Published in final edited form as:
Proc IEEE Int Conf Data Min. 2009 ; : 968–973. doi:10.1109/ICDM.2009.141.

A Fully Automated Method for Discovering Community Structures in High Dimensional Data

Jianhua Ruan
Department of Computer Science University of Texas at San Antonio One UTSA Circle, San Antonio, TX 78249 jruan@cs.utsa.edu

Abstract

Identifying modules, or natural communities, in large complex networks is fundamental in many fields, including social sciences, biological sciences and engineering. Recently several methods have been developed to automatically identify communities from complex networks by optimizing the modularity function. The advantage of this type of approaches is that the algorithm does not require any parameter to be tuned. However, the modularity-based methods for community discovery assume that the network structure is given explicitly and is correct. In addition, these methods work best if the network is unweighted and/or sparse. In reality, networks are often not directly defined, or may be given as an affinity matrix. In the first case, each node of the network is defined as a point in a high dimensional space and different networks can be obtained with different network construction methods, resulting in different community structures. In the second case, an affinity matrix may define a dense weighted graph, for which modularity-based methods do not perform well. In this work, we propose a very simple algorithm to automatically identify community structures from these two types of data. Our approach utilizes a *k*-nearest-neighbor network construction method to capture the topology embedded in high dimensional data, and applies a modularity-based algorithm to identify the optimal community structure. A key to our approach is that the network construction is incorporated with the community identification process and is totally parameter-free. Furthermore, our method can suggest appropriate preprocessing/normalization of the data to improve the results of community identification. We tested our methods on several synthetic and real data sets, and evaluated its performance by internal or external accuracy indices. Compared with several existing approaches, our method is not only fully automatic, but also has the best accuracy overall.

Keywords

community structure; modularity; image clustering

I. Introduction

Complex network structures have drawn much interest lately in many fields, ranging from biological sciences, engineering, to social sciences. (See [1] for reviews.) In the framework of network analysis, a system is modeled as a graph, where the vertices are the elements of the system, and the edges represent certain relationships between pairs of vertices. It is well known that networks of real-world problems often have natural community structures, i.e. highly connected subnetworks that are not expected by chance [1], [2]. The identification

and characterization of communities are fundamental in understanding the organization and dynamics of complex systems, and have become one of the key problems in analyzing complex networks.

In order to identify such communities, many graph theoretical algorithms have been developed recently. Among them, three types of methods have received broad attention: Markov Cluster algorithm (MCL) [3], Affinity Propagation (AP) [4], and modularity-optimization algorithms [5], [2]. Unlike previous graph partitioning approaches that require a user to specify the number of partitions (explicitly or implicitly), MCL and AP are relatively parameter free. MCL is a graph cluster algorithm based on simulation of (stochastic) flow in graphs [3], and requires a single parameter called *inflation*. AP is a clustering algorithm that groups data points by passing messages between them. In AP, a user needs to determine a *prior*, preferably for every node. Although the parameters in both MCL and AP can be set relatively easily, still, one has to supply a series of parameters and manually choose one based on inspection of the modules identified, possibly with the assistance of a different data source.

In the last few years, several modularity-based methods have been developed in the statistical physics domain, pioneered by Newman and his colleagues [5], [2]. These approaches distinguish themselves from conventional graph partitioning algorithm in that they are aimed at optimizing an objective function called *modularity*, or *Q* for short; therefore, no parameter is required at all. The modularity function, which we will define later in Methods, has been shown to be effective in measuring the overall quality of community structures in many cases [6], [5], [7], [8]. On the other hand, several recent studies have suggested that the modularity function has a so-called resolution limit, i.e., it tends to miss small yet genuine communities if they are attached to a large community via a small number of edges [9], [10]. A few algorithms have been developed to address this problem [9], [11], [8].

A common problem with the modularity-based community identification algorithms is that they all assume the network structure has been explicitly given and is correct. In addition, these algorithms work best if the network edges are unweighted and/or sparse, because of an intrinsic limitation of the modularity function (discussed later in Methods).

