows: Sect. 2 presents a brief description of the technique of graph embedding via dissimilarity. Section 3 describes the instance selection method used in our proposal. Experimental results are presented in Sect. 4. Finally, Sect. 5 presents some conclusions and future work.

2 Graph Embedding via Dissimilarity

Graph embedding consists in transforming a graph into a feature vector of suitable dimensionality [10]. The main motivation of this approach is to overcome the lack of algorithmic tools in the graph domain and taking advantage the existing statistical pattern recognition techniques, preserving the representational power the graphs provide.

Given a labeled set of sample graphs $T = \{g_1, \ldots, g_m\}$ from some graph domain G, and a dissimilarity measure $d(g_i, g_j)$. Graph embedding consists in selecting a set $P = \{p_1, \ldots, p_n\} \subseteq T$ with $n < m$ prototype graphs (see Sect. 3), and by computing the dissimilarity measure of a input graph g with each of the

n prototypes in P an n-dimensional feature vector $(d(g, p_1), ..., d(g, p_n))$ that represents the graph g is built. Based on this technique a set of graphs can be transformed into n-dimensional feature vectors allowing the use of the different techniques of the statistical approach to pattern recognition.

Definition 1 (Graph Embedding). Given a graph domain G, a dissimilarity function $d : G \times G \longrightarrow \mathbb{R}$, and $P = \{p_1, ..., p_n\} \subseteq G$ a prototype graph set. The mapping $\varphi_n^P : G \rightarrow \mathbb{R}^n$ is defined as:

$$\varphi_n^P(g) = (d(g, p_1), ..., d(g, p_n))$$

In this work, a dissimilarity measure based on the Graph Edit Distance (GED) was used, which defines the dissimilarity, or distance between two graphs as the minimum number of edit operations (insertion, deletion and substitution of nodes or edges) necessary to transform a graph into another [13]. A variety of algorithms have been proposed in the literature to calculate the edit distance between two graphs [13–18], in this work we use the approach based on bipartite graphs; initially presented in [15], improved in [16] and proposed as a toolkit in [17]. Although other improvements have been introduced for this approach [18], not all improvements have good performance for some application domains.

3 Prototype Selection

The graph embedding method described in Sect. 2 strongly depends on the selected prototypes, since they determine the quality of the embedding, and the number of prototypes determines the dimensionality of the feature vectors. A good prototype selection is very important because it should include as much descriptive information of the original graph dataset as possible [10]; furthermore, a small number of prototypes is desirable. The selected prototypes should avoid redundancy or even noise from the set of graphs.

In the literature, different prototype selection methods have been proposed. Strategies as those presented in [8,10,11] show good results, the most remarkable are those that consider those discrimination between classes; however, its main characteristic is that the number of prototypes is determined by the user. Other strategies optimize the size of the prototype set over a validation set [9,12]. Although these strategies infer automatically the number of prototypes to select, their selection cost is high in comparison with those proposed in [8,10,11]. On the other hand, to the best of our knowledge, strategies based on instance selection have not been previously used for graph embedding. Therefore, in this work, we propose and empirically evaluate the use of an instance selection method for prototype selection for graph embedding.

The following section describes the instance selection method that we used in this paper. This method, based on clustering, aims to retain prototypes that are located in the decision border and some others in the interior of each class (non-border prototypes).

3.1 Prototype Selection by Clustering (PSC)

The Prototype Selection by Clustering (PSC) method [19] was proposed for n-dimensional vector spaces, and among the different selection methods shown in the literature, PSC is one of the most recent; which has a fast runtime compared with other instances selection methods [19]. Thus, we adapted PSC to be used for prototype selection by using the dissimilarity measure $d(g_i, g_j)$ proposed in [17].

