# COMPLESSITÀ NEI SISTEMI SOCIALI

LORENZO DALL'AMICO
*ISI Foundation*

Note per il corso in Fisica dei sistemi complessi

Università di Torino
April 24, 2024

Lorenzo Dall’Amico: *Complessità nei sistemi sociali*

# CONTENTS

1 TEMPORAL GRAPHS 1
- 1.1 Why temporal graphs? 1
- 1.2 Representing temporal graphs 2
- 1.3 Measuring proximity graphs 5
- 1.4 Properties of temporal graphs 8
- 1.5 Conclusion 12
- 1.6 References 12

2 EPIDEMICS ON NETWORKS 13
- 2.1 Epidemics 13
- 2.2 Epidemic modeling on networks 16
- 2.3 Extensions 22
- 2.4 References 23

3 CAVITY METHOD 25
- 3.1 Sparse and tree-like graphs 25
- 3.2 Factorizing probability distributions on trees 27
- 3.3 The non-backtracking matrix 31
- 3.4 An application to epidemics 32
- 3.5 Efficiently computing the spectral radius of $B$ 34
- 3.6 Conclusion 35
- 3.7 References 35

4 GRAPH FOURIER TRANSFORM 37
- 4.1 The Fourier transform 37
- 4.2 The graph Fourier transform 38
- 4.3 Graph signal processing 44
- 4.4 Conclusion 48
- 4.5 References 49

5 COMMUNITY DETECTION 51
- 5.1 Community detection 51
- 5.2 Optimization approaches 55
- 5.3 Inference in the DCSBM 59
- 5.4 Spectral clustering 64
- 5.5 Conclusion 75
- 5.6 References 76

6 GRAPH EMBEDDINGS 77
- 6.1 Graph embeddings 77
- 6.2 Word2Vec 78
- 6.3 Node2Vec 81
- 6.4 Conclusion 82
- 6.5 References 83

# ACRONYMS

| | |
|---|---|
| F2F | *Face-to-face* |
| ER | *Erdős-Rényi* |
| NMF | *naïve mean field* |
| BP | *belief propagation* |
| SIR | *Susceptible-Infected-Recovered model* |
| GW | *Galton Watson tree* |
| GFT | *Graph Fourier transform* |
| CD | *Community detection* |
| AMI | *Adjusted mutual information* |
| DCSBM | *Degree corrected stochastic block model* |
| SC | *Spectral clustering* |

# SYMBOLS

- $\delta_{a,b}$ is the Kroeneker delta equal to 1 if $a = b$ and equal to 0 otherwise.
- $\Lambda(M)$ is the set of eigenvalues of a matrix $M$. $\lambda_i(M)$ is the $i$-th (smallest or largest, according to the context) eigenvalue of $M$
- The spectral radius of $M$ (largest eigenvalue) is denoted with $\rho(M)$.
- With the notation $\mathbf{1}_n$ we denote the all-ones vector of size $n$.
- The entry-wise Hadamard product is denoted with $\circ$.
- The set of the first $n$ integers is denoted with $[n]$.
- We adopt the Landau notation for the asymptotic behavior of variables. In particular $x = O_n(y)$ is equivalent to $\lim_{n\to\infty} \frac{x}{y} = c$ for some finite $c$. The notation $x = o_n(y)$ instead means $\lim_{n\to\infty} \frac{x}{y} = 0$.
- The set of neighbors of a node $i$ on a graph is $\partial i = \{j \in \mathcal{V} \,:\, A_{ij} = 1\}$

# 1 TEMPORAL GRAPHS

1.1 Why temporal graphs? . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Representing temporal graphs . . . . . . . . . . . . . . . . . 2
1.3 Measuring proximity graphs . . . . . . . . . . . . . . . . . . 5
1.4 Properties of temporal graphs . . . . . . . . . . . . . . . . . 8
1.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . 12
1.6 References . . . . . . . . . . . . . . . . . . . . . . . . . . 12

## 1.1 WHY TEMPORAL GRAPHS?

Graphs are an essential mathematical tool to represent interacting systems. When using static graphs, the interactions between two nodes is often represented as a single Boolean variable determining whether or not those two nodes interacted with one another. Yet, we can think of many examples in which this variable should depend on time. Think of a graph in which an interaction between two people is a phone call. In most instants, for most people, no interactions are recorded at all, and, in all cases at most one interaction per time may occur. So, how would we determine the Boolean interaction variable? One approach would be to aggregate time as determine that two people interacted if they had enough phone calls during a specific time window. In this way, connections are created between people that frequently call each other, but we loose an important piece of information: the order of events. Imagine an event like the fire of Notre Dame de Paris: in few moments people witnessing the fire spread the news to their contacts who, themselves reported to others in chain. If we had no idea of what happened and what is the content of the calls or messages, we could actually retrieve the geographical location from which the burst of information was initiated by retracing backwards the chain of events. If instead we use a static representation of the graph as we did earlier, all this information would be lost. A temporal graph is then an object capable of representing the interactions between the elements of a system together with a time stamp. Besides communication graphs, other notable examples are face-to-face proximity graphs, biological graphs, ecological graphs and many others.

*A pictorial representation of a chain of calls*

As we will show in the remainder, in some cases it is necessary to keep the temporal dimension into account if one wants to understand the process happening on top of a graph. This is due to the fact that links may have

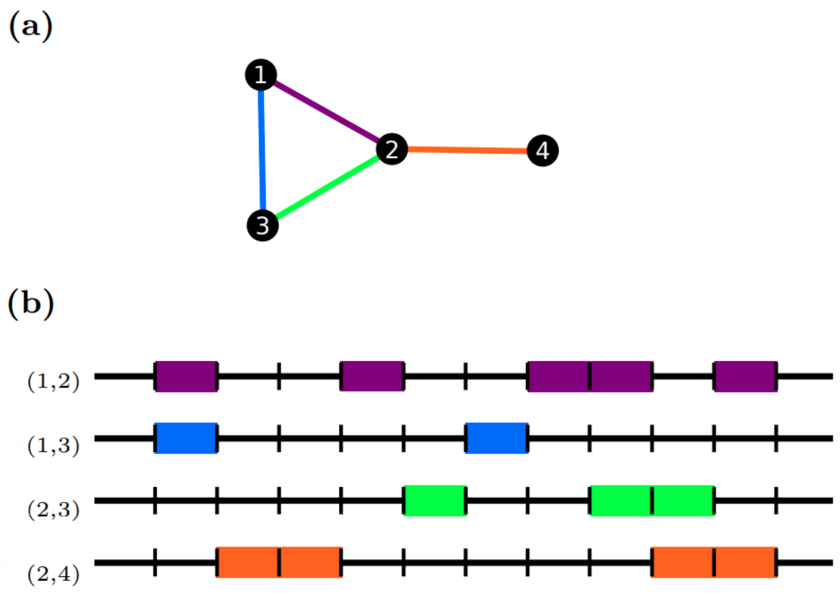


Figure 1.1: **A pictorial representation of a temporal graph**. (a) A graph with 4 nodes with $\mathcal{V} = \{1, 2, 3, 4\}$ and $\mathcal{E} = \{(1, 2), (1, 3), (2, 3), (2, 4)\}$. (b) the temporal activation patterns of each edge (with color code). The $x$ axis represents time. Picture taken from *Gauvin et al., Randomized reference models for temporal graphs.*

a causal relation (like the phone calls in the Notre Dame example) or simply because the aggregated static graph may be non representative of the interactions at any time stamp. We will then discuss how to mathematically define and represent temporal graphs, a relevant example of how to measure them and show some peculiar properties of temporal graphs.

## 1.2 REPRESENTING TEMPORAL GRAPHS

Let us consider a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$ connecting the nodes. Given a suited definition of interaction for our problem and a time window, $\mathcal{E}$ is the set of all pairs of nodes $(i, j)$ that interacted at least once in the considered time window. We now want to add the notion of *when* these interactions occurred. Each edge $(i, j)$ can appear multiple times and for each interaction we can consider a time $t$ at which the interaction begun and a time duration $\tau$.[1] We can then represent a temporal graph as a sequence of *temporal edges* in the form $(i, j, t, \tau)$. Figure 1.1 gives a pictorial representation of the temporal edges of a graph with 6 nodes. We now provide a more formal definition of a temporal graph.

1 We could equivalently replace $\tau$ with the time $t_e$ at which the interaction ended.

**Temporal graphs**

A temporal graph is a tuple $\mathcal{G}(\mathcal{V}, \mathcal{E}_t)$, where $\mathcal{V}$ denotes the set of $n$ nodes and $\mathcal{E}_t$ of temporal edges. Each $e \in \mathcal{E}_t$ can be written as $(i, j, t, \tau)$ where $i, j \in \mathcal{V}$, $t$ is a time-stamp and $\tau \in \mathbb{R}^+$ is a positive interaction duration, implying that the link between $i$ and $j$ was active from $t$ to $t + \tau$. If a node has at least one connection at time $t$ we say it is *active* at time $t$ and it is inactive otherwise.

