Chapter 6
Network-Based Unsupervised Learning

Abstract In this chapter, we present representative state-of-the-art unsupervised learning techniques that rely on networked environments to conduct the learning process. In a typical unsupervised task, no external knowledge is presented to the algorithm. As such, the learning process is guided by the provided data, since no prior knowledge about the existing groups is supplied. For network-based methods, the learning procedure is performed by navigating in networks that are constructed from the input data set according to some similarity criterion. As networks naturally embody topological information of data relationships, network-based methods take advantage over algorithms that make use of raw, vector-based data. Moreover, network-based methods can be conceived as a general solution for unsupervised learning tasks even for data sets that are not represented by networks. In this case, we can apply network formation techniques on that data set to generate a network from the input data. Once the network is constructed, all of the network-based techniques described in this chapter can effectively be employed.

6.1 Introduction

In this chapter, we shift our attention to network-based unsupervised methods. The data representation as networks enables us to systematically investigate the topology and function of data relationships using well-understood graph-theoretical concepts that can be employed to uncover structural and dynamical properties of the underlying constructed network.

One of the main tasks of unsupervised learning is data clustering. In essence, data clustering can be considered as a community detection problem once a network is constructed from the original data set. In this transformation, each vertex corresponds to a data item and connections are established according to a certain similarity measure. The clusters in a community detection task are often denominated communities. A community is defined as a subgraph whose vertices are densely connected within itself, but sparsely connected with the remainder of the network. Figure 6.1 illustrates typical processes in data clustering and in community detection. In the former, unstructured or raw data are received by a data clustering procedure that finds similar groups in accordance with a similarity criterion. In the latter, the community detection procedure uncovers communities in the network.

T.C. Silva, L. Zhao, *Machine Learning in Complex Networks*,
DOI 10.1007/978-3-319-17290-3_6

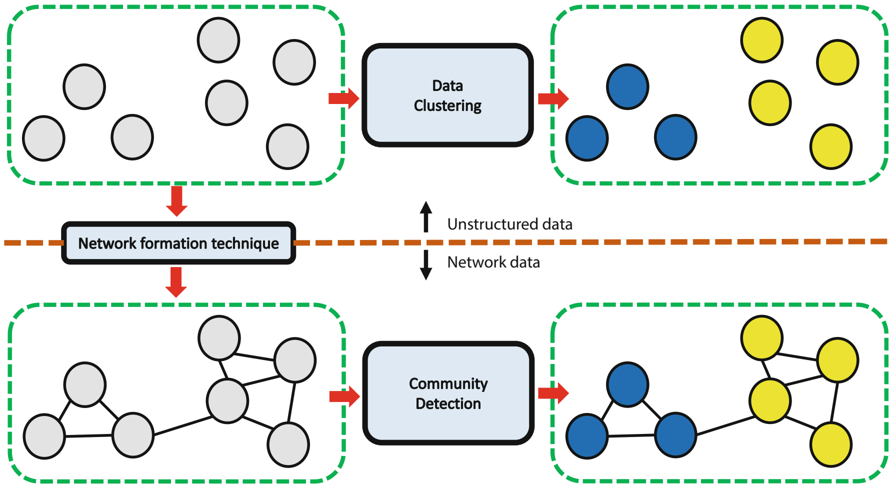

Fig. 6.1 Similarities between data clustering and community detection tasks. The *dotted horizontal line* represents the frontier of unstructured data and networked data. A network formation method interfaces between unstructured and networked data. Note that each of the data items is represented by a vertex in the networked domain

Topological information of the data, such as direct or indirect neighborhoods, can be readily employed by the community detection method. Observe that the network formation method serves as interface between unstructured and networked data.

Network-based methods are specially useful when we deal with clusters of arbitrary shape, proximity, orientation, and varying densities [36]. Since in unsupervised learning methods we usually do not know how the clusters are shaped nor how many of them exist, network-based methods stand as good candidates for tasks related to data clustering. Consider that we use as input the data set depicted in Fig. 6.2a in the schematic shown in Fig. 6.1. For the data clustering method in unstructured data, we choose the well-known K-Means procedure with a number of clusters calibrated to 2. For the community detection task in networked data, we use the Chameleon technique [36], which is a network-based unsupervised learning method that we discuss in this chapter. We employ the k-nearest neighbor technique with $k = 7$ as the network formation technique that interfaces between unstructured and networked data. The clustering result for the K-Means technique is displayed in Fig. 6.2b, while the outcome of Chameleon is portrayed in Fig. 6.2c. While K-Means has difficulty in clustering arbitrary-shaped clusters due to its strong bias on circular-shaped items, network-based methods can provide robust results as they are guided by the network topology in the learning process. This is because network-based methods use the network topology to derive its decisions, in a way that we do not need assumptions about the data distribution nor about the number of clusters or communities. Consequently, we prevent the insertion of wrong biases over the data distributions that can severely hamper the quality of the learning process.

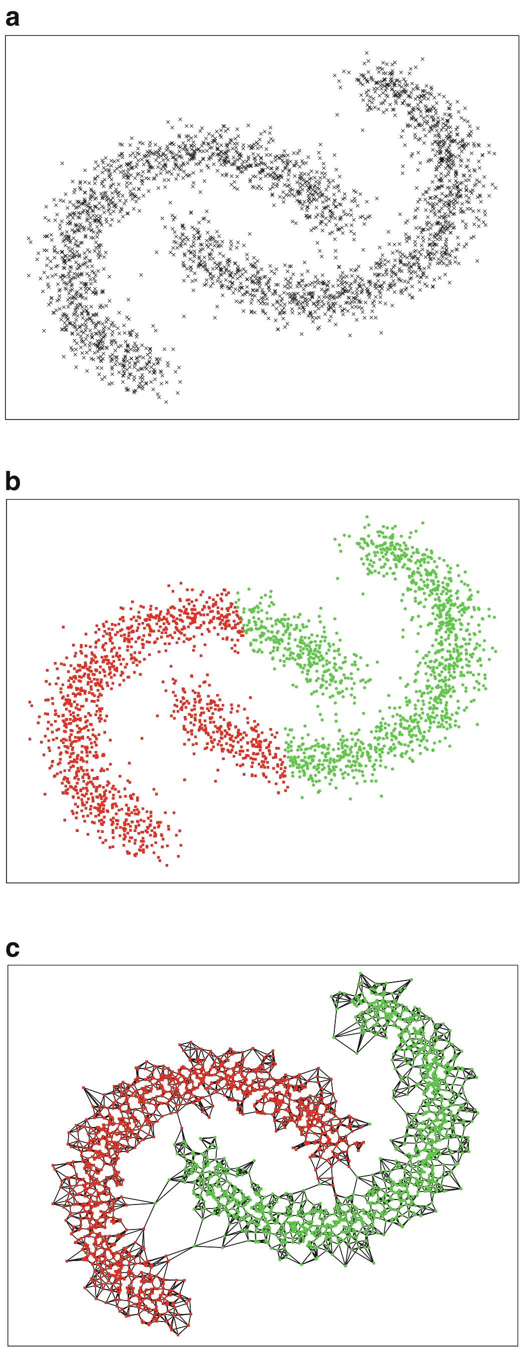

Fig. 6.2 Comparison of vector- and network-based methods in data clustering and community detection tasks, respectively. We use the K-Means algorithm with $K = 2$ in Fig. 6.2b. In Fig. 6.2c, we first construct the network from the unstructured data in Fig. 6.2a using $k = 7$ and then apply the Chameleon. (**a**) Initial state (vector-based data); (**b**) Results for vector-based learning method; (**c**) Results for network-based learning method

6.2 Community Detection

In this section, we introduce the main concepts of community detection, as well as a brief description of the related state-of-the-art techniques. In addition, we present some broadly accepted community detection benchmarks.

6.2.1 Relevant Concepts and Motivations

Complex networks are found in fields as diverse as the Internet, the World Wide Web, food webs, and biological and social organizations [7]. Even though the main features of complex networks have been properly described at the microscale level, such as strict-local properties of network vertices, and also at the macroscale level, such as global properties of the entire network, some of the characteristics lying at a mesoscale level are still elusive.

Nonetheless, modern science related to networks brought a substantial advance in understanding complex networks. One of the features evident and prominent in complex networks is the presence of mesoscale structures called *communities*. These communities can carry functional, relational, or even social common concepts. Though the formal definition of a community is controversial in the literature, the essence of a community is straightforward: each community is defined as a subgraph whose vertices are densely interconnected, and, at the same time, these vertices have few links with the remainder of the network. Figure 6.3 portrays a network in which four well-defined communities can be observed, because the quantity of edges between members of the same community is perceptively larger than the number of edges connecting different communities. The community detection task in complex networks has become an important topic in graphs and data mining [16, 22, 57]. In graph theory, community detection corresponds to the graph partitioning problem, which is an NP-complete problem [22].

The study of community detection is very important for understanding various phenomena in complex networks [32]. Modular structure introduces important heterogeneities in complex networks. Each module, for example, can have different local statistics [55]; some modules may have many connections, while other modules may be sparse. When there is large variation among communities, global values of statistical measures can be misleading. The presence of modular structure may also alter the way in which dynamical processes (e.g. spreading processes and synchronization [3]) unfold on the network. In biological networks, communities correspond to functional modules in which module members function coherently to perform essential cellular tasks. Both metabolic networks [69] and protein phosphorylation networks [35], for instance, have modular structures.

A promising computational approach to discovering functions of genes and proteins is to identify functional modules in biological networks. Since modules are sets of genes or proteins that perform biological processes together, it is

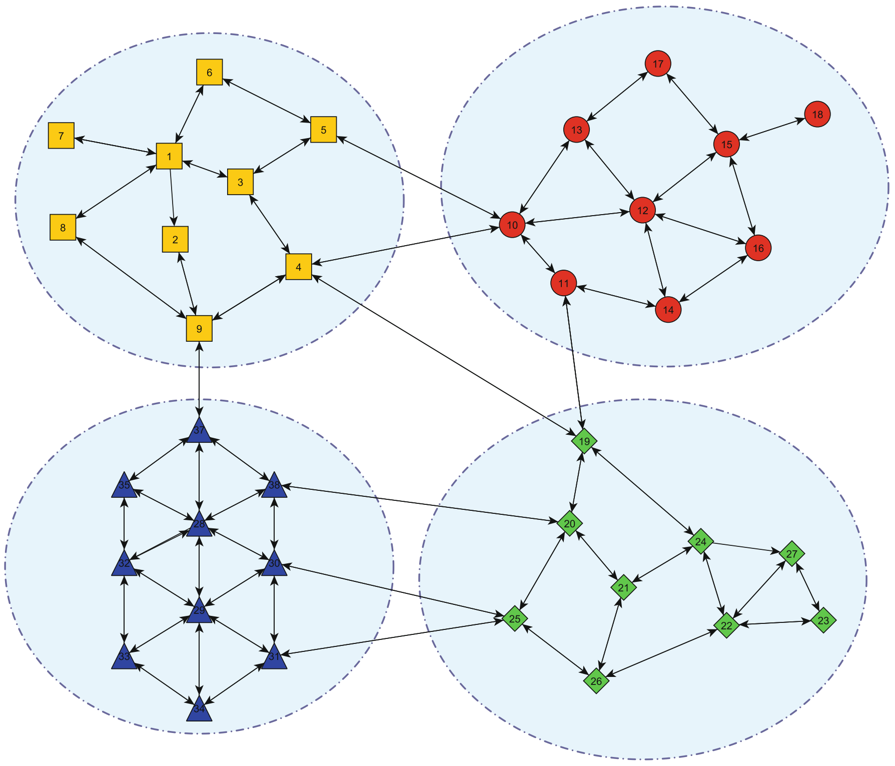

Fig. 6.3 A network that presents four well-defined communities. The *different vertices' colors or formats* denote the communities to which each of them belong

possible to classify proteins with unknown functions by determining to what module they belong [63]. Correct identification of functional modules has also important biotechnological and drug design applications. In many cases, the deletion of a certain function may be necessary and this can be achieved by removing the entire functional module.

Several distinct ways of detecting modules in complex networks have been proposed [22]. One popular approach considers communities as sets of adjacent motifs [63], other methods are inspired by information theory [71], message passing [26], or Bayesian principles [34, 58]. A widely used class of algorithms is based on the optimization of a quantity called modularity [57].

