5 COMMUNITY DETECTION

5.1 COMMUNITY DETECTION

Real-world networks often have groups of nodes that are more densely connected among themselves than with others: we refer to this recurrent topological property as presence of a *community structure*. Let us list some examples of communities in real-world networks:

- In human social networks edges indicate an interaction between individuals. Depending on the system under consideration, one can find communities that represent groups of friends, people that speak a common language, colleagues of people from a same party. A practical example is a co-authorship network of researchers, in which an edge

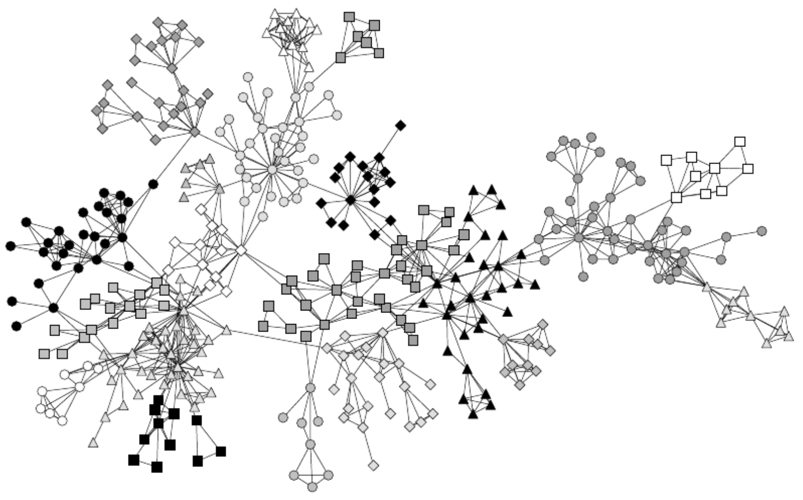


Figure 5.1: **A graph with communities**. Each community is marker coded. Picture taken from *Newman, Netwoks.*

between two nodes (researchers) is drawn if they co-authored a paper and communities correspond naturally to research fields.

Some examples of graph with communities

- Social networks such as Facebook or, more generally the Web have a community structure made of groups of related pages or accounts.
- In biology, groups of molecules form functional modules that can be represented as communities of a network.
- If one identifies the nodes with the words of a dictionary and edges represent co-occurrence in a text, then communities indicate words that are related, such as *milk* and *cow*.

More generally, given a set of objects (say images) that can be related, one can design a graph in which each node corresponds to an element and an edge represents the relation between pairs of objects. Communities then correspond to categories of objects (such as images of dogs vs images of cats). *Community detection* (CD) is the task of identifying these communities on a given graph. Given the broad range of systems that can be modeled with networks, CD has important applications in categorization. We now give a broad overview of some building blocks of CD that need to be considered to have a full picture of the problem at hand.

5.1.1 DEFINING COMMUNITIES

A relevant problem of CD is that, although it appears as an intuitive task, strictly speaking it is ill-defined. In fact there exist no shared consensus in the scientific literature on what a *community* really is. Every CD algorithm is based on a particular definition – that can be more or less explicit – of communities and, from an operational viewpoint, one can define *communities* as the output of a particular CD algorithm.

Communities are ill-defined

In the remainder we will consider CD only for undirected and unweighted graphs, with the additional requirement that communities are not overlapping. This means that the output of a CD algorithm can be represented in the form of a vector $\boldsymbol{\ell} \in [k]^n$, where k is the number of communities and n the number of nodes. This labeling vector associates each node to a unique class. Denoting with $\mathcal{V}_a = \{i \in \mathcal{V} \ : \ \ell_i = a\}$ the set of all nodes with label a, we have that

$$\mathcal{V} = \bigcup_{a=1}^{k} \mathcal{V}_a \, ;$$
$$\mathcal{V}_a \cap \mathcal{V}_b = \emptyset \qquad \text{for } a \neq b.$$

It has to be noted, however, that also this requirement can be questioned and there are many algorithms that instead consider a more general concept of overlapping communities. An example of a non-overlapping node partition on a graph is shown in Figure 5.1.

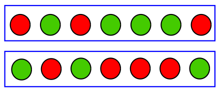
5.1.2 THE NUMBER OF COMMUNITIES

Community detection is an inherently unsupervised task, meaning that the input of CD is a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ without any additional information. Consequently, a problem is related to the fact that the number of communities itself needs to be determined. This is a significant challenge, as we will see in the remainder. Intuitively, the problem is related to the fact that comparing partitions with the same number of communities may be a rather straightforward task, but not the same can be said for partitions into a different number of groups. As a consequence, some algorithms require the number of communities k is required as an algorithm input and leave the problem of finding a reasonable k to the user. Other algorithms adopt greedy strategies to compare partitions with different numbers of communities and suffer for several limitations for this very reason, as we will see in the remainder. Generally speaking, very few methods are known to reliable estimate k and they leverage specific definitions of communities. For this reason, when one is performing CD, it should be well aware of the fact that different algorithms may yield very different responses and a holistic vision may help one have a clearer picture. But how do we compare the outputs of different algorithms?

Determining the number of communities is a difficult task

5.1.3 COMPARING PARTITIONS

The output of a CD algorithm is a node partition, *i.e.* a subdivision of the graph nodes into sets. Importantly, the naming of the subsets that we provide (say a and b, 0 and 1, *red* and *blue*) are only meaningful when considering different nodes in the same partition. If instead we consider two different partitions, the naming we use can be interchanged (what we used to call b is now a) or even be completely different (we call them c and d). As a consequence, partitions cannot be directly compared and more refined strategies need to be adopted.


Two different naming strategies of the same partition

A powerful metric to compare different partitions is based on the mutual information. The mutual information between two random variables X and Y quantifies how much is known of X if Y is given. In the example on the side bar, knowing that a point is red in the first partition implies that it is green in the second. The metric we will adopt in the following is the *Adjusted mutual information* (AMI) that is a rescaled version of the mutual information so that: $\text{AMI} = 1$ if the partitions are equivalent; $\text{AMI} = 0$ if the partitions contain as much information of one another as a random guesser. Notably, the AMI also allows us to compare partitions with a different number of communities.

5.1.4 COMPUTATIONAL COMPLEXITY

As a final remark, we should be aware that CD is a practical task that is executed by algorithms that should be fast, in order to be deployable. The measure of speed of an algorithm is its computational complexity, *i.e.* how many operations the algorithm performs with respect to the number of input variables. For CD there are two main quantities of interest that determine the computational complexity: the number of edges m of $\mathcal{G}(\mathcal{V}, \mathcal{E})$ and the number of communities k. A first remarkable distinctions should be made between polynomial algorithms and NP-hard ones. Polynomial algorithms require a number of operations scaling as $\mathcal{O}(m^\alpha k^\beta)$ for some $\alpha, \beta \geq 0$, while NP-hard problems likely[1] require an number of operations that goes to infinity faster than any polynomial. In practical terms, NP-hard problems cannot be solved unless for very small input graphs, since the number of operations required may even scale exponentially with the number of nodes. In practice "NP complete" should sound to your ears as "impossible to solve". Polynomial-time algorithms are the only ones that can used, but polynomial does not mean fast. In fact, on an ordinary personal computer, running an algorithm with complexity $\mathcal{O}(n^3)$ may become prohibitive in terms of time and memory for $n > 10^4$. In practice, fast algorithms that can be applied to large graphs should scale linearly with the number of edges m.

Large graphs require algorithms with a computational complexity scaling linearly with m

We now proceed providing an overview of some popular approaches to CD, alongside with their strengths and limitations.

1 Even if it is not proved, it has be conjectured that NP-hard problems do not admit a polynomial time solution.

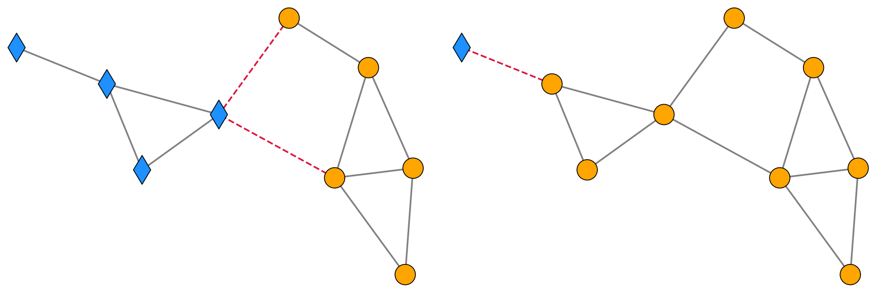


Figure 5.2: **Graph cut evaluation**. Two partitions of the same graph: the one on the left has a graph cut equal to 2, while the one on th right has a graph cut equal to 1.

5.2 OPTIMIZATION APPROACHES

Defining communities as the solution of an optimization problem consists in identifying a quality function assessing how satisfactory a given class partition is on a graph. Such function should depend on the partition $\boldsymbol{\ell}$ and the graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, in the form of its adjacency matrix A.