*Temporal graphs*

One can generalize the concept of adjacency matrix to the temporal setting by letting

$$\tilde{A}_{ij}^{(t)} = \begin{cases} 1 & \text{if } \exists\, e = (i, j, t_0, \tau) \in \mathcal{E}_t \text{ s.t. } t \in [t_0, t_0 + \tau] \\ 0 & \text{else.} \end{cases}$$

*Temporal adjacency matrix*

In this representation, however, time is kept as a continuous variable and, even for a finite observation time, we obtain an infinite number of adjacency matrices. For this reason, the snapshot representation – that uses a discrete notion of time – may be more suited. In particular, we assume that the interaction duration $\tau$ is a multiple of a unit $\Delta t$ that sets the temporal resolution of the graph. Let us define the concept of snapshot graphs, pictorially visualized in Figure 1.2.

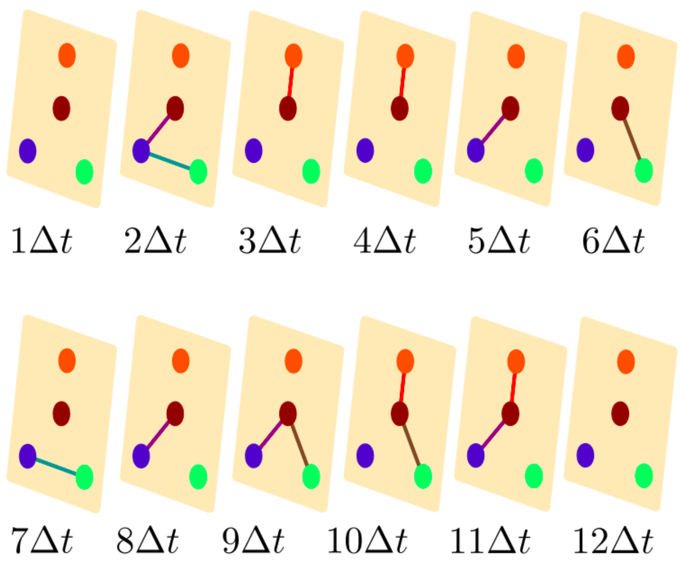


Figure 1.2: **A pictorial representation of a snapshot graph**. This plot represents the same graph of Figure 1.1 in which time was already discretized for convenience. Each *slice* corresponds to a different time step in which the edges progressively are activated. Picture taken from *Gauvin et al., Randomized reference models for temporal graphs.*

*Snapshot graphs*

**Snapshot graphs**

A snapshot graph is a tuple $\mathcal{G}(\mathcal{V}, \mathcal{E}_t)$, where $\mathcal{V}$ denotes the set of $n$ nodes and $\mathcal{E}_t$ of temporal edges. Each $e \in \mathcal{E}_t$ can be written as $(i, j, t)$ where $i, j \in \mathcal{V}$, $t$ is a *discrete* time-stamp and models an instantaneous interaction between $i, j$ at time $t$.

With this representation at hand, one can obtain the adjacency matrix representation of a temporal graph as follows

*The snapshot adjacency matrix*

$$A_{ij}^{(t)} = \begin{cases} 1 & \text{if } (i,j,t) \in \mathcal{E}_t \\ 0 & \text{else.} \end{cases}$$

In this way we obtain a sequence of $T$ adjacency matrices $\{A^{(t)}\}_{t \in 1,\dots,T}$, where $T$ is the total number of snapshots. A natural question that poses is how which of the two representations is more appropriate and how much information is lost when choosing to discretize time. We address this point in the following remark.

*Discretizing time*

**Discretizing time**

If we want to choose a discrete representation of time, the natural questions that arise are how to choose $\Delta t$, the minimal interaction duration and how much do we loose in performing this simplification. The first point we want to raise is that any measured quantity (including time) is not truly continuous and it is bounded by a resolution that makes time discrete by design. So, in all cases, one can say $\Delta t$ is the measurement instrument resolution and the snapshot representation is always appropriate. Yet, if $\Delta t$ is much smaller than the total observation time, the number of time frames $T$ – even if finite – will tend to be very large, hence untractable. So, setting a longer $\Delta t$ might be more appropriate in some cases and the choice of a good $\Delta t$ is necessarily problem-dependent because $\Delta t$ determines the scale at which we consider interactions to be *simultaneous*.

*The coupling between the process and graph dynamics*

Let us make two examples to make this point clearer. Suppose we have two spreading phenomena: in one case the propagation of a piece of information (such as the fire of Notre Dame) and in the other a flu-like illness transmission. In the former case, the propagation of the information from one person to the other moves very fast and so $\Delta t$ must be small, in the order of seconds/minutes to capture the rapid dynamics of the news propagation. If we consider the flu-like propagation, instead, we know that a person, after being infected, is not typically able to infect someone for a couple of days, hence we can set $\Delta t$ of the order of one day, assuming that one cannot change its own infectious state in the course of a day. So, summarizing, the proper time aggregation depends on the time-scale of the dynamic process happening on top of the graph. If this process if much slower than the temporal evolution then we can simply aggregate the graph. If instead the two time scales (of the process and of the graph evolution) are similar, then we have an interesting coupling that must be taken into account.

To conclude this remark, there is still a quantity we want to preserve when we aggregate time, that is the cumulative interaction duration: what do we do with all interactions so that $\tau < \Delta t$? Also in this case the answer depends on the problem under consideration. Take for instance the flu propagation with a time aggregation of 24 hours. If an infectious individual has a interaction with a susceptible one, the interaction duration is key to determine the probability of infection: an hour-long interaction is much more likely to propagate the disease than a minute-long interaction and all interactions will be shorter than $\Delta t$ in this case. To preserve this piece of information, we might want to associate a weight to each edge, representing the cumulative interaction duration. We relate the continuous time and snapshot adjacency matrices as follows

$$W_{ij}^{(t)} = \int_t^{t+\tau_0} dt' \, \tilde{A}_{ij}^{(t')}.$$

*The weighted aggregated graph*

If this representation may seem very reasonable, it must be noted that is not the only admissible one: in the case of the propagation of sexual diseases, for instance, the interaction duration is not relevant and may simply want to keep a Boolean representation of the edges, without attributing any weight.

Now that we have introduced some of the main concepts related to temporal graphs, let us consider the specific set of proximity graphs as a case study, first describing a method to measure these graphs and then using the open source, real data to describe some relevant properties.

## 1.3 MEASURING PROXIMITY GRAPHS

In proximity graphs the edges represent a *Face-to-face* (F2F) contact between two persons. The interest of this type of graphs resides in the fact that F2F interactions are the vehicle of human communication and of infectious diseases and, more generally, they quantify how humans interact with one another. Measuring *Face-to-face* (F2F) proximity graphs is, however, a very challenging task. Among the most used approaches to quantify them we have the use of questionnaires in which the interactions one has are self-reported. It was shown that this method is biased towards long interactions – in the sense that short interactions tend to be forgotten – and can achieve a low temporal resolution. A very important contribution to the field of measur-

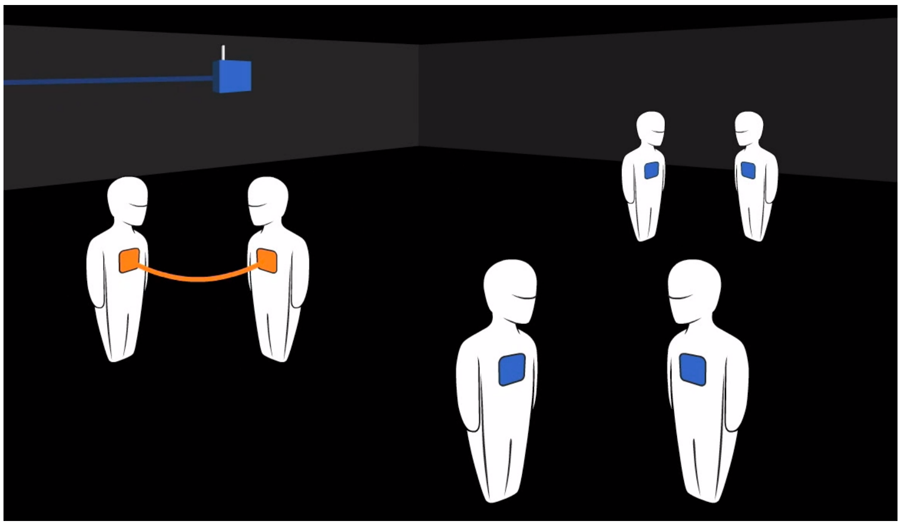
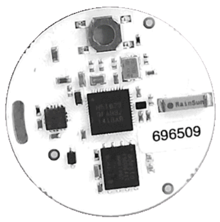


Figure 1.3: **Pictorial example of the use of SocioPatterns proximity sensors**. Six people in a room wearing a proximity sensor. The orange ones (with the line), indicate a recorded F2F interactions between two individuals. Picture taken from http://www.sociopatterns.org/.

ing F2F proximity graphs was made with the creation of the SocioPatterns collaboration by the ISI Foundation.