Another important aspect related to community structure is of the hierarchical organization displayed by most networked systems in the real world [22]. Real networks are usually composed of communities including smaller communities, which in turn include even smaller communities, and so on. The human body offers a paradigmatic example of hierarchical organization: it is composed by organs, organs are composed by tissues, tissues by cells, etc. Another example is represented by business firms, which are characterized by a pyramidal organization, going from the workers to the president, with intermediate levels corresponding to work groups, departments and management. Other example is the network formed by all human acquaintances. While at a local scale we expect to find many communities formed by families and friends, on a larger scale, the expected communities turn into cities, regions, followed by countries, and, finally, probably continental areas. The generation and evolution of systems organized in interrelated stable subsystems are much quicker than unstructured systems. One evidence corroborating this fact is that it is much easier to assemble the smallest sub-parts of a structured system first and then use them as building blocks to build up larger structures, until the entire system is assembled. In view of these examples, it is clear that the study of community presence in networks plays an important role in understanding natural concepts encountered in various branches of science.

Another interesting topic is of overlapping communities. We have seen that the identification of modules and their boundaries enables us to classify vertices according to their structural positions in those modules. So, vertices with a central position in their clusters, i.e., which share large numbers of edges with other group partners, may have important functions of control and stability within the group. Notwithstanding, vertices lying at the boundaries between these modules also play an important role of mediation and lead the relationships and exchanges between different communities. These kinds of vertices are termed as overlapping vertices [22].

Formally, overlapping vertices are defined as those vertices that are members of more than one community or class at the same time [63]. For example, in a network of semantic association concepts [38], the term "brilliant" may be a member of several classes, such as the one representing the concepts related to "light," to "astronomy," "color," and so on [63]. In a social network, each person naturally belongs to the company where he/she works and also to the group representing the members of his/her family. Given this scenario, the discovery of overlapping vertices and communities is important for data analysis in general.

6.2.2 Mathematical Formalization and Fundamental Assumptions

Unsupervised learning methods are guided exclusively by the intrinsic structure of the data items throughout the learning process, i.e., without any sort of external knowledge. Consider that $\mathscr{X} = \{x_1, x_2, \ldots, x_N\}$ is a data set, where $N = |\mathscr{X}|$ is the total number of data items involved in the learning process. Techniques that are members of the network-based unsupervised learning paradigm always accept as input a network. In this respect, we can face the following scenarios:

- The items in the data set are already in the network format, i.e., the vertex set $\mathscr{V}$ coincides with the set of data items $\mathscr{X}$ and the set of edges $\mathscr{E}$ is given. In this case, no preprocessing is needed. Well-known examples that already are in the form of networks include: WWW, Internet, transport and financial networks. Data sets of this type are inherent candidates to serve as input to network-based unsupervised learning methods.
- The items in the data set are presented in a raw, vector-based format. Normally, $\mathscr{X} = \mathscr{V}$, but we can also use compacted or expanded sets of $\mathscr{X}$ to build up $\mathscr{V}$. The edge set $\mathscr{E}$ is unknown and must be estimated using a network construction technique. Normally, the set of edges is constructed according to some similarity criteria that are imposed by the network construction process. Figure 6.1 illustrates this process. In Chap. 4, we have presented several manners to deal with this problem. Here, we assume that there exists such a function of network formation technique that simply transforms the vector-based format to a network.

Suppose the network $\mathscr{G} = \langle \mathscr{V}, \mathscr{E} \rangle$ is obtained from the input data items. Then, the unsupervised learning problem is now posed in a network-based form. Recall that data clustering turns into a community detection task when the network structure of the data distribution is well-conditioned.

Though intuitive at first sight, the problem of community detection is actually not well defined. The main elements that make up the community detection task *per se*, that is, the concepts of community and partition, are not rigorously defined. In view of that, one must accept some degree of arbitrariness or common sense [22]. In fact, some ambiguities are hidden and there are often many equally legitimate ways of resolving them. It is not surprising, thus, that there are plenty of recipes in the literature and that people do not even try to ground the problem on shared definitions.

One point that is at least common sense in the literature is of the identification of the structural constraint for the existence of communities. In this regard, the existence of structural and well-defined communities is only possible when graphs are sparse. Sparseness arises when the number of edges E is of the order of the number of vertices V in non-weighted graphs, i.e. $E = \mathscr{O}(V)$. If $E \gg V$, the distribution of edges among the vertices is too homogeneous for communities to make sense. In this case, the problem turns into something rather different, close to data clustering, as the network structure does not convey relevant information to identify the community structures. The main difference between a community detection and data clustering task is that, while communities in graphs are related, explicitly or implicitly, to the concept of edge density (inside versus outside the community), in data clustering communities are sets of points which are "close" to each other, with respect to a measure of distance or similarity, defined for each pair of points [22].

6.2.3 *Overview of the State-of-the-Art Techniques*

Given that the task of accurately solving a problem of community detection is NP-complete, many efforts have been expended towards the development of approximate and efficient solutions. Some of these solutions include the spectral method [54], the betweenness-based technique [57], modularity greedy optimization [52], detection of communities based on the Potts model [70], synchronization [3], information theory [24], and random walks [92]. A thorough review on this topic is presented in [22].

Regarding the techniques which aim at detecting overlapping vertices and communities, various methods have been proposed in the literature [19, 43, 59, 63, 77, 79, 90]. In the research in [90], the authors combine the idea of the modularity function Q, spectral relaxation, and fuzzy C-Means clustering in order to build a new modularity function based on a generalized Newman and Girvan's Q function, which is an approximate mapping of the network vertices into the Euclidean space. In the study in [63], the community structure is uncovered by means of a k-clique percolation and the overlaps among communities are guaranteed by the fact that one vertex can participate in more than one clique. However, the k-clique percolation method gives rise to an incomplete cover of the network, i.e., some vertices may not belong to any community. In addition, the hierarchical structure may not be revealed for a given k. In contrast, the investigation in [43] introduces an algorithm that concomitantly finds both overlapping communities and the hierarchical structure based on a fitness function and a resolution parameter. In turn, the research in [19] proposes a method to recognize the overlapping community structure by partitioning a graph built from the original network. A perceptive drawback of the majority of these techniques resides in the fact that the detection of the overlapping characteristics of the input network is performed as a separated or dedicated process apart from the standard community detection technique. In this way, additional computational time is required. As a result, the whole process may have high computational complexity.

6.2.4 *Community Detection Benchmarks*

In this section, we introduce two community detection benchmarks, which are frequently used for comparing different competing techniques.

Benchmark of Girvan and Newman [28] This benchmark uses an agglomerative method that groups V initially isolated vertices into M communities. This is managed by creating links between two vertices with probability p_{in}, if they belong to the same community, or with probability p_{out}, if they belong to distinct communities. The values of p_{in} and p_{out} can be arbitrarily chosen to control the number of intracommunity and intercommunity links, z_{in} and z_{out}, respectively, for an arbitrary average network degree $\bar{k}$. On the basis of these parameters, we are able to define the fraction of intracommunity links $z_{\text{in}}/\bar{k}$ and, likewise, the fraction of intercommunity links $z_{\text{out}}/\bar{k}$. The quantity $z_{\text{out}}/\bar{k}$ defines the mixture of the communities, i.e., as $z_{\text{out}}/\bar{k}$ increases, the communities become more mixed and harder to be identified.

The benchmark works by varying the mixture of communities, i.e., $z_{\text{out}}/\bar{k}$, for a fixed network comprising V vertices and M communities. For each run, the community detection accuracy is registered. After all of the runs have been properly performed, a curve is plotted in a two-dimensional graph. This curve serves the purpose of comparing the community detection performance of a control algorithm in relation to competing techniques.

Benchmark of Lancichinatti et al. [42] The Girvan-Newman's benchmark in its original form suffers from several drawbacks, among which we can highlight:

- Each community has necessarily a random network topology. Therefore, the vertices have similar degrees and therefore have trivial link relationships; and
- Communities are forced to be of the same size.

Motivated by the fact that real-world networks are characterized by heterogeneous distributions of vertex degree, whose tails often decay as power laws, the benchmark of Lancichinatti et al. generates artificial networks with properties that overcome the size homogeneity of communities and the random network topology of the Girvan-Newman's benchmark.

The constructed networks assume that both degree and community size distributions follow a power law function, with exponents γ and β, respectively. Typical values of real-world networks are: $2 \leq \gamma \leq 3$ and $1 \leq \beta \leq 2$. Moreover, a mixing parameter μ is employed to interconnect communities in the following manner: each vertex shares a fraction $1 - \mu$ of its links with other vertices of the same community and a fraction μ with vertices of other communities.

The benchmark process consists in varying the mixing parameter μ and evaluating the normalized mutual information index, which is a similarity measure of partitions borrowed from the information theory [16] that measures the mutual dependence of different random variables.

6.3 Representative Network-Based Unsupervised Learning Techniques

In the following, we present representative techniques that are members of the network-based unsupervised learning.

6.3.1 Betweenness

A natural strategy to identify communities in a network is to detect and subsequently remove those edges that connect vertices of different communities, so that the communities eventually get disconnected from each other. In this case, the number of network components represents the number of communities. This is the philosophy of divisive algorithms. The crucial point resides in finding useful properties of intercommunity edges that could allow for their identification.

The most popular algorithm is that proposed by Girvan and Newman [28, 57]. In the edge removal process, the algorithm selects edges according to the values of edge centrality, estimating the importance of edges according to some property or process running on the network. The steps of the algorithm are:

1. Computation of the centrality for all of the edges;
2. Removal of the edge with the largest centrality: in case of ties with other edges, one of them is picked at random;
3. Recalculation of centralities on the modified network (network without that removed edge);
4. Iteration of the cycle from Step 2.

Girvan and Newman focused on the concept of betweenness, which is a variable expressing the frequency of the participation of edges to a process. They considered three alternative definitions: geodesic edge betweenness, random-walk edge betweenness and current-flow edge betweenness. In the following we shall refer to them as edge betweenness, random-walk betweenness and current-flow betweenness, respectively.

The betweenness of an edge is the number of shortest paths between all of the vertex pairs that run along that edge. It is an extension to edges of the popular concept of site betweenness, introduced by Freeman in 1977 [25] and expresses the importance of edges in processes like information spreading, where information usually flows through shortest paths. It is intuitive that intercommunity edges have large values of edge betweenness, because many shortest paths connecting vertices of different communities pass through them. As in the calculation of vertex betweenness, if there are two or more geodesic paths with the same endpoints that run through an edge, the contribution of each of them to the betweenness of the edge must be divided by the multiplicity of the paths, as one assumes that the signal/information propagates equally along each geodesic path.

In random-walk betweenness, one could imagine that signals flow across random rather than geodesic paths. In this case, the betweenness of an edge is given by the frequency of passages of a random walker across that edge. A random walker moving from a vertex follows each adjacent edge with equal probability. The algorithm works by first choosing a pair of vertices at random, say $s \in \mathscr{V}$ and $t \in \mathscr{V}$. The walker starts at s and keeps moving until it finally reaches t, where it stops. We then compute the probability that each edge in the network is crossed by that random walker. We perform this process for every given pair of network vertices s and t and take the average values. In this process, it is meaningful to compute the net crossing probability, which is proportional to the number of times the walk crossed an edge in one direction. In this way one neglects back and forth passages that are accidents of the random walk and tell us nothing about the centrality of that edge.

In current-flow betweenness, the network is considered as a resistor network, with edges having unit resistance. If a voltage difference is applied between any two vertices, each edge carries some amount of current, that can be calculated by solving Kirchoff's equations. The procedure is repeated for all of the possible vertex pairs: the current-flow betweenness of an edge is the average value of the current carried by the edge. It is possible to show that this measure is equivalent to random-walk betweenness, as the voltage differences and the random walks net flows across the edges satisfy the same equations [53].

In practical applications, the Girvan-Newman algorithm with edge betweenness gives better results than adopting the other centrality measures and is also much faster to compute than current-flow or random walk betweenness [51]. Nevertheless, the algorithm is still quite slow and is not applicable to large-scale graphs. In the original version of the Girvan-Newman algorithm [28], the authors had to deal with the entire hierarchy of partitions, as they had no procedure to say which partition is the best. In a successive refinement [57], they incorporate the process of selecting the best partition into the algorithm by employing the largest value of modularity.

Chen and Yuan [10] pointed out that considering all of the possible shortest paths in the evaluation of the edge betweenness may lead to unbalanced partitions, with communities of very different sizes. In order to overcome that problem, they proposed to count only non-redundant paths, i.e. those paths whose endpoints are all different from each other: the resulting betweenness yields better results than standard edge betweenness for mixed clusters on the benchmark graphs of Girvan and Newman.

6.3.2 Modularity Maximization

The scientific community considers the modularity algorithm as a seminal work in community detection. This class of algorithms relies on the fact that maximizing modularity is a good strategy for obtaining well-established communities. Before we discuss some representative methods that maximize modularity, we first recap the concept of network modularity, which has already been introduced in Definition 2.50.

