Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Pattern Recognition Letters 24 (2003) 1107–1113

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Validation indices for graph clustering

Simon Günter, Horst Bunke *

Department of Computer Science, Inst. Fur Informatik und angewandte Mathematik, University of Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland

Abstract

In this paper, a new clustering algorithm for the domain of graphs is introduced. Also a number of cluster validation indices are reviewed. These indices aim at finding the optimal number of clusters automatically. The suitability of the considered indices is experimentally evaluated.

Keywords: Structural pattern recognition; Graphs; Clustering; Graph clustering; Cluster validation indices; Self organizing map

1. Introduction

Clustering is the process of dividing a set of given patterns into groups, or clusters, such that all patterns in the same group are similar to each other, while patterns from different groups are dissimilar. Clustering problems are ubiquitous in pattern recognition. For a survey on the most important clustering methods used in pattern recognition see, for example Dubes (1993). However, almost all clustering algorithms published in the literature are based on pattern representations in terms of feature vectors. A mapping from the domain of graphs to feature vectors by means of a neural net and a subsequent clustering procedure in the feature space, which is based on similarity constraints, has been proposed in De Mauro et al. (2001). But there are only a few works where symbolic data structures, in particular graphs, have been directly used in clustering (Englert and Glantz, 2000; Seong et al., 1993; Riano and Serratosa, 2000; Lus et al., 2001). This lack of clustering algorithms working on symbolic data structures is an unfortunate restriction, because symbolic data structures have a higher representational power than feature vectors.

Many clustering algorithms require the number of clusters being given beforehand. To overcome this problem, various cluster validation indices have been proposed. These indices allow to measure the quality of a clustering. Hence, a clustering algorithm can be executed several times, with a different number of clusters in each run, and the clustering that optimizes the considered index is selected as the final result. Another alternative is to change the number of clusters dynamically during the execution of the clustering algorithm. For example, in the Rival Penalized Competitive

* Corresponding author. Tel.: +41-31-631-4451; fax: +41-31-631-3965.
E-mail addresses: sguenter@iam.unibe.ch (S. Günter), bunke@iam.unibe.ch (H. Bunke).

0167-8655/03/$ - see front matter
PII: S0167-8655(02)00257-X

Learning Scheme, clustering is started with a number of clusters larger than the expected value and the location of cluster centers is changed in a competitive way (Xu et al., 1993). Whenever a cluster center represents an empty set of input patterns it is deleted. Thus the number of clusters can be dynamically changed. Similar strategies have been discussed in Fritzke (1999). Notice, however, that all of these techniques were exclusively developed for feature vector representations. Another approach to dealing with an unknown number of clusters has been proposed in Lus et al. (2001). Here the clustering task is formulated as a maximum likelihood estimation problem, which is solved by means of the EM-algorithm. This method can be applied without prior knowledge of the optimal number of clusters.

In the present paper two issues will be addressed. First, a graph clustering algorithm that was recently introduced by the authors (Günter, 2000; Günter and Bunke, 2002) will be reviewed. Secondly, a number of cluster validation indices will be proposed to evaluate the quality of graph clusterings. It will be shown through experiments that these indices are suitable to find the optimal number of clusters automatically.

The paper is organized as follows. In Section 2 the new graph clustering algorithm will be described. The following section reviews a set of cluster validation indices. Experimental results will be presented in Section 4, and conclusions drawn in Section 5.

2. A new graph clustering algorithm

For all issues related with graph matching, in particular with graph edit distance, the reader is referred to our previous work (Messmer and Bunke, 1998).