*The SocioPatterns collaboration*

SocioPatterns was formed in 2008 and developed wearable proximity sensors that are capable to *measure* temporal proximity graphs. Their functioning is based on the transmission and exchange of information packets using radio-frequency electromagnetic waves. Briefly speaking, each device is associated with a code and continuously switches from a *listener* to a *speaker* mode. When it is in the *speaker* mode it emits an information packet containing its own code and the power at which the signal was emitted. When it is in listener mode, instead, the device intercepts the packets emitted by the "speakers" and records on its memory the code and the power declared, the time stamp at which this interaction occurs as well as the power of the received signal. The devices have to be worn on the chest of the participants and, by design, record F2F proximity. Figure 1.3 shows a demo of the functioning of the SocioPatterns proximity sensors.



*A proximity sensor*

Inside the memory of each sensor we then have a list of entries of the type $(j, \text{pow}_{\text{tr}}, \text{pow}_{\text{rec}}, t)$, where $j$ is the code of the sensor that emitted the signal, $\text{pow}_{\text{tr}}$ is the transmission power, $\text{pow}_{\text{rec}}$ is the received power and $t$ is the time-stamp that has a temporal resolution of 20 seconds. Looking at the difference $\text{pow}_{\text{tr}} - \text{pow}_{\text{rec}}$ one can measure the attenuation of the signal and filter out the interactions that are too attenuated, thus keeping only those that happened at a distance within approximately 2 meters. From this we can create a snapshot graph with $\Delta t = 20\ s$ as described above.

The SocioPatterns sensors have been used in several contexts, including schools, hospitals, offices and rural African villages among others. They constitute a well known benchmark of temporal graph measurement that has

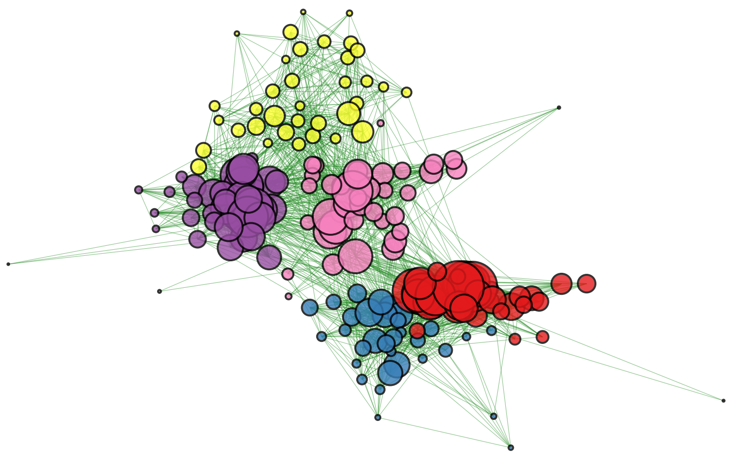


Figure 1.4: **The *School* dataset**. Pictorial representation of the *School* dataset (see Table 1.1) aggregated over all the observation time. The size of each node is the determined by the total interaction time of that node, while the color is determined by the class the student belongs to.

been used in many applications and many of the collected datasets are publicly available at http://www.sociopatterns.org/datasets/. We will make use of some of these data to study some relevant properties of temporal graphs. Table 1.1 summarizes some descriptive properties of the considered graphs, while Figure 1.4 shows the aggregated graph collected in a high school.

| **Name** | $n$ | **Observation time** | **Description** |
|---|---|---|---|
| School | 180 | from a Monday to the Tuesday of the following week in November 2012. | interactions between students in a high school in Marseilles, France belonging to 5 classes. |
| Office | 92 | June 24 to July 3, 2013 | interactions between individuals measured in an office building in France |
| Village | 86 | between 16th December 2019 and 10th January 2020 | interactions between the people of Mdoliro village in Dowa district in the Central Region of Malawi. |
| Conference | 405 | June 4-5, 2009 | interactions at the SFHH conference in Nice |

Table 1.1: **Summary statistics of the *SocioPatterns* temporal networks**. The first column indicates the name used in these notes. The column indexed by $n$ indicates the number of nodes appearing in the graph. The column *Observation time* describes the experiment duration, while *Description* provides a few details on the context of the data collection. For more information, refer to the SocioPatterns website.

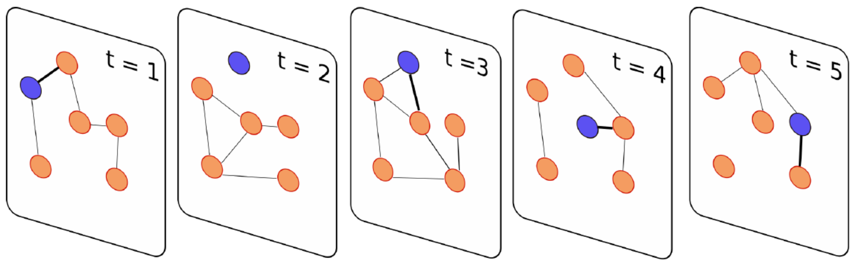


Figure 1.5: **Time respecting paths**. Five snapshots of a temporal graph in which unoccupied nodes are depicted in orange at each time, while the currently occupied node is in blue. A larger width is used to highlight the edge that causes the transition.

## 1.4 PROPERTIES OF TEMPORAL GRAPHS

We now proceed to describe some important concepts that characterize temporal graphs and use the four aforementioned datasets to show them on empirical data.

### TIME-RESPECTING PATHS

When we consider a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, we define a path on it as an ordered sequence of nodes $\{i_1, i_2, \ldots, i_T\}$ for that, for all $p \in [T]$, $i_p \in \mathcal{V}$ and for all $p \in [T-1]$, $(i_p, i_{p+1}) \in \mathcal{E}$. In words, every step of a path allows one to only move from a node to one of its neighbors. When we consider a temporal graph, instead, we must generalize the concept of path, to encode the role played by time, introducing the *time-respecting paths*.

*Time respecting paths*

**Time respecting paths**

Given a temporal graph $\mathcal{G}(\mathcal{V},\mathcal{E}_t)$, we denote a time respecting path as $\{i_1(t_1), i_2(t_2), \ldots, i_T(t_T)\}$ if it satisfies the following conditions

- For all $p \in [T]$, $i_p \in \mathcal{V}$: the path is defined on the graph nodes.
- For all $p \in [T-1]$, $t_p < t_{p+1}$: these two times indicate the beginning of the residency on the respective nodes and time must be increasing.
- For all $p \in [T-1], \exists\, t_0, \tau$ s.t. $(i_p, i_{p+1}, t_0, \tau) \in \mathcal{E}_t$ and $t_{p+1} \in [t_0, t_0+\tau]$: the transition between one node and the other can only take place at a time at which the two nodes are connected.

This definition is given for a continuous time representation but it can simply be adapted to the discrete one. For this case, we give a simple representation of a time respecting path in Figure 1.5.

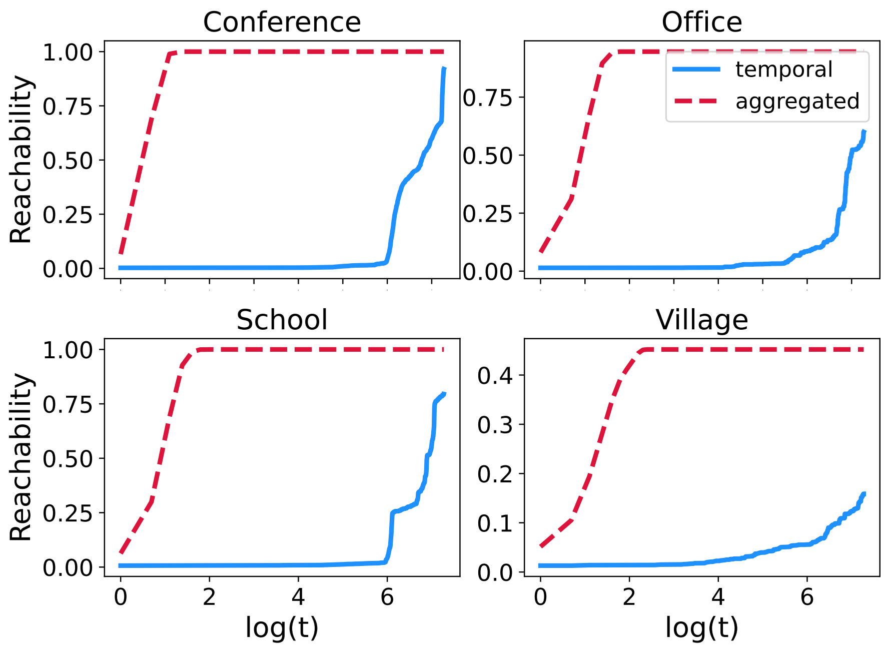


Figure 1.6: **Time-respecting vs aggregate reachability**. Each plot refers to one of the 4 *SocioPatterns* datasets described in Table 1.1, considering the first 8 hours of measurements. The red dashed line is the average of the reachability matrix $R_t$ defined in Equation (1.1) as function of time, using for all $t$ $A_t$ the weighted aggregated matrix over all the observation period. The blue continuous line, instead, is obtained from the snapshot adjacency matrices and encodes time-respecting paths.