The modularity measure quantifies how good a particular division of a network is [13, 55] and is designed to measure the strength of division of a network into modules (also called groups, clusters or communities). Generally, it ranges from 0 to 1. When the modularity is near 0, it means that the network does not present community structure, suggesting that the links are disposed at random in the network. As the modularity grows, the community structure gets more and more defined, that is, the mixture between communities gets smaller and therefore the fraction of links inside communities is larger than that between different communities. Mathematically, the network modularity is given by:

$$Q = \frac{1}{2E} \sum_{i,j \in \mathscr{V}} \left(\mathbf{A}_{ij} - \frac{k_i k_j}{2E} \right) \mathbb{1}_{[c_i = c_j]}, \tag{6.1}$$

in which E represents the total number of edges in the network; $\mathbf{A}_{ij}$ indicates the edge weight linking i to j; k_i stands for the degree of the vertex i; c_i is the community of vertex i; and $\mathbb{1}_{[c_i=c_j]}$ indicates the Kronecker's Delta or the indicator function, which produces 1 if $c_i = c_j$ and 0, otherwise. Essentially, the modularity captures how well the network structure fits to a given set of communities. In the computation, random chances are canceled out by subtracting the edge quantity that is expected within a community from an equivalent random network.

Modularity has been used to compare the quality of the partitions obtained by different methods, but has also been used as an objective function to be optimized [52]. Unfortunately, exact modularity optimization is a problem that is computationally hard [6] and so approximation algorithms are necessary when dealing with large networks.

The first proposed method to perform modularity optimization was done by Clauset et al. [13]. Since then, several other versions have been proposed [6, 11, 31, 66, 83]. The greedy algorithm proposed by Clauset et al. may produce modularity values that are significantly lower than what can be found by using, for instance, simulated annealing [31]. Moreover, the method proposed in [13] has a tendency to produce super-communities that contain a large fraction of the vertices, even on synthetic networks that have no significant community structure. This artefact also has the disadvantage to slow down the algorithm considerably and makes it inapplicable to networks of more than a million vertices. The Louvain method [6] is the fastest modularity optimization algorithm proposed so far. In addition, the mechanism underlying the Louvain algorithm circumvents the undesired effect of unbalanced communities encountered in Clauset et al. by introducing tricks in order to balance the size of the communities being merged, thereby speeding up the running time and making it possible to deal with networks that have a few million vertices.

In the following, we first discuss the traditional modularity optimization method proposed by Clauset et al. [13] and then the Louvain method [6].

6.3.2.1 Clauset et al. Algorithm

At each time step of the modularity maximization, the algorithm of Clauset et al. [13] chooses to merge two communities that lead to the largest increase in the modularity Q, i.e., it finds the largest modularity increment ΔQ. In the initial step, the increment in the network modularity if communities i and j are joined is:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2E} - \frac{k_i k_j}{(2E)^2}, & \text{if } i \text{ and } j \text{ are connected.} \\ 0, & \text{otherwise.} \end{cases} \tag{6.2}$$

Two communities, say i and j, are merged, in such a way that their merge causes the largest increment (or the least decrement) of the modularity at a particular step. The algorithm is agglomerative and each vertex represents a community in the initial configuration. If one wants to stop the merges when the network configuration reaches its maximum modularity, one can use the stop criterion as follows: once a negative increment is encountered in this greedy process, the maximum global value associated to the modularity has been reached and subsequent merges will only monotonically decrease the modularity of the network. Therefore, by looking at the signal of ΔQ_{ij} at each iteration, it is sufficient to know when to stop merging. In addition, no restrictions on the communities to be merged are specified by the original model.

A major advantage of the modularity greedy algorithm is that no model selection is required, as no parameters need to be adjusted. Moreover, we have a nice stopping criterion for the algorithm due to the behavior of the modularity curve.

A drawback of the original modularity algorithm is in its resolution limit. Several studies have shown that it is unable to detect very small communities [23, 39, 41]. Roughly speaking, the modularity compares the number of edges inside a community with the expected number of edges that one would find in the community if the network were a random network with the same number of vertices, each of which with the same degree, but with edges randomly reattached. This random null model implicitly assumes that each vertex can get attached to any other vertex of the network. Such assumption is however unreasonable if the network is very large, as the horizon of a vertex includes a small part of the network, ignoring most of it. Moreover, this null model implies that the expected number of edges between two groups of vertices decreases if the size of the network increases. So, if a network is large enough, the expected number of edges between two groups of vertices in the modularity's null model may be smaller than one. If this happens, a single edge between the two communities would be interpreted by modularity as a sign of a strong correlation between these two communities, and the modularity optimization procedure would lead to the merge of them, independently of the communities' features. So, even weakly interconnected complete graphs, which have the highest possible density of internal edges, and represent the best identifiable communities, would be merged by the modularity optimization process if the network is sufficiently large. For this reason, optimizing modularity in large networks would fail to identify small communities, even when they are well defined. This bias is inevitable for methods like modularity optimization, which rely on a global null model.

6.3.2.2 Louvain Algorithm

The Louvain algorithm [6] is divided into two phases that are repeated iteratively. Assume that we start with a weighted network of V vertices. First, we assign a different community to each vertex. So, in this initial partition, there are as many communities as there are vertices. Then, for each vertex i, we consider the neighbors j of i and we evaluate the gain of modularity that would take place by removing i from its community and by placing it in the community of j. Vertex i is then placed in the community for which this gain is maximum, but only if this gain is positive. If no positive gain is possible, vertex i stays in its original community. This process is applied repeatedly and sequentially for all of the vertices until no further improvement can be achieved. When the equilibrium is reached, the first phase of the Louvain algorithm is then complete. Note that a vertex may be, and often is, considered several times in this community flipping process. This first phase stops when a local maximum of the modularity is attained, i.e., when no individual move can improve the modularity. One should also observe that the output of the algorithm depends on the order in which the vertices are processed. Preliminary results on several test cases seem to indicate that the ordering of the vertices does not have a significant influence on the achieved maximum modularity. However, the ordering can influence the computation time. The problem of choosing an order is thus worth studying since it could give good heuristics to enhancing the computation time.

Part of the efficiency of the algorithm results from the fact that the gain in modularity ΔQ obtained by moving an isolated vertex i into a community m can easily be computed by:

$$\Delta Q = \left[\frac{\Sigma_{\text{in}} + s_{i,\text{in}}}{2E} - \left(\frac{\Sigma_{\text{tot}} + s_i}{2E}\right)^2\right] - \left[\frac{\Sigma_{in}}{2E} - \left(\frac{\Sigma_{\text{tot}}}{2E}\right)^2 - \left(\frac{s_i}{2E}\right)^2\right], \tag{6.3}$$

in which Σ_{in} is the sum of link weights inside community m, Σ_{tot} is the sum of link weights incident to vertices in community m, s_i is the sum of the link weights incident to vertex i (in-strength), $s_{i,\text{in}}$ is the sum of link weights from i to vertices in community m, and E is the sum of link weights in the network. A similar expression is used to evaluate the change of modularity when i is removed from its community. In practice, one therefore evaluates the modularity change by removing i from its community and then by moving it into a neighboring community.

The second phase of the algorithm consists in building a new network whose vertices are now the communities found during the first phase. To do so, the weights of the links between the new vertices are given by the sum of the link weights vertices in the corresponding two communities [4]. Links between vertices of the same community lead to self-loops for this community in the new network. Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate.

This simple algorithm has several advantages. First, the procedure is intuitive and easy to implement, and the outcome is unsupervised. Moreover, the algorithm is extremely fast, i.e., computer simulations on large ad-hoc modular networks suggest that its complexity is linear on typical and sparse data. This is due to the fact that the possible gains in modularity are easy to compute with the above formula and that the number of communities decreases drastically just after a few passes so that most of the running time is concentrated on the first iterations. The so-called resolution limit problem of modularity is also circumvented due to the intrinsic multi-level nature of the algorithm.

6.3.3 Spectral Bisection Method

Spectral graph theory is concerned with graph properties such as its characteristic polynomial, eigenvalues, and eigenvectors of matrices associated to the adjacency matrix or the Laplacian matrix of the graph. We define the spectrum of a finite graph $\mathscr{G}$ as the spectrum of the adjacency matrix $\mathbf{A}$, that is, its set of eigenvalues and their multiplicities together with the set of orthonormal eigenvectors. The Laplace spectrum of a finite undirected graph without loops is the spectrum of the Laplace matrix $\mathbf{L}$.

An undirected network with real-valued edges, for example, has a symmetric adjacency matrix and therefore has real eigenvalues. The set of all of these eigenvalues and the corresponding complete set of orthonormal eigenvectors make up the graph spectrum. While the adjacency matrix depends on the vertex labeling or ordering, its spectrum is graph invariant. The spectral bisection method is one type of algorithm that falls into this category.

Spectral methods for graph partitioning have been known to be robust but computationally expensive.

The use of spectral methods to compute cuts in graphs was first considered by Donath and Hoffman [18] who first suggested using the eigenvectors of adjacency matrices of graphs to find partitions. Fiedler [12] associated the second smallest eigenvalue of the Laplacian matrix with its connectivity and suggested partitioning the graph by splitting vertices according to their values in the corresponding eigenvector. Thus, the eigenvector corresponding to the second smallest eigenvalue (i.e., the algebraic connectivity) of the Laplacian matrix of a graph $\mathscr{G}$ is termed as the Fiedler vector, while the corresponding eigenvalue, the Fiedler value. Since then, spectral methods for computing and analyzing graph properties have received increasing attention by the community [2, 37, 56, 91].

In one of these spectral methods [54], the spectral bisection method defines the cut size R of a graph partition into two groups as:

$$R = \frac{1}{2} \sum_{i,j \in \mathscr{V}} \mathbf{A}_{ij} \mathbb{1}_{[c_i \neq c_j]}, \tag{6.4}$$

in which the indicator function makes sure that only those edges crossing different communities are considered in the computation of the cut size R.

Consider the index vector s, whose component s_i is $+1$ if vertex i is in one group and -1 if it is in the other group:

$$\mathbf{s}_i = \begin{cases} +1, & \text{if vertex } i \text{ belongs to group 1.} \\ -1, & \text{if vertex } i \text{ belongs to group 2.} \end{cases} \tag{6.5}$$

Then, R can be rewritten as:

$$R = \frac{1}{4} \sum_{i,j \in \mathcal{V}} (1 - s_i s_j) \mathbf{A}_{ij}. \tag{6.6}$$

As the degree of vertex i is $k_i = \sum_{j \in \mathcal{V}} \mathbf{A}_{ij}$, then we have $\sum_{i,j \in \mathcal{V}} \mathbf{A}_{ij} = \sum_{i \in \mathcal{V}} k_i = \sum_{i \in \mathcal{V}} s_i^2 k_i = \sum_{i,j \in \mathcal{V}} s_i s_j k_i \mathbb{1}_{[i=j]}$.

Then, R can be rewritten as:

$$R = \frac{1}{4} \sum_{i,j \in \mathcal{V}} s_i s_j (k_i \mathbb{1}_{[i=j]} - \mathbf{A}_{ij}). \tag{6.7}$$

In matrix form, we have:

$$R = \frac{1}{4} s^T \mathbf{L} s, \tag{6.8}$$

in which s^T is the transpose of s and $\mathbf{L} = k_i \mathbb{1}_{[i=j]} - \mathbf{A}_{ij}$ is the Laplacian matrix.

Let us write s as a linear combination of the orthonormal eigenvectors v_i of the Laplacian:

$$s = \sum_{i \in \mathcal{V}} a_i v_i, \tag{6.9}$$

in which $a_i = v_i^T s$. The normalization implies $s^T s = V$ and $\sum_{i \in \mathcal{V}} a_i^2 = V$, where V is the number of network vertices. Then, we have:

$$R = \sum_{i \in \mathcal{V}} a_i v_i^T \mathbf{L} \sum_{j \in \mathcal{V}} a_j v_j = \sum_{i,j \in \mathcal{V}} a_i a_j \lambda_j \mathbb{1}_{[i=j]} = \sum_{i \in \mathcal{V}} a_i^2 \lambda_i, \tag{6.10}$$

in which λ_i is the eigenvalue of $\mathbf{L}$ corresponding to the eigenvector v_i and we have made use of $v_i^T v_j = \mathbb{1}_{[i=j]}$.

Assume that the eigenvalues are labeled in increasing order $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_V$. The task of minimizing R can then be equivalently equated as the task of choosing nonnegative quantities a_i^2, in a way to place larger weights to the components that correspond to the smallest eigenvalues in the sum of R.