5.2.1 SOME DEFINITIONS

The simplest method to define such a cost function consists in literally counting how many edges fall among nodes in the same different communities and minimize it.[2] We then introduce the graph cut

The graph cut

$$\mathcal{Q}_A^{\mathrm{Cut}}(\boldsymbol{\ell}) = \frac{1}{2}\sum_{a=1}^{k}\sum_{i\in\mathcal{V}_a}\sum_{j\notin\mathcal{V}_a} A_{ij}, \tag{5.1}$$

where we recall that $\mathcal{V}_a = \{i \in \mathcal{V} \ : \ \ell_i = a\}$ is the set of all nodes with label p. The goal is to find $\boldsymbol{\ell}$ that minimizes $\mathcal{Q}_A^{\mathrm{Cut}}$, under the constraint $\boldsymbol{\ell} \neq \mathbf{1}_n$, which prevents all nodes from being assigned to the same cluster. Yet, for how reasonable this function may seem, it is too simple and it tends to create partitions in which a single node is isolated from all others, as shown in Figure 5.2. Even if these partitions are so that nodes in the same community are more connected with one another than with other nodes, they do not encode a common sense meaning of *community*.

To cope with this problem, we must introduce the requirement that the community is *sufficiently* large. We do so introducing the *ratio cut* (RCut), in Equation (5.2) and the *normalized cut* (NCut) in Equation (5.3).

$$\mathcal{Q}_A^{\mathrm{RCut}}(\boldsymbol{\ell}) = \frac{1}{2}\sum_{a=1}^{k}\frac{1}{|\mathcal{V}_a|}\sum_{i\in\mathcal{V}_a}\sum_{j\notin\mathcal{V}_a} A_{ij} \tag{5.2}$$

2 Note that, since the number of edges of a given graph its fixed, we are also maximizing the number of edges between nodes in the same community.

Normalized cut

$$\mathcal{Q}_A^{\mathrm{NCut}}(\boldsymbol{\ell}) = \frac{1}{2}\sum_{a=1}^{k} \frac{\sum_{i\in\mathcal{V}_a}\sum_{j\notin\mathcal{V}_a} A_{ij}}{\sum_{i\in\mathcal{V}_a}\sum_{j\in\mathcal{V}} A_{ij}}, \tag{5.3}$$

Another quality function that is very popular is the *modularity*:

Modularity

$$\mathcal{Q}_A^{\mathrm{Mod}}(\boldsymbol{\ell}) = \frac{1}{4|\mathcal{E}|}\sum_{i,j\in\mathcal{V}} \left(A_{ij} - \frac{d_i d_j}{2|\mathcal{E}|}\right)\delta(\ell_i,\ell_j), \tag{5.4}$$

The modularity attributes a large score to configurations in which nodes in the same community are connected by a *greater than expected*[3] number of edges. In fact, for a fixed degree sequence, $d_i d_j / 2|\mathcal{E}|$ is the probability that nodes i and j are connected if edges were placed at random. The modularity has subsequently been exploited to define CD algorithms in which the label assignment is obtained maximizing the $\mathcal{Q}_A^{\mathrm{Mod}}(\boldsymbol{\ell})$. In practice, a modularity equal to 0.4 is considered to be a good partition, even if we will discuss how this metric should be taken with caution. Since modularity maximization is one of the most popular methods for CD, we will look at it in more detail, describing a modularity maximization algorithm.

5.2.2 LOUVAIN ALGORITHM

Given its popularity, we here will discuss one algorithm to optimize the modularity cost function, that is called *Louvain algorithm*, named after the university of origin of the researchers that introduced it. The first, fundamental observation is that *Modularity* (but also *RCut*, *NCut*) optimization is an NP hard problem. We thus need to find some greedy algorithm to find a reasonable approximation of this maximum.

The *Louvain* algorithm indeed defines an approximate strategy to maximize the modularity and it is composed of two steps. The first starts from a configuration in which each node is set in a different community and then each *single* nodes is moved to the community of one of its neighbors if there is a gain in modularity in doing so. When no gains in the modularity can be obtained, step 1 is concluded. Communities are then represented as nodes and the edges are given a weight according to how many links run between one community and the other. The process is iterated until there is no gain in merging nodes. Figure 5.3 summarizes the steps of *Louvain* algorithm.

The main advantage of *Louvain* algorithm is related to its speed. since the computation of the modularity variation can be performed very efficiently. Letting $m = 2|\mathcal{E}|$ for simplicity, we can rewrite the modularity as follows:

$$\mathcal{Q}_A^{\mathrm{Mod}}(\boldsymbol{\ell}) = \frac{1}{m}\sum_{i,j\in\mathcal{V}} \left(A_{ij} - \frac{d_i d_j}{m}\right)\delta(\ell_i,\ell_j)$$

3 In other words, the modularity compares the realization of the matrix A with a typical realization of a *null model*, called *configuration model*.

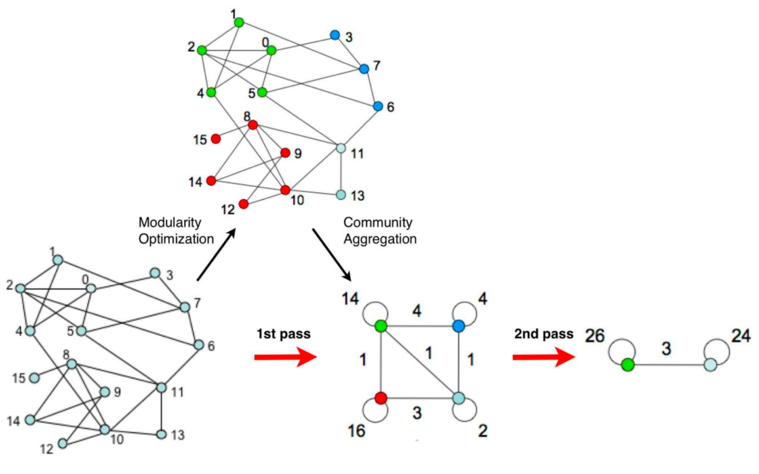


Figure 5.3: **Schematic representation of Louvain algorithm**. Each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the found communities are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible. Picture taken from *Blondel et al, Fast unfolding of communities in large networks.*

Rewriting the modularity in a simpler form

$$\begin{aligned}
&\overset{(a)}{=} \frac{1}{m}\sum_{a=1}^{k}\sum_{b=1}^{k}\sum_{i\in\mathcal{V}_a}\sum_{b\in\mathcal{V}_b}\left(A_{ij}-\frac{d_id_j}{m}\right)\delta(a,b)\\
&= \frac{1}{m}\sum_{a=1}^{k}\sum_{i,j\in\mathcal{V}_a}\left(A_{ij}-\frac{d_id_j}{m}\right)\\
&\overset{(b)}{=} \frac{1}{m}\sum_{a=1}^{k}\sum_{i\in\mathcal{V}_a}\left(d_i^{(a)}-\frac{d_i\Sigma_{\text{tot}}^{(a)}}{m}\right)\\
&\overset{(c)}{=} \sum_{a=1}^{k}\frac{\Sigma_{\text{in}}^{(a)}}{m}-\left(\frac{\Sigma_{\text{tot}}^{(a)}}{m}\right)^2, \qquad (5.5)
\end{aligned}$$

where in (a) we denoted with $\mathcal{V}_a = \{i : \ell_i = a\}$, in (b) we denoted with $d_i^{(a)}$ the number of connections node i has with nodes in community a and in (b,c) $\Sigma_{\text{in}}^{(a)}, \Sigma_{\text{tot}}^{(a)}$ are the number of connections among nodes[4] in community a and the total number of connections nodes in community a have. From Equation (5.5), we can write the change of modularity obtained by moving a node i (that forms a community on its own) to class a to which one of its neighbors belongs to:

$$\Delta\mathcal{Q}_A^{\text{Mod}}(\boldsymbol{\ell}) = \underbrace{\left[\frac{\Sigma_{\text{in}}^{(a)}+d_i^{(a)}}{m}-\left(\frac{\Sigma_{\text{tot}}^{(a)}+d_i}{m}\right)^2\right]}_{\text{new modularity}} - \underbrace{\left[\overbrace{\frac{\Sigma_{\text{in}}^{(a)}}{m}-\left(\frac{\Sigma_{\text{tot}}^{(a)}}{m}\right)^2}^{\text{modularity from } a} - \overbrace{\left(\frac{d_i}{m}\right)^2}^{\text{modularity from } i}\right]}_{\text{old modularity}}$$

4 Note that each edge is counted twice.

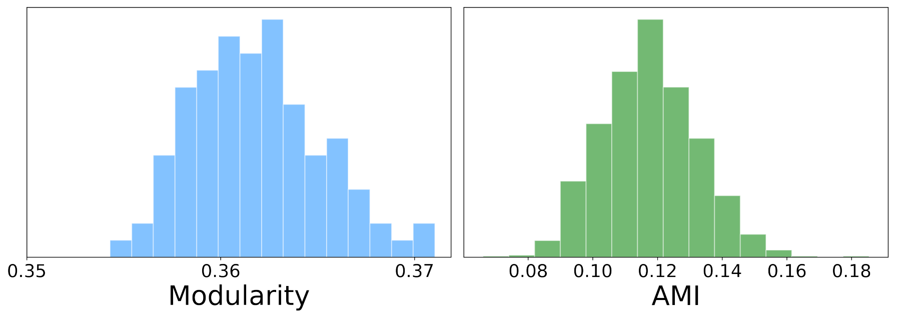


Figure 5.4: **Evaluation of Louvain algorithm**. The Louvain algorithm was run 100 times on a random graph with communities. The left plot reports the histogram of the modularity obtained after each run. In the right plot we selected 1000 random pairs and computed for all of them the AMI and plotted the histogram.

Note that the order in which the merging are attempted matters to determine the outcome and this is the random component of this algorithm. Figure 5.3 summarizes the two steps of the Louvain algorithm.