In practice, however, many networks are defined implicitly by the attributes of their vertices/nodes. For example, in a genetic network, each gene can be defined by its expression levels under various conditions; in a document network, each document is defined by a list of words and their frequencies in the document. In these cases, each node can be represented as a point in a high dimensional space. In order to identify modular structures in such systems, traditional clustering algorithms are usually applied, which tend to ignore the topological structure underlying the system [12]. Alternatively, there have been several approaches which first construct a graph from the data by connecting pairs of nodes that are similar, and then search for dense subgraphs. Examples include DBScan [13], Chameleon [14], and SPARCL [15]. These types of approaches often require a number of parameters to be set by the user both to construct the network and to identify dense subgraphs.

In addition, sometimes a network may be given as a weight matrix W, where W_{ij} represents the affinity between nodes i and j based on some measurement. For example, in a collaboration network, the affinity between two researchers may be the number of papers they have co-authored, normalized by the total number of papers published by each author. This is another type of high dimensional data if one considers each node as being defined by its affinity to all the other nodes. However, the affinity measure is not necessarily a metric, and therefore the data points may not have a direct geometric meaning. In such cases, although one can still apply traditional clustering algorithm, their performance is often suboptimal. Spectral clustering algorithms, such as the one proposed in [16], often works well by treating the affinity matrix as a network with weighted edges. Similar to many clustering algorithms, however, it requires the number of clusters to be determined either explicitly or implicitly.

Inspired by the current development of community identification algorithms, we believe it will be highly desirable to have an algorithm to automatically identify community structures from these types of high dimensional data. One idea would be to apply modularity optimization algorithm directly to the affinity matrix or similarity matrix derived from attribute values. However, as mentioned, the modularity function does not extend well to affinity/similarity matrix, which is a dense weighted network. Another idea would be to construct a sparse unweighted network from the data, and then apply modularity optimization to find community structures. However, the construction of the network is often done in an arbitrary way, and is not incorporated into the process of community identification. Different network construction methods or parameters will result in different networks and usually different community structure as well.

In this work, we propose a very simple method that incorporates network construction and modularity-based community identification. Given a set of high dimensional points or an affinity matrix, we construct a series of nearest neighbor networks, and then apply a modularity-based algorithm to search for the optimal community structure in each network. At the same time, we compare the community structures obtained from different networks, and select the one with the highest *absolute modularity*, which is essentially modularity corrected for the biases caused by different network densities. As absolute modularity is a global measurement of community strengths, this simple idea can also be used to guide the pre-processing/normalization step such that the resulting network has the best community structure.

We tested our methods on a large number of synthetic and real data sets, and evaluated our method by internal or external accuracy indices. Compared with several existing approaches, including *k*-means, MCL and AP, our method is not only fully automatic, but also has the best accuracy overall.

The paper is organized as follows. In Section 2, we introduce our method for constructing networks and identifying communities. We present experimental results in Section 3 and end with some concluding remarks in Section 4.

II. Methods

In this section, we first provide a brief overview of the modularity function, and two modularity-based algorithms. One algorithm, called *Qcut*, has been shown to be able to optimize modularity efficiently and effectively [8]. The other algorithm, *HQcut*, is an extension of *Qcut* for addressing the resolution limit problem [8]. We then present our algorithm, based on these two algorithms, to automatically identify the optimal community structure from high dimensional points or affinity matrices.

A. Community identification via modularity optimization

1) Modularity—Given an unweighted network with N vertices and M edges, and a partition that divides the vertices into k communities, the modularity function is defined as

$$Q=\sum_{i=1}^{k}\left(\frac{e_{ii}}{M}-\left(\frac{a_i}{2M}\right)^2\right), \quad (1)$$

where e_{ii} is the number of edges within community i, and a_i is the total degree for the vertices in community i [2]. The Q function measures the fraction of edges falling within communities, subtracted by what would be expected if the edges were randomly placed. A larger Q value indicates stronger community structures. If a partition gives no more intra-community edges than would be expected by chance, $Q \leq 0$. For a trivial partition with a single cluster, $Q = 0$. Given the definition of Q, the community discovery problem is to find a partition of the network that optimizes Q.