The sum of every row (and column) of the Laplacian matrix is zero: $\sum_{j\in\mathcal{V}} \mathbf{L}_{ij} = \sum_{j\in\mathcal{V}} (k_i \mathbb{1}_{[i=j]} - \mathbf{A}_{ij}) = \sum_{j\in\mathcal{V}} k_i - k_i = 0$. Thus, the vector $(1, 1, \ldots, 1)$ is always an eigenvector of the Laplacian with eigenvalue zero. The Laplacian is symmetric and hence its eigenvalues are all squares of real vectors, i.e., all eigenvalues of the Laplacian are nonnegative, i.e., $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_V$.

Since the eigenvectors are orthogonal, a good approximate solution can be obtained by choosing s to be as close to parallel with v_2 as possible, i.e., minimizing:

$$|v_2^T s| = |\sum_{i\in\mathcal{V}} v_i^{(2)} s_i| \leq \sum_{i\in\mathcal{V}} |v_i^{(2)}|. \tag{6.11}$$

A simple choice for defining the clusters ($+1$ or -1) is:

$$s_i = \begin{cases} +1, & \text{if } v_i^{(2)} \geq 0. \\ -1, & \text{if } v_i^{(2)} < 0. \end{cases} \tag{6.12}$$

6.3.4 Community Detection Using Particle Competition

This technique is proposed in [68]. The evolution of this model is very similar to various natural and social processes, such as resource competition, territory exploration by animals, election campaigns, etc. In this model, particles explore a network by combining roles of random and deterministic moving. The investigation of the behavior of this technique reveals that the introduction of a certain level of randomness can yield a big gain in the learning process. This phenomenon is analogous to stochastic resonance in which the performance of a nonlinear deterministic system can be largely enhanced by a certain level of noise. The study shows that learning techniques consisting of only deterministic rules are insufficient. This is because the number of rules required to completely describe even a very specific environment can be prohibitively high. In a dynamical environment, the situation gets worse because the system should keep acquiring new knowledge over time. In this way, a certain level of randomness or chaos is essential for the learning process. The random term models the "I don't know" state and serves as a novelty finder. It can also help learning agents, like particles in this model, in escaping from traps in the physical or learning spaces.

The technique relies on the competition of several particles in a networked environment to identify communities. These particles navigate in the network with the purpose of dominating new vertices, while also trying to defend their previous dominated territory. In the long run, the subsets of vertices that each particle dominates represent the communities.

Two dynamical variables for the j-th particle, denoted by ρ_j, are maintained:

- $\rho_j^v(t)$: it represents the vertex that particle ρ_j is visiting at time t.

- $\rho_j^{\omega}(t) \in [\omega_{\min}, \omega_{\max}]$: it indicates the exploration potential of particle ρ_j at time t, where $\omega_{\min}$ and $\omega_{\max}$ are scalars that define the minimum and maximum potential that each particle can reach in the learning process, respectively.

The update rules that govern the movement and the exploration potentials of the particles are given by:

$$\rho_j^{\omega}(t+1) = \begin{cases} \rho_j^{\omega}(t) & \text{if } v_i^{\rho}(t) = 0 \\ \rho_j^{\omega}(t) + (\omega_{\max} - \rho_j^{\omega}(t))\Delta_{\rho} & \text{if } v_i^{\rho}(t) = \rho_j \neq 0 \\ \rho_j^{\omega}(t) - (\rho_j^{\omega}(t) - \omega_{\min})\Delta_{\rho} & \text{if } v_i^{\rho}(t) \neq \rho_j \neq 0 \end{cases}, \tag{6.13}$$

in which Δ_{ρ} controls the exploration level variation that each particle gains or loses, depending on the nature of the vertex that it visits. Specifically, if it visits an already dominated vertex, then the particle's exploration level is strengthened; otherwise, it is decremented. The location of particle ρ_j at $t+1$, $\rho_j^{v}(t+1)$, is determined by sampling from a mixture of deterministic and random walk distributions.

Each vertex v_i in the network is represented by three scalar variables:

- $v_i^{\rho}(t)$: it defines the proprietary particle of the vertex v_i at time t.
- $v_i^{\omega}(t)$: it indicates the level of domination imposed by proprietary particle $v_i^{\rho}(t)$ on vertex v_i at time t.
- $v_i^{\gamma}(t)$: it symbolizes whether or not vertex v_i is being visited by any of the particles at time t.

With the help of these variables, the dynamical behaviors of the quantities related to the vertices in the network are governed by the following set of equations:

$$v_i^{\rho}(t+1) = \begin{cases} \rho_j & \text{if } v_i^{\gamma}(t) = 1 \text{ and } v_i^{\omega}(t) = \omega_{\min} \\ v_i^{\rho}(t) & \text{otherwise} \end{cases}, \tag{6.14}$$

$$v_i^{\omega}(t+1) = \begin{cases} v_i^{\omega}(t) & \text{if } v_i^{\gamma}(t) = 0 \\ \max\{\omega_{\min}, v_i^{\omega}(t) - \Delta_v\} & \text{if } v_i^{\gamma}(t) = 1 \text{ and } v_i^{\rho}(t) \neq \rho_j \\ \rho_j^{\omega}(t+1) & \text{if } v_i^{\gamma}(t) = 1 \text{ and } v_i^{\rho}(t) = \rho_j \end{cases}, \tag{6.15}$$

in which Δ_v denotes the exploration level fraction lost by a vertex, if a rival particle visits it.

The detection algorithm begins by putting K particles into random vertices. At the beginning of the dynamical process, each particle ρ_j and each vertex v_i have their potentials set to $\rho_j^{\omega}(0) = \omega_{\min}$ and $v_i^{\omega}(t) = \omega_{\min}$, respectively. At each iteration, each particle travels to a neighboring vertex, in accordance with a movement policy that consists in a combination of deterministic and random walks. In the former, the particle randomly visits the neighbors of the currently visited vertex. In the latter, the particle prefers to visit vertices that are already being dominated by the same particle. In the following, we illustrate cases that can be faced when a particle j, ρ_j, is on the process of choosing the next vertex to visit:

- If the visited vertex v_i still does not belong to any particles, then initially $v_i^{\rho}(t) = 0$. In this case, such vertex starts to be dominated by the visiting particle, i.e., $v_i^{\rho}(t) = \rho_j$. The particle's potential $\rho_j^{\omega}(t)$ is not altered and the vertex's potential receives the particle's potential: $v_i^{\omega}(t) = \rho_j^{\omega}(t)$;
- If the visited vertex is dominated by the same particle, the visiting particle's potential is incremented and v_i receives the new potential of that particle: $v_i^{\omega}(t) = \rho_j^{\omega}(t)$;
- If the visited vertex belongs to a rival particle, then the particle's and the vertex's potentials are weakened. If the particle's potential $\rho_j^{\omega}(t)$ reaches a value lower than $\omega_{\min}$, then this particle is reset to a new randomly chosen vertex. If the potential of the vertex $v_j^{\omega}(t)$ reaches a value lower than $\omega_{\min}$, then the vertex becomes no longer owned by the previous particle, i.e., it regresses to the free, non-dominated state: $v_j^{\omega}(t) = 0$.

Thus, the vertex's level of domination increases if it is visited by the same particle that dominates it at the present moment. In contrast, during the visit of a rival particle, the domination level imposed by the current dominating particle on that vertex is weakened. If this domination is not strong enough, the proprietary particle loses its domination over that vertex. In the long run, it is expected that each particle will dominate a community in the network.

The model proposed in [68] has two noticeable features: (1) high community detection rates and (2) low computational complexity. However, in its original form, only a procedure of particle competition is introduced, without any formal definitions. This precludes any further analyses or predictions on the model's behavior. In Chap. 9, we show a rigorous model for particle competition that is governed by a stochastic competitive dynamical system. That same model is also adapted to a semi-supervised learning environment in Chap. 10, where we also investigate the relevant problem of imperfect learning.

6.3.5 Chameleon

This is a well-known method in the network-based community for data clustering [36]. In general, existing clustering algorithms use static models of the clusters and do not use information about the nature of individual clusters as they are merged or divided. Furthermore, while some schemes ignore the information about the aggregate interconnectivity of data items in two clusters, other schemes ignore information about the closeness of two clusters as defined by the similarity of the closest items across two clusters. By only considering either interconnectivity or closeness, these algorithms can easily select and merge the wrong pair of clusters.

Chameleon is an agglomerative hierarchical clustering algorithm that employs both interconnectivity and closeness features in identifying the most similar pairs of clusters. It is designed to overcome the major limitation of learning methods that assume a static, user-supplied interconnectivity model. Such models are inflexible and can easily lead to incorrect merge decisions when the model under- or overestimates the interconnectivity of the data set or when different clusters exhibit different interconnectivity characteristics. For that, Chameleon uses a combined approach to model the degree of interconnectivity and closeness between each pair of clusters. This approach considers the internal and adaptive characteristics of the clusters themselves. Thus, it does not depend on a static, user-supplied model and can automatically adapt to the internal characteristics of the merged clusters.

Given a vector-based data set, Chameleon first constructs a network using the k-nearest neighbors method, that is, each data sample is represented by a vertex and it is connected to the other k most similar data samples using a similarity metric. Then, Chameleon finds the initial partition of the network using an algorithm that partitions the network into several communities in a way to minimize the edge cut. Since each edge in the k-nearest neighbor graph represents the similarity among data points, a partitioning that minimizes the edge cut effectively minimizes the relationship (affinity) among data points across the partitions. After finding subclusters, Chameleon switches to an algorithm that repeatedly combines these small subclusters, using the cluster similarity measures, which determine the similarity between pairs of clusters by looking at their Relative Interconnectivity (RI) and Relative Closeness (RC). The definitions of these two internal indices are given in the following.

- **Relative interconnectivity.** Relative interconnectivity between clusters C_i and C_j, denoted as $RI(C_i, C_j)$, is defined as the absolute interconnectivity between C_i and C_j normalized with respect to the internal interconnectivity of the two clusters C_i and C_j. The absolute interconnectivity between a pair of clusters C_i and C_j, symbolized as $EC(C_i, C_j)$, is defined as the sum of the weight of the edges that connect vertices in C_i to vertices in C_j. This is essentially the edge-cut of the cluster containing both C_i and C_j such that the cluster is broken into C_i and C_j. The internal interconnectivity of a cluster C_i can be easily captured by the size of its min-cut bisector $EC(C_i)$, which is the weighted sum of edges that partition the graph into two roughly equal parts. Thus, the relative interconnectivity between C_i and C_j is:

$$RI(C_i, C_j) = \frac{|EC(C_i, C_j)|}{\frac{|EC(C_i)|+|EC(C_j)|}{2}}. \tag{6.16}$$

- **Relative closeness.** The closeness of clusters of C_i and C_j, $RC(C_i, C_j)$, is the average weight of the edges that connect vertices in C_i to those in C_j. It provides a good measure of the affinity between the data items along the interface layer of the two clusters. At the same time, this measure is tolerant to outliers and noise. To get a cluster's internal closeness, we take the average of the edge weights across a min-cut bisection that splits the cluster into two roughly equal parts. The relative closeness between a pair of clusters is the absolute closeness normalized with respect to the internal closeness of the two clusters:

$$RC(C_i, C_j) = \frac{\bar{S}_{EC}(C_i, C_j)}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{EC}(C_i) + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{EC}(C_j)}, \tag{6.17}$$

where $\bar{S}_{EC}(C_i)$ and $\bar{S}_{EC}(C_j)$ are the average weights of the edges that belong in the min-cut bisector of clusters C_i and C_j, and $\bar{S}_{EC}(C_i, C_j)$ is the average weight of the edges that connect vertices in C_i and C_j. Terms $|C_i|$ and $|C_j|$ are the number of data points in each cluster. This equation also normalizes the absolute closeness of the two clusters by the weighted average of the internal closeness of C_i and C_j. This feature discourages merges of small sparse clusters into large dense clusters.

Chameleon selects pairs to merge for which both RI and RC are high. That is, it selects clusters that are well interconnected as well as close together. The merge scheme implemented in Chameleon uses a function to combine the relative interconnectivity and relative closeness. For this purpose, Chameleon selects the pair of clusters that maximizes

$$RI(C_i, C_j) \times RC(C_i, C_j)^{\alpha}, \tag{6.18}$$

where α is a user-specified parameter. If $\alpha > 1$, then Chameleon gives a higher importance to the relative closeness, and when $\alpha < 1$, it gives a higher importance to the relative interconnectivity.

The algorithm is well suited for applications in which the volume of the available data is large. For large V, the worst-case time complexity of the algorithm is $\mathcal{O}(V(\log_2 V + M))$, where M is the number of clusters formed after completion of the first phase of the algorithm.