Computational complexity

The evaluation of $\Delta\mathscr{Q}_A^{\text{Mod}}(\ell)$ is performed in a constant number of operations, *i.e.* independent of n, k and it has to be performed at most $|\mathcal{E}|$ times (each node for each of its neighbors). After the first iteration this operation becomes even faster, and the algorithm's complexity is hence $\mathcal{O}(|\mathcal{E}|)$ making it particularly appealing for large networks.

5.2.3 PITFALLS OF OPTIMIZATION APPROACHES

Defining communities according to a score function may seem a particularly good strategy because it gives a common sense definition of communities, expressing a property of a good class assignment. This approach has, however, strong algorithmic limitations.

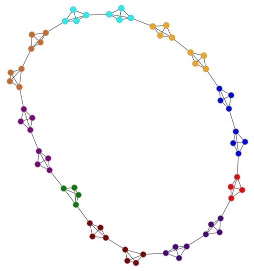
- Optimization problems such as NCut, RCut and the modularity maximization are NP hard and only approximate solutions can be obtained by efficient algorithms. For this reason, we must inderline that the output of the Louvain algorithm (or any other modularity maximization technique) is **not** guaranteed to be the maximum modularity partition.
- There exists a large number of structurally different configurations having values of $\mathscr{Q}_A^\bullet$ value very close to the maximum. Running multiple times an approximate algorithm looking for the optimum of $\mathscr{Q}_A^\bullet$ may output similar results in terms of the score, but corresponding to rather different label assignments, as shown in Figure 5.4. This makes the use of greedy algorithms, such as *Louvain*, potentially unreliable.

- Talking about the modularity, it is known that even in the presence of well defined clusters (such as *cliques*), optimizing the modularity score over the number of clusters k, small communities may be joined together, causing the so-called *resolution limit*. This is a consequence of the fact that the values of $\mathscr{Q}_A^{\text{Mod}}$ are not directly comparable for different values of k. Even if formal results have been derived for the popular modularity, similar effects are expected to be seen also for other cost functions. To circumvent this problem it was introduced a generalized modularity, depending on a positive regularizer γ:

$$\mathscr{Q}_A^{GMod}(\boldsymbol{\ell};\gamma) = \frac{1}{4|\mathcal{E}|}\sum_{i,j\in\mathcal{V}}\left(A_{ij} - \gamma\frac{d_i d_j}{2|\mathcal{E}|}\right)\delta(\ell_i,\ell_j).$$

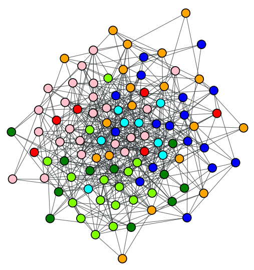
  Tuning the value γ it is possible to identify communities at different length scales, but this requires an *ad hoc* solution, depending, in general, on the underlying graph.

The resolution limit on a ring of cliques *graph: the colour is the assignment obtained maximizing the modularity*



- From the optimization perspective, communities can be defined even on graphs with no community structure, such as *Erdős-Rényi* (ER) random graphs. This is not only a philosophical problem, related to the fact that a good CD algorithm should be capable of detecting whether or not communities are present on the graph. In fact, considering for instance the modularity, one expects that on ER graphs, any partition satisfies $\mathscr{Q}_A^{\text{Mod}} \approx 0$. It has however been shown that high modularity partitions can be found on ER graphs, evidencing that the modularity maximization may lead to over-fitting.

An ER *graph with a partition in* 34 *communities and modularity* 0.52



These problems altogether are severe limitations of the optimization approach to CD and justify the adoption of a different strategy, based on inference from the *Degree corrected stochastic block model* (DCSBM). We will show how Bayesian inference is able to overcome the aforementioned limitations of optimization and how it is able to motivate their origin.

5.3 INFERENCE IN THE DCSBM

The Bayesian approach relies on the formulation of an inference problem from a generative model of the network. In fact, we suppose that there exists a model $\mathbb{P}(A|\boldsymbol{\ell})$ creating the adjacency matrix given the node partition into communities and our goal is to estimate $\boldsymbol{\ell}$ from an observation of A. The Bayes formula then reads

The Bayes formula

$$\overbrace{\mathbb{P}(\boldsymbol{\ell}|A)}^{\text{Posterior}} = \frac{\overbrace{\mathbb{P}(A|\boldsymbol{\ell})}^{\text{Likelihood}}\ \overbrace{\mathbb{P}(\boldsymbol{\ell})}^{\text{Prior}}}{\underbrace{P(A)}_{\text{Evidence}}}\,.$$

In CD we generally suppose a uniform prior, since no information is available on the community structure. We now define the DCSBM model to generate a random graph with communities.

5.3.1 THE DEGREE CORRECTED STOCHASTIC BLOCK MODEL

The DCSBM can be seen as a generalization of the configuration model.

The degree corrected stochastic block model

Definition 5.1. *Let $\boldsymbol{\ell} \in \{1,\dots,k\}^n$ be the class label vector, where k is the number of classes and $\mathbb{P}(\ell_i = a) = \pi_a$ and let $C \in \mathbb{R}^{k\times k}$ be a symmetric matrix with positive elements. Let $\theta \in \Theta = [\theta_{\min}, \theta_{\max}]$ be a random variable that encodes the intrinsic node connectivity, distributed according to ν, satisfying $\mathbb{E}[\theta] = 1, \mathbb{E}[\theta^2] = \Phi = O_n(1)$. For each node, θ_i is drawn independently at random from ν.*

The entries of the matrix $A_{ij} = A_{ji}$ are set to one independently at random with probability

$$\mathbb{P}(A_{ij} = 1) = \min\left(\theta_i\theta_j \frac{C_{\ell_i,\ell_j}}{n}, 1\right),$$

and are equal to zero otherwise.

Interpreting the DCSBM

From a simple computation, one can see that the expected average degree of node i is proportional to θ_i, *i.e.* $\mathbb{E}[d_i] \propto \theta_i$. Consequently the vector $\boldsymbol{\theta} = (\theta_1,\dots,\theta_n)$ can be used to produce any degree distribution on the graph. The matrix C is the class affinity matrix, generating the community structure. In fact, if the diagonal elements of C are larger than the off-diagonal ones,[5] it is more likely to be connected to someone in your own community than to someone in another community. The vector $\boldsymbol{\pi} \in \mathbb{R}^k$ is defined so that π_a is the expected fraction of nodes with label a.

Given the generative model of Definition 5.1 the labels are indisputably defined by the vector $\boldsymbol{\ell}$ and the number of classes by k. Before discussing how to perform inference according to this method, let us consider a very important result about inference in the DCSBM.

5.3.2 INFORMATION THEORETIC LIMITS IN THE DCSBM

Inference cannot be performed for any values of the entries of C. Suppose the extreme case in which all entries of C are equal: the resulting graph is just a realization of the configuration model in which communities do not exist

5 Note that, if the converse is true, *i.e.* nodes get connected more often to nodes in a different community (*e.g.* adjective and nouns in a text), then we talk about *disassortativity* but communities are still defined.

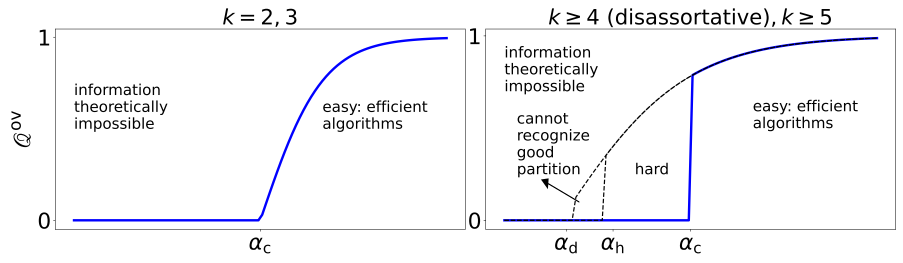


Figure 5.5: Schematic representation of the phase transition in the sparse DCSBM. The y axis represents the performance of reconstruction. Picture adapted from *Moore, The Computer Science and Physics of Community Detection.*

and hence cannot be retrieved. Actually, it was proved that there should be a minimal distance between the probability of connection within and across community to allow the graphs to be statistically distinguishable. We consider the case of $k = 2$ communities in which the diagonal entries of C are c_{in}, while the off-diagonal are c_{out} in which we have a proper phase transition, determined by the existence of a *detectability threshold.* We formulate this formally in the following theorem.

The detectability threshold

Theorem 5.1. *Consider a graph generated by the DCSBM with $k = 2$ communities. Let the diagonal entries of C be c_{in} and the off-diagonal ones be c_{out} and denote with $c = (c_{\text{in}} + c_{\text{out}})/2$ the expected average degree. Let the control parameter α be*

$$\alpha = (c - c_{\text{out}})\sqrt{\frac{\Phi}{c}}, \tag{5.6}$$

then detection is feasible if and only if $\alpha > 1$.

In the case of $k > 2$ there are conjectures that state the existence of three regions: *undetectable* in which it is impossible to make reconstruction; *hard* in which if we initialize the Bayes estimator to the ground truth we could obtain the good solution, but not for an arbitrary intial condition; *easy* in which the communities can be recovered, as summarized in Figure 5.5.