Given a sequence of adjacency matrices, we now define the *reachability matrix* $R_t$ as follows

*Reachability matrix*

$$R_t = \text{sign}\left[\prod_{t'=1}^{t}(A_{t'} + I_n)\right], \tag{1.1}$$

where $I_n$ is the identity matrix, the sign function has to be considered entry-wise, while the product has to be taken from right to left, *i.e.* $\prod_{t=1}^{3} A_t = A_3A_2A_1$. The entry $R_{t,ij}$ equals 1 if there exists a time-respecting path of length smaller or equal to $t$ that allows one to go from $i$ to $j$.

*Time-respecting paths are not symmetric*

An important fact related to this matrix is that it is not necessarily symmetric. This comes from the fact that the product of matrices (such as the $A_t$'s) is symmetric only if the matrices commute. This is not the case in general and it implies that if there is a time-respecting path from $i$ to $j$, that does not imply that there exists also a time-respecting path from $j$ to $i$.

By taking the average of the reachability matrix, we also have a measure of how well its nodes are connected. Figure 1.6 compares the reachability on the 4 real temporal graphs described above with the one obtained on their aggregated version and clearly shows that temporal graphs have a lower reachability. This is because the valid time-respecting paths are constrained

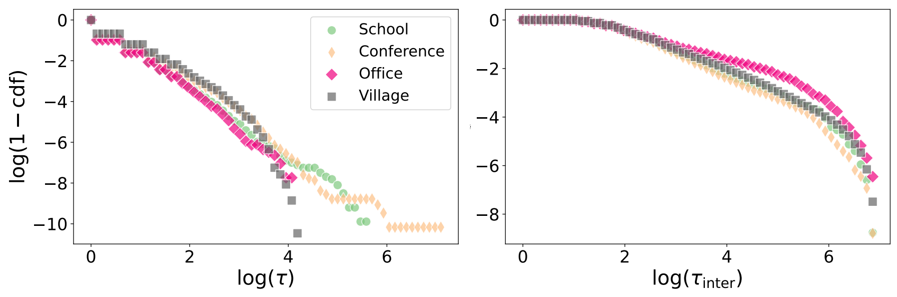


Figure 1.7: **Event and inter-event duration distributions**. The figures show the scatter plot in log-log scale of the interaction duration (left) and inter-event duration (right) distributions vs the $1 - cdf$, *i.e.* the complementary of the cumulative density function. Each line refers to one of the datasets described in Table 1.1 and is color and marker coded.

and are only a subset of all the possible paths that one can perform on the aggregate version of the graph. This is an important ingredient to consider when coupling a dynamic process with the graph, because only a fraction of all possible paths can actually take place.

## DURATION DISTRIBUTION AND BURSTINESS

We now focus on a very peculiar aspect of contact graphs, that is the contact duration distribution. It has been observed in many instances (with no apparent exception) that this distribution is very broad and follows approximately a power law decay that appears to be a universal behavior. Figure 1.7 (left plot) shows in log-log scale 1 minus the cumulative density function vs the interaction duration and confirms this trend, since the relation is approximately linear in the logarithmic scale. This is an important observation, because it tells us that very long interactions are much more common than what we would expect for a thin tail distribution, such as the Poisson. The consequence is that, if we have a process that needs a minimal time of interaction to consider the interaction to be valid, then, in practice, even if the threshold is very large, there will be valid interaction edges with high probability. On the other hand, we also know that most of the distribution is concentrated around small values.

$1 - \mathrm{cdf}(x) = \mathbb{P}(\tau \geq x)$

A similar behavior is observed for the inter-event duration distribution. We define the inter-event duration as the time elapsed between two successive interactions of the same pair of nodes $(ij)$. Since this distribution is broad, we say that the interaction dynamics is *bursty*, *i.e.* that typically we have an alternation of time intervals in which the activity is very low and some in which it is very high. To best understand the effect that a bursty dynamics may have on a process, let us consider the following example.

Suppose we have a quantity $\mathcal{Q}$ that is increased by one unit every time there is a interaction and it is decreased by a factor $\alpha$ for each time step in which no interaction occurs. If $\mathcal{Q}$ exceeds a threshold value $\mathcal{Q}_{\text{th}}$, then some

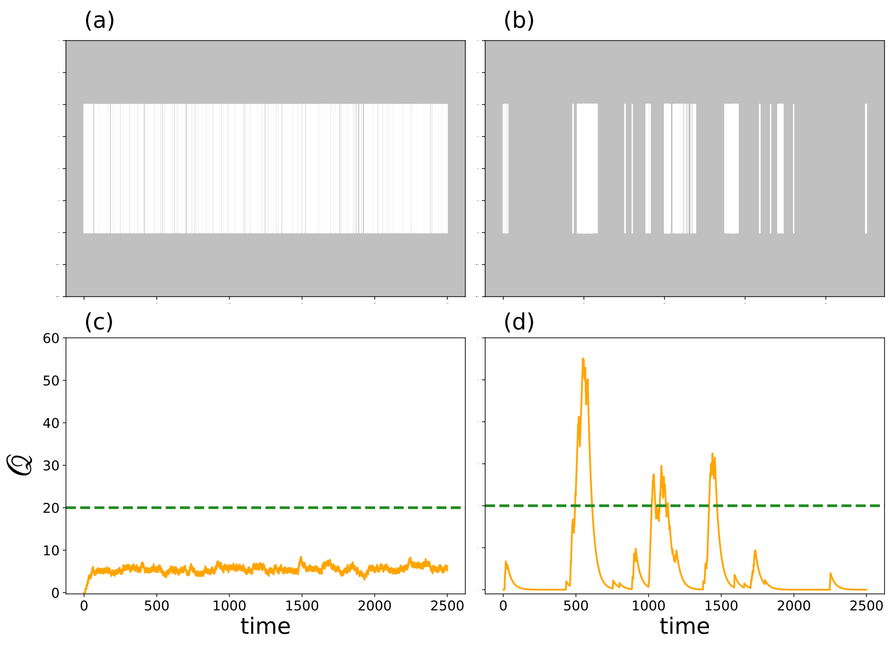


Figure 1.8: **The effect of bursty dynamics**. The first row represents a temporal time series for (a) a Poisson process and $(b)$ the interaction times of a node of the *Conference* graph (see Table 1.1. Each horizontal line indicates an active time. The second line displays the dynamics of a quantity $\mathcal{Q}$ evolving according to the process described in the main text for $(c)$ the Poisson dynamics of $(a)$, and for $(d)$ the bursty dynamics of $(b)$. The horizontal dashed line indicates an arbitrary selected threshold value that triggers some process when $\mathcal{Q} > \mathcal{Q}_{\text{th}}$. Image adapted from *Holme, Saramaki, Temporal networks.*

process is triggered otherwise it is not. In Figure 1.8 we compare this dynamical process on a Poisson temporal series made of 362 interaction events with one extracted from an individual activity pattern of the *Conference* graph. The bottom plots clearly evidence that the bursty dynamics, being highly concentrated in some time regions, allows one to go beyond the threshold several times, while this does not happen to the Poisson distribution.

A method to measure the burstiness level of a time series is as follows

$$B = \frac{s-m}{s+m},$$

where $s$ and $m$ are the standard deviation and mean of the inter-event duration distribution, respectively. For a periodic process, $s = 0$ and $B = -1$, while for the burstiest of processes, $s \to \infty$ and $B \to 1$. In our case, the Poisson dynamics of Figure 1.8(a) has $B = -0.45$, while the one of Figure 1.8(b) has $B = 0.70$.

## 1.5 CONCLUSION

Temporal networks are a powerful tool to model complex dynamical systems. Empirical networks often show very broad distribution of the interaction duration as well as bursty dynamics. These features are of great importance to some dynamical processes that may unfold on networks and the temporal framework is a relevant generalization of static graphs. However, the dynamic component of graph evolution must always be compared with the typical time scale of the process unfolding over the graph in order to understand whether it is necessary to have an additional layer of complexity given by time or, more in general, to choose an appropriate time discretization to perform the analysis.

## 1.6 REFERENCES

- P. Holme, J. Saramäki, *Temporal networks.* Physics reports, 519(3), 97-125 2012.
  This is *the* reference for an introduction to temporal graphs.