The good performance of the Chameleon is recognized when applied to low-dimensional spaces. However, the performance of Chameleon in high-dimensional spaces is still not thoroughly clarified [87]. The time complexity of the Chameleon algorithm in high-dimensional spaces is $\mathcal{O}(V^2)$.

6.3.6 Community Detection by Space Transformation and Swarm Dynamics

We describe the technique introduced in [17, 61], which is based on collective dynamics. Much interest has been spent in the study of collective motion of biological entities, like schools of fish, flocks of birds, herds of hoof animals or swarms of insects. Swarm behavior is a collective behavior exhibited by animals of similar size that aggregate together, perhaps milling about the same spot or perhaps moving *en masse* or migrating in some direction. The swarm approach seeks methods consisting of a large number of simple and locally interacting agents that collectively present macroscopically complex organizations [29, 81]. Swarm behavior techniques have been successfully applied to solve various optimization problems [14].

The community detection algorithm using space transformation and swarm dynamics uses collective dynamics in a networked environment and consists of two serial steps. In the first step, the method determines how data items are represented as a network. In the second step, it detects clusters or communities by partitioning that constructed network using rules built on neighborhood agreements. This is a divisive hierarchical algorithm, in which we initially consider the entire network as a large cluster and we split it into smaller clusters, until each vertex corresponds to a cluster. Due to its hierarchical nature, we can illustrate the algorithm's result using a dendrogram, a special kind of tree where each vertex represents a cluster. A horizontal cut on the dendrogram represents a partition of the data set.

We summarize these two steps in the following:

1. *Network formation*: In this step, a weighted complete network is constructed using the input data set, in which each vertex represents a data sample. Then, a non-weighted network is generated using the k-NN method, i.e., each vertex is connected to its k most similar vertices. The similarity is determined by calculating the Euclidean distance between pairs of data samples.[1]
2. *Angle's updating rule:* After the network is constructed, the algorithm organizes the vertices on a circle. The displacement of vertices is conducted in a random manner. Thus, each vertex v_i has an initial angle $\theta_i(t = 0)$ that is randomly chosen over the range $[0, 2\pi)$. While the angle's updating rule approximates vertices that belong to the same cluster, it also separates vertices that belong to different clusters. At each time step t, the method updates the angle of each vertex according to the angles of its neighbors. We define the angle's updating rule by the following equation:

$$\theta_i(t+1) = \theta_i(t) + \eta_i(t)\left[\frac{\sum_{j\in\mathcal{N}(v_i)} \mathbf{A}_{ij}\theta_j(t)}{\sum_{j\in\mathcal{N}(v_i)} \mathbf{A}_{ij}} - \theta_i(t)\right], \tag{6.19}$$

in which $\mathcal{N}(v_i)$ is the set of neighbors of vertex v_i, $\eta_i(t)$ is the moving rate of v_i at time step t, and $\mathbf{A}_{ij}$ is the weight that represents the influence of neighbor v_j on v_i.

The edge weight $\mathbf{A}_{ij}$ aims at approximating vertices that belong to the same cluster. It is composed of two parts: $CN(v_i, v_j)$ and $SN(v_i, v_j)$. Mathematically, $\mathbf{A}_{ij}$ is expressed as:

$$\mathbf{A}_{ij} = CN(v_i, v_j) \times SN(v_i, v_j). \tag{6.20}$$

The idea of the term $CN(v_i, v_j)$ is to model physical proximity between v_i and v_j. As such, it gives more importance to vertex v_j the closer v_i and v_j are. This

[1]See Chap. 4 for a thorough review on network formation methods and similarity functions.

kind of behavior can effectively be captured by modeling $CN(v_i, v_j)$ according to the following rule:

$$CN(v_i, v_j) = e^{-\alpha d(v_i, v_j)}, \tag{6.21}$$

in which parameter α controls for the penalization decay rate of the Euclidean distance $d(v_i, v_j)$ from v_i to v_j. The algorithm can change the relative importance of a neighbor by adjusting α. The angle's updating rule can also be applied to non-weighted networks. In this case, $CN(v_i, v_j) = 1$ for all of the pairs of neighbors v_i and v_j.

In contrast to that, the term $SN(v_i, v_j)$ models the topology similarity between v_i and v_j. The hypothesis is: whenever two vertices belong to the same cluster, they are likely to share a large number of common neighbors. With that in mind, we can write $SN(v_i, v_j)$ as follows:

$$SN(v_i, v_j) = \frac{c(v_i, v_j)}{|\mathscr{N}(i)|}, \tag{6.22}$$

in which $c(v_i, v_j)$ is the number of common neighbors shared by v_i and v_j and $|\mathscr{N}(i)|$ is the number of neighbors of i. In this way, $SN(v_i, v_j)$ yields large values for vertices that share a large portion of common neighbors, regardless of the physical distance. Conversely, if they share only a small fraction of common neighbors, $SN(v_i, v_j)$ outputs small values.

Intuitively, the term $CN(v_i, v_j)$ forces angles of neighbor vertices to approximate to that of v_i and $SN(v_i, v_j)$ stops such an approximation between pairs of vertices that possibly belong to different clusters. However, these two mechanisms still cannot eliminate interference between different groups, which may cause the angles of all of the network vertices to approach each other. To mitigate this problem, one solution is to reduce the moving rate $\eta_i(t)$ in (6.19) as a function of how quickly the angles change as follows:

$$\eta_i(t) = \exp - \left(\frac{\beta}{\sigma(v_i)} \right), \tag{6.23}$$

in which $\sigma(v_i)$ is the standard deviation of the angle distribution and β is a user-defined parameter to scale the updating process of $\eta_i(t)$ as a function of $\sigma(v_i)$.

The moving rate parameter $\eta_i(t)$ decreases as the standard deviation $\sigma(v_i)$ among angles decreases. At the beginning, each angle takes a random value. In this way, the standard deviation of the angles distribution $\sigma(v_i)$ is expected to be high in such a way that $\eta_i(t)$ assumes large values, say $\eta_i(t) \approx 1$. In this situation, angles of neighboring vertices approximate freely to form angle bands. As time progresses, $\sigma(v_i)$ and consequently $\eta_i(t)$ assume smaller values. When $\eta_i(t)$ reaches a very small value, say $\eta_i(t) \approx 0$, all of the angles remain steady and a stable state is reached.

To illustrate the algorithm, we use a random clustered network with three unbalanced communities. We inspect the evolution of the angle's update process in Fig. 6.4. In this case, the algorithm identifies three communities in the network as there are three perceptive angle bands in the time series.

Now, we see the performance of the method in a real-world data set, which is a social network describing the associations (interactions) among dolphins [46]. This network has 62 vertices and 159 edges without weights. It presents two well-known communities, formed by 21 and 41 elements, respectively. Inspecting how the vertices' angles are updated in Fig. 6.5, it is possible to identify two distinct groups or communities of angles. In Fig. 6.6, we see the same simulation results but through a dendrogram perspective, where the color of each vertex indicates the community to which it originally belongs.

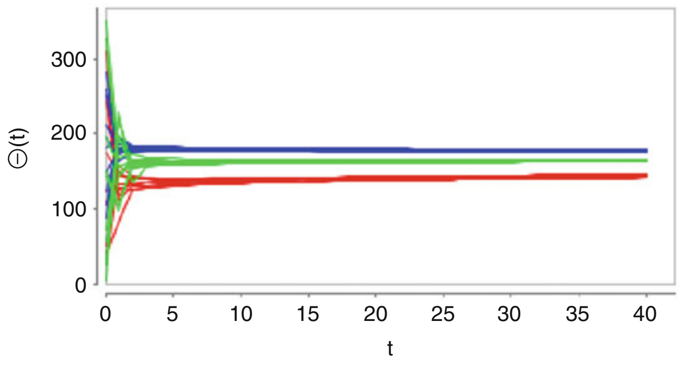

Fig. 6.4 Evolution of the angle's updating process. In the first iterations, the vertices' angles are disordered due to the random arrangements. After some iterations, they converge to stable subgroups. Reproduced from [62] with permission from the author

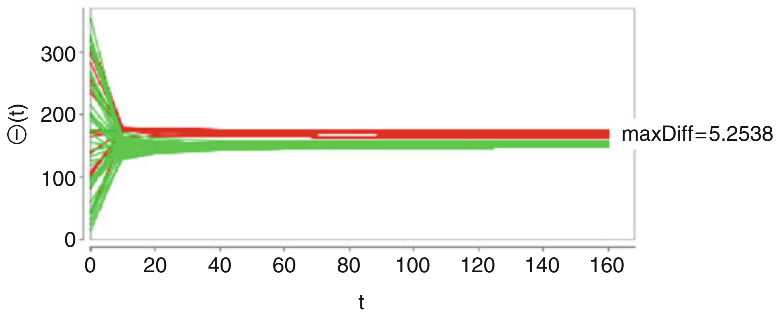

Fig. 6.5 Evolution of the angle's updating process for the social network presented in [46]. Two communities can be clearly identified by inspecting the time series. Reproduced from [62] with permission from the author

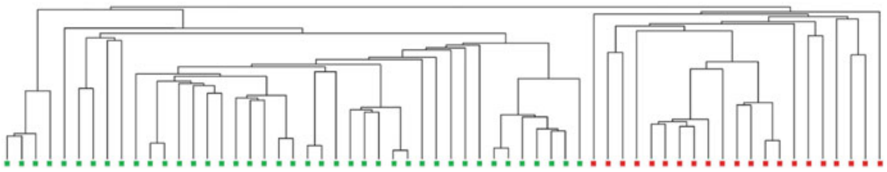

Fig. 6.6 Dendrogram showing the community detection results for the social network presented in [46]. The *dendrogram* reveals the division of data into two original communities, indicated by 41 *green* elements and 21 *red* elements. Reproduced from [62] with permission from the author

6.3.7 Synchronization Methods

Physicists have given increasing attention to the dynamics of a diversity of complex systems. In special, several studies have investigated the paradigmatic analysis of large populations of coupled oscillators [40, 64, 78, 85]. The emergence of synchronization patterns in these systems is closely related to the underlying topology of interactions. In this section, we discuss methods that rely on dynamical processes towards synchronization. In this respect, these models show different patterns over time that are intrinsically connected to the hierarchical organization of communities in complex networks. The ubiquity of synchronization phenomena in the real world makes this approach interesting from a physical and biological perspectives [3].

One of the most successful attempts to understanding synchronization phenomena comes from Kuramoto [40], who analyzed a model of phase oscillators coupled by the sine of their phase differences. The model is rich enough to display a large variety of synchronization patterns and sufficiently flexible to be adapted to many different contexts [1].

The Kuramoto model consists of V coupled phase oscillators, in which the phase of the i-th unit, denoted by $\theta_i(t)$, evolves in time according to the following dynamic:

$$\frac{d\theta_i}{dt} = \omega_i + \sum_{j \in \mathscr{V}} \mathbf{A}_{ij} \sin(\theta_j - \theta_i), \tag{6.24}$$

for $i \in \mathscr{V}$. The term ω_i stands for the natural frequency of the i-th oscillator and $\mathbf{A}_{ij}$ describes the coupling between units. The coupling weights are extracted from a network, in which each vertex is an oscillator and edge weights denote the coupling strength between different oscillators.