The main idea of PSC is to retain border prototypes and some non-border prototypes. For this, PSC initially divides the graph set T in small regions to find border and non-border prototypes into these regions, instead of finding them over the whole set T. In [19], originally *K-Means* was used to create these regions, however, in this work, we use the *K-Medoids* algorithm [20]. This clustering algorithm works similar to *K-Means*, but the major difference is on how the cluster centers are computed in each iteration. In this case, the median determines the center of each cluster A_j, and it is computed as follows:

$$center(A_j) = arg \min_{g_1 \in A_j} \sum_{g_2 \in A_j} d(g_1, g_2) \qquad (1)$$

PSC creates C clusters from a graph set T using *K-Medoids*. Subsequently, for each cluster A_j, it is necessary to determine if it is homogeneous or not. A pseudo-code description of this algorithm can be seen in Algorithm 1.

Algorithm 1. $PSC(T, C)$

```
Input: $T = (g_1, \ldots, g_m)$: A graph set; $C$: Number of clusters
Output: $P = (p_1, \ldots, p_n)$: Prototype set
1  $P \leftarrow \emptyset$;
2  $Clusters \leftarrow K\text{-}Medoids(T, C)$;
3  foreach cluster $A_j$ in $Clusters$ do
4      if $A_j$ is homogeneous then
5          $c \leftarrow$ the center of $A_j$;
6          $P \leftarrow P \cup \{c\}$;
7      else
8          $C_M \leftarrow$ the mayority class in $A_j$;
9          foreach class $C_K$ in $A_j$, where $C_K \neq C_M$ do
10             foreach prototype $p_i$ belonging to class $C_K$ do
11                 Let $p_c \in C_M$ be the nearest prototype $p_i$;
12                 $P \leftarrow P \cup \{p_c\}$;
13                 Let $p_M \in C_K$ be the nearest prototype to $p_c$;
14                 $P \leftarrow P \cup \{p_M\}$;
```

A cluster A_j is considered homogeneous if all the prototypes in the cluster belong to the same class. Therefore, these prototypes are non-border and they can be represented by the center of the cluster, discarding all other prototypes in the cluster (steps 4–6).

A cluster A_j is non-homogeneous if it has prototypes from two or more classes. PSC analyses the non-homogeneous clusters in order to find prototypes located in the border of the classes. For this purpose, PSC finds the most frequent class C_M (majority class) in each non-homogeneous cluster A_j. Then, the border prototypes in C_M are the most similar prototypes in A_j that belong to different class C_K (see Algorithm 1, steps 10–12). The border prototypes for C_K are obtained in the same way (steps 13–14).

As result, PSC obtains the most representative prototype for each homogeneous cluster (centers) and the border prototypes from non-homogeneous clusters. The number of clusters C to build in PSC, will be discussed in the next section.

4 Experimental Results

This section shows an experimental evaluation on 10 databases taken from IAM repository [21]. Table 1 describes the main features of each database divided into training, validation, and test, sets. The main purpose of these experimental tests is to evaluate the prototype selection obtained through PSC for graph embedding. In our experiments, we include an assessment in terms of accuracy, number of selected prototypes and runtime.

For each database, the training set is used to build and train the classifier, while the validation set was used to determine the appropriate number of prototypes, and the test set was used for evaluation. In order to carry out the classification task we use Support Vector Machines (SVM) with a Radial Basis

Table 1. Summary of characteristics of the graphs datasets used in our experiments. The size of training (Tr), validation (Va), testing (Te), number of classes (#Cls), average (Avg.) and maximum (Max.) of nodes and edges, as well as whether or not the classes are balanced (Bal.), are shown.

Dataset	Size (Tr,Va,Te)	#Cls.	Nodes (Avg.)	Edges (Avg.)	Nodes (Max.)	Edges (Max.)	Bal.
Letter low	750,750,750	15	4.7	3.1	8	6	Yes
Letter medium	750,750,750	15	4.7	3.2	9	7	Yes
Letter high	750,750,750	15	4.7	4.5	9	9	Yes
GREC	286,286,528	22	11.5	12.2	25	30	Yes
Fingerprints	500,300,2000	4	5.4	4.4	26	24	No
Protein	200,200,200	6	32.6	62.1	126	149	Yes
AIDS	250,250,1500	2	15.7	16.2	95	103	No
Mutagenicity	1500,500,1500	2	30.3	30.8	417	112	Yes
Coil-RAG	2400,500,1000	100	3	3	11	13	Yes
Coil-DEL	2400,500,1000	100	21.5	54.2	77	222	Yes

Function (RBF) kernel, provided by the WEKA Toolkit [22]. This classifier was chosen since it has been used in recent works as [11], and it has had a remarkable performance in n-dimensional vector spaces.