5.3.3 DCSBM BAYESIAN INFERENCE

Using the uniform prior and assuming $\theta_i\theta_j C_{\ell_i,\ell_j}/n < 1$ for all i, j,[6] the posterior distribution reads:

$$\begin{aligned}\mathbb{P}(\boldsymbol{\ell}|A) &\propto \prod_{(ij)\in\mathcal{E}} \theta_i\theta_j \frac{C_{\ell_i,\ell_j}}{n} \cdot \prod_{(ij)\notin\mathcal{E}} \left(1 - \theta_i\theta_j\frac{C_{\ell_i,\ell_j}}{n}\right) \\ &\propto \exp\left\{\sum_{(ij)\in\mathcal{E}} \log\left(C_{\ell_i,\ell_j}\right) + \sum_{(ij)\notin\mathcal{E}} \log\left(1 - \theta_i\theta_j\frac{C_{\ell_i,\ell_j}}{n}\right)\right\}. \end{aligned} \tag{5.7}$$

6 This can be done without loss of generality in the limit for $n \to \infty$.

Note that the vector $\boldsymbol{\theta}$ can be easily estimated from the degree distribution and can be used as a known variable. Obtaining the marginal node probability from Equation (5.7), one can assign to each node the label maximizing the node marginal. A possible way to accomplish this task is to sample from the distribution (5.7) using Monte Carlo Markov chains. The Bayes optimal procedure is, however, typically quite expensive from the computational viewpoint. Luckily, for sparse graphs, the asymptotically exact expression of $\mathbb{P}_i(\ell_i|A)$ can be efficiently obtained using the cavity method discussed in Chapter 3. The computational complexity of the cavity method for CD scales as $O(|\mathcal{E}|k^2)$, making it computationally efficient in the sparse regime in which $c = O_n(1)$ (or, equivalently, $|\mathcal{E}| = O_n(n)$). The main interest in the cavity method, however, comes from the fact that it is asymptotically exact on sparse graphs.

5.3.4 OPTIMIZATION VS MODEL BASED: A STATISTICAL PHYSICS PERSPECTIVE

To conclude this section, let us relate the model-based and optimization-based approaches.

As it was described in Section 5.2, defining communities as the solution to an optimization problem makes a requirement on what a good class partition should be like, with no hypothesis on the underlying graph. This makes it a seemingly good way of performing CD on arbitrary graphs. On the opposite, designing an algorithm for CD inspired from DCSBM gives a good mathematical control and, in some cases, information theoretic guarantees. Nevertheless, the model-based approach relies on some assumptions that are not necessarily verified on arbitrary graphs. This may lead to thinking of the inference approach as a mere mathematical exercise. This section on the opposite argues that the model-based approach should be generally preferred to the optimization one. In fact, the latter actually relies on some implicit hypothesis on the matrix A and its limitations can be clearly interpreted from a Bayesian perspective.

To simplify the discussion, let us consider the $k = 2$ class DCSBM. In this case, letting $\sigma_i = 1$ if $\ell_i = 1$ and $\sigma_i = -1$ if $\ell_i = 2$, the posterior probability $\mathbb{P}(\boldsymbol{\ell}|A) \equiv \mathbb{P}(\boldsymbol{\sigma}|A)$ of Equation (5.7) can be rewritten for $n \to \infty$ as

$$\mathbb{P}(\boldsymbol{\sigma}|A) \propto \exp\left\{ \sum_{(ij)\in\mathcal{E}} \log\left(C_{\ell_i,\ell_j}\right) - \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\notin\partial i} \theta_i\theta_j \frac{C_{\ell_i,\ell_j}}{n} \right\}$$

$$\propto \exp\left\{ \sum_{(ij)\in\mathcal{E}} \log(c_{\text{in}})\frac{1+\sigma_i\sigma_j}{2} + \log(c_{\text{out}})\frac{1-\sigma_i\sigma_j}{2} - \frac{1}{2}\sum_{i\in\mathcal{V}}\sum_{j\notin\partial i}\theta_i\theta_j\left[\frac{c_{\text{in}}}{n}\frac{1+\sigma_i\sigma_j}{2} + \frac{c_{\text{out}}}{n}\frac{1-\sigma_i\sigma_j}{2}\right]\right\}$$

$$\propto \exp\left\{ \sum_{(ij)\in\mathcal{E}} \underbrace{\frac{1}{2}\log\left(\frac{c_{\text{in}}}{c_{\text{out}}}\right)}_{\beta}\sigma_i\sigma_j - \sum_{i\in\mathcal{V}}\sigma_i \underbrace{\sum_{j\notin\partial i}\theta_i\theta_j\frac{c_{\text{in}}-c_{\text{out}}}{4n}\sigma_j}_{h_i(\sigma)}\right\}$$

$$\propto \exp\left\{ \sum_{(ij)\in\mathcal{E}} \beta\sigma_i\sigma_j - \sum_{i\in\mathcal{V}} h_i(\sigma)\sigma_i\right\} \equiv e^{-\beta\mathcal{H}(\sigma)}. \tag{5.8}$$

Equation (5.8) precisely corresponds to the Boltzmann distribution for the Ising Hamiltonian with local fields, depending on the configuration $\boldsymbol{\sigma}$. The Bayesian approach is equivalent to finding the magnetization $\boldsymbol{m} = \langle\boldsymbol{\sigma}\rangle_\beta$, associated to the Hamiltonian $\mathcal{H}(\boldsymbol{\sigma})$. Finding the ground state of $\mathcal{H}(\boldsymbol{\sigma})$, *i.e.* the configuration $\boldsymbol{\sigma}$ corresponding to its minimum, instead is equivalent to finding the maximum of the generalized modularity $\mathscr{Q}_A^{GMod}(\boldsymbol{\ell};\gamma)$, in fact, in the large n limit for sparse graphs $\sum_{j\notin\partial i} \approx \sum_{j\in\mathcal{V}}$ and thus

$$\frac{\mathbb{E}\left[\mathcal{H}(\boldsymbol{\sigma})\right]}{\beta} = \sum_{i,j\in\mathcal{V}}\left(A_{ij} - \frac{2(c_{\text{in}}-c_{\text{out}})}{(c_{\text{in}}+c_{\text{out}})\beta}\frac{\mathbb{E}[d_i]\mathbb{E}[d_j]}{m}\right)\sigma_i\sigma_j,$$

which is closely related to the regularized modularity.

This observation puts us in position to make two very important remarks. The most questionable assumption of the DCSBM is that of generating edges independently at random. In terms of log-likelihood, this translates into a sum over all graph edges, as shown in Equation (5.8). This sum is the same appearing in $\mathscr{Q}_A^{\text{GMod}}$, $\mathscr{Q}_A^{\text{Mod}}$, $\mathscr{Q}_A^{\text{RCut}}$ and $\mathscr{Q}_A^{\text{NCut}}$ that can be associated to a generative model (different from the DCSBM) in which the edges of $\mathcal{G}(\mathcal{V},\mathcal{E})$ are also generated independently at random, evidencing how the functions $\mathscr{Q}_A^\bullet$ rely on some "silent" assumptions on the matrix A.

Furthermore, from a statistical physics perspective, the functions $\mathscr{Q}_A^\bullet$ (eventually taken with a negative sign) can generally be considered as Hamiltonians, *i.e.* cost functions associated to a given label configuration. In all cases (also for the DCSBM), the Hamiltonian is what *defines* communities from a *microscopic* perspective. Stating which definition is the best is a difficult task that we are not going to investigate. However it should be remarked that what is *essentially* different in the optimization and inference approaches is how to retrieve the communities from the Hamiltonian: in one case they are obtained from the marginals of the Boltzmann distribution, in the other from the ground state energy. The Hamiltonian, or equivalently the cost $\mathscr{Q}_A^\bullet$ is what *defines* the concept of communities. The optimization approach, however, only takes into account the minimum of the Hamiltonian, disregarding the rest of its profile. Consider two functions $\mathscr{Q}_A^\bullet$, one being convex and the other having multiple minima with similar values of $\mathscr{Q}_A^\bullet(\boldsymbol{\ell})$. These two settings are clearly different: in the first the label assignment is uniquely defined by minimum of $\mathscr{Q}_A^\bullet$, whereas, in the second, several configurations could be considered as almost equally good community structures. Taking

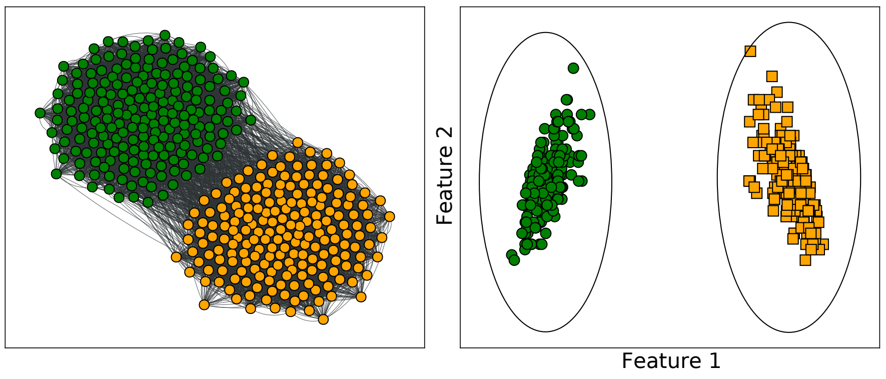


Figure 5.6: **Left**: a graph with two communities, highlighted with a different color code. **Right**: a 2 dimensional node embedding of the graph on the left. The embedding dimensions are here denoted with as *features*.

only the minimum of the Hamiltonian means to disregard all other configurations that may have, instead, a potentially great importance. The Bayesian approach does not consider exclusively the minimum of $\mathscr{Q}^{\bullet}_{A}$, but the whole energy landscape, giving a generally richer description of the problem.