In particular, some works have shown that highly interconnected sets of oscillators synchronize more easily that those with sparse connections [48, 60]. This scenario suggests that, for a complex network with nontrivial connectivity patterns, starting from random initial conditions, those highly interconnected units forming local clusters will synchronize first. Then, in a sequential process, larger and larger spatial structures will do the same until we reach a final state in which the entire

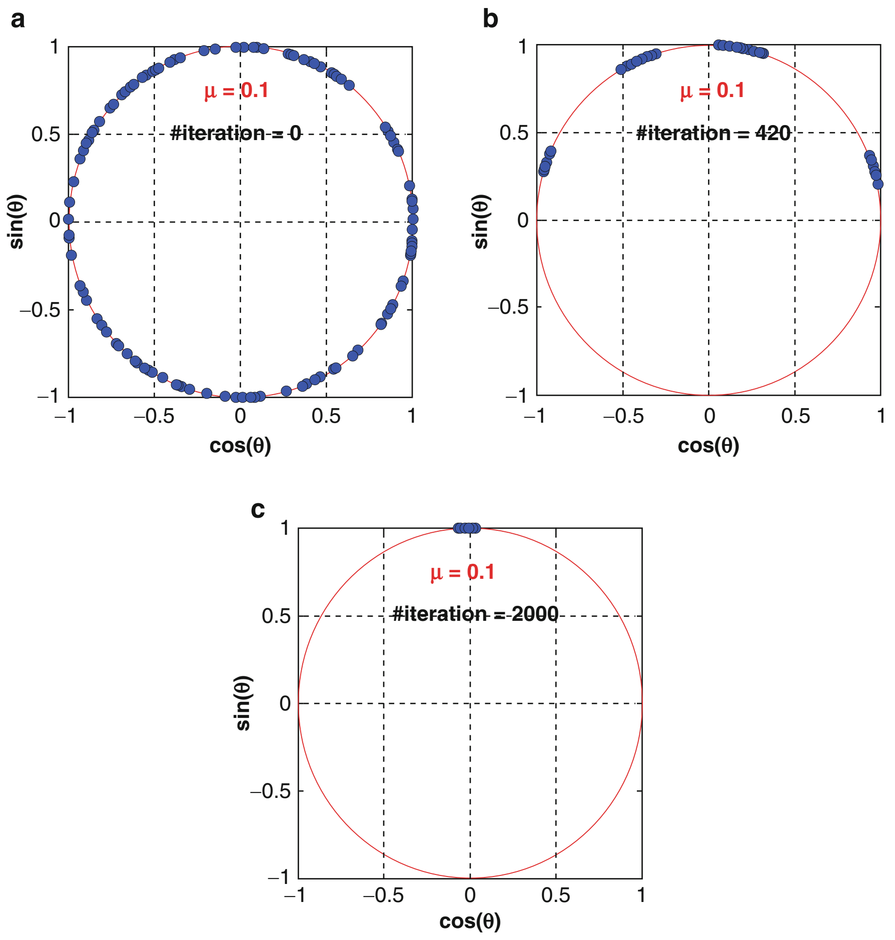

Fig. 6.7 States assumed by the population of coupled oscillators. (**a**) Random initial configuration; (**b**) Intermediate state (four communities); (**c**) Final global synchronization state

population has the same phase. This process is expected to occur at different time scales whenever clear community structures exist. Thus, the dynamical route towards the global attractor reveals different topological structures, presumably those which represent communities.

For an artificial random clustered network with four communities, Figs. 6.7a–c show, respectively, the initial configuration of the oscillators, the formation of four synchronized communities of oscillators, and the global synchronization state.

Li et al. [44] has shown that communities are delineated by interface or overlapping vertices [63], in which the oscillating frequency is intermediate among different modules, in such a way that synchronization techniques cannot clearly group these interface vertices into a single community. From this reported shortcoming, Wu et al. [86] has developed an alternative method that is capable of detecting these overlapping vertices. Contrasting to the result of Arenas et al. [3], in which the stable state is necessarily reached only by a global synchronization, in the research of Wu et al. [86], the synchronization may occur within modules. Thus, after the method synchronizes the oscillators between different communities, we can understand the phases that are in the valley between different modules to be the overlapping vertices. In order to do so, besides the global coupling supplied by the traditional Kuramoto model, another type of coupling, which is negative, between oscillators that are not connected is applied. The network dynamic can be mathematically expressed as:

$$\frac{d\theta_i}{dt} = \omega_i + \frac{K_p}{V} \sum_{j \in \mathscr{V}} \mathbf{A}_{ij} \sin(\theta_j - \theta_i) - \frac{K_n}{V} \sum_{j \in \mathscr{V}} (1 - \mathbf{A}_{ij}) \sin(\theta_j - \theta_i). \tag{6.25}$$

In this adapted format, the phases of the interconnected oscillators i and j are modeled by a positive coupling (with coupling strength K_p) in accordance with the original expression in (6.24). Thus, their phases evolve together. Non-connected vertices in the network, in contrast, tend to have opposite phases, on account of the negative coupling forced by K_n. In summary, after reaching the dynamic equilibrium, oscillators that make up the same community in the network will indicate similar phase values. Opposed to that, oscillators that represent overlapping vertices will have their phases in-between different modules [44].

6.3.8 *Finding Overlapping Communities*

The community structure is a fundamental property of most real-world networks, i.e., it is commonly observed that groups of vertices are densely interconnected. It would be oversimplifying, however, if we assumed that communities are well-defined partitions over the entire network. This is a strong assumption that may not be fulfilled in many cases. First, it is very natural for a vertex to participate in more than one community at a time; i.e., communities often overlap. Second, some vertices might not participate in any community; i.e., we might have outliers [33]. An outlier is not necessarily solitary, and it might have some negligible connection with some communities. Finally, some vertices of a community might be special in the sense that they are linked with almost all of the others. In the literature, these vertices are known as hubs, leaders, or centers. Since many real-world networks are huge, the analysis usually starts from the identification of the underlying communities possibly with overlapping characteristics. Needless to say, the community structure will greatly benefit from the simultaneous detection of hubs and outliers [9].

We can easily find overlapping communities in real-world networks. A person, for instance, can be member of a social network, of his/her family community, and also of his/her institutional community. In community detection or network-based data clustering, the detection of overlapping communities is specially interesting for fuzzy clustering.

In this section, we present some popular overlapping community detection techniques.

6.3.8.1 Clique Percolation

The most popular community detection with overlapping vertices is the Clique Percolation Method (CPM) [63]. CPM relies on the assumptions that communities consist of overlapping sets of fully connected subgraphs and that it is unlikely the existence of cliques in intercommunity edges. The general idea of the method is to detect communities by searching for adjacent cliques. It begins by identifying network cliques of size k, termed as k-cliques. Once these have been identified, a new collapsed graph is constructed in such a way that each vertex represents each of these k-cliques. Two vertices in the collapsed graph are connected if the k-cliques that represent them share $k - 1$ members. In this case, we say that these two k-cliques are adjacent. The union of adjacent k-cliques is called k-clique chain. Finally, a k-clique community is the largest connected subgraph obtained by the union of a k-clique and of all k-cliques that are connected to it.

Since a vertex can be in multiple k-cliques simultaneously, the identification of overlapping communities is possible. CPM is suitable for networks with densely connected parts. Empirically, small values of k (typically between 3 and 6) often give good results [30, 42, 63].

CPM has been extended to weighted, directed, and bipartite graphs. For weighted graphs, in principle, one can follow the standard procedure of thresholding the edge weights, and of applying the method on the resulting graph, treating them as non-weighted. Farkas et al. [20] has proposed to threshold the weight of cliques, defined as the geometric mean of the weights of all edges of the clique. The value of the threshold is chosen slightly above the critical value at which a giant k-clique community emerges, in order to get the richest possible variety of clusters.

CPM has a notable drawback in that it assumes that the network has a large number of cliques [22]. As such, CPM may fail to give meaningful covers for graphs with few cliques, like technological networks and some social networks. In contrast, if the network presents many cliques, the method may deliver trivial community structure, like a cover consisting of the entire network as a single giant cluster. A more fundamental issue is the fact that the method does not look for actual communities, consistent with the shared notion of dense subgraphs, but for subgraphs “containing” many cliques, which may be quite different objects than communities. (For instance, they could be “chains” of cliques with low internal edge density.) Another problem is that there are considerable fractions of vertices in real networks that are left out of the communities, like leaves or singletons. One could think of some postprocessing procedure to include them in the communities, but for that it is necessary to introduce a new criterion, outside the framework that inspired the method. Furthermore, besides empirical work, it is not clear *a priori* which values of k one has to choose to identify meaningful structures. Finally, the criterion to choose the threshold for weighted networks and the definition of directed k-cliques are also rather arbitrary.

6.3.8.2 Bayesian Nonnegative Matrix Factorization Algorithm

This method has been described in various works [9, 21, 67, 75]. It relies on a centrality matrix of vertices and a degree matrix of communities. The importance of a vertex to a community is represented by its centrality. The centrality matrix, hence, carries the vertices' importance in each community. An element of the degree matrix of communities, which is diagonal, indicates the degree of the community, and is equivalent to the summation of the expected degree of all vertices of that community. The algorithm then learns these two quantities by the multiplicative updating rule using a nonnegative matrix factorization style. These matrices enable us to rank each vertex's centrality in each community, and use the community degree as a cutting off criterion. Since the communities are retrieved independently, when we are working on a new community, we do not need to care whether or not its vertices belong to previously identified communities. The overlapping communities are thus handled naturally. The importance of a hub in a community ensures that it gets ranked at the top of the community. After all of the communities have been decided, those vertices that have not been included in any of them are declared as outliers. In summary, this algorithm is capable of identifying overlapping communities as well as detecting hubs and outliers simultaneously.

Mathematically, Bayesian nonnegative matrix factorization is an adaptation of the nonnegative matrix factorization technique used in machine learning for dimensionality reduction and feature extraction [89]. This technique factorizes the matrix $\mathbf{V} \in \mathbb{R}_+^{V \times V}$ into two matrices $\mathbf{W} \in \mathbb{R}_+^{V \times M}$ and $\mathbf{H} \in \mathbb{R}_+^{M \times V}$, whose elements are nonnegative, such that $\mathbf{A} \approx \mathbf{WH}$. Within the context of community detection, $\mathbf{A}$ is the adjacency matrix of the network, V is the number of vertices and M is the pre-defined number of communities. Each element of the i-th line or the j-th column of matrix $\mathbf{W}$ is the statistical dependence between a vertex i to community j. Due to matrix multiplication, the traditional nonnegative matrix factorization procedure is inefficient with respect to time and memory restrictions. In [9], a hybrid optimization algorithm that relies on a Bayesian optimization process is proposed. In essence, this algorithm optimizes an objective function expressed in terms of the above-mentioned matrices and user-supplied parameters $\beta \in \mathbb{R}^M = [\beta_1, \ldots \beta_M]$, which represent the importance of communities on the interactions of the adjacency matrix. The algorithm involves consecutive updates of $\mathbf{W}$, $\mathbf{H}$, and β until these parameters achieve convergence or until a maximum number of iterations is processed. Matrices $\mathbf{W}$, $\mathbf{H}$ and the user-supplied parameters β are calculated as follows:

$$\mathbf{H} = \left(\frac{\mathbf{H}}{\mathbf{W}^{\mathrm{T}}\mathbf{1} + \mathbf{BH}}\right) \cdot \left[\mathbf{W}^{\mathrm{T}}\left(\frac{\mathbf{V}}{\mathbf{WH}}\right)\right], \tag{6.26}$$

$$\mathbf{W} = \left(\frac{\mathbf{W}}{\mathbf{1H}^{\mathrm{T}} + \mathbf{WB}}\right) \cdot \left[\left(\frac{\mathbf{V}}{\mathbf{WH}}\right)\mathbf{H}^{\mathrm{T}}\right], \tag{6.27}$$

$$\beta_i = \frac{V + a - 1}{\frac{1}{2}\left(\sum_i \mathbf{W}_{ik}^2 + \sum_j \mathbf{W}_{ij}^2\right) + b}, \tag{6.28}$$

in which a and b are fixed parameters of a Gamma distribution and matrices $\mathbf{W}$, $\mathbf{H}$ are initialized with random values. Finally, the columns of $\mathbf{W}$ (or lines $\mathbf{H}$) containing elements with only zero values are removed and the number of communities is given by the number of columns of $\mathbf{W}$ (or the number of lines of $\mathbf{H}$) obtained after the removal.

6.3.8.3 Fuzzy Partition Algorithm

The fuzzy partition algorithm is introduced in [49]. The procedure runs as a constrained optimization problem. In that study, the expression "overlapping vertices" is conceived as "bridges" in the context of social networks, in which it is very common to find individuals that are members of multiple communities at the same time. In a social network context, "bridges" then can be defined as those vertices that cross structural holes between discrete groups of people [8]. It is therefore important to define a quantity that measures the commitment of a vertex to several communities in order to obtain a more realistic view of these networks.

The intuitive meaning of a bridge vertex may differ in different types of networks that exist beyond sociometrics. In protein interaction networks, proteins with multiple roles can be seen as bridge vertices. In cortical networks containing brain areas responsible for different modalities, the cortical areas that assume integrative roles and that provide higher level processing of sensory signals are the bridges vertices. In word-association networks, words with multiple meanings are likely to be bridges.

The overlapping condition is modeled via a fuzzy partition algorithm. A convenient representation of a given partition is the partition matrix $\mathbf{U} = [u_{ik}]$, where i indexes for the fuzzy membership across clusters k, for the data items. In this way, matrix $\mathbf{U}$ has V columns and M rows, where M is the number of subsets or clusters. We observe that $u_{ik} = 1$ if and only if vertex k belongs to the i-th subset in the partition; otherwise, it is zero. For a complete partitioning algorithm, $\sum_{i=1}^{M} u_{ik} = 1$, $\forall k \in \{1, \ldots, V\}$ must hold. The size of community i can then be calculated as $\sum_{k=1}^{V} u_{ik}$, and for any meaningful partition, we can assume that $0 < \sum_{k=1}^{V} u_{ik} < V$. These partitions are traditionally called hard or crisp partitions, because a vertex can belong to one and only one of the detected communities.