4.1 Baseline Methods

The graph embedding strategy proposed by Riesen and Bunke [8–11] can work without making a previous selection of prototypes. However, as it was mentioned previously, the dimensionality of the vectors in the embedding depends on the number of prototypes. So a small set of prototypes is desirable since it will lead to lower computational cost. For us, it is important to make a comparison between the prototypes selected by PSC and using all prototypes in the dataset, being this our first baseline method.

Our second baseline method is the Spanning Prototype Selection Discriminative (SPS-D) proposed in [11]. This method was chosen because it shows very good results, and is one of the most recent methods reported in the literature. The main idea of SPS-D consists in selecting those prototypes that are located in the center of each class. Subsequently, SPS-D selects the farthest prototype to the prototypes already selected, trying to cover the whole training set. This method, considers discrimination between classes by assigning a weight of compactness intra-class and separation inter-class. For our experiments, the weights assigned to each dataset were fixed as in [11].

4.2 Experimental Comparison

In order to compare PSC and SPS-D, we carry out an experiment by using different percentages of retention. A number of prototypes per class was designed for SPS-D, according to retention levels from 5 to 95 % (see Fig. 1). The retention percentage in PSC is of interest, because this method uses the number of clusters C as its unique parameter. In [19], the authors fixed for C an even number of clusters per class for instance selection. However, in our experiment C was set trying to get a number of prototypes similar to the number of prototypes used by SPS-D for each retention level.

Table 2 shows the classification results after generating the graph embedding using all the prototypes of the training set, as well as the prototypes selected by SPS-D and PSC. This table shows the accuracy on the test set, the number of selected prototypes, and the retention percentage for each method. The best accuracy results appear boldfaced.

The experimental results show that PSC gets better average accuracy than SPS-D, we can observe that PSC's average accuracy was not better than the one obtained by using all graphs of the training set. However, a Friedman test [23] on the classification results with a significance level $\alpha = 0.5$ and 2 degrees of freedom distributed according to chi-square, shows that there is not a statistical significant difference between PSC's results and the two baseline methods. It is important to comment that these results were performed with the same parameter values for the SVM classifier, we did not perform a custom configuration for each database as it was the made in previous works [10,11].

Table 2. Accuracy of SVM applied over the graph embedding using all graphs of the training set, and the prototypes selected by SPS-D and PSC. Acc.: Accuracy, Prots.: number of prototypes, Ret.: retention percentage.

	All Graphs			SPS-D			PSC		
Dataset	Acc.	Prots.	Ret.	Acc.	Prots.	Ret.	Acc.	Prots.	Ret.
Letter low	**99.20**	750	100.00	99.07	226	30.13	**99.20**	272	36.27
Letter Medium	94.67	750	100.00	**94.80**	706	94.13	94.40	152	20.27
Letter High	**92.80**	750	100.00	92.40	301	40.13	91.60	302	40.27
GREC	**98.30**	286	100.00	98.11	67	23.43	**98.30**	156	54.55
Fingerprint	80.45	500	100.00	80.60	299	59.80	**81.05**	286	57.20
Protein	65.00	200	100.00	61.50	189	94.50	**66.00**	194	97.00
AIDS	99.27	250	100.00	**99.40**	26	10.40	**99.40**	50	20.00
Mutagenicity	**66.54**	1500	100.00	66.37	1125	75.00	65.17	484	32.27
Coil-RAG	**94.80**	2400	100.00	94.70	2201	91.71	94.70	2202	91.75
Coil-DEL	95.00	2400	100.00	**95.20**	1401	58.38	**95.20**	1802	75.08
Average	**88.60**		100.00	88.21		57.76	88.50		52.46