The definition of Q *per se* can be easily extended to weighted networks, by replacing vertex degree with total edge weight. However, Q is ill-defined for weighted *and dense* network. This is because, unlike an unweighted network, a weighted network cannot be randomly rewired and yet remain its degree sequence. Therefore, the second term in the above formula does not reflect the expected fraction of edges falling within communities. This limitation, along with the resolution limit of the modularity function, creates a problem when one wants to identify communities via modularity optimization from dense weighted networks.

Below, we briefly describe an algorithm to optimize modularity, and an algorithm to address the resolution limit problem. We will later utilize these two algorithms to identify community structures from dense weighted networks.

2) Qcut—*Qcut* is an spectral-based modularity optimization algorithm developed in [8]. This algorithm combines recursive graph partitioning with local search to balance between efficiency and accuracy. It has been shown to outperform several other modularity-optimization methods in terms of both efficiency and accuracy [8]. The code is written in MATLAB and freely available from the authors.

3) HQcut—*HQcut* is another algorithm developed by the same authors as *Qcut*. *HQcut* successfully solved the resolution limit problem of the modularity function by iteratively applying *Qcut* to each community that has been identified, as long as the partitioning seems statistically significant [8].

B. Our algorithm

Assume that we are given a sample S containing n data points, X_1, …, X_n, which are i.i.d. vectors in R^d. Our objective is to construct an unweighted and undirected graph, where each data point is a vertex of the graph, and two vertices are connected if the distance between their respective vectors is small enough. Distances can be measured by any metric that is deemed appropriate by the user, and is usually specific to applications. Euclidean distance is used throughout this work. Given the distance measure, we construct a mutual kNN graph [17]. Briefly, for each point X_i in S, let $N_k(X_i)$ be the k nearest neighbors of X_i, the network $kNN(S, k)$ is defined as follows: X_i and X_j are connected if $X_i \in N_k(X_j)$ AND $X_j \in N_k(X_i)$.

Several alternative methods can be used as well. For example, one can define a network based on a distance threshold cutoff: X_i and X_j are connected if the distance between X_i and X_j is smaller than a threshold, ε. Another alternative approach is to connect X_i and X_j if $X_i \in N_k(X_j)$ OR $X_j \in N_k(X_i)$ (called asymmetric kNN in [17]). Each method has some advantages and disadvantages, and a thorough discussion is out of the scope in this work. Briefly, compared to the threshold-based approach, the mutual kNN approach is better capable of identifying clusters of different shapes and sizes, and compared to the asymmetric kNN approach, the mutual kNN approach is less prone to noises in data and can identify outliers [17]. In this work, we always use the mutual kNN approach.

Given a network constructed with one of the methods above, we can then apply *Qcut* or *HQcut* to identify community structures. However, as the final community structure depends on the network topology, which is determined by the single parameter, k, it is critical to determine a good k, preferably in an automated way. In this work, we propose the following fully automated approach.

The idea is simple - when a good k is chosen, there should be relatively more intra-community edges than would be expected. Therefore, the modularity score should be high. On the other hand, it is known that even random sparse networks may have some modularity by chance, and the sparser a network is, the higher modularity it may have by chance [18]. Therefore, we search for the k that gives the largest *absolute modularity*, defined as the difference between the modularity of the real network and the modularity of an appropriate random network. In order to obtain a random network that has the same density as the real network, we randomize the real network by randomly shuffle the ends of all edges. With this approach, not only the random network has the same overall density as the real network, but each node in the random network has exactly the same degree as in the real network. The pseudocode of our algorithm is shown in Figure 1.

