

Coring Method for Clustering a Graph

Thang V. Le
Dept. of Computer Science
Rutgers University
thang@cs.rutgers.edu

Casimir A. Kulikowski
Dept. of Computer Science
Rutgers University
kulikows@cs.rutgers.edu

Ilya B. Muchnik
DIMACS
Rutgers University
muchnik@dimacs.rutgers.edu

Abstract

Graph clustering partitions a graph into subgraphs with strongly interconnected nodes, while nodes belonging to different subgraphs are weakly connected. In this paper, we propose a new clustering method applicable to either weighted or unweighted graphs in which each cluster consists of a highly dense core region surrounded by a region with lower density. We have developed a highly efficient and robust method to identify nodes belonging to dense cores of clusters. The set of the nodes is then divided into groups, each of which is the representative of one cluster. These groups are finally expanded into complete clusters covering all the nodes of the graph. Experiments with both synthetic and real datasets for gene expression analysis and image segmentation yield very encouraging results.

1. Introduction

Let us consider an undirected proximity graph $G = (V, E, W)$, where V is a set of nodes, E is a set of edges, W is a matrix with entry w_{ij} being the weight of the edge between nodes i and j . In proximity graphs, V represents a set of data objects, $w_{ij} \geq 0$ represents the similarity of the objects i and j . A higher value of w_{ij} reflects a higher degree of similarity. A proximity graph is the natural representation for a dataset if pairwise relationships between data objects are explicitly provided. If the data is represented in a feature space, a proximity graph can be derived by computing pairwise similarities based on Euclidean or other distance metrics between data points in the feature space, so that the dataset can be analyzed by graph-based data analysis approaches.

Graph clustering methods aim to discover dense subgraphs in an unweighted graph, or highly weighted subgraphs in a weighted graph, such that the sum of edge weights inside subgraphs is high, while the sum of weights of edges connecting different subgraphs is low. Thus, applying a graph clustering method to a

proximity graph will produce a set of subgraphs, such that each subgraph corresponds to a group of similar objects, which are dissimilar to objects of groups corresponding to other subgraphs. There are different approaches for clustering a graph such as maximum cliques, thresholding minimum spanning tree, random walk, and spectral clustering. In this paper, we propose a method that finds clusters of a graph by first detecting nodes in the densest regions of clusters, then identifying the cores of clusters, and lastly assigning all other nodes to the nearest cluster. Details of the method are described next in Section 2. Sections 3 and 4 present experimental results on gene expression analysis and image segmentation. Advantages of the method are briefly discussed in Section 5.

2. Coring method for clustering a graph

2.1. The greedy procedure of layered clustering

Our graph clustering method is derived from the layered clustering approach in [1]. Layered clusters are chain-nested subgraphs such that the density of a subgraph is higher than the densities of any subgraphs which are not part of it. This approach finds layered clusters of a proximity graph by a greedy procedure which iteratively removes the least similar node, and then measures the density of the graph. The sequence of density values produced by this procedure is used to find the densest subgraph and layered clusters. The similarity of a node i with a subgraph H is defined by $\sum_{j \in H} w_{ij}$. The density of a subgraph H is defined as the minimum similarity of the nodes of H : $\min_{i \in H} \sum_{j \in H} w_{ij}$. The layered clustering approach produces as the output chain-nested subgraphs but it does not partition the graph into disjoint subgraphs, so it does not provide a clustering solution. A previously proposed clustering method based on [1] has several issues as shown in [2].