5.4 SPECTRAL CLUSTERING

Networks are complex mathematical entities that are hard to represent hence to deal with. The difficulty of defining communities is a direct consequence of this complexity of representation. A powerful method to perform network analysis that can also be adapted to CD is to perform an embedding, *i.e.* in providing a representation of the network in a Euclidean space.

Node embedding

Given a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, a node embedding consists in identifying a mapping $f : \mathcal{V} \to \mathbb{R}^d$, where d is the embedding dimension. Each node i is associated with a vector $\boldsymbol{x}_i \in \mathbb{R}^d$, called *embedding vector*.

An embedding is defined so that nodes that are structurally similar (for instance, that belong to the same community) are represented with similar vectors. In Figure 5.6 we give a pictorial representation of a node embedding applied to a graph with two communities. Once the graph has been represented as a set of points in a Euclidean space, CD simply translates into *clustering*, *i.e.* the objective of grouping together points in a high dimensional space. Several algorithms exist to accomplish this task and can be divided in two groups: those in which partitions are attributed solving an optimization problem such as *k-means*, *k-medoids*, *expectation maximization*; those in which boundaries between clusters are drawn where the density of points is minimal, such as *DBSCAN* and *OPTICS*.

Clustering

In this section we introduce *Spectral clustering* (SC) that is one of the most well studied class of algorithms to perform embeddings for CD and it has strong relations with both the optimization and Bayesian approaches.

5.4.1 VANILLA SPECTRAL CLUSTERING

In SC algorithms the embeddings are obtained with the eigenvectors of a suited graph matrix representation, M. Defining what *suited* means is a hard task and several possible choices are plausible. One possibility would be $M = A$, the adjacency matrix, while one of the most popular (but not necessarily one of the best) if the graph Laplacian matrix $L = D - A$. SC benefits from solid theoretical foundations and high explainability. All SC algorithms can be reduced to a very similar structure[7] that we summarize in Algorithm 5.1. Here the number of clusters k is required as an input but, as we will see in the next sections, SC algorithms provide methods to estimate k in a theoretically well grounded way. The basic intuition of SC is that *some* eigenvectors of M are *informative* and carry information about the graph community structure. These eigenvectors are typically considered to be as many as the number of communities.

Algorithm 5.1 : Spectral clustering

```
  Input : Dataset with $n$ items, $k$ number of clusters
  Output : $\ell \in \{1, \dots, k\}^n$ label assignment
1 begin
2     Define suited matrix representation of the dataset $M \in \mathbb{R}^{n \times n}$;
3     Stack the $k$ largest (or smallest) eigenvalues of $M$ in the columns
        of $X \in \mathbb{R}^{n \times k}$ (Embedding);
4     Estimate community labels $\ell$ with a small dimensional clustering
        algorithm performed on the rows of $X$ (Clustering) ;
5     return $\ell$
6 end
```

The complexity of Algorithm 5.1 scales with $\mathcal{O}(|\mathcal{E}|k^2)$ that is the number of operations required to compute the eigenvectors. The clustering step typically requires a lower number of operations. This complexity thus scales well with the matrix size, but may become prohibitive when considering graphs with a very large number of communities. We proceed motivating Algorithm 5.1, describing its relation with other techniques adopted for CD.

7 Note that this is not a strict rule and there exist some SC that follow a structure that is similar to the one of Algorithm 5.1, but it is not exactly the same.

5.4.2 THE RELATION WITH OPTIMIZATION APPROACHES

SC has a strong relation with optimization algorithms. We explicitly derive this formal relation for the *RatioCut* optimization problem and briefly overview the results for other optimization functions.

Lemma 5.1. *Consider a node partition $\boldsymbol{\ell}$ on an undirected and unweighted graph with adjacency matrix A. Let $L = D - A$ be its associated graph Laplacian matrix. Let $H \in \mathbb{R}^{n\times k}$ be the matrix with entries $H_{ia} = \delta_{\ell_i,a}/\sqrt{V_a}$, where $V_a = |\mathcal{V}_a|$. Then*

$$\mathcal{Q}_A^{\mathrm{RCut}}(\boldsymbol{\ell}) = \frac{1}{2}\mathrm{Tr}(H^T L H).$$

Proof. Exploiting Lemma 4.1, we can write

$$\begin{aligned}
\left(H^T L H\right)_{aa} &\overset{(a)}{=} \boldsymbol{h}_a^T L \boldsymbol{h}_a \\
&= \frac{1}{2}\sum_{i,j\in\mathcal{V}} A_{ij}\left(H_{ia} - H_{ja}\right)^2 \\
&= \frac{1}{2}\sum_{\alpha=1}^{k}\sum_{\beta=1}^{k}\sum_{i\in\mathcal{V}_\alpha}\sum_{j\in\mathcal{V}_\beta} A_{ij}\left(\frac{\delta_{\alpha,a}}{\sqrt{V_a}} - \frac{\delta_{\beta,a}}{\sqrt{V_a}}\right)^2 \\
&= \frac{1}{2V_a}\sum_{\alpha=1}^{k}\sum_{\beta=1}^{k}\sum_{i\in\mathcal{V}_\alpha}\sum_{j\in\mathcal{V}_\beta} A_{ij}\left(\delta_{a,\alpha} + \delta_{a,\beta} - 2\delta_{a,\alpha}\delta_{a,\beta}\right) \\
&\overset{(b)}{=} \frac{1}{V_a}\left[\sum_{\beta=1}^{k}\sum_{i\in\mathcal{V}_a}\sum_{j\in\mathcal{V}_\beta} A_{ij} - \sum_{i\in\mathcal{V}_a}\sum_{j\in\mathcal{V}_a} A_{ij}\right] \\
&= \frac{1}{V_a}\sum_{\beta\neq a}\sum_{i\in\mathcal{V}_a}\sum_{j\in\mathcal{V}_\beta} A_{ij} \\
&= \frac{1}{V_a}\sum_{i\in\mathcal{V}_a}\sum_{j\notin\mathcal{V}_a} A_{ij}\,,
\end{aligned}$$

where in (a) in denoted with $\boldsymbol{h}_a$ the a-th column of H and in (b) we exploited that A is symmetric. To conclude the proof, we simply recall the definition of the *RatioCut*

$$\begin{aligned}
\mathcal{Q}_A^{\mathrm{RCut}}(\boldsymbol{\ell}) &= \frac{1}{2}\sum_{a=1}^{k}\frac{1}{V_a}\sum_{i\in\mathcal{V}_a}\sum_{j\notin\mathcal{V}_a} A_{ij} \\
&= \frac{1}{2}\sum_{a=1}^{k}\left(H^T L H\right)_{aa} \\
&= \frac{1}{2}\mathrm{Tr}\left(H^T L H\right)\,.
\end{aligned}$$

□

Lemma 5.1 puts in direct relation the optimization of the *RatioCut* function with the graph Laplacian matrix. Yet, however we may formulate it, this problem is still NP-hard, hence it cannot be easily solved. The SC approach, however, allows one to obtain an approximate solution, by relaxing the optimization problem from a discrete set to the whole real axis. First notice that $H^TH = I_k$, then the relaxation of the *RatioCut* optimization problem is obtained by solving

Relaxed optimization

$$X = \underset{H \in \mathbb{R}^{n\times k}\,:\, H^TH = I_k}{\arg\min} \quad \mathrm{Tr}(H^TLH)\,. \tag{5.9}$$

The solution of this optimization problem is the matrix X storing in its columns the k eigenvectors of L with smallest eigenvalues. We stress that this is not an exact solution because the optimization problem is not run over all the *discrete* values that H can take, but over all the real space. To summarize, with reference to Algorithm 5.1, here we choose $M = L$ and extract the k smallest eigenvalues of L to obtain the embedding.

Similarly to the derivation detailed above, one can show that the *Normalized Cut* can be approximated by SC using the k eigenvectors associated with the smallest eigenvalues of $L^{\mathrm{rw}} = I_n - D^{-1}A$, or equivalently the k largest of $D^{-1}A$. Often, instead of considering L^{rw} the matrix $L^{\mathrm{sym}} = I_n - D^{-1/2}AD^{-1/2}$ is preferred. This matrix has the same eigenvalues of L^{rw} and its eigenvectors are closely related but it has the advantage of being symmetric. By relaxing the modularity, instead, one can define a SC algorithm that exploits the eigenvectors of the modularity matrix $A - \frac{dd^T}{2|\mathcal{E}|}$ that are closely related to the ones of the adjacency matrix itself.