# 2 EPIDEMICS ON NETWORKS

2.1 Epidemics . . . 13
2.1.1 Epidemic modeling . . . 13
2.1.2 The epidemic threshold . . . 14
2.2 Epidemic modeling on networks . . . 16
2.2.1 The state evolution equation . . . 16
2.2.2 Naïve mean field . . . 17
2.2.3 The reproductive number with naïve mean field . . . 18
2.2.4 Graph structure and reproductive number . . . 19
2.2.5 From theory to practice: epidemic mitigation . . . 21
2.3 Extensions . . . 22
2.4 References . . . 23

## 2.1 EPIDEMICS

Epidemiology is a branch of science that studies the determinants, distribution and dynamics of a propagation process in a population. Given its relation to a whole population, epidemiology is, by design complex and of great interest when shaping public health policies. We will focus on infectious disease epidemiology, *i.e.* on illnesses that can be transmitted from one person to the other. Notable examples include the bubonic plague, smallpox, the Spanish flu, HIV, influenza and, of course, Covid. Even though we will mainly have in mind "illness" propagation, a similar if not identical mathematical framework can be adopted to anythings that propagates through interactions, such as opinions, computer viruses or information.

Let us now describe in deeper detail some fundamental elements of epidemics and their modeling.

### 2.1.1 EPIDEMIC MODELING

The basic medical observation that we want to model and capture is that when a person is "sick" – for instance it is positive to influenza – and it is in contact with a person who is not – and never was – then it can pass the infection over to the healthy person. The meaning of "contact" depends on the disease we are considering: think to the extreme difference there

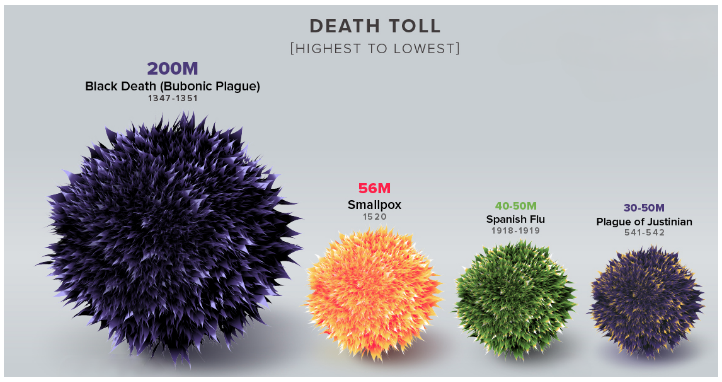
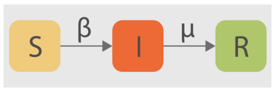


Figure 2.1: **The deadliest pandemics in history**.
Source: visualcapitalist.com/history-of-pandemics-deadliest

is between the HIV and the flu transmissions. Nonetheless, the dynamic processes we have in mind are very similar. We introduce the *Susceptible-Infected-Recovered model* (SIR), a cornerstone of epidemic modeling.

We define three possible states every individual can be in: $S$, susceptible; $I$, infected; $R$ recovered. Susceptible people are healthy individuals that can contract the disease. Infectious ones are those who currently carry the disease and can spread it out when they interact with a susceptible individual. The recovered people, instead, are those who used to be infectious and now can no longer be infected. There are two important parameters that take part to this model: $\beta$ that is the probability *per unit time* to infect a susceptible person and $\mu$, the probability *per unit time* to recover. We can summarize the SIR model with the following equation



*The SIR model*

$$S + I \xrightarrow{\beta} 2I$$
$$I \xrightarrow{\mu} R$$

The parameters $\beta, \mu$ are disease-dependent and tell us how easily the infection runs across the population. Intuitively, if $\beta \gg \mu$ we are in a situation in which people get infected at a much faster pace than they recover. As a consequence, the epidemic will swiftly spread across the population. On the opposite, if $\mu \ll \beta$ people simply recover very quickly and the epidemic dies out. This concept, that we delineated in intuitive terms, is the so-called epidemic threshold that determines the necessary and sufficient condition for an epidemic spreading to occur and that we now more formally define.

### 2.1.2 THE EPIDEMIC THRESHOLD

We consider a population in which everybody is susceptible. This condition is a stationary state, because no infection can occur among susceptible people, or, in other words

$$S + S \longrightarrow 2S.$$

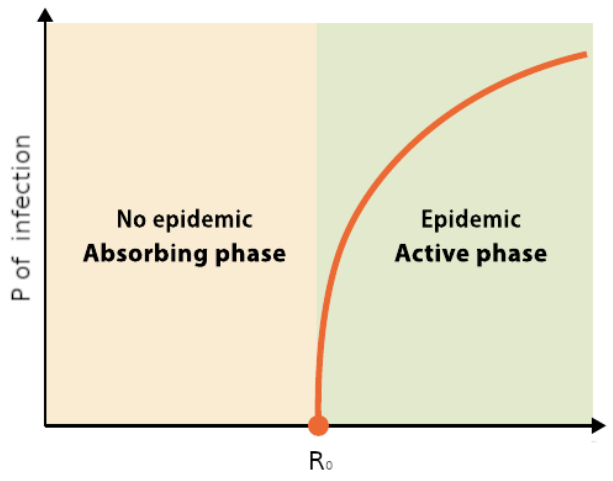
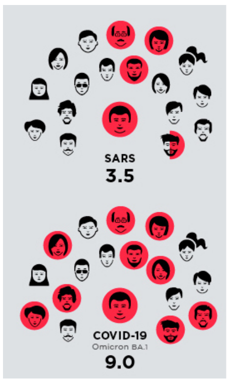


Figure 2.2: **Epidemic threshold phase diagram**. Below the critical value of $R_0$ we are in an absorbing phase in which there is no epidemic, while for $R_0$ larger than the critical value, the probability that each node has of being infected is non-null. Source: Satorras, Castellano, Van Mieghem, Vespignani. *Epidemic processes in complex networks.*

We now perturb this state introducing few infectious individuals and ask ourselves whether the system will fall back to an equilibrium having most people being unaffected by the disease, or, if it will spread hitting a large portion of the population. The simple approach to understand this problem lies in the following question

*At the beginning of the spreading,*
*how many people do I infect before recovering?*

If the answer is "*more than one*", then we have a cascade in which the propagation grows exponentially fast. If, on the opposite it is "*less than one*" than it dies out because, in most cases, every individual recovers before infecting someone else. We call this the *reproductive number*. As shown in Figure 2.2 it is the control parameter of a phase transition between a disease-free and an epidemic state.



*The effect of $R_0$, some numbers*
*Source:visualcapitalist.com/history-of-pandemics-deadliest/*

Let us stress two important facts. The first one is that $R_0$ is defined *at the beginning* of the epidemic. Notation-wise, this is why we call it $R_0$, while $R_t$ is the reproductive number at a given time $t$. This is important to say because the exponential spreading can occur only on a short time scale and then saturate due to the finite population size. The second fact is that the question refers to a non-better specified "*I*", implying that the answer is the same for everybody. Of course, if we think of our everyday life, the probability of infection vary a lot across individuals, according to their sociability, the interaction with children, the spaces they occupy *etc.* So, of course, we want an *average* answer and we will study it in the next section.

## 2.2 EPIDEMIC MODELING ON NETWORKS

As we mentioned, the epidemic spreading propagates through contacts that are well modeled by the edges of a graph. We now want to formally define the epidemic threshold transition for an arbitrary graph. This is of key importance to understand if, given the structure of the network and the disease parameters, the spread will touch a large fraction of individuals or not.

### 2.2.1 THE STATE EVOLUTION EQUATION

We consider an agent-based model in which every individual $i \in \mathcal{V}$ is associated to a discrete variable $x_i(t) \in \{S, I, R\}$ determining the state $i$ is in. When a susceptible person $i$ is in contact with an infectious one $j$ ($A_{ij} = 1$) for a time $dt$, then it gets infected with a probability $\beta dt$. An infected person recovers with a probability $\mu dt$. We can write the probability of being infected at the time-step $t + dt$ as a function of $t$ as follows:

*The infected state equation of the SIR on a graph*

$$\mathbb{P}(x_i(t+dt) = I) = \mathbb{E}\Bigg[ \underbrace{\delta[x_i(t) = I](1-\mu)}_{i \text{ was infected and did not recover}} \\ \underbrace{+\, \delta[x_i(t) = S]\left(1 - \prod_{j\in\mathcal{V}} \left(1 - \beta dt \cdot \delta[x_j(t) = I]\right)^{A_{ij}}\right)}_{i \text{ was susceptible and got infected}} \Bigg]. \tag{2.1}$$

This equation features two terms: the case in which $i$ was susceptible and got infected and the can in which it was infected and did not recover. Note that the probability of being infected is written as 1 minus the probability of not being infected. We can simplify this equation in the limit for $dt \to 0$, focusing on the term describing the probability of not being infected.