2.2. The dense core assumption

We assume that every cluster of the input graph has

a region of high density called a ‘cluster core’, surrounded by sparser regions (non-core). The nodes in cluster cores are denoted as ‘core nodes’, the set of core nodes as the ‘core set’, and the subgraph consisting of core nodes as the ‘core graph’. For a graph satisfying the above core assumption, the greedy procedure used in the layered clustering method exposes two useful properties: (i) the least similar node is removed at each iteration, so the nodes of the graph are arranged in an order where nodes in sparser regions are placed ahead of nodes in denser regions. In other words, the procedure progresses from outer layers to inner layers of the graph, (ii) density values drop significantly when the procedure removes nodes belonging to cluster cores of the graph. We have taken advantage of these properties in order to design a new clustering method for graphs in which each cluster has a dense core. For each node i of $H \subseteq V$, the local density at i is defined as $d(i, H) = (\sum_{j \in H} w_{ij}) / |H|$. The node with the minimum local density in H is referred to as the weakest node of H : $\operatorname{argmin}_{i \in H} d(i, H)$. We define the minimum density of H as $D(H) = \min_{i \in H} d(i, H)$ to measure the local density of the weakest node of H .

Let us consider the greedy procedure that iteratively computes $D(G)$ and removes weakest nodes from G . By analyzing the variation of the minimum density value D , we can identify core nodes located in the dense cores of clusters. That is, if the weakest node is in a sparse region, the D value will increase when this node is removed, in other words, the next weakest node to be removed will be in a region with higher density. On the other hand, if the removal of the weakest node causes a significant drop in D value, then this node is highly connected with a set of stronger nodes in a high density region. It is potentially a core node because its removal greatly reduces the density of nodes around it.

Fig. 1 shows an example of a weighted graph of 25 nodes and 126 edges with two dense cores. Edge weights are inversely proportional to normalized edge lengths and computed by $w_{ij} = 1 - (d_{ij} / \max_{x,y \in V} d_{xy})$, where d_{xy} is the Euclidean distance between points x and y on the plane. By continuously removing the weakest nodes, we gradually go to the first cluster core on the left, while D values tend to increase. At some point, when we remove nodes from the core ($t=8 \rightarrow 13$), D values drop precipitously because the cluster core collapses. After the first cluster is fully removed, D values raise but then drop again when the second cluster core collapses ($t=21 \rightarrow 25$). Thus the decreases of D values indicate core nodes of cluster cores.

2.3. The coring method

Our method clusters a proximity graph in four steps.

Step 1: Compute the density variation sequence.

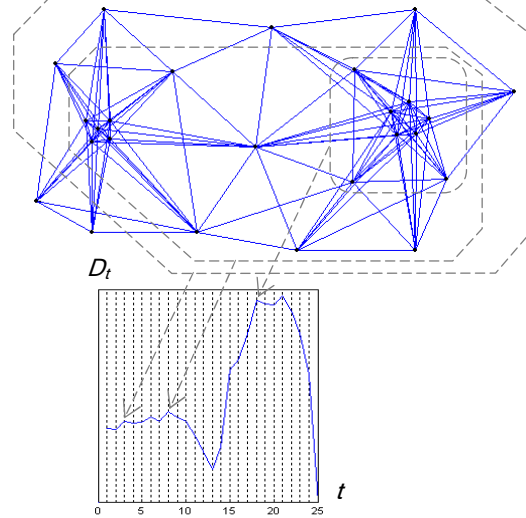


Figure 1. A graph and its D value sequence.

This is an iterative procedure that computes the sequence of density variation. At each iteration t , we compute the minimum density D_t , and the set of the weakest nodes M_t . Then, M_t is removed from the graph. When the graph becomes empty, the procedure returns the sequences of D_t s and M_t s, which contain D values and the nodes relating to each D value, respectively.

Step 2: Identify the set of core nodes. Core nodes are identified based on the sequences of D_t s and M_t s produced by step 1 and two parameters: $\alpha \in [0, 1)$ and $\beta \in \mathbb{N}$. Elements of M_t are selected to be core nodes if the corresponding D_t satisfies two conditions:

(1) Its rate of decrease R_t is greater than α . That is, $R_t = (D_t - D_{t+1}) / D_t > \alpha$.