In Günter (2000) and Günter and Bunke (2002) a new graph clustering algorithms is introduced. This algorithm will be briefly reviewed in the present section. It is derived from self-organizing map (som), as described in Kohonen (1997). While the 'classical' som-algorithm is based on vectorial pattern representations, the new clustering algorithm works on graphs. Hence it can be considered a generalization that includes the original som as a special case.

som-algorithm

(1) **input:** a set of patterns, $X = \{x_1, \ldots, x_N\}$
(2) **output:** a set of prototypes, $Y = \{y_1, \ldots, y_M\}$
(3) **begin**
(4) initialize $Y = \{y_1, \ldots, y_M\}$ randomly
(5) **repeat** select $x \in X$ randomly
(7) find y^* such that $d(x, y^*) = \min\{d(x, y) | y \in Y\}$
(8) **for** all $y \in N(y^*)$ **do**
(9) $y = y + \alpha(x - y)$
(10) reduce learning rate α
(11) **until** termination condition is true
(12) **end**

Fig. 1. The som-algorithm.

A pseudo-code description of the classical som-algorithm is given in Fig. 1. The algorithm can serve two purposes, either clustering or mapping a high-dimensional pattern space to a lower-dimensional one. In the present paper we focus on its application to clustering. Given a set of patterns, X, the algorithm returns a prototype y_i for each cluster i. The prototypes are sometimes called *neurons*. The number of clusters, M, is a parameter that must be provided a priori. In the algorithm, first each prototype y_i is randomly initialized (line 4). In the main loop (lines 5–11) one randomly selects an element $x \in X$ and determines the neuron y^* that is nearest to x. In the inner loop (lines 8 and 9) one considers all neurons y that are within a neighborhood $N(y^*)$ of y^*, including y^*, and updates them according to the formula in line 9. The effect of neuron updating is to move neuron y closer to pattern x. The degree by which y is moved towards x is controlled by the parameter α, which is called the *learning rate*. It has to be noted that α is dependent on the distance between y and y^*, i.e. the smaller this distance is the larger is the change on neuron y. After each iteration through the repeat-loop, the learning rate α is reduced by a small amount, thus facilitating convergence of the algorithm. It can be expected that after a sufficient number of iterations the y_i's have moved into areas where many $x_j's$ are concentrated. Hence each y_i can be regarded a cluster center. The cluster around center y_i consists of exactly those patterns that have y_i as closest neuron. More detail about this algorithm can be found in Kohonen (1997).

In the original version of the som-algorithm all x_j and y_i are feature vectors (Kohonen, 1997). To make the algorithm applicable in the graph domain, two new concepts are needed. First, a graph distance measure has to be provided in order to find graph y^* that is closest to x (see line 7). For this purpose the use of graph edit distance (Messmer and Bunke, 1998) has been proposed in Günter (2000) and Günter and Bunke (2002). Notice that graph edit distance has metric properties (Bunke and Günter, 2001), which is very important for clustering algorithms. Secondly, a graph updating procedure implementing line 9 has to be found. Such a procedure has been described in Günter (2000), Günter and Bunke (2002) and Bunke and Günter (2001).

The purpose of the updating procedure is to change graph y so as to make it more similar to graph x. Thus the edit distance, $d(x,y)$, between x and y is decreased by a certain amount β. The procedure described in Günter and Bunke (2002) and Bunke and Günter (2001) to accomplish the task is based on computing the edit distance, $d(x,y)$, between x and y. Edit distance computation not only yields a distance value, but also a sequence of edit operations, $\delta = e_1, \ldots, e_n$, that transform x into y with minimum cost (Messmer and Bunke, 1998). If we drop some of these edit operations from δ, we obtain a new sequence, δ', that transforms y into another graph y'. Now it can be shown that y' is more similar to x by the amount β, where β is the cost of the edit operations dropped from sequence δ.

With these new concepts the som-algorithm becomes in fact applicable in the domain of graphs. For further details, including the initialization procedure (line 4), definition and computation of neighborhood $N(y^*)$ (line 8), and termination (line 11) (see Günter, 2000; Günter and Bunke, 2002).

3. Cluster validation indices

Some clustering algorithms, for example the methods described in Lus et al. (2001), dynamically obtain the number of clusters during execution, but many clustering algorithms, including the original and the graph based version of som, require the number of clusters given as an input parameter. This is a potential problem as this number is often not known. To overcome the problem, a number of cluster validation indices have been proposed. A cluster validation index is a number that indicates the quality of a given clustering. Hence if the correct number of clusters is not known, one can execute a clustering algorithm multiple times, varying the number of clusters in each run from some minimum to some maximum value. For each clustering obtained under this procedure the considered validation index is computed. Eventually, the clustering that yields the best index value is returned as the final result.