The generalization of the hard partition follows by allowing u_{ik} to attain any real value from the interval [0, 1]. The constraints imposed on the partition matrix remain the same.

It should be observed that a meaningful partition should group vertices that are somehow similar to each other in the same community. It is reasonable to assume that an edge between vertex v_1 and v_2 implies the similarity of v_1 and v_2, and likewise, the absence of an edge implies dissimilarity. Define $s(\mathbf{U}, i, j)$ as a similarity function that respects the following restrictions:

- $s(\mathbf{U}, i, j) \in [0, 1]$;
- $s(\mathbf{U}, i, j)$ is continuous and differentiable $\forall u_{ij}, i \in \{1, \ldots, c\}, j \in \{1, \ldots, N\}$;
- $s(\mathbf{U}, i, j)$ increases as i and j are more similar. Therefore, $s(\mathbf{U}, i, j)$ assumes its maximum value, $s(\mathbf{U}, i, j) = 1$, when i and j are as similar as possible. Conversely, $s(\mathbf{U}, i, j) = 0$ when i and j are totally dissimilar.

As a shorthand, consider that $s_{ij} = s(\mathbf{U}, i, j)$. Suppose we have a prior assumption about the actual similarity of vertices i and j, denoted by $\tilde{s}_{ij}$. Define the fitness of a given partition $\mathbf{U}$ of the graph by quantifying how precisely it approximates the prescribed similarity values to s_{ij}:

$$D_G(\mathbf{U}) = \sum_{i=1}^{V} \sum_{j=1}^{V} w_{ij} (\tilde{s}_{ij} - s_{ij})^2, \tag{6.29}$$

in which w_{ij} are optional weights. Say that $\mathbf{W} = \left[w_{ij}\right]$, $\mathbf{S}(\mathbf{U}) = \left[s_{ij}\right]$, and $\tilde{\mathbf{S}}(\mathbf{U}) = \left[\tilde{s}_{ij}\right]$. From now on, we assume that $\tilde{\mathbf{S}} = \mathbf{A}$, the adjacency matrix of the graph, which is in accordance with our assumption that the similarity of connected vertex pairs should be close to 1 and the similarity of disconnected vertex pairs should be close to zero. Consider that the similarity function s_{ij} is given as follows:

$$s_{ij} = \sum_{i=1}^{V} \sum_{k=1}^{M} u_{ki} u_{kj} = \mathbf{U}^T \mathbf{U}. \tag{6.30}$$

The community detection problem in this framework boils down to the optimization of $D_G(\mathbf{U})$ defined in accordance with (6.29). We note that the goal is to find a matrix $\mathbf{U}$ such that it minimizes $D_G(\mathbf{U})$. The number of clusters c, the weight matrix $\mathbf{W}$ and the desired similarity matrix $\mathbf{S}$, which is often the adjacency matrix of the network, are supplied by the user. This is a nonlinear constrained optimization problem. Although there exists a set of necessary conditions that restrict the set of possible matrices $\mathbf{U}$ worth evaluating [73], the computationally most feasible approach to optimize $D_G(\mathbf{U})$ is to use a gradient-based iterative optimization method (e.g., simulated annealing).

Consider the following objective function:

$$D_G(\mathbf{U}) = \sum_{i=1}^{V}\sum_{j=1}^{V} w_{ij}(\tilde{s}_{ij} - s_{ij})^2 + \sum_{i=1}^{V} \lambda_i \left(\sum_{k=1}^{M} u_{ki} - 1 \right), \tag{6.31}$$

in which $\lambda = [\lambda_1, \ldots, \lambda_N]$ are Lagrangian multipliers that simply force the total membership degree for each vertex to be 1 (complete partitioning).

Now we need to find $\mathbf{S}$ to minimize $D_G(\mathbf{U})$ satisfying the above constraints. The partial derivative of $D_G(U)$, with respect to u_{kl} is therefore:

$$\frac{\partial D_G(\mathbf{U})}{\partial u_{kl}} = 2 \sum_{i=1}^{V} (e_{il} + e_{li}) \left(\frac{1}{M} - u_{ki} \right), \tag{6.32}$$

in which $e_{ij} = w_{ij}(\tilde{s}_{ij} - s_{ij})$.

The simplest gradient-based algorithm for finding a local minimum of D_G is then the following:

1. Start from an arbitrary random partition $\mathbf{U}(0)$ and let $t = 0$.
2. Calculate the gradient vector of D_G according to (6.32) and the current $\mathbf{U}(t)$.
3. If $\max_{k,l} \left| \frac{\partial D_G(U)}{\partial u_{kl}} \right| < \epsilon$, stop the iteration and declare $\mathbf{U}(t)$ a solution.
4. Otherwise, calculate the next partition in the iteration with the following equation:

$$u_{ij}^{(t+1)} = u_{ij}^{(t)} + \alpha^{(t)} \frac{\partial D_G(U)}{\partial u_{ij}}, \tag{6.33}$$

   in which $\alpha^{(t)}$ is a small step size constant chosen appropriately.
5. Increase t and continue from step 2.

6.3.9 Network Embedding and Dimension Reduction

Dimension reduction is an important pre-processing in data analysis and machine learning. It can be considered as a procedure to produce a compact low-dimensional encoding of a given high-dimensional data set [47, 74, 84]. Dimension reduction is specially interesting when we deal with data sets that have many more variables than data samples. For example, microarray data sets usually are composed by thousands of variables (genes) in dozens of samples. The most famous dimension reduction technique is the Principal Component Analysis (PCA) that dates back to Karl Pearson in 1901 [65]. The basic idea is to find a new coordinate system via a linear or a nonlinear transformation in which the input data can be expressed with many less variables without a significant loss. Isomap [80] was originally proposed as a generalization of multidimensional scaling [15]. An alternative method known as Locally Linear Embedding (LLE) [72] was developed that solved a consecutive pair of linear least square optimizations. The kernel method, including graph kernel method, has also been proposed for nonlinear dimension reduction by performing linear operations on kernel mapping functions. Graph kernel methods for data analysis and machine learning are an active research topic and are not covered in this book. The interested readers may refer to [5, 27, 45, 50, 76, 82]. In this book, we just present one technique on this topic. In [88], a graph embedding method has been proposed and is briefly reviewed in the following paragraph.

Consider that it is given a data set $\mathscr{X} = \{x_1, x_2, \ldots, x_V\}$. Each data sample is described by P attributes, that is, a feature vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{iP})^T$. Consider $\mathbf{X}$ as the matrix whose columns denote each data item in $\mathscr{X}$. The goal of the technique is to perform dimensionality reduction in the data items' feature vectors to a smaller number P' of projected attributes. For example, the feature dimension P of images is usually very high, and transforming the data from the original high-dimensional space to a low-dimensional space can alleviate the curse of dimensionality problem. To accomplish that, a technique should find a mapping function F that transforms each feature vector $x \in \mathbb{R}^P$ into the desired low-dimensional representation y, so that $y = F(x)$, $y \in \mathbb{R}^{P'}$. By using an underlying network to find such function F, the dimensionality reduction process can be viewed as a graph-preserving criterion of the following form:

$$\begin{aligned} Y^* &= \arg\min_Y \sum_{\substack{i,j \in \mathscr{V} \\ i \neq j}} \mathbf{A}_{ij} \|y_i - y_j\|^2 \\ &= \arg\min_Y Y^T \mathbf{L} Y, \end{aligned} \tag{6.34}$$

constrained to $Y^T\mathbf{B}Y = d$. In this formulation, d is a constant vector, $\mathbf{A}$ is the adjacency matrix of the network, $\mathbf{B}$ is the constraint matrix, and $\mathbf{L}$ is the Laplacian matrix. Recall that the Laplacian matrix can be found via the following operation:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{6.35}$$

in which:

$$\mathbf{D}_{ii} = \sum_{\substack{j \in \mathscr{V} \\ j \neq i}} \mathbf{A}_{ij}, \tag{6.36}$$

$\forall i \in \mathscr{V}$.

The constraint matrix $\mathbf{B}$ can be viewed as the adjacency matrix of a penalty network $\mathbf{A}^P$, so that $\mathbf{B} = \mathbf{L}^P = \mathbf{D}^P - \mathbf{A}^P$. The penalty network conveys information about which vertices should not be linked together, that is, which instances should be far apart after the dimensionality reduction process. The similarity preservation property from the graph-preserving criterion has a twofold explanation. For larger similarity between samples x_i and x_j, the distance between y_i and y_j should be smaller to minimize the objective function. Likewise, smaller similarity between x_i and x_j should lead to larger distances between y_i and y_j for minimization. Assume that the low-dimensional attribute space can be found by using a linear projection such as $Y = \mathbf{X}^T w$, where w is the projection vector. The objective function in (6.34) becomes:

$$\begin{aligned} w^* &= \arg\min_w \sum_{\substack{i,j \in \mathcal{V} \\ i \neq j}} \mathbf{A}_{ij} \| w^T x_i - w^T x_j \|^2 \\ &= \arg\min_w w^T \mathbf{X}^T \mathbf{L} \mathbf{X} w, \end{aligned} \tag{6.37}$$

constrained to $w^T \mathbf{X}^T \mathbf{L} \mathbf{X} w = d$. By using the Marginal Fisher Criterion and the penalty network constraint, Eq. (6.35) becomes:

$$w^* = \arg\min_w \frac{w^T \mathbf{X}^T \mathbf{L} \mathbf{X} w}{w^T \mathbf{X} \mathbf{L}^P \mathbf{X}^T w}, \tag{6.38}$$

which can be solved by the generalized eigenvalue problem by using the equation $\mathbf{X} \mathbf{L} \mathbf{X}^T w = \lambda \mathbf{X} \mathbf{L}^P \mathbf{X}^T w$.

6.4 Chapter Remarks

Clustering is the unsupervised grouping of patterns, such as observations, data items, or feature vectors. The clustering task has been addressed in many contexts and by researchers in many disciplines; this diversity reflects its broad appeal and usefulness as one of the steps in exploratory data analysis. Intuitively, patterns within the same cluster are more similar to each other than they are to a pattern belonging to a different cluster. Clustering is useful in several exploratory tasks, such as in data mining, document retrieval, image segmentation, and pattern classification. Often, there is little prior information (e.g., statistical models) available about the data, and the learning algorithm must make as few assumptions about the data as possible. It is under these restrictions that clustering methodology is particularly appropriate for the exploration of interrelationships among the data points.

In this chapter, we have focused on data clustering in a networked environment, which is often termed as community detection. The study of community detection is very important for understanding various phenomena in complex networks. Modular structure introduces important heterogeneities in complex networks. Each module, for example, can have different local statistics; some modules may have many connections, while other modules may be sparse. When there is large variation among communities, global values of statistical measures can be misleading. The presence of modular structure may also alter the way in which dynamical processes unfold on the network. In biological networks, communities correspond to functional modules in which module members function coherently to perform essential cellular tasks. Hence, the development of efficient community detection methods stands as an important topic in the agenda for the complex network and machine learning communities. Due to that importance, this chapter has dedicated a great part of it to the study of several representative community detection algorithms. For each of them, we have explained the main idea behind the community detection mechanism and also the potentialities and shortcomings of the methods. Community detection benchmarks have also been explored.

The topic of detection of overlapping communities has also been discussed. We can easily find overlapping communities in real-world networks. A person, for instance, can belong to a social network, in his/her family community, and also in his/her institutional community. In network-based unsupervised learning, the detection of overlapping communities is especially interesting for fuzzy clustering. Some representative methods have also been explored.