(2) $\exists k \in \mathbb{N}$ such that $D_t \in \{D_{k+1}, D_{k+2}, \dots, D_{k+\beta}\}$, where $D_{k+1}, D_{k+2}, \dots, D_{k+\beta}$ are β consecutive density values that also satisfy the condition (1).

Therefore, elements of M_t are core nodes if the corresponding D_t has the rate of decrease R_t greater than α , and D_t is among at least β successive D values which also have a rate of decrease greater than α . Parameter α specifies the minimum rate of decrease for D values of core nodes; β controls the minimum size of a group of successive core nodes on the M_t sequence. The range of β in our experiments here is $\beta \in \{2, 3, 4\}$.

Step 3: Partition the core set into cluster cores. The set of core nodes identified in step 2 is partitioned into groups representing cluster cores. The number of obtained groups automatically determines the number of clusters. For sparse graphs, we can find all the connected components of the core graph, and then each component is regarded as a cluster core. When the graph is highly connected, the core graph may be connected in only one component. In this case, before finding connected components, we can apply a pre-processing such as: (i) for unweighted graphs, step 1 is

applied again to the core graph to get a smaller set of core nodes, (ii) for weighted graphs, the histogram of edge weights can be scanned to determine a threshold to remove weak edges connecting core groups. In fact, step 3 solves another clustering problem, and any graph clustering method may be used for partitioning the core set. In highly connected weighted graphs, hierarchical clustering is an excellent choice for this step: The dendrogram can be built for the set of core nodes; if core groups are well separated, they can be easily extracted from the dendrogram. Note that partitioning the core set is easier than clustering the original graph because we have a smaller set of nodes, and the separation of cluster cores in the core set is much better than that of clusters in the original graph.

Step 4: Expand core groups into full clusters. This final step is similar to solving a supervised classification problem where unlabeled data is classified based on a training set of previously labeled data. Here we take advantage of a property of step 1 of going from outer to inner layers, that is, nodes in the sequence M_t are ranked from sparse to dense regions. Therefore, scanning the sequence backwards will go from the centers to the boundaries of clusters. While scanning the sequence, non-core nodes are assigned to the most similar cluster, thus expanding cluster cores to full clusters. Each non-core node n of M_t is assigned to the cluster $C^* = \operatorname{argmax}_C S(n, C)$, where $S(n, C)$ measures the degree of similarity of a node n with a cluster C . For weighted graphs, it can be defined by $S(n, C) = \operatorname{average}_{i \in C, w_{ni} > 0} w_{ni}$, or $S(n, C) = \max_{i \in C} w_{ni}$. For unweighted graphs, we can use $S(n, C) = \operatorname{average}_{i \in C} w_{ni}$, or $S(n, C) = \sum_{i \in C} w_{ni}$. If a node m is found to be not connected to any known clusters, i.e., $\forall C: S(m, C) = 0$, then a new cluster C^+ will be created to accommodate m . This can happen if the parameters are set to so high values that representatives of some clusters are excluded from the core set. To reveal the structure of clusters, we can compute a confidence degree for each node when assigning it to a cluster. Let $C1$ be the most similar cluster and $C2$ be the second most similar cluster of node n . The confidence degree of n is defined as $\operatorname{confidence}(n) = 1 - (S(n, C2)/S(n, C1))$.

3. Gene expression analysis experiment

Clustering applications to gene expression analysis were demonstrated in [3]. Here we tackle the problem of tissue clustering which aims to find connections between gene expressions and statuses of tissues. In other words, we want to see if it is possible to predict the status of a tissue based on its gene expressions. The dataset used in this experiment is publicly available at microarray.princeton.edu/oncology/affydata/index.html. This data contains 62 samples including 40 tumor and

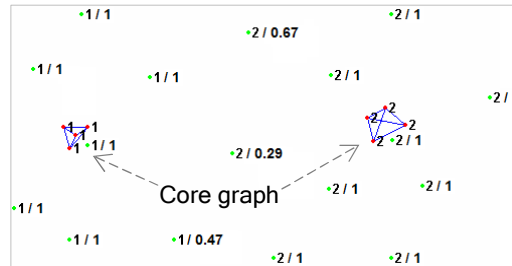


Figure 2. Cluster labels and confidence degrees of the clustering result for the graph in Fig. 1.