In this section we review a few cluster validation indices that have been proposed in the literature. It is not intended to provide an exhaustive coverage of all known indices. Only a few indices that are intuitively plausible, easy to implement and included in the standard literature on clustering (Dubes, 1993; Theodoridis and Koutroumbas, 1998; Jain et al., 1999) have been taken into account. All indices considered below have been proposed for vectorial pattern representations only. However, as it will become evident below, they can be applied to graph based representations as well if graph edit distance is adopted as distance measure.

3.1. Davies–Bouldin index (Davies and Bouldin, 1979)

This index, Davies–Bouldin (DB), is defined as follows:

$$\mathrm{DB} = \frac{1}{M}\sum_{i=1}^{m} \max_{j=1,\ldots,M;\, j\neq i} (d_{ij}), \text{ where } d_{ij} = \frac{\sigma_i + \sigma_j}{d(c_i, c_j)}$$

In the formulas, M is the number of clusters, σ_i is the average distance of all patterns in cluster i to their cluster center c_i and $d(c_i, c_j)$ is the distance of cluster centers c_i and c_j. Hence d_{ij} is small if clusters i and j are compact and theirs centers are far away from each other. Consequently, DB will have a small value for a good clustering. It is easy to see that $\mathrm{DB} \in [0, \infty]$. This index has a small computational complexity, and can be expected to work

well if the clusters are of spherical shape in the feature space.

3.2. Dunn index (Dunn, 1974)

Let $d_{\min}$ denote the smallest distance between two elements from different clusters, and $d_{\max}$ the largest distance of two patterns from the same cluster. Then the Dunn index, D, is given by

$$D = \frac{d_{\min}}{d_{\max}}$$

Clearly, $D \in [0, \infty]$ with larger values of D indicating better clusterings. Obviously, compact clusters that are well separated in the feature space manifest themselves in small values of $d_{\max}$ and large values of $d_{\min}$, leading to a small value of D. This index is easy to implement and has a low computational complexity, but it is vulnerable to outliers as only two distances are used.

3.3. C index (Hubert and Schultz, 1976)

This index is defined as follows:

$$C = \frac{S - S_{\min}}{S_{\max} - S_{\min}}$$

In this formula, S is the sum of distances over all pairs of patterns from the same cluster. Let l be the number of those pairs. Then $S_{\min}$ is the sum of the l smallest distances if all pairs of patterns are considered (i.e. if the patterns can belong to different clusters). Similarly $S_{\max}$ is the sum of the l largest distances out of all pairs. It is easy to see that the numerator in the formula will be small if pairs of patterns with a small distance are in the same cluster. Hence a small value of C indicates a good clustering. The denominator serves the purpose of normalization, causing $C \in [0, 1]$. This index is appropriate when the clusters have similar sizes.

3.4. Goodman–Kruskal index (Goodman and Kruskal, 1954)

For this index all possible quadruples (q, r, s, t) of input patterns are considered. Let $d(x, y)$ be the distance of pattern x and y. A quadruple is called *concordant* if one of the following two conditions is true:

- $d(q, r) < d(s, t)$, q and r are in the same cluster, and s and t are in different clusters.
- $d(q, r) > d(s, t)$, q and r are in different clusters, and s and t are in the same cluster.

By contrast, a quadruple is called *disconcordant* if one of following two conditions is true:

- $d(q, r) < d(s, t)$, q and r are in different clusters, and s and t are in the same cluster.
- $d(q, r) > d(s, t)$, q and r are in the same cluster, and s and t are in different clusters.