References

1. Acebrón, J.A., Bonilla, L.L., Vicente, P.C.J., Ritort, F., Spigler, R.: The kuramoto model: A simple paradigm for synchronization phenomena. Rev. Mod. Phys. **77**, 137–185 (2005)
2. Alpert, C.J., Kahng, A.B., Yao, S.Z.: Spectral partitioning with multiple eigenvectors. Discret. Appl. Math. **90**(1-3), 3–26 (1999)
3. Arenas, A., Guilera, A.D., Pérez Vicente, C.J.: Synchronization reveals topological scales in complex networks. Phys. Rev. Lett. **96**(11), 114102 (2006)
4. Arenas, A., Duch, J., Fernández, A., Gómez, S.: Size reduction of complex networks preserving modularity. New J. Phys. **9**(6), 176 (2007)
5. Borgwardt, K.M.: Graph kernels. Ph.D. thesis, Ludwig-Maximilians-Universitöt München, Germany (2007)
6. Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: On modularity clustering. IEEE Trans. Knowl. Data Eng. **20**(2), 172–188 (2008)
7. Buchanan, M.: Nexus: Small Worlds and the Groundbreaking Theory of Networks. W.W. Norton, New York (2003)
8. Burt, R.S.: Structural holes: the social structure of competition. Harvard University Press, Cambridge, MA (1992)
9. Cao, X., Wang, X., Jin, D., Cao, Y., He, D.: Identifying overlapping communities as well as hubs and outliers via nonnegative matrix factorization. Sci. Rep. **3**, 2993 (2013)
10. Chen, J., Yuan, B.: Detecting functional modules in the yeast protein–protein interaction network. Bioinformatics **22**(18), 2283–2290 (2006)
11. Chen, M., Kuzmin, K., Szymanski, B.: Community detection via maximization of modularity and its variants. IEEE Trans. Comput. Soc. Syst. **1**(1), 46–65 (2014)
12. Chung, F.R.K.: Spectral Graph Theory. CBMS Regional Conference Series in Mathematics, vol. 92. American Mathematical Society, Providence, RI (1997)
13. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Phys. Rev. E **70**(6), 066111+ (2004)
14. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
15. Cox, T.F., Cox, M.: Multidimensional Scaling. Chapman & Hall/CRC, London/Boca Raton (2000)

16. Danon, L., Díaz-Guilera, A., Duch, J., Arenas, A.: Comparing community structure identification. J. Stat. Mech. Theory Exp. **2005**(09), P09008 (2005)
17. de Oliveira, T., Zhao, L.: Complex network community detection based on swarm aggregation. In: International Conference on Natural Computation, vol. 7, pp. 604–608. IEEE, New York (2008)
18. Donath, W.E., Hoffman, A.J.: Lower bounds for the partitioning of graphs. IBM J. Res. Dev. **17**(5), 420–425 (1973)
19. Evans, T.S., Lambiotte, R.: Line graphs, link partitions, and overlapping communities. Phys. Rev. E **80**(1), 016105 (2009)
20. Farkas, I., Ábel, D., Palla, G., Vicsek, T.: Weighted network modules. New J. Phys. **9**(6), 180 (2007)
21. Févotte, C., Bertin, N., Durrieu, J.L.: Nonnegative matrix factorization with the itakura-saito divergence: with application to music analysis. Neural Comput. **21**(3), 793–830 (2009)
22. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**, 75–174 (2010)
23. Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proc. Natl. Acad. Sci. **104**(1), 36–41 (2007)
24. Fortunato, S., Latora, V., Marchiori, M.: Method to find community structures based on information centrality. Phys. Rev. E **70**(5), 056104 (2004)
25. Freeman, L.C.: A set of measures of centrality based upon betweenness. Sociometry **40**, 35–41 (1977)
26. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. Science **315**, 972–976 (2007)
27. Gärtner, T.: A survey of kernels for structured data. SIGKDD Explor. **5**(1), 49–58 (2003)
28. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. Proc. Natl. Acad. Sci. USA **99**(12), 7821–7826 (2002)
29. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science **286**, 531–537 (1999)
30. Gregory, S.: Finding overlapping communities in networks by label propagation. New J. Phys. **12**(10), 103018 (2010)
31. Guimera, R., Sales-Pardo, M., Amaral, L.: Modularity from fluctuations in random graphs and complex networks. Phys. Rev. E **70**, 025101 (2004)
32. Gulbahce, N., Lehmann, S.: The art of community detection. BioEssays **30**(10), 934–938 (2008)
33. Gupta, M., Gao, J., Aggarwal, C., Han, J.: Outlier detection for temporal data: a survey. IEEE Trans. Knowl. Data Eng. **26**(9), 2250–2267 (2014)
34. Hofman, J.M., Wiggins, C.H.: Bayesian approach to network modularity. Phys. Rev. Lett. **100**(25), 258701+ (2008)
35. Jin, J., Pawson, T.: Modular evolution of phosphorylation-based signalling systems. Philos. Trans. R. Soc. Lond. Ser. B Biol. Sci. **367**(1602), 2540–55 (2012)
36. Karypis, G., Han, E.H., Kumar, V.: Chameleon: hierarchical clustering using dynamic modeling. Computer **32**(8), 68–75 (1999)
37. Kawamoto, T., Kabashima, Y.: Limitations in the spectral method for graph partitioning: detectability threshold and localization of eigenvectors. Phys. Rev. E **91**, 062803 (2015)
38. Kiss, G.R., Armstrong, C., Milroy, R., Piper, J.R.I.: An associative thesaurus of English and its computer analysis. In: The Computer and Literary Studies. University Press, Edinburgh (1973)
39. Kumpula, J.M., Saramäki, J., Kaski, K., Kertész, J.: Limited resolution in complex network community detection with Potts model approach. Eur. Phys. J. B **56** (2007)
40. Kuramoto, Y.: Chemical Oscillations, Waves, and Turbulence. Springer, New York (1984)
41. Lancichinetti, A., Fortunato, S.: Limits of modularity maximization in community detection. Phys. Rev. E **84**, 066122 (2011)
42. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E **78**(4), 046110(1–5) (2008)

43. Lancichinetti, A., Fortunato, S., Kertész, J.: Detecting the overlapping and hierarchical community structure in complex networks. New J. Phys. **11**(3), 033015 (2009)
44. Li, D., Leyva, I., Almendral, J.A., Sendina-Nadal, I., Buldu, J.M., Havlin, S., Boccaletti, S.: Synchronization interfaces and overlapping communities in complex networks. Phys. Rev. Lett. **101**(16), 168701 (2008)
45. Liu, W., Principe, J.C., Haykin, S.: Kernel Adaptive Filtering: A Comprehensive Introduction. Wiley, New York (2010)
46. Lusseau, D.: The emergent properties of a dolphin social network. Proc. R. Soc. B Biol. Sci. **270**(Suppl 2), S186–S188 (2003)
47. Ma, Y., Zhu, L.: A review on dimension reduction. Int. Stat. Rev. **81**(1), 134–150 (2013)
48. Moreno, Y., Vazquez-Prada, M., Pacheco, A.F.: Fitness for synchronization of network motifs. Physica A **343**, 279–287 (2004)
49. Nepusz, T., Petróczi, A., Négyessy, L., Bazsó, F.: Fuzzy communities and the concept of bridgeness in complex networks. Phys. Rev. E **77**, 016107 (2008)
50. Neuhaus, M., Bunke, H.: Bridging the Gap Between Graph Edit Distance and Kernel Machines. World Scientific, River Edge, NJ (2007)
51. Newman, M.E.J.: Analysis of weighted networks. Phys. Rev. E **70**, 056131 (2004)
52. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E **69**(6), 066133 (2004)
53. Newman, M.E.J.: A measure of betweenness centrality based on random walks. Soc. Networks **27**, 39–54 (2005)
54. Newman, M.E.J.: Finding community structure in networks using the eigenvectors of matrices. Phys. Rev. E **74**(3), 036104 (2006)
55. Newman, M.E.J.: Modularity and community structure in networks. Proc. Natl. Acad. Sci. **103**(23), 8577–8582 (2006)
56. Newman, M.E.J.: Spectral methods for community detection and graph partitioning. Phys. Rev. E **88**, 042822 (2013)
57. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. Lett. **69**, 026113 (2004)
58. Newman, M.E.J., Leicht, E.A.: Mixture models and exploratory analysis in networks. Proc. Natl. Acad. Sci. USA **104**(23), 9564–9569 (2007)
59. Nicosia, V., Mangioni, G., Carchiolo, V., Malgeri, M.: Extending the definition of modularity to directed graphs with overlapping communities. J. Stat. Mech. Theory Exp. **2009**(03), 03024 (2009)
60. Oh, E., Rho, K., Hong, H., Kahng, B.: Modular synchronization in complex networks. Phys. Rev. E **72**, 047101 (2005)
61. de Oliveira, T., Zhao, L., Faceli, K., de Carvalho, A.: Data clustering based on complex network community detection. In: IEEE Congress on Evolutionary Computation, pp. 2121–2126. IEEE, New York (2008)
62. Oliveira, T.B.S.: Clusterização de dados utilizando técnicas de redes complexas e computação bioinspirada (2008). Master Thesis. Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo (USP)
63. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**(7043), 814–818 (2005)
64. Panaggio, M.J., Abrams, D.M.: Chimera states: coexistence of coherence and incoherence in networks of coupled oscillators. Nonlinearity **28**(3), R67 (2015)
65. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philos. Mag. **2**(6), 559–572 (1901)
66. Pons, P., Latapy, M.: Computing communities in large networks using random walks. J. Graph Algorithms Appl. **10**, 284–293 (2004)
67. Psorakis, I., Roberts, S., Ebden, M., Sheldon, B.: Overlapping community detection using bayesian non-negative matrix factorization. Phys. Rev. E **83**, 066114 (2011)
68. Quiles, M.G., Zhao, L., Alonso, R.L., Romero, R.A.F.: Particle competition for complex network community detection. Chaos **18**(3), 033107 (2008)

69. Ravasz, E., Somera, A.L., Mongru, D.A., Oltvai, Z.N., Barabási, A.L.: Hierarchical organization of modularity in metabolic networks. Science **297**(5586), 1551–1555 (2002)
70. Reichardt, J., Bornholdt, S.: Detecting fuzzy community structures in complex networks with a potts model. Phys. Rev. Lett. **93**(21), 218701(1–4) (2004)
71. Rosvall, M., Bergstrom, C.T.: An information-theoretic framework for resolving community structure in complex networks. Proc. Natl. Acad. Sci. **104**(18), 7327–7331 (2007)
72. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science **290**, 2323–2326 (2000)
73. Ruszczyński, A.P.: Nonlinear optimization. Princeton University Press, Princeton, NJ (2006)
74. Sarveniazi, A.: An actual survey of dimensionality reduction. Am. J. Comput. Math. **4**, 55–72 (2014)
75. Schmidt, M.N., Winther, O., Hansen, L.K.: Bayesian non-negative matrix factorization. In: Adali, T., Jutten, C., Romano, J.M.T., Barros, A.K. (eds.) Independent Component Analysis and Signal Separation. Lecture Notes in Computer Science, vol. 5441, pp. 540–547. Springer, Berlin, Heidelberg (2009)
76. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York (2004)
77. Shen, H., Cheng, X., Cai, K., Hu, M.B.: Detect overlapping and hierarchical community structure in networks. Physica A **388**(8), 1706–1712 (2009)
78. Strogatz, S.H.: Sync: The Emerging Science of Spontaneous Order. Hyperion, New York (2003)
79. Sun, P.G., Gao, L., Shan Han, S.: Identification of overlapping and non-overlapping community structure by fuzzy clustering in complex networks. Inf. Sci. **181**, 1060–1071 (2011)
80. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290**(5500), 2319–2323 (2000)
81. Topaz, C.M., Andrea, Bertozzi, L.: Swarming patterns in a two-dimensional kinematic model for biological groups. SIAM J. Appl. Math. **65**, 152–174 (2004)
82. Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. J. Mach. Learn. Res. **11**, 1201–1242 (2010)
83. Wakita, K., Tsurumi, T.: Finding community structure in mega-scale social networks: [extended abstract]. In: Proceedings of the 16th International Conference on World Wide Web, WWW '07, pp. 1275–1276 (2007)
84. Wang, F., Sun, J.: Survey on distance metric learning and dimensionality reduction in data mining. Data Min. Knowl. Disc. **29**(2), 534–564 (2015)
85. Winfree, A.T.: The Geometry of Biological Time. Springer, Berlin (2001)
86. Wu, Z., Duan, J., Fu, X.: Complex projective synchronization in coupled chaotic complex dynamical systems. Nonlinear Dyn. **69**(3), 771–779 (2012)
87. Xu, R., II, D.W.: Survey of clustering algorithms. IEEE Trans. Neural Netw. **16**(3), 645–678 (2005)
88. Yan, S., Xu, D., Zhang, B., Zhang, H.J., Yang, Q., Lin, S.: Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 40–51 (2007)
89. Zarei, M., Izadi, D., Samani, K.: Detecting overlapping community structure of networks based on vertex-vertex correlations. J. Stat. Mech. Theory Exp. **11**, P11013 (2009)
90. Zhang, S., Wang, R.S., Zhang, X.S.: Identification of overlapping community structure in complex networks using fuzzy C-Means clustering. Physica A **374**(1), 483–490 (2007)
91. Zhang, X., Nadakuditi, R.R., Newman, M.E.J.: Spectra of random graphs with community structure and arbitrary degrees. Phys. Rev. E **89**, 042816 (2014)
92. Zhou, H.: Distance, dissimilarity index, and network community structure. Phys. Rev. E **67**(6), 061901 (2003)