$$\begin{aligned}
&\lim_{dt\to 0} \prod_{j\in\mathcal{V}} \left(1 - \beta dt \cdot \delta[x_j(t) = I]\right)^{A_{ij}} \\
&\overset{(a)}{=} \lim_{dt\to 0} \exp\left\{ \sum_{j\in\mathcal{V}} A_{ij} \log\left(1 - \beta dt \cdot \delta[x_j(t) = I]\right) \right\} \\
&\overset{(b)}{=} \lim_{dt\to 0} \exp\left\{ -\beta dt \sum_{j\in\mathcal{V}} A_{ij}\delta[x_j(t) = I] \right\} \\
&\overset{(c)}{=} \lim_{dt\to 0} 1 - \beta dt \sum_{j\in\mathcal{V}} A_{ij}\delta[x_j(t) = I],
\end{aligned}$$

where in $(a)$ we used the identity $x = e^{\log x}$; in $(b)$ we performed the expansion $\log(1+x) = x + o(x)$ and in $(c)$ the expansion $e^x = 1 + x + o(x)$. Substituting this expression in Equation (2.1) we obtain

$$\begin{aligned}\mathbb{P}(x_i(t+dt) = I) &= \mathbb{E}\left[\delta(x_i(t) = I)\right](1-\mu) + \beta dt \sum_{j\in\mathcal{V}} A_{ij}\mathbb{E}\left[\delta[x_i(t) = S]\delta[x_j(t) = I]\right] \\ &= \mathbb{P}\left(x_i(t) = I\right)(1-\mu) + \beta dt \sum_{j\in\mathcal{V}} A_{ij}\mathbb{P}\left(x_i(t) = S, x_j(t) = I\right).\end{aligned}$$

Taking the first term on the right hand-side to the left and dividing by $dt$, we obtain the derivative of the probability that reads

$$\partial_t \mathbb{P}(x_i(t) = I) = \beta \sum_{j\in\mathcal{V}} A_{ij}\mathbb{P}\left(x_i(t) = S, x_j(t) = I\right) - \mu\mathbb{P}(x_i(t) = I). \tag{2.2}$$

Following the same passages, we get the evolution equations for all three states. Note that, $\partial_t\mathbb{P}(x_i(t) = S) + \partial_t\mathbb{P}(x_i(t) = I) + \partial_t\mathbb{P}(x_i(t) = R) = 0$, because the probability of being in one of the three states sums up to one.

SIR model on a graph

$$\begin{aligned}\partial_t\mathbb{P}(x_i(t) = S) &= -\beta \sum_{j\in\mathcal{V}} A_{ij}\mathbb{P}\left(x_i(t) = S, x_j(t) = I\right) \\ \partial_t\mathbb{P}(x_i(t) = I) &= \beta \sum_{j\in\mathcal{V}} A_{ij}\mathbb{P}\left(x_i(t) = S, x_j(t) = I\right) - \mu\mathbb{P}(x_i(t) = I) \\ \partial_t\mathbb{P}(x_i(t) = R) &= \mu\mathbb{P}(x_i(t) = I).\end{aligned} \tag{2.3}$$

In order to obtain the reproductive number we must study the stability of this system of equations that, however, is still non-linear and hard to study because it involves the marginal distributions $\mathbb{P}\left(x_i(t) = S, x_j(t) = I\right)$ for which we do not have an explicit expression. To cope with this problem, we adopt the simple naïve mean field approximation.

## 2.2.2 NAÏVE MEAN FIELD

The *naïve mean field* (NMF) approximation consists in considering all variables as independent *i.e.* in factorizing the marginals as follows

$$\mathbb{P}(x_i(t) = S, x_j(t) = I) = \mathbb{P}(x_i(t) = S)\mathbb{P}(x_j(t) = I).$$

*The NMF approximation*

This approximation greatly simplifies the problem. In fact, in (2.3) we have defined the evolution of $3n$ equations concerning the node marginal probabilities, but there are $2|\mathcal{E}|$ equations (where $\mathcal{E}$ is the set of edges) that are unspecified. Factorizing the probabilities with naïve mean field, we simply get rid of these terms. Let us first make some comments about this approximation, its limits and when we expect to be a good method to proceed.

**Some notes on the naïve mean field approximation**

Given its simplicity NMF is a commonly adopted strategy to first tackle a problem, but is it accurate? In other words, we are asking to what extent we can assume that the event that $i$ is susceptible is independent from the event that its neighbor $j$ is infected. Given the context of the model, the answer seems necessarily to be negative since the contagion is transmitted through the contacts.

The most relevant setting in which NMF should be considered is that of *dense* networks, *i.e.* those in which every node has a large degree. Suppose we have a fully connected network: the fact that the edge $A_{ij}$ exists is simply irrelevant because all edges exist. One can show that indeed, in this setting the NMF approximation becomes asymptotically exact and, in general, the denser the network is the more the NMF approximation is accurate. An example of how to go beyond this approximation is discussed in chapter 3.

With NMF approximation at hand, Equation (2.2) turns into

$$\partial_t \mathbb{P}(x_i(t) = I) = \beta \mathbb{P}(x_i(t) = S) \sum_{j \in \mathcal{V}} A_{ij} \mathbb{P}(x_j(t) = I) - \mu \mathbb{P}(x_i(t) = I).$$

We now linearize this equation around the stationary state $\mathbb{P}(x_i(t) = S) = 1$ to get the reproductive number.

### 2.2.3 THE REPRODUCTIVE NUMBER WITH NAÏVE MEAN FIELD

As we explained before, we want to see the effect of perturbing the stationary state in which everybody is susceptible by adding a small probability of being infected. For simplicity, we denote $\mathbb{P}(x_i(t) = I) := p_i(t)$ and move to a vector form of the equations. We let $P_i(x_i(t) = S) = 1$ and obtain

*The linearization around the diseases-free point under the NMF approximation*

$$\partial_t \boldsymbol{p}(t) = (\beta A - \mu I_n) \boldsymbol{p}(t).$$

If we want that $\partial_t p_i(t) < 0$ for all $i$ and all $t$, we must impose that $\beta\rho(A) - \mu < 0$, where $\rho(A)$ denotes the spectral radius of $A$. If this condition is satisfied, then we end up in the disease-free region. If on the opposite $\beta\rho(A) - \mu > 0$, the probability of being infected grows at each time step and the virus has a broad diffusion on the network. We thus obtain the following value for the reproductive number

**Reproductive number with the NMF approximation**

$$R_0 = \frac{\beta\rho(A)}{\mu} \tag{2.4}$$

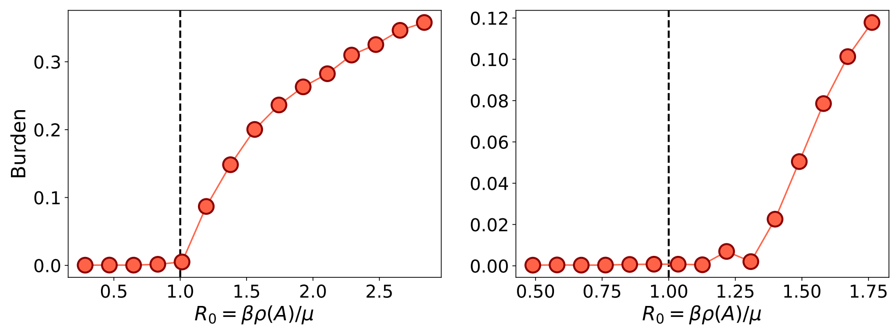


Figure 2.3: **Epidemic threshold: theoretical prediction versus simulated data**. We run a SIR model for different $\beta$ values on a dense graph (left panel) and on a sparse one (right panel) generated from the random configuration model. We plot the burden (*i.e.* the fraction of non-susceptible individuals) as a function of $R_0$ as predicted by the NMF approximation.

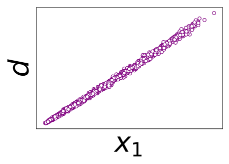
In Figure 2.3 we compare this prediction with an empirical simulation on a dense (left panel) and on a sparse one (right panel), evidencing the goodness of the approximation only in the former case. Now, as a last step, we give some simple results relating the spectral radius of $A$ with its structure.

### 2.2.4 GRAPH STRUCTURE AND REPRODUCTIVE NUMBER

Studying the spectral properties of the adjacency matrix for different generative models is a problem of great interest, that however goes beyond the scope of this course. Here we provide two simple examples with intuitive and non-rigorous arguments to characterize the value of $\rho(A)$ and understand the role of density and degree heterogeneity in determining the threshold.