Obviously, a good clustering is one with many concordant and few disconcordant quadruples. (Notice that some quadruples may be neither concordant nor disconcordant.) Let S^+ and S^- denote the number of concordant and disconcordant quadruples, respectively. Then the following quantity, called the Goodman–Kruskal index (GK), can be used to assess a given clustering:

$$\mathrm{GK} = \frac{S^+ - S^-}{S^+ + S^-}$$

Apparently, large values of GK indicate a good clustering. Notice that the denominator normalizes the index such that $\mathrm{GK} \in [-1, 1]$. This index can be expected to be robust against outliers because quadruples of patterns are used for its computation. However, its drawback is a high computational complexity, which restricts its applicability to problem of rather moderate size.

4. Experimental results

The graph clustering algorithm and the cluster validation indices described in Sections 2 and 3, respectively, were experimentally evaluated. In the experiments, graph representations of capital characters were used. In Fig. 2, 15 characters are shown, each representing a different class. The characters are composed of straight line segments. In the corresponding graphs, each line segment is represented by a node with the coordinates of the

Fig. 2. Fifteen characters each representing a different class.

endpoints in the image plane as attributes. No edges are included in this kind of graph representation. The edit costs are defined as follows. The cost of deleting or inserting a line segment is proportional to its length, while substitution costs correspond to the difference in length of the two considered line segments. This kind of representation will be called R1 in the following.

In addition to R1, a second representation, called R2 below, was considered. Under R2, the nodes represent locations where either a line segment ends, or where the end points of two different line segments coincide with each other. The attributes of a node represent its location in the image. There is an edge between two nodes if the corresponding locations are connected by a line in the image. No edge attributes are used in this representation. The deletion and insertion cost of a node is a constant, while the cost of a node substitution is proportional to the distance of the corresponding points in the image plane. The deletion and insertion of an edge also have constant cost. As there are no edge labels, edge substitutions will never be needed under representation R2.

For each of the 15 prototypical characters shown in Fig. 2, 10 distorted versions were generated. Examples of distorted A's are shown in Fig. 3. The degree of distortion of the other characters is similar to Fig. 3. As a result of the distortion procedure, a sample set of 150 characters were obtained. Although the identity of each sample was known, this information was not used in the experiments described below. In other words, only *unlabeled* samples were used in the experiments.

Fig. 3. Ten distorted versions of character A.

The graph clustering algorithm described in Section 2 was run on a set of 150 graphs representing the (unlabeled) sample set of characters, with the number of clusters set to 15. As the algorithm is non-deterministic, a total of 10 runs were executed. The cluster centers obtained in one of these runs are shown in Fig. 4. Obviously, all cluster centers are correct in the sense that they represent meaningful prototypes of the different character classes. In all other runs similar results were obtained for both representation R1 and R2, i.e., in none of the runs an incorrect prototype was generated. Also all of the 150 given input patterns were assigned to their correct cluster center.

From these experiments it can be concluded that the new graph clustering algorithm is able to produce a meaningful partition of a given set of graphs into clusters and find an appropriate prototype of each cluster, if the correct number of clusters is known. The purpose of the experiments described below is to investigate the usefulness of the cluster validation indices described in Section 3 in order to automatically find the correct number of clusters.

The graph clustering algorithm was executed multiple times increasing the number of clusters from 2 to 24. Then all of the four considered indices were computed for each of the resulting clusterings. Graphs showing the values of the indices as a function of the number of clusters are depicted in Figs. 5 and 6. Notice that the optimal number of clusters manifests itself in a small value of the DB and the C index, while it corresponds to a large value of the Dunn and GK index. It is evident from Figs. 5 and 6 that, for each of the considered indices, the correct number of clusters always corresponds to an optimal index value.

Fig. 4. Cluster centers obtained in one of the experimental runs.