Compared to existing algorithms, our algorithm has a few advantages. First, by constructing a mutual kNN graph, it can capture clusters of various sizes, shapes, and density, and is robust with respect to noises. Second, our algorithm automatically determines the best k in order to achieve the optimal modularity. Furthermore, in real applications, it is often desired to normalize the data before clustering, in order to eliminate systematic biases caused by technical or other reasons. For example, gene expression data needs to be normalized to eliminate biases introduced by experimental protocols or equipments. Many normalization techniques have been developed and it is often unclear to the user which one should be

performed. Assuming that a good normalization technique should preserve the modular structure of the data, the absolute modularity measure we defined here can be used to choose a good normalization strategy prior to clustering - one can try several normalization methods and choose the one with the highest absolute modularity.

The most time-consuming part of our algorithm is in constructing the kNN graph, which can be done in $O(n^2)$ time. Note that we did not search all possible k's. We have found that the algorithm is typically robust with respect to a large range of values of k. We choose k to be a power of 2; therefore only $\log_2(n)$ iterations of *Qcut* needs to be executed. This significantly reduces running time without sacrificing the performance of the algorithm.

III. Experimental Results

We extensively tested our method on several types of data sets, including a large number of synthetic affinity matrices, synthetic gene expression microarray data, synthetic high dimensional data with irregular shapes, as well as an image data set and a large-scale gene expression microarray data set. All experimental results show that the performance of our algorithm is superior to a number of existing algorithms. Due to space constraints, we omitted the synthetic and real microarray test cases here.

In this work, clustering accuracy is measured using Jaccard Index, which is defined based on the number of correctly identified intra-community vertex pairs [19]. Given a true community structure, C_t, and a predicted community structure, C_p, let S_1 be the set of vertex pairs in the same community of C_t, and S_2 the set of vertex pairs in the same community of C_p. The Jaccard Index is defined as

$$JI(C_t, C_p) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}.$$

The value of Jaccard Index is in [0, 1], with one being the most accurate. The results using two other accuracy measurements, adjusted Rand Index [19] and Variation of Information [19], are similar, but are omitted here for clarity.

A. Evaluation using synthetic data

1) Community structure in affinity matrices—We first tested the case where the network is given as a general affinity matrix W, where W_{ij} represents the affinity between nodes i and j based on some measurement. The affinity in this example has no physical meaning and is a not a metric. To identify community structures for this network, one can either apply a cutoff to remove some edges with lower weights, or following the same idea in this work, construct a mutual kNN graph by connecting two nodes if their affinity is within each other's top k. Here we test whether our algorithm can be utilized to derive an optimal mutual kNN graph in order to recover the community structures in this type of networks.

In this experiment, we first generate an unweighted and undirected network with 500 nodes, divided into 10 equalsized communities. Each node has a 0.3 probability to connect to

another node in the same community, and a 0.03 probability to connect to a node outside of the community. On average, each node has 15 within-community edges and approximately 12 between-community edges. Let A denote the adjacency matrix of the network, i.e., $A_{ij} = 1$ if and only there is an edge between nodes i and j. We then create an affinity matrix W using the following formula:

$W = A + c * R$, where c is a scale factor, and R a symmetric matrix with uniformly distributed random numbers between 0 and 1. We set c to be between 0.5 and 2.5.

We applied our algorithm to each matrix in order to derive an optimal kNN graph and identify an optimal community structure simultaneously. To compare, we also applied the community discovery algorithm (*HQcut*) directly to W. Figure 2(a) shows that the optimal network structure for an example affinity matrix with $c = 1$ is obtained when $k = 32$. Figure 2(b) shows the clustering accuracy of the two methods. Clearly, for c between 1.5 and 2, our algorithm in this work significantly outperforms *HQcut*. For c values lower than 1.5, both algorithms work equally well, and for c values higher than 2, both methods suffer from low accuracy.

2) Community structure in high dimensional data—Furthermore, to test the performance of our algorithm on high dimensional data points, we used a synthetic data generator SynDECA (http://cde.iiit.ac.in/soujanya/syndeca/) to generate a large number of data sets. Each data set contains 1000 points, divided into 20 clusters, with about 5% of uniformly distributed noisy points (outliers). SynDECA can generate regular (circle, eclipse, square and rectangle) as well as random/irregular shapes. The dimensions of these data sets are between 10 and 100.