5.4.3 A RANDOM MATRIX PERSPECTIVE

The previous section justified SC from the perspective of optimization algorithms. We now show its relation with random generative models. We will here consider the particular case of a graph generated from the DCSBM of Definition 5.1 model with $k = 2$ classes and and a homogeneous degree distribution. This choice is only made for simplicity. We will then study the spectral properties of the adjacency matrix of a graph generated from this model. We use in particular the notation $C_{ab} = c_{\mathrm{in}}$ if $a = b$ and c_{out} otherwise. The first step consists in writing the random matrix A as the sum of its expectation and white noise:

$$A = \underbrace{\mathbb{E}[A]}_{\text{expectation}} + \underbrace{X}_{\text{white noise}}\,.$$

The expectation $(\mathbb{E}[A])_{ij} = C_{\ell_i,\ell_j}/n$ is a low rank matrix with only two eigenvalues that are non-zero. The leading one equal to $c = (c_{\mathrm{in}} + c_{\mathrm{out}})/2$ (the expected average degree) with eigenvector $\mathbf{1}_n$ and the second one equal to $(c_{\mathrm{in}} - c_{\mathrm{out}})/2$ with eigenvector $\boldsymbol{\sigma}$, where $\sigma_i = 1$ if $i \in \mathcal{V}_1$ and $\sigma_i = -1$

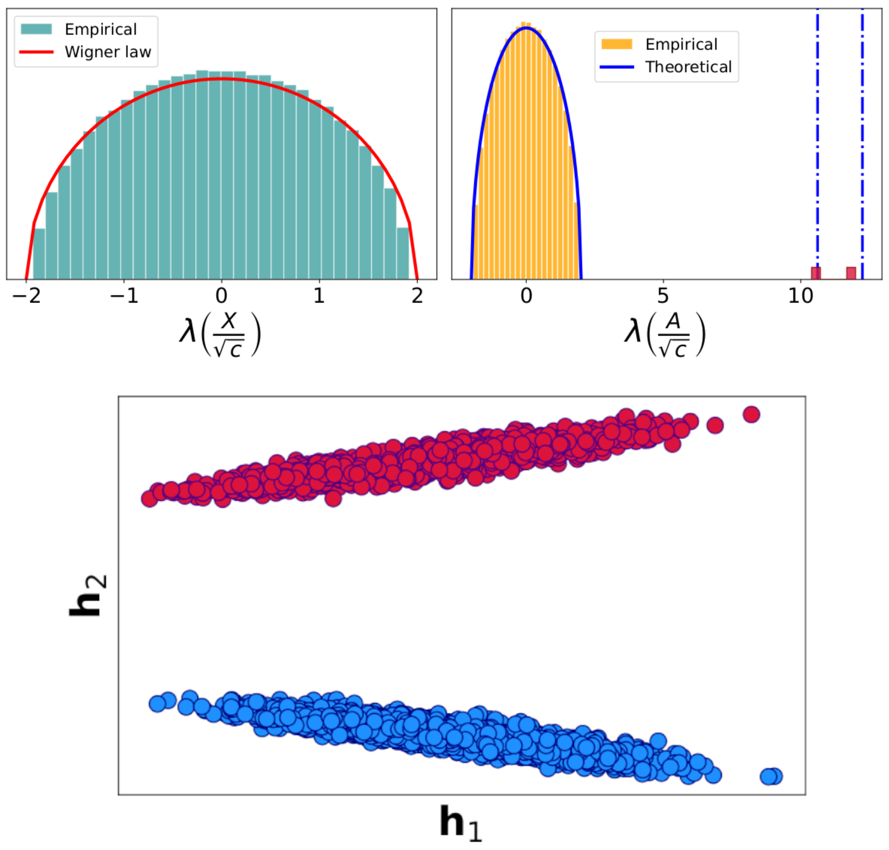


Figure 5.7: **Visualization of spectral properties of the adjacency matrix**. *Top left panel*: spectrum of the centered adjacency matrix X. *Top right panel*: spectrum of the adjacency matrix A with a zoom inset on the largest isolated eigenvalues. *Bottom panel*: two dimensional embedding obtained from the eigenvectors associated to the two largest eigenvalues of A that correspond to the isolated ones shown in the inset. The color code refers to the ground truth community label of the generative model.

else. The matrix X instead has zero mean, finite variance and is a noise matrix. If $c \to \infty$ – *i.e.* , if we are in a dense regime – the law of the eigenvalues of X converges in distribution to the semi-circle Wigner law, bounded between -2α and 2α, as shown in the left panel of Figure 5.7. Now, we have two matrices, $\mathbb{E}[A]$ and X of which we know the spectral properties, but what can we say about their sum? Bauer-Fike theorem comes to help us finding an answer

Theorem 5.2 (Bauer-Fike theorem for Hermitian matrices)**.** *Consider a Hermitian matrix $\tilde{A}$ and matrix X. Letting μ be an eigenvalue of $\tilde{A} + X$, then there exists an eigenvalue λ of $\tilde{A}$ so that*

$$|\lambda - \mu| \leq \rho(X),$$

where $\rho(\cdot)$ denotes the spectral radius.

In simple words, this theorem provides a bound to how much the eigenvalues of a perturbed matrix can differ from those of its unperturbed version. From this we know that the maximal distance between the eigenvalues of A and the eigenvalues of $\mathbb{E}[A]$ is, at most, equal to $\rho(X) = 2\sqrt{c} + o_n(\sqrt{c})$ with high probability. The spectrum of the matrix A will be composed by two eigenvalues coming from $\mathbb{E}[A]$ that are close to the eigenvalues of $\mathbb{E}[A]$. We call these eigenvalues *isolated*. The remaining ones are the *bulk* eigenvalues and follow the Wigner semi-circle law.

Isolated and bulk eigenvalues

The eigenvectors associated with the *isolated eigenvalues* of A will be strongly correlated with the eigenvectors of $\mathbb{E}[A]$. These eigenvectors, however, are piece-wise contact and in particular, have the same value for all nodes in the same community. Recall in particular the definition of σ. These eigenvectors thus project each node to a point in a low dimensional space that depends on the community structure, as shown in the lower panel of Figure 5.7. It is very important to stress that all this argument holds if the expected average degree c goes to infinity. In this case, the isolated eigenvalues are of order $\mathcal{O}(c)$ and the perturbation is of order $\mathcal{O}(\sqrt{c})$. Consequently, the relative variation scales as $\mathcal{O}(c^{-1/2})$ and vanishes only for graphs that are sufficiently dense. As we will discuss in the next paragraph, this is not the case for sparse graphs.

Remark

The argument we made can be extended to an arbitrary number of communities k and shows why in SC the embedding dimension is often chosen to be equal to the number of classes. In fact, the number of isolated *informative* eigenvalues equals the rank of $\mathbb{E}[A]$ that is equal to k. Now, the fact that these eigenvalues are isolated – *i.e.* far from all others – is fundamental for SC to work well. If this is not the case, it means that the noise – represented by the bulk – "covers" the information contained in the isolated eigenvalues and reconstruction is not feasible.

In conclusion, the random matrix approach motivates SC by showing that the low-rank mesoscale structure of a graph with communities can be recovered from few eigenvectors of a proper graph matrix representation. We showed the argument flow for $k = 2$ classes and a homogeneous degree distribution, but everything can be extended to a more general scenario. Moreover, the same approach can be used – even if it is mathematically more challenging – to the use of other matrices, such as $D - A$, or $D^{-1}A$, with results that are qualitatively very similar.

5.4.4 SPECTRAL CLUSTERING IN SPARSE GRAPHS

As we mentioned in the previous section, the random matrix approach works well when considering dense graphs, but the theoretical results do not hold

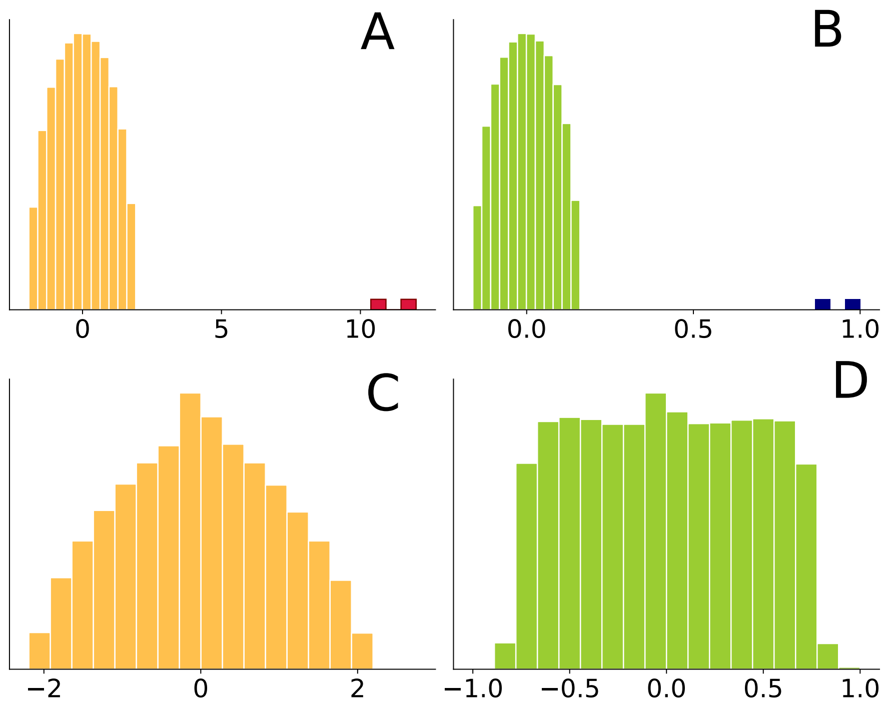


Figure 5.8: **Spectral behavior of graph matrix representations in the sparse and dense regimes**. We consider two graphs with two communities with a large (panels *A* and *B*) and a small (panels *C* and *D*) average degree. In the panels *A* and *C* we show the spectrum of the adjacency matrix A and in panels *B* and *D* the spectrum of $D^{-1/2}AD^{-1/2}$. In the case of dense graphs (*A* and *B*), we highlighted (and zoomed) the two isolated eigenvalues. For sparse graphs, these eigenvalues do not exist.

in the sparse regime in which the expected average degree is independent of the graph size, *i.e.* $c = \mathcal{O}_n(1)$. Now, this is not only a theoretical limitation as shown in Figure 5.8 in which we show that the well behaved spectral behavior of A and $D^{-1/2}AD^{-1/2}$ in the dense regime is not replicated in the sparse one. In the sparse regime that characterizes most real-world networks, SC is known to be hard to deploy. Yet, we here provide three choices of M – referring to Algorithm 5.1 – that recently proved to be very efficient to perform SC in sparse (but also dense) graphs.