#### *Erdős Renyi random graph*

The spectral behavior of the *Erdős-Rényi* (ER) random graph changes dramatically according to whether its expected average degree grows with its size or not. Letting $p$ be the probability of being connected, then we can define two different regimes: the dense one in which $\log(n)/pn = o_n(1)$ and the sparse one in which the opposite is true, *i.e.* $pn/\log(n) = o_n(1)$. In words, if the average degree grows faster than $\log(n)$ we say the network to be *dense*. This is a game-changer because, under this hypothesis, the degree distribution is concentrated, *i.e.* , for all large $n$, with probability one



*The leading eigenvalue of $A$, $x_1$ against the degree vector on a random dense graph*

$$\max |d_i - np| = o_n(np).$$

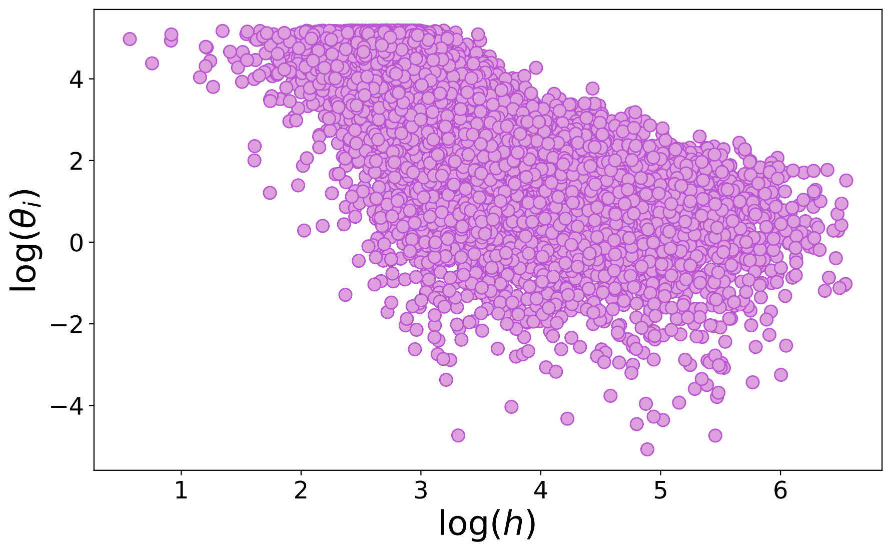


Figure 2.4: **Hitting time as a function of the expected degree**. We consider a random graph generated from the configuration model and denote with $\theta_i$ the expected degree of node $i$. In the plot we show the scatter plot of $\theta_i$ against the hitting time $h$, defined as the number of iterations after which the node got infected in a SIR simulation.

Again, in words, this means that a dense ER graph is quasi regular. In this case, we can heuristically[1] write the following equation

$\rho(A) = \langle d\rangle + o_n(d)$
*on dense ER graphs*

$$(A\mathbf{1}_n)_i = \sum_{j\in\mathcal{V}} A_{ij} = d_i \approx np \approx \langle d\rangle(\mathbf{1}_n)_i.$$

This implies that $\mathbf{1}_n$ is a close approximation of the leading eigenvector[2] and the average degree is an approximation of $\rho(A)$. From this result, we obtain that in denser networks an epidemic spreading runs faster, as one could reasonably expect.

Let us now consider the case in which the graph is generated from a configuration model with an arbitrary degree distribution.

## *Configuration model*

Once again we operate under the assumption of being in a sufficiently dense regime and derive a heuristic expression of the leading eigenvector of $A$, which we suppose in this case to be $\boldsymbol{d}$, the degree vector.

$\rho(A) = \frac{\langle d^2\rangle}{\langle d\rangle} + o_n(d)$

*for dense random graphs with an arbitrary degree distribution*

$$(A\boldsymbol{d})_i = \sum_{j\in\mathcal{V}} A_{ij}d_j \approx \sum_{j\in\mathcal{V}} \frac{d_i d_j^2}{2|\mathcal{E}|} = d_i\frac{\langle d^2\rangle}{\langle d\rangle}.$$

We thus get that $\rho(A) = \langle d^2\rangle/\langle d\rangle$ implying that a broad degree distribution makes the spreading run even faster on the network. This is because of the

1 For simplicity we derive this result heuristically, but it can be formally proved.
2 Due to Perron-Frobenius theorem.

role played by hubs, *i.e.* nodes with a high degree. Since they have a lot of connections, they are very likely to get infected in the earlier stages of the epidemic and then become super-spreaders. Notably, the value of $\rho(A)$ that we just may diverge for scale-free networks in which the second moment of the degree distribution goes to infinity, making the transition go to zero. Figure 2.4 shows for each node the infection hitting time (*i.e.* the time it takes to get infected) as a function of their degree. The plot evidences a strong negative correlation, confirming the intuition that nodes with a large degree are the first to get infected and then they are responsible of the spreading. Note that in practice the second moment of the degree may only diverge in the asymptotic theoretical limit. All real world networks are finite and so is $\langle d^2 \rangle$. Nonetheless, in real-world settings, it can be overwhelmingly large and drive the epidemic to unfold very fast on the network.

### 2.2.5 FROM THEORY TO PRACTICE: EPIDEMIC MITIGATION

Let us now discuss some basic facts about epidemic mitigation based on our results. Suppose we have a vaccine and we add a fourth compartment to our model, that of vaccinated people that behaves exactly like the recovered one. From our analytical view-point, we can still deploy the results we obtained, because vaccinated people simply do not take part to the process, since they cannot change their compartment. For this reason it is as if they were not part of the network.

We ask ourselves how vaccination impacts the epidemic spread. To answer this question we add a Boolean variable $s_i$ the equals 0 if $i$ is vaccinated and cannot transmit the disease and is 1 otherwise. The matrix that determines the epidemic threshold is then now

$$W = A \circ (\boldsymbol{s}\boldsymbol{s}^T).$$

Using a non-rigorous argument, we will see how the vaccination determines the epidemic threshold.

**Methodological remark**

The approach we used to study $\rho(A)$ was non rigorous but leads to the correct result because we assume $A$ to be dense. By vaccinating a large portion of the population, instead $W$ becomes sparse by design and not only the method we adopt but also the result is incorrect. Formally, we can only see what is the effect of vaccination on the spreading, assuming that a small fraction of the population has been vaccinated and $W$ is still dense. This heuristic result, however, gives us some important intuition that we can verify numerically and that can be rigorously proved with other more rigorous methods.

Let $\tilde{d}_i = d_i s_i$. Then

$$(W\tilde{\boldsymbol{d}})_i = \sum_{j\in\mathcal{V}} A_{ij} s_i s_j \tilde{d}_j \approx = \sum_{j\in\mathcal{V}} \frac{\tilde{d}_i d_j^2 s_j}{2|\mathcal{E}|} = \tilde{d}_i \frac{\boldsymbol{s}^T \boldsymbol{d}^2}{\boldsymbol{1}_n^T \boldsymbol{d}},$$

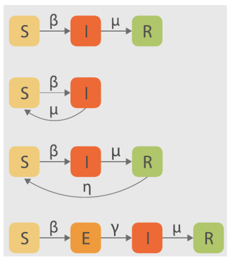
so $\frac{\boldsymbol{s}^T \boldsymbol{d}^2}{\boldsymbol{1}_n^T \boldsymbol{d}}$ is a close approximation of $\rho(W)$. Now, if $s_i$ is Bernoulli random variable with probability $p$, we get

$$\rho(W) \approx p \frac{\langle d^2 \rangle}{\langle d \rangle}.$$

So, the vaccination decreases the $R_0$ and allows one to stay below the epidemic threshold. From a simple observation, however, one sees that this is not the optimal strategy. In fact, in we fix $\boldsymbol{s}^T \boldsymbol{1}_n$, *i.e.* the number of vaccines, and attempt to minimize $\boldsymbol{s}^T \boldsymbol{d}^2$ one immediately sees that the solution lies in vaccinating the nodes with the highest degree, *i.e.* the *hubs*. This target immunization significantly helps in improving the mitigation effectiveness.

## 2.3 EXTENSIONS

In the previous sections we only considered the SIR to model an epidemic spreading. While this is one the most relevant models in epidemiology, it must be mentioned that several alternatives exist. The simplest one is the SI in which individuals cannot recover and is equivalent to the SIR for $\mu = 0$. In this case one can see that no epidemic threshold exists and, for how little is the transmission parameter, the epidemic will certainly involve all the population sooner or later. Different is the case of the SIS model in which an infected individual recovers but is once again susceptible. This model accounts for the fact that having experienced an infection does not imply one is immune, in some cases. From a mathematical perspective this slightly changes things: as we commented already, in the SIR, an infected individual cannot have been infected by a susceptible neighbor. On the opposite in the SIS this can happen: a infected individual can have been infected by someone who *now* is susceptible but that recovered. Actually, the difference between these two regimes cannot be understood from the NMF approximation because it is indeed not able to capture this dynamic. If one adopts a more refined approximation strategy, however, it is indeed possible to see that in the two cases the epidemic threshold varies.



Other models realistically add more compartments to the equations. Some of the most common are the *exposed*, *vaccinated* and *dead* compartments. Exposed people are those who already contracted the virus but that are still not contagious. After some time, they turn infected. The addition of these or other compartments may take into account of more complex medical and behavioral factors. On top of this, the model parameters may add further depth. We assumed $\beta, \mu$ to be constant and equal for all individuals, while one may assume that they depend, for instance, on age or mask-wearing.