We applied *HQcut*, the algorithm in this work, and k-means algorithm to these data sets. For k-means, the number of clusters is set to 21. In addition, we constructed kNN graphs using all possible values of k, and applied *HQcut* to each resulting network to find the one with the best clustering accuracy (instead of the best absolute modularity). This represents the optimal clustering accuracy that can be achieved by applying *HQcut* to the best kNN graphs (optimal-kNN). Figure 2(c) shows the clustering accuracy of the four approaches. The accuracy of our algorithm is clearly much higher than k-means. *HQcut* has the worst accuracy in these test cases, because of the intrinsic limitation of the modularity function. Furthermore, the accuracy of our algorithm is very close to optimal-kNN, indicating that the simple strategy for choosing the best k is fairly effective.

B. Real application in image clustering

For a real application, we applied our algorithm to cluster a set of face images taken from the UCI KDD Archive (data set name: CMU face images). The data set consists of 624 black and white face images of 20 people. Each people has 28 to 32 images, taken with varying pose (straight, left, right, up), expression (neutral, happy, sad, angry), eyes (wearing sunglasses or not), and size. Figure 3(a) shows some example face images for 4 people. Some images for the same people look quite different, even to human eyes. We therefore ask whether it is possible to automatically separate the images for each person into a single cluster.

To apply our algorithm, we first converted each image, which has size 64×60, into a vector of 3840 dimensions. We therefore obtained a matrix of 624 rows and 3840 columns, where each row represents an image and each column represents a pixel. In image clustering, it is often useful to normalize the vectors before applying a distance measure to compare images. Many different normalization approaches exist and it is often not clear which normalization approach should be performed. As mentioned earlier, we can use the modularity measure to help us determine an appropriate normalization strategy.

We first clustered the images without any normalization. Second, we normalized the images so that each vector has a unit length. Third, we normalized each pixel using quantile normalization. With this normalization, the histogram of each pixel becomes identical, and therefore gives equal emphasis to all pixels during clustering. We applied our algorithm to the three matrices, and compared the identified clusters to the actual clusters, where the images of one individual are put in one cluster.

Interestingly, the three normalization methods resulted in very different modularity and, correspondingly, very different clustering accuracy (Table 1). The matrix without normalization and the one with length normalization both had a modularity 0.62, and clustering accuracy 0.54 and 0.53, respectively. In comparison, the quantile normalization resulted in much higher modularity and clustering accuracy.

The optimal kNN graph, constructed with $k = 32$ on the quantile normalized data, is shown in Figure 3(b), where different clusters identified by our algorithm is shown with different colors. The nodes in yellow are not in a same cluster; rather, they highlight the ones that have been misclustered. On the top left corner, a small group of photos was clustered in the gray cluster to its right, while it should belong to the green cluster below it. On the top right corner, a small set of photos was identified as a separate cluster, which actually contains photos belonging to the two neighboring clusters (red and blue, corresponding to the first and third person in Figure 3(a), respectively). Also, a photo on the bottom left corner was identified as a singleton. A closer inspection shows that the image is contaminated, possibly due to problematic camera setup. It is interesting that our algorithm correctly singled it out as an outlier.

To compare, we also tested several other algorithms for clustering the images. We first applied *k*-means to the quantile normalized matrix, to find 18 to 22 clusters. The best clustering accuracy 0.70 was obtained with 20 clusters. We also tested another algorithm, the Affinity Propagation algorithm (AP) [4]. This algorithm works on affinity matrices. Therefore, we computed the Euclidean distance matrix and converted it into an affinity matrix by multiplying each value by −1. The algorithm also depends on a single parameter called *prior*, which implicitly determines the number of clusters that will be generated. We varied a wide range of values for prior. The best clustering accuracy we obtained with AP is 0.82. Finally, we applied the Markov Clustering algorithm (MCL), another popular graph clustering algorithm [3]. This algorithm works on sparse graphs, and requires a single parameter, inflation. Therefore, we take the optimal network returned by our algorithm, and varied the inflation parameter from 1.2 to 2.0, with an incremental at 0.05. The best clustering accuracy, 0.86, is achieved at I = 1.8. This indicates that (1) the optimal network

returned by our algorithm captures the modularity of the data well, so that the MCL algorithm can also successfully identify the correct clusters, and (2) the community structure identified by out algorithm without any parameter tuning is possibly optimal, with an accuracy comparable to that of MCL which requires extensive parameter tuning.