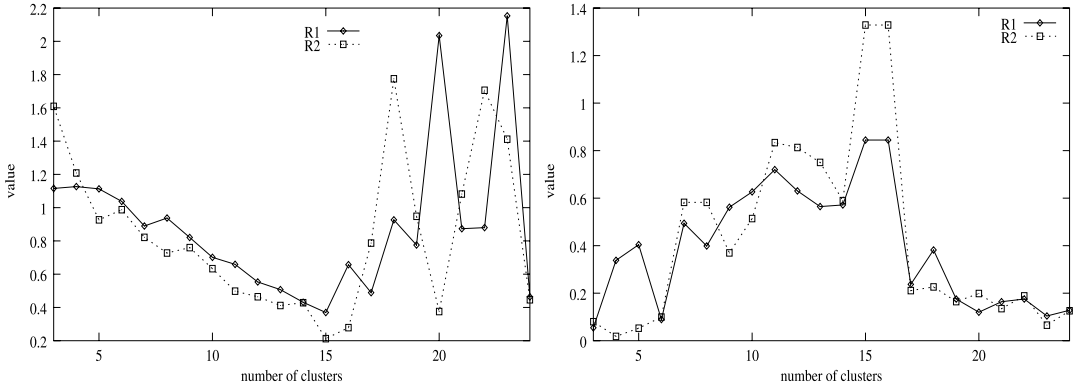

Fig. 5. DB index (left) and Dunn index (right) as a function of the number of clusters.

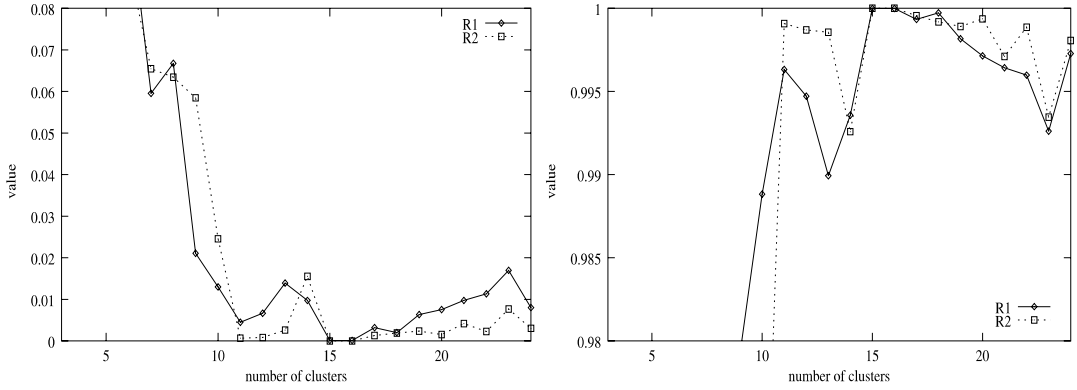

Fig. 6. *C* index (left) and GK index (right) as a function of the number of clusters.

However, there are several ties or near-ties. For example, the minimum of the *C* index is achieved for the correct value of 15, but also for 16 under both representation R1 and R2. In order to make the automatic determination of the optimal number of clusters more robust and eliminate such ties, a simple voting strategy was implemented. The number of clusters with the best, the second best and the third best result for an index receives 3, 2 and 1 points, respectively. (Notice that the best values are large for Dunn and GK, but small for DB and the *C* index.) Then the sum of the points for all indices were calculated for each number of clusters. The result is shown in Fig. 7. Obviously, the combination index shown in Fig. 7 is a clear improvement over all individual indices. The correct number of clusters (15) is represented by a very sharp peak for both representation R1 and R2. Ties or near-ties do not occur any longer.

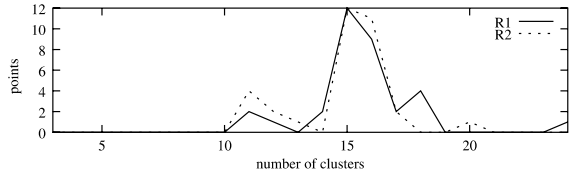

Fig. 7. Score resulting from the combination of all indices.

5. Conclusions

Clustering has become a mature subfield of pattern recognition, but the clustering of graphs is still a widely unexplored area. In the present paper

a new graph clustering algorithm is reviewed. It is an extension of the well-known som-algorithm into the domain of graphs. The main contribution of the paper is an extension of a number of cluster validation indices from feature vector representations to the graph domain. These cluster validation indices allow the application of the proposed graph clustering algorithm without any prior knowledge of the number of clusters. The applicability of the new graph clustering algorithm and the cluster validation indices has been demonstrated through a series of experiments.