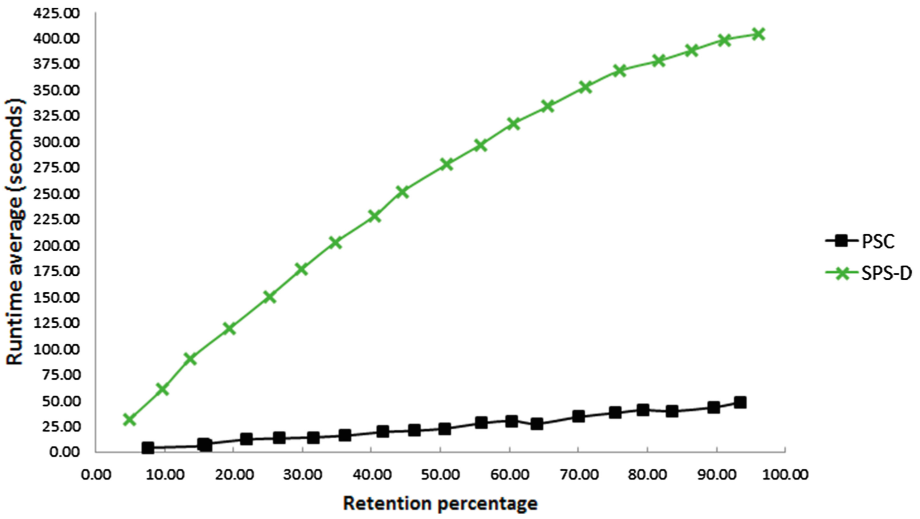

Fig. 1. Average runtime spent by the PSC and SPS-D for prototype selection on the 10 graphs datasets.

The optimal number of prototypes used to perform the graph embedding is also of interest. The results show that PSC get competitive classification results with a lower retention. In the last row of Table 2, we can see that average retention obtained by PSC is lower compared with the one of SPS-D. These findings are important because the use of a small number of prototypes implies a lower computational cost in the classification stage in many real applications.

A second experiment is related to the runtime of SPS-D and PSC, the results are shown in Fig. 1. This experiment shows the average runtime on the ten datasets regarding the percentage of selected prototypes. The runtimes that appear in Fig. 1 do not include the time for computing the distance matrix involved in the graph embedding, since both methods use this matrix.

In Fig. 1, we can observe that PSC is faster than SPS-D. We can also notice that the percentage of retention of PSC does not coincide with that of SPS-D because the number of clusters for PSC was assigned trying to select the same number of prototypes as SPS-D, but it was not always possible. Finally, we can see that PSC obtains very good runtimes with competitive classification results.

5 Conclusions

In graph embedding, especially in techniques based on dissimilarities, the search for a suitable set of prototypes proves to be an important task. A small number of prototypes will reduce the dimensionality of the feature vectors and therefore leads to lower computing costs in real world applications.

The main contribution of our work is to show that instance selection methods are useful for selecting prototypes for graph embedding. In this paper, we have evaluated the PSC instance selection method for the selection of prototypes for graph embedding via dissimilarity. PSC selects graphs located on the border of the class, with the aim of preserving the discrimination, and some prototypes that are non-borders. The experimental results show that the use of PSC for selecting prototypes for graph embedding allows obtaining competitive classification results compared to SPS-D, a state of the art prototype selection method for graph embedding, but requiring less runtime.

As it was described in this paper, an instance selection method was evaluated for selecting prototypes for the graph embedding via dissimilarity, however, given the promising results, as future, we will conduct a full study including other instance selection methods.

Acknowledgment. This work was partly supported by the National Council of Science and Technology of Mexico (CONACyT) through the project grant CB2008-106366; and the scholarship grant 298513.