IV. Conclusions

In this work, we have introduced a very simple yet robust method for identifying communities from high dimensional data and affinity matrices. By constructing a mutual kNN network, our method is able to detect clusters of various sizes, shapes and densities, and is robust to noises. As the construction of the network is integrated with a modularity-based community discovery algorithm, the optimal parameter for network construction is determined automatically. In addition, using absolute modularity as a criterion, our method can be used to suggest the most appropriate preprocessing/normalization for the data.

We tested our methods on a large number of synthetic and real data sets. Experimental results show that our method is highly effective. Even though it is fully automatic, it outperformed a number of existing algorithms that requires user-defined parameters.

Acknowledgments

This work was supported in part by a UTSA faculty research award and a NIH grant 1SC3GM086305-01 to JR.

REFERENCES

[1]. Newman M. The structure and function of complex networks. SIAM Review. 2003; 45:167–256.

[2]. Newman M, Girvan M. Finding and evaluating community structure in networks. Phys Rev E. 2004; 69:026113.

[3]. Dongen SV. Graph clustering via a discrete uncoupling process. SIAM Journal on Matrix Analysis and Applications. 2008; 30:121–141.

[4]. Frey BJJ, Dueck D. Clustering by passing messages between data points. Science. 2007; 315:972–976. [PubMed: 17218491]

[5]. Newman M. Modularity and community structure in networks. Proc Natl Acad Sci USA. 2006; 103:8577–82. [PubMed: 16723398]

[6]. Danon L, et al. Comparing community structure identification. J. Stat. Mech. 2005:P09008.

[7]. Guimera R, Amaral LN. Functional cartography of complex metabolic networks. Nature. 2005; 433:895–900. [PubMed: 15729348]

[8]. Ruan J, Zhang W. Identifying network communities with a high resolution. Physical Review E. 77(1):2008.

[9]. Fortunato S, Barthelemy M. Resolution limit in community detection. Proc Natl Acad Sci USA. 2007; 104:36–41. [PubMed: 17190818]

[10]. Muff S, Rao F, Caflisch A. Local modularity measure for network clusterizations. Physical Review E. 2005; 72:056107.

[11]. Arenas A, Fernández A, Gomez S. Analysis of the structure of complex networks at different resolution levels. New Journal of Physics. 2008; 10:053039–+.

[12]. Jain AK, Murty MN, Flynn PJ. Data clustering: A review. ACM Comput. Surv. 1999; 31:264–323.

[13]. Ester, M.; peter Kriegel, H.; S, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. AAAI Press; 1996. p. 226-231.

[14]. Karypis G, Han E-HS, Kumar V. Chameleon: Hierarchical clustering using dynamic modeling. Computer. 1999; 32:68–75.

[15]. Chaoji, V.; Al Hasan, M.; Salem, S.; Zaki, M. Sparcl: Efficient and effective shape-based clustering; Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on; 2008; p. 93-102.

[16]. Ng A, Jordan M, Weiss Y. On spectral clustering: Analysis and an algorithm. NIPS. 2001:849–856.