22 normal colon tissues. Each sample consists of a vector of 2000 gene expressions. We will set aside the sample labels (tumor/normal) and cluster the samples based on the similarities between their gene expressions. Ideally, we want to partition the sample set into two clusters such that one contains only tumor tissues and the other contains only normal tissues. The proximity graph constructed from the gene expression vectors is a complete graph of 62 nodes. Because relative values are more important than absolute values in gene expressions, edge weights that reflect the pairwise similarities of samples are computed based on the Pearson correlation coefficient [4]. Specifically, the weight function is defined by:

$$w_{ij} = \frac{1}{2000} \sum_{k=1}^{2000} \frac{1}{s_i s_j} (i_k - m_i)(j_k - m_j),$$

where i_k and j_k are gene expressions of samples i and j ; m_i, m_j, s_i, s_j are means and standard deviations of i_k s and j_k s. Steps 1 and 2 of the coring method identify 12 core nodes. The dendrogram of these core nodes shown in Fig. 3 exposes two well-separated groups, one contains 10 nodes and the other has 2 nodes. Thus, we cut the dendrogram at the height of 0.1 to obtain two cluster cores. Expanding these cluster cores yields two clusters. One has 40 samples consisting of 37 tumor and 3 normal tissues. The other contains 22 samples consisting of 3 tumor and 19 normal tissues.

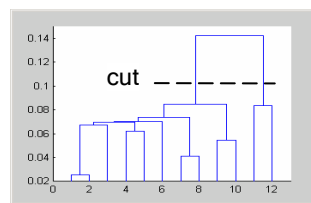


Figure 3. Dendrogram of the core graph.

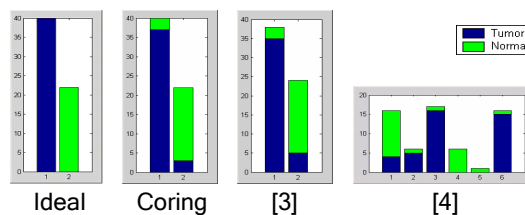


Figure 4. Clustering result comparison.

Fig. 4 shows the comparison of clustering results by the coring method, [3] and [4]. The result of [4] consists of 6 clusters, but joining clusters 1, 4 and 5 into one group of normal tissues and 2, 3 and 6 into another group of tumor tissues yields a clustering similar to the result of [3].

4. Image segmentation experiment

We consider the problem of partitioning a grayscale image into regions of nearby pixels that have similar intensities. Based on an image, we can construct a proximity graph where each node represents a pixel. The weight of an edge between two nodes reflects the pairwise similarity of the corresponding pixels, i.e., the likelihood that those two pixels belong to the same segment. Therefore, edge weights are computed based on the intensities and locations of pixels. Specifically, we use the following weight function described in [5]:

$$w_{ij} = e^{-\left(\frac{I(i)-I(j)}{\sigma_I}\right)^2 - \left(\frac{d_{ij}}{\sigma_d}\right)^2} \text{ if } d_{ij} < r; \text{ but } w_{ij} = 0 \text{ if } d_{ij} \geq r,$$

where $I(i) \in [0,1]$ is the intensity of pixel i ; d_{ij} is the distance in pixels between i and j . Edges only exist between nodes whose pixels are apart by a distance less than r . Here we use $\sigma_I = 0.1$, $\sigma_d = 7$, $r = 10$ for images of sizes around 200×300 . Segmentations of an image are obtained by clustering its proximity graph. In Fig. 5, the second column shows the segmentation results by our method on some images of the Berkeley dataset [6]. By tuning α and β , we can adjust the segmentation such as the number of segments, which is from the number of connected components in the core graph.