The non-backtracking matrix

As we intuitively hinted in Chapter 2, the adjacency matrix naturally appears when we perform the *naïve mean field* (NMF) approximation of a probability distribution, while in Chapter 3 we showed that the non-backtracking matrix naturally appears when using the *belief propagation* (BP) or *cavity* approximation. The NMF is appropriate on dense graphs and – intuitively – the spectrum of A is well behave for these graphs and $M = A$ is a good choice for SC. This is not true for sparse graphs, as we briefly showed in the previous section. It can be shown that the non-backtracking matrix, instead, is a good

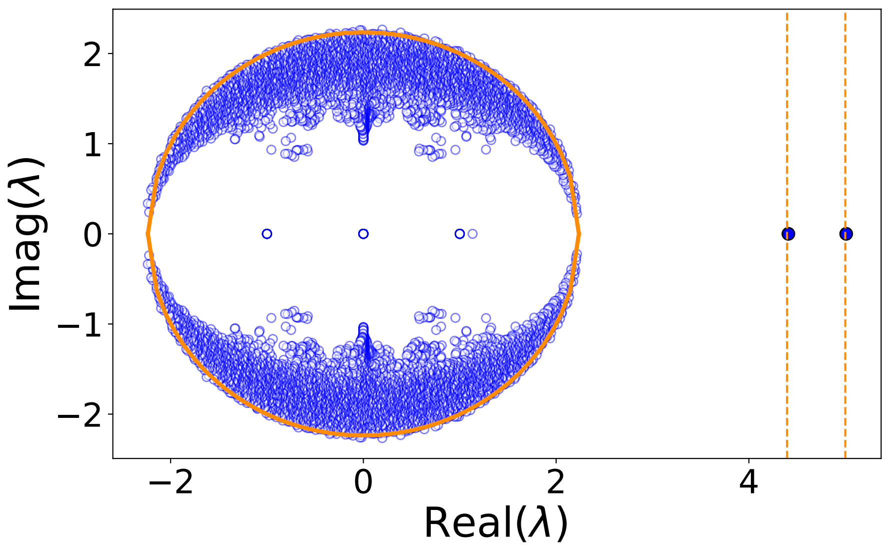


Figure 5.9: **Non-backtracking matrix spectrum in the complex plane**. Scatter plot of the real vs imaginary part of the eigenvalues of B for a graph obained from the DCSBM model with $k = 2$ communities in the sparse regime. The blue dots are the eigenvalues of the matrix, while the orange lines are the theoretic prediction as per Theorem 5.1.

choice for M, given that BP is asymptotically exact on sparse graphs. Let us first recall the definition of the non-backtracking matrix $B \in [0,1]^{2|\mathcal{E}|\times 2|\mathcal{E}|}$:

The non-backtracking matrix

$$B_{(ij),(kl)} = \delta_{jk}(1-\delta_{ik}). \tag{5.10}$$

Each index of B corresponds to *directed* edge of $\mathcal{G}(\mathcal{V},\mathcal{E})$ even if the graph is undirected. The entries are non zero when they correspond to two edges that are adjacent but they are non-backtracking, *i.e.* $B_{(ij),(ji)} = 0$. The spectrum of the matrix B can be divided, even in the sparse regime, into isolated and bulk eigenvalues, as shown in Figure 5.9. Note that, since B is not Hermitian, its eigenvalues are defined on the complex plane. In particular, all isolated eigenvalues are real, while the bulk one may have a non-zero imaginary part. We can state this result formally as per the following theorem.

Theorem 5.3. *Let $\mathcal{G}(\mathcal{V},\mathcal{E})$ be a graph generated from the DCSBM of Definition 5.1. Denote with $\lambda_p(\cdot)$ the p-th largest eigenvalue of a matrix and with $\Pi = \mathrm{diag}(\boldsymbol{\pi}) \in \mathbb{R}^{k\times k}$. Suppose that:*

- $C\Pi\mathbf{1}_k = c\mathbf{1}_k$ *where $c = \mathcal{O}_n(1)$ is the expected average degree*
- *all eigenvalues of $C\Pi$ are so that $\lambda_p(C\Pi)\Phi > c\Phi$.*

Then, the following relations are satisfied with high probability:

$$\forall\, p \in [k], \quad \lambda_p(B) = \lambda_p(C\Pi)\Phi + o_n(1)$$
$$\forall\, p > k, \quad |\lambda_p(B)| \leq \sqrt{c\Phi} + o_n(1).$$

Let us take a moment to comment this theorem. Firstly, one can easily verify that the assumption $C\Pi \mathbf{1}_k = c\mathbf{1}_k$ implies that the expected average degree c does not depend on the class, regardless of its size and of how they are connected. Secondly, the bound shows that the position of the isolated eigenvalues of B only depend on $C\Pi$ hence on the expectation of A,[8] while the bulk eigenvalues are confined by a circle in the complex plane. This bound is tight for $n \to \infty$ and $c = \mathcal{O}_n(1)$ so it works well also in the sparse regime. Finally, in the case of $k = 2$ communities of equal size, $\lambda_1(C\Pi) = c = \frac{c_{\text{in}}+c_{\text{out}}}{2}$ and $\lambda_2(C\Pi) = c = \frac{c_{\text{in}}-c_{\text{out}}}{2}$. The eigenvalue $\lambda_2(B)$ is isolated if

$$\frac{c_{\text{in}} - c_{\text{out}}}{2}\Phi \geq \sqrt{c\Phi},$$

that implies

$$(c - c_{\text{out}})\sqrt{\frac{\Phi}{c}} \geq 1,$$

that is precisely the *detectability threshold* of the DCSBM as per Theorem 5.1. Consequently, the non-backtracking matrix can be used to detect communities as soon as theoretically possible. To conclude, we must still solve a problem: the size of B is larger than n, hence we need to make a pre-processing on the eigenvectors before to obtained an embedding $X \in \mathbb{R}^{n\times k}$. Recalling the definition of $T \in \mathbb{R}^{n\times 2|\mathcal{E}}$ given in Chapter 2, $T_{a,(ij)} = \delta_{ia}A_{ij}$, for any $\boldsymbol{g} \in \mathbb{R}^{2|\mathcal{E}|}$ we let $\boldsymbol{g}^{\text{in}} = T\boldsymbol{g}$. By construction the vector $\boldsymbol{g}^{\text{in}} \in \mathbb{R}^n$ and it can be used to define a SC algorithm with the eigenvectors of B. Moreover, still referring to Chapter 2, we recall that $\boldsymbol{g}^{\text{in}}$ can be extracted by the first n entries of the matrix B_{p} defined in Equation (3.11) that has size $2n \times 2n$ and it can thus be efficiently computed on large graphs.

$$\underbrace{\begin{pmatrix} A & -I_n \\ D - I_n & 0 \end{pmatrix}}_{B_{\text{p}}} \begin{pmatrix} \boldsymbol{g}^{\text{in}} \\ \boldsymbol{g}^{\text{out}} \end{pmatrix} = \gamma \begin{pmatrix} \boldsymbol{g}^{\text{in}} \\ \boldsymbol{g}^{\text{out}} \end{pmatrix}.$$

The Bethe-Hessian matrix

We now show that B_{p} and the vector $\boldsymbol{g}^{\text{in}}$ are strongly related with an $n \times n$ matrix called *Bethe-Hessian*[9]

$$\begin{aligned} A\boldsymbol{g}^{\text{in}} - \boldsymbol{g}^{\text{out}} &= \gamma\boldsymbol{g}^{\text{in}} \\ (D - I_n)\boldsymbol{g}^{\text{in}} &= \gamma\boldsymbol{g}^{\text{out}} \end{aligned}$$

that leads to

$$A\boldsymbol{g}^{\text{in}} - \frac{1}{\gamma}(D - I_n)\boldsymbol{g}^{\text{in}} = \gamma\boldsymbol{g}^{\text{in}}$$

8 As an exercise, try to show that $\forall\, p \in [k], \lambda_p(C\Pi) = \lambda_p(\mathbb{E}[A])$.

9 This names comes from the fact that it can be interpreted as the Hessian matrix of the Bethe free energy of an Ising model on $\mathcal{G}(\mathcal{V}, \mathcal{E})$ at the paramagnetic point.