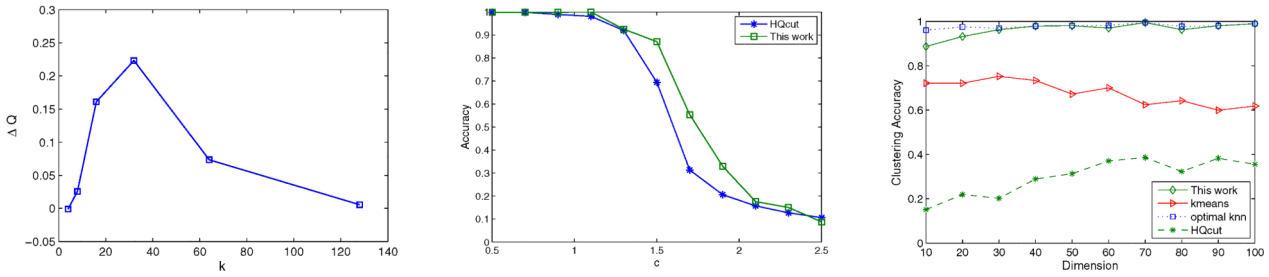
[17]. Maier M, Hein M, von Luxburg U. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. Theor. Comput. Sci. 2009; 410(19):1749–1764.

[18]. Guimera R, Sales-Pardo M, Amaral LN. Modularity from fluctuations in random graphs and complex networks. Physical Review E. 2004; 70:025101.

[19]. Meila, M. ICML '05: Proceedings of the 22nd international conference on Machine learning. ACM Press; New York, NY, USA: 2005. Comparing clusterings: an axiomatic view; p. 577-584.

for i = 1 to $\lfloor \log_2 n \rfloor$ **do**
  $k = 2^i$;
  Construct a graph G_k = *kNN*(S, k);
  Run *Qcut* to find the optimal community structure for G_k, denoted by $C(G_k)$;
  Randomize G_k to obtain G_k^r;
  Run *Qcut* to find the optimal community structure for G_k^r, denoted by $C(G_k^r)$;
  $\Delta Q_k = Q(C(G_k)) - Q(C(G_k^r))$;
end for
$k^* = \arg\max_k \Delta Q_k$;
G^* = *kNN*(S, k^*);
Apply *HQcut* to G^* to obtain final community structure.

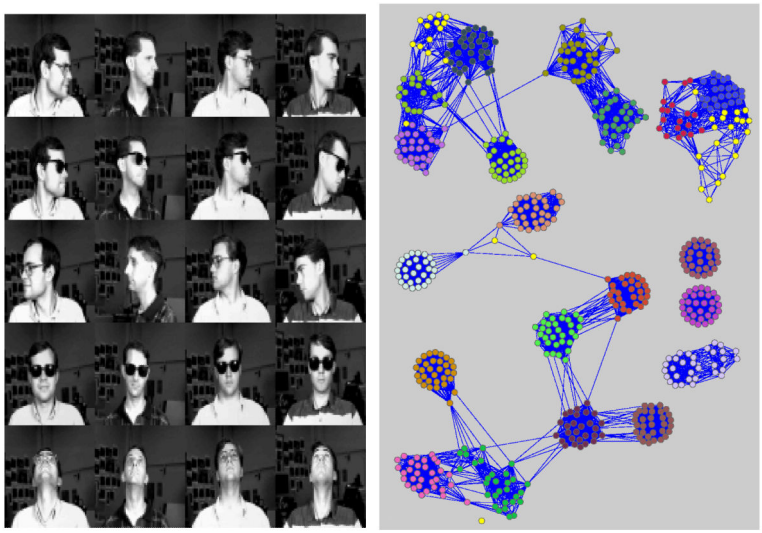
Figure 1.
Pseudocode of our algorithm.



(a) (b) (c)

Figure 2.

Results on synthetic data. (a) Modularity as a function of *k* for a synthetic affinity matrix. (b) Clustering accuracy on synthetic affinity matrices. (c) Clustering accuracy on synthetic high dimensional data points having irregular shapes. In (b) and (c), each point is an average of 10 data sets.



(a) (b)

Figure 3.

Clusters among face images. (a) Some example face images. (b) A kNN graph connecting all face images, and the identified community structure. Best viewed in color.

Table I

Effect of Normalization to Image Clustering

Normalization metod	Modularity	Accuracy
None	0.62	0.54
Length normalization	0.62	0.53
Quantile normalization	0.74	0.86