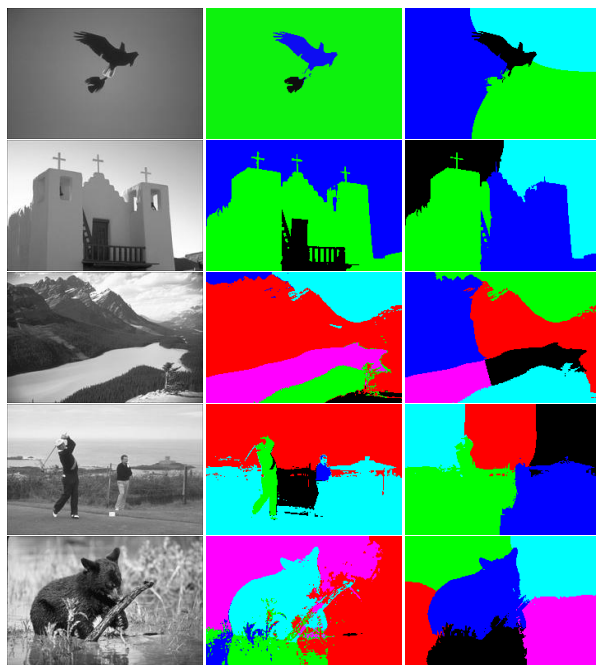


Figure 5. Image segmentation comparison.

A well known spectral clustering method based on minimum normalized cuts is presented in [5]. An implementation by its authors is available on the web. One of the problems with minimizing normalized cuts is that the normalizing factors in this criterion function make it favor clusters with similar sizes. In Fig. 5, the last column shows the segmentations by normalized cuts where we see large segments tend to break into several parts of similar sizes. Clustering on the same proximity graphs, our method produces much better segmentations. It also has a superior running time. Specifically, our implementation with the step 1 using a Fibonacci heap needs only 1 second versus about 30 seconds for the normalized cut method for clustering a weighted graph of 60×10^3 nodes and 10^7 edges.

5. Conclusions

We have described a novel method for clustering a graph in which clusters have a strongly connected core. Experiments with synthetic graphs and proximity graphs built from gene expressions and images have shown good clustering results. The method is simple and fast. Input parameters have a clear interpretation and can be adjusted to yield a wide spectrum from coarse to fine-grained clustering. In practice, searching the parameter space for a good setting is relatively easy. Among four steps of the algorithm, step 1 has the highest complexity (from $O(|V| \log |V|)$ to $O(|V|^2)$), but it is executed just once for all the settings of α and β . If a parameter is changed, we only have to re-execute steps 2, 3, and 4, which are very fast. Core nodes are extracted from cores of clusters, so they can represent informative data objects and also make the method robust to noise. We have seen that the set of core nodes and segmentation results remain stable even if noise is added to input images. The method can be extended for clustering directed graphs if we define a density function that takes into account edge directions.

6. References

- [1] B. Mirkin and I. Muchnik, Layered clusters of tightness set functions, *Applied Math. Letters*, 15:147-151, 2002.
- [2] T. Le, C. Kulikowski and I. Muchnik, Coring method for clustering a graph, *DIMACS Technical Report*, 2008.
- [3] U. Alon, N. Barkai, D. Notterman, K. Gish, S.Ybarra, D. Mack and A. Levine, Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide array, *Proc. Natl. Acad. Sci. USA*, 96(12):6745-6750, 1999.
- [4] A. Ben-Dor, R. Shamir and Z. Yakhini, Clustering gene expression patterns, *Journal of Computational Biology*, 1999.
- [5] J. Shi and J. Malik, Normalized cuts and image segmentation, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8):888-905, 2000.
- [6] <http://www.cs.berkeley.edu/projects/vision/bsds>.