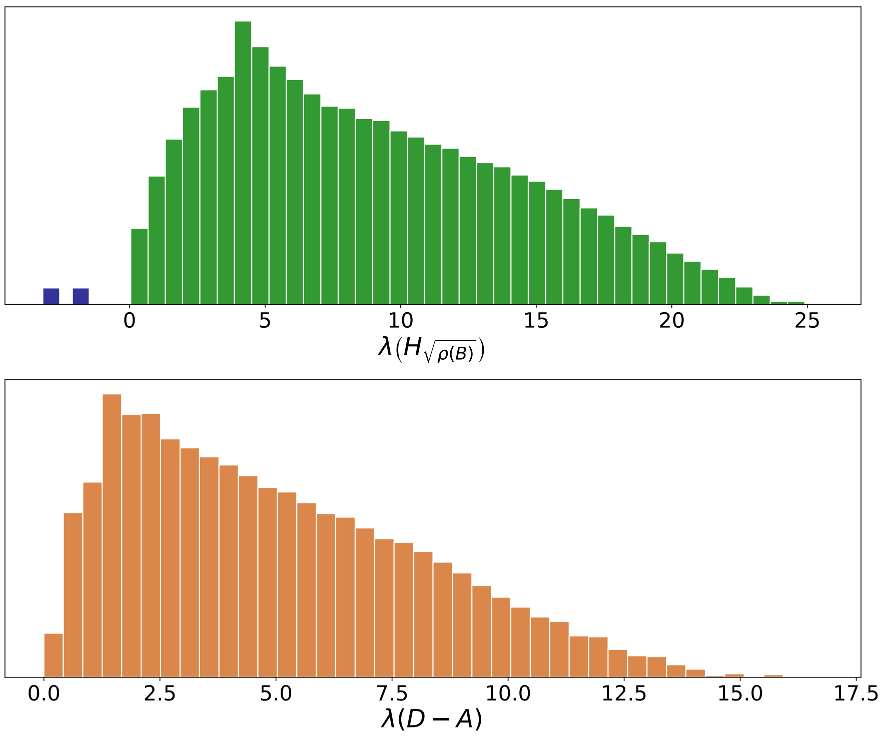


Figure 5.10: **Spectra of the Bethe-Hessian and Laplacian matrices on a sparse graph with communities**. The top row shows the spectrum of $H_{\sqrt{B}}$ for a graph with $k = 2$ communities generated from the DCSBM and expected average degree equal to 5. In blue we evidence the two isolated *negative* eigenvalues. For the same graph, the bottom plot shows the histogram of the eigenvalues of the graph Laplacian $L = D - A = H_1$.

and thus

$$\underbrace{\left[(\gamma^2 - 1)I_n + D - \gamma A\right]}_{H_\gamma} \boldsymbol{g}^{\text{in}} = 0. \tag{5.11}$$

The matrix H_γ is the *Bethe-Hessian* matrix and $\boldsymbol{g}^{\text{in}}$ is an eigenvector of H_γ is γ is an eigenvalue of B. Note that for $H_{\gamma=1} = D - A$, the graph Laplacian. Like the matrix L, the informative eigenvectors are associated with the smallest eigenvalues of H_γ and it has been shown that for some choices of γ, the algorithmic threshold of a SC algorithm based on H_γ coincides with the theoretical detectability threshold of Theorem 5.1. A particularly interesting choice is the one $\gamma = \sqrt{\rho(B)}$ that corresponds to the radius of the bulk of B, at least for random networks. For this choice SC provably achieves the detectability threshold and, interestingly, only the isolated informative eigenvalues of H_γ are negative, as shown in Figure 5.10. This provides us with a method for estimating the number of communities that is based on counting the number of negative eigenvalues of $H_{\sqrt{\rho(B)}}$ and then use the related eigenvectors to perform SC.

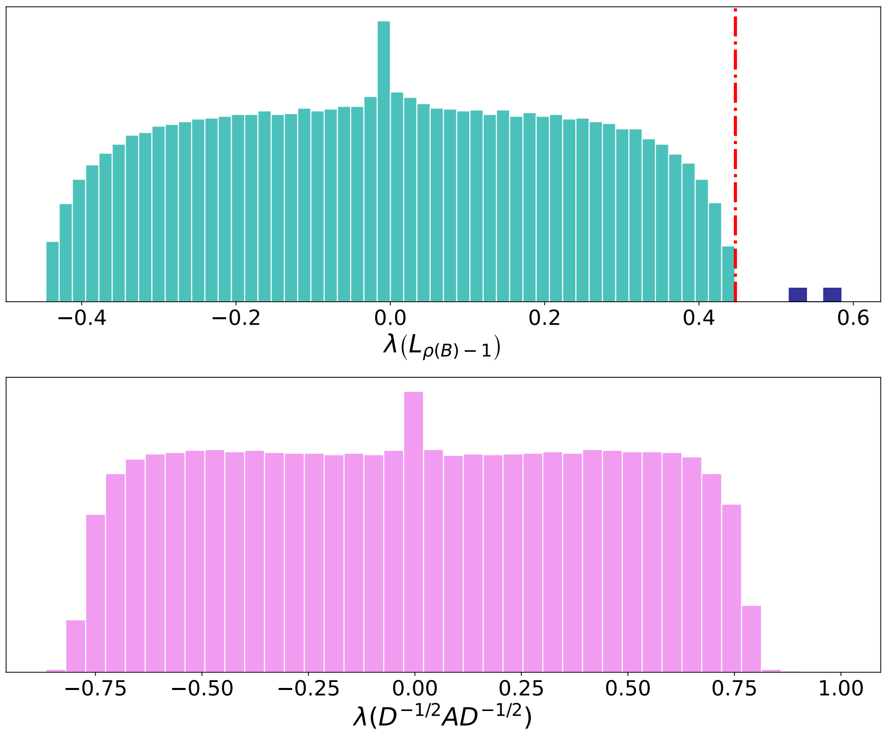


Figure 5.11: **Spectra of the regularized and non-regularized symmetric Laplacian matrices on a sparse graph with communities**. The top row shows the spectrum of $L^{\rm sym}_{\sqrt{B}}$ for a graph with $k = 2$ communities generated from the DCSBM and expected average degree equal to 5. In blue we evidence the two isolated eigenvalues, while the red dash-dotted line is located at $\rho(B)^{-1/2}$, the expected right edge of the bulk. For the same graph, the bottom plot shows the histogram of the eigenvalues of the graph symmetrized Laplacian $L^{\rm sym} = D^{-1/2}AD^{-1/2} = L^{\rm sym}_1$.

The regularized Laplacian matrix

As a final method, we still introduce one more matrix can can be used to for SC in sparse graphs. Let $D_\tau = D + \tau I_n$, then, starting from the Bethe-Hessian matrix

$$\begin{aligned}
\left[(\gamma^2-1)I_n + D - \gamma A\right] \boldsymbol{g}^{\rm in} &= 0 \\
D_{\gamma^2-1}\boldsymbol{g}^{\rm in} &= \gamma A \boldsymbol{g}^{\rm in} \\
\underbrace{D^{1/2}_{\gamma^2-1}\boldsymbol{g}^{\rm in}}_{\boldsymbol{y}} &= \gamma D^{-1/2}_{\gamma^2-1} A \boldsymbol{g}^{\rm in} \\
\boldsymbol{y} &= \gamma D^{-1/2}_{\gamma^2-1} A D^{-1/2}_{\gamma^2-1} \boldsymbol{y} \\
L^{\rm sym}_\gamma \boldsymbol{y} &= \frac{1}{\gamma}\boldsymbol{y},
\end{aligned}$$

where we introduce the normalized Laplacian matrix $L^{\rm sym}_\gamma = D^{-1/2}_{\gamma^2-1}AD^{-1/2}_{\gamma^2-1}$. Also in this case, the choice $\gamma = \sqrt{\rho(B)}$ provides good performances for spectral clustering and this is a valid choice for SC in sparse graphs. Figure 5.11 compares the spectrum of $L^{\text{sym}}_{\sqrt{\rho(B)}}$ with the one of L^{sym}_{1} that simply corresponds to the classical symmetric Laplacian matrix.

5.4.5 FINAL REMARKS ON SPECTRAL CLUSTERING

SC is a very relevant class of algorithms for CD and beyond. Several matrices can be deployed to obtained meaningful representations of the graph and this is at the same time a strength and a weakness of this approach. There is not a unique matrix "to rule them all" and if a specific choice does not produce good results, then one can look in the literature for different proposals, once the problem has been identified. Another notable point is the solid theoretical framework that can be used to characterize these algorithms, making them particularly appealing. At the same time, these results are typically derived for specific generative models (such as the DCSBM) and may not generalize well to real-world graphs, for which additional caution is needed. Finally, the computational complexity of spectral algorithms typically scales as $\mathcal{O}(|\mathcal{E}|k^2)$, the number of operations required to compute k eigenvectors of a matrix with $|\mathcal{E}|$ non-zero entries. This complexity allows one to use these algorithms on very large sparse graphs (due to the linear scaling with $|\mathcal{E}|$) but they are unsuited when approaching graphs with a large number of communities.

5.5 CONCLUSION

CD is a very important task in graph data mining but it is equally very challenging. This is primarily due to the fact that defining communities is hard per se and then, given a definition of *community* it is often hard to find an efficient algorithm to detect them. Whenever approaching CD it is important to remind that different algorithms may look for substantially different definitions of communities and may be suited on some graphs, but not on others due to their limitations. In this chapter we gave a non-extensive overview of some of the most significant methods together with their limitations. This brief introduction should raise in you a the need critical thinking when deploying a CD algorithm so to identify its potential weaknesses and interpret its results. This can be done, in practice, by deploying simultaneously several CD algorithms and compare the results to obtain an complete overview of the problem at hand. This is the approach that we will follow in the notebooks.

5.6 REFERENCES

- Fortunato: *Community detection in graphs*
  This is a fundamental (even if not so recent) review of CD algorithms. It has very interesting insights to frame the problem on a broad picture
- Von Luxburg: *A tutorial on spectral clustering*
  This is another very important review on spectral clustering, relating it to optimization problems and with some useful interpretations of SC.
- Moore: *The computer science and physics of community detection: Landscapes, phase transitions, and hardness*
  This is am article with a rather pedagogical intent on inference in the DCSBM and its relation with optimization problems with a statistical physics perspective.