Only drawings consisting of straight line segments are considered in this paper. But it is straightforward to apply the approach to graph representations of other types of patterns. Examples are line drawings with curved segments (Foggia et al., 1999) or patterns composed of regions (De Mauro et al., 2001). Future research will also address the development of additional graph clustering algorithms and techniques to find the optimum number of clusters automatically.

References

Bunke, H., Günter, S., 2001. Weighted mean of a pair of graphs. Computing 67 (3), 209–224.

Davies, D., Bouldin, D., 1979. A cluster separation measure. IEEE Trans. Pattern Recognit. Machine Intell. 1 (2), 224–227.

De Mauro, C., et al., 2001. Similarity learning for graph-based image representations, in: Jolion, J.-M., Kropatsch, W., Vento, M. (Eds.), Proc. 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, 2001, pp. 250–259.

Dubes, R., 1993. Cluster analysis and related issues. In: Chen, C., Pau, L., Wang, P. (Eds.), Handbook of Pattern Recognition and Computer Vision. World Scientific, pp. 3–32.

Dunn, J., 1974. Well separated clusters and optimal fuzzy partitions. J. Cybernet. 4, 95–104.

Englert, R., Glantz, R., 2000. Towards the clustering of graphs, in: Proc. 2nd IAPR-TC-15 Workshop on Graph-based Representations. Austrian Computer Society, Austria, 2000, pp. 125–133.

Foggia, P., Sansone, C., Tortorella, F., Vento, M., 1999. Definition and validation of a distance measure between structural primitives. In: Pattern Analysis and Applications 2, vol. 3. Springer, pp. 215–227.

Fritzke, B., 1999. Growing self-organizing networks—history, status quo and perspectives. In: Oja, E., Kaski, S. (Eds.), Kohonen Maps. Elsevier, pp. 131–144.

Günter, S., 2000. Graph clustering using Kohonen's method. Master's thesis. University of Bern (in German), 2000.

Günter, S., Bunke, H., 2002. Self-organizing map for clustering in the graph domain. Pattern Recognition Lett. 23, 401–417.

Goodman, L., Kruskal, W., 1954. Measures of associations for cross-validations. J. Am. Stat. Assoc. 49, 732–764.

Hubert, L., Schultz, J., 1976. Quadratic assignment as a general data-analysis strategy. Br. J. Math. Stat. Psychol. 29, 190–241.

Jain, A., Murty, M., Flynn, P., 1999. Data clustering: a review. ACM Comput. Surveys 31, 264–323.

Kohonen, T., 1997. Self-Organizing Map. Springer.

Lus, B., Robles-Kelly, A., Torsello, A., Wilson, R., Hancock, E., 2001. Clustering shock trees. In: Jolion, J.-M., Kropatsch, W., Vento, M. (Eds.), Proc. 3rd IAPR-TC15 Workshop on Graph-Based Representations in Pattern Recognition, 2001, pp. 217–228.

Messmer, B., Bunke, H., 1998. A new algorithm for error tolerant subgraph isomorphism. IEEE Trans. Pattern Recognit. Machine Intell. 20, 493–505.

Riano, D., Serratosa, F., 2000. Unsupervised synthesis of function described-graphs, in: Proc. 2nd IAPR-TC-15 Workshop on Graph-based Representations. Austrian Computer Society, Austria, 2000, pp. 165–171.

Seong, D., Kim, H., Park, K., 1993. Incremental clustering of attributed graphs. IEEE Trans. Systems Man Cybernet., 1399–1411.

Theodoridis, S., Koutroumbas, K.K., 1998. Pattern Recognition. Academic Press.

Xu, L., Krzyzak, A., Oja, E., 1993. Rival penalized competitive learning for clustering analysis, rbf net, and curve detection. IEEE Trans. Neural Networks 4, 636–649.