References

1. Bunke, H., Riesen, K.: Recent advances in graph-based pattern recognition with applications in document analysis. Pattern Recogn. **44**(5), 1057–1067 (2011)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Information Science and Statistics. Springer, New York (2006)
3. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recognit. Artif. Intell. **18**(3), 265–298 (2004)
4. Harchaoui, Z., Bach, F.: Image classification with segmentation graph kernels. IEEE Computer Society (2007)

5. Sharma, H., Alekseychuk, A., Leskovsky, P., Hellwich, O., Anand, R., Zerbe, N., Hufnagl, P.: Determining similarity in histological images using graph-theoretic description and matching methods for content-based image retrieval in medical diagnostics. Diagn. Pathol. **7**(1), 134 (2012)
6. Han, P.Y., San, H.F., Yin, O.S.: Face recognition using a kernelization of graph embedding. World Acad. Sci. Eng. Technol. **6**(2), 460–464 (2012)
7. Marcialis, G., Roli, F., Serrau, A.: Graph-based and structural methods for fingerprint classification. In: Kandel, A., Bunke, H., Last, M. (eds.) Applied Graph Theory in Computer Vision and Pattern Recognition. SCI, vol. 52, pp. 205–226. Springer, Heidelberg (2007)
8. Riesen, K., Neuhaus, M., Bunke, H.: Graph embedding in vector spaces by means of prototype selection. In: Escolano, F., Vento, M. (eds.) GbRPR. LNCS, vol. 4538, pp. 383–393. Springer, Heidelberg (2007)
9. Bunke, H., Riesen, K.: Graph classification based on dissimilarity space embedding. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Loog, M. (eds.) SSPR & SPR 2008. LNCS, vol. 5342, pp. 996–1007. Springer, Heidelberg (2008)
10. Riesen, K., Bunke, H.: Graph classification based on vector space embedding. Int. J. Pattern Recognit. Artif. Intell. **23**(06), 1053–1081 (2009)
11. Borzeshi, E.Z., Piccardi, M., Riesen, K., Bunke, H.: Discriminative prototype selection methods for graph embedding. Pattern Recogn. **46**(6), 1648–1657 (2013)
12. Livi, L., Rizzi, A., Sadeghian, A.: Optimized dissimilarity space embedding for labeled graphs. Inf. Sci. **266**, 47–64 (2014)
13. Gao, X., Xiao, B., Tao, D., Li, X.: A survey of graph edit distance. Pattern Anal. Appl. **13**(1), 113–129 (2010)
14. Vento, M.: A one hour trip in the world of graphs, looking at the papers of the last ten years. In: Kropatsch, W.G., Artner, N.M., Haxhimusa, Y., Jiang, X. (eds.) GbRPR 2013. LNCS, vol. 7877, pp. 1–10. Springer, Heidelberg (2013)
15. Riesen, K., Bunke, H.: Approximate graph edit distance computation by means of bipartite graph matching. Image Vis. Comput. **27**(7), 950–959 (2009)
16. Fankhauser, S., Riesen, K., Bunke, H.: Speeding up graph edit distance computation through fast bipartite matching. In: Jiang, X., Ferrer, M., Torsello, A. (eds.) GbRPR 2011. LNCS, vol. 6658, pp. 102–111. Springer, Heidelberg (2011)
17. Riesen, K., Emmenegger, S., Bunke, H.: A novel software toolkit for graph edit distance computation. In: Kropatsch, W.G., Artner, N.M., Haxhimusa, Y., Jiang, X. (eds.) GbRPR 2013. LNCS, vol. 7877, pp. 142–151. Springer, Heidelberg (2013)
18. Riesen, K., Bunke, H.: Improving bipartite graph edit distance approximation using various search strategies. Pattern Recogn. **48**(4), 1349–1363 (2015)
19. Olvera-Lpez, J., Carrasco-Ochoa, J., Martnez-Trinidad, J.: A new fast prototype selection method based on clustering. Pattern Anal. Appl. **13**(2), 131–141 (2010)
20. Kaufman, L., Rousseeuw, P.: Clustering by Means of Medoids. Reports of the Faculty of Mathematics and Informatics, Faculty of Mathematics and Informatics (1987)
21. Riesen, K., Bunke, H.: Iam graph database repository for graph based pattern recognition and machine learning. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Georgiopoulos, M., Anagnostopoulos, G.C., Kwok, J.T., Loog, M. (eds.) SSPR & SPR 2008. LNCS, vol. 5342, pp. 287–297. Springer, Heidelberg (2008)
22. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
23. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)