It is worth mentioning that we only talked about simple contagion, *i.e.* the process in which one infected individual passes the disease over to a susceptible one. Thinking however of epidemiology in a broader sense, this is not the only possible alternative. Contagion may occur, for instance, only if one is exposed several times to an infected individual, each one passing a "piece". Only when all "pieces" are passed one becomes infected. Alternatively one can imagine contagion as a process in which it is necessary to have several infectious people interacting *at once* for the disease to be transmitted. We talk in these cases of *complex contagion*.

## 2.4 REFERENCES

- A.L. Barabasi, *Network Science, Chapter 10*, networksciencebook.com/chapter/10
  This chapter gives a wide and detailed view of epidemics on networks and can be used as a reading to get a bigger picture of the problem that we only treated in a very schematic way. From the analytical viewpoint, the book gives more results on the dynamics and considers also other models but does not use the NMF as it was presented here, but rather presents the degree-based mean field approach that is closely related to NMF.

- Pastor-Satorras, Castellano, Van Mieghem, Vespignani, *Epidemic processes in complex networks*, Reviews of modern physics 87.3 (2015): 925, arxiv.org/pdf/1408.2701.pdf
  This is a long but very important review. The intent of this review is less pedagogical that the Barabasi's book, but it takes a broader look at the problem from a technical perspective and summarizes different results.

- P. Van Mieghem, *Exact Markovian SIR and SIS epidemics on networks and an upper bound for the epidemic threshold*, arxiv.org/pdf/1402.1731.pdf
  This article details some rigorous results based on the NMF approximation with a notation very similar to the one adopted in this chapter.

# 3 CAVITY METHOD

3.1 Sparse and tree-like graphs . . . . . . . . . . . . . . . . . . . . 25
3.1.1 Locally tree-like graphs . . . . . . . . . . . . . . . . . 25
3.1.2 Conditional independence . . . . . . . . . . . . . . . . 26
3.2 Factorizing probability distributions on trees . . . . . . . . . 27
3.2.1 Cavity method on tree-like graphs . . . . . . . . . . . 30
3.3 The non-backtracking matrix . . . . . . . . . . . . . . . . . . . 31
3.4 An application to epidemics . . . . . . . . . . . . . . . . . . . 32
3.5 Efficiently computing the spectral radius of $B$ . . . . . . . . 34
3.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
3.7 References . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

## 3.1 SPARSE AND TREE-LIKE GRAPHS

In the previous lecture we introduced the *naïve mean field* (NMF) approximation to study the epidemic threshold on a graph. We saw, however, that this approximation is appropriate only for dense graphs, while most of real world graphs are (luckily) sparse. We here investigate an alternative approach that is well suited for sparse random graphs and that builds an approximation based on the locally tree-like[1] structure of a sparse *Erdős-Rényi* (ER) graph.

### 3.1.1 LOCALLY TREE-LIKE GRAPHS

Let us first introduce the concept of *rooted* graph $\mathcal{G}_i(\mathcal{V},\mathcal{E})$ that is a graph in which a particular node $i \in \mathcal{V}$ (the *root*) is specified. Denote with $\mathfrak{B}_i(t)$ the ball of radius $t$ around the node $i$, *i.e.* the sub-graph made by the set of all nodes that can be reached from $i$ in at most $t$ steps and the corresponding edges. If the law of $\mathfrak{B}_i(t)$ under uniformly random sampling of the root admits a limit $\mathcal{L}$, then we call it *local weak limit*. In words, $\mathcal{L}$ is the asymptotic local distribution of $\mathcal{G}(\mathcal{V},\mathcal{E})$ as seen from a random vertex. A relevant result concerning sparse ER graphs[2] is that they locally converge to a tree, as more formally stated in Property 3.1.

1 We recall that an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$is said to be a tree if it is connected and it does not contain any cycle.
2 But actually also other sparse random graphs.

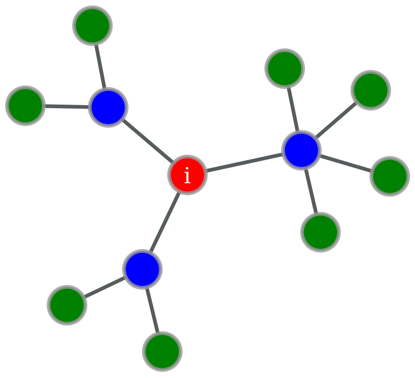


Figure 3.1: A toy example of a Poisson GW tree. In red the root $i$, in blue the first generation of nodes, in green the second.

*Local convergence to a tree*

**Property 3.1** (Convergence to Poisson *Galton Watson tree* (GW) tree of ER)**.** *A sparse ER random graph with $n \to \infty$ rooted at $i$ with average degree $d = O_n(1)$ converges locally to a Poisson GW tree so obtained: consider the node $i$ as the root and generate $d_i$ neighbours (called* sons*), where $d_i$ is a Poisson random variable with parameter $d$ and iteratively repeat the operation for each son.*

As a consequence of this property, a sparse ER graph locally *looks like* a tree and hence, with high probability, there are no cycles of finite size.[3] Figure 3.1 displays an example of a Poisson GW tree, rooted at $i$.

Let us now describe a fundamental property of probability distributions defined over the nodes of a tree, namely, *conditional independence.*

## 3.1.2 CONDITIONAL INDEPENDENCE

Consider three random variables $x, y, z$. We say that $x$ and $y$ are conditionally independent given $z$ if

*Conditional independence*

$$\mathbb{P}[x, y \mid z] = \mathbb{P}[x \mid z]\mathbb{P}[y \mid z]. \tag{3.1}$$

Note that two random variables are independent if we can write $\mathbb{P}[xy] = \mathbb{P}[x]\mathbb{P}[y]$, that is what we did in the NMF approximation. Conditional independence holds only on the conditional probabilities and in general $\mathbb{E}[xy] \neq \mathbb{E}[x]\mathbb{E}[y]$. Now, the relation between conditional independence and trees is that all variables associated to the neighbors of a same node are conditionally independent given the value of their common neighbor, or, more formally

$$\forall j \neq k \in \partial i, \quad \mathbb{P}[x_j, x_k \mid x_i] = \mathbb{P}[x_j \mid x_i]\mathbb{P}[x_k \mid x_i].$$

Let us try to understand why. Suppose there is a piece of news that is propagating on the network through contacts. If a node knows it, then with some

3 Recall the local convergence definition is given in the asymptotic limit of $n \to \infty$. Finite cycles will exist, but their size will depend on $n$ (for instance they may grow as $\log(n)$) and thus diverge in the large $n$ limit.

probability it will talk about it to its neighbors that will also be aware of it from that moment on. Now, let us consider a node $i$ (as the red one in Figure 3.1) and two of its neighbors $j, k$ (in blue, same figure). If we let $x_i = 1$ if $i$ knows the piece of information and $x_i = 0$ otherwise, then, clearly $\mathbb{P}[x_j, x_k] \neq \mathbb{P}[x_j]\mathbb{P}[x_k]$. If the variables were independent, the notion of $x_j$ would not allow me to say anything about $x_k$, but it turns out that if I know $x_j$ I can tell something about $x_k$. The two random variables bring information one of the other because there is a (short) path connecting them and the piece of information may flow from one node to the other. However, since we are considering a tree, there is only one such path, that is the one going through $i$. If we suppose to know $x_i$, then knowing also $x_j$ does not add any information when trying to predict $x_k$, because the only influence $j$ has on $k$ is through $i$. This is the effect of conditional independence.

Notably, conditional independence implies the following relation that we will exploit later on.

$$\begin{aligned}\mathbb{P}(x_i, x_j, x_k) &= \mathbb{P}(x_j, x_k|x_i)\mathbb{P}(x_i)\\ &= \mathbb{P}(x_j|x_i)\mathbb{P}(x_k|x_i)\mathbb{P}(x_i)\\ &= \frac{\mathbb{P}(x_i, x_j)\mathbb{P}(x_i, x_k)}{P(x_i)}\end{aligned} \tag{3.2}$$

Knowing that a sparse ER random graph asymptotically "looks like" a tree, we can now simplify our analysis exploiting conditional independence and the graph structure to introduce the *cavity method*, or *belief propagation*.

## 3.2 FACTORIZING PROBABILITY DISTRIBUTIONS ON TREES

As we saw in Chapter 2, a difficulty of studying processes on a graph is to compute the edge marginal probability distributions that cannot be simply assumed to factorize as the product of the node marginals. The cavity method builds on the fact that the edge marginals can be exactly calculated on tree with a recursive formula, as stated in Lemma 3.1.

**Lemma 3.1.** *Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a tree and let $\mu(\boldsymbol{x})$ be a probability distribution defined on $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that can be written as*

$$\mu(\boldsymbol{x}) = \prod_{(ij)\in\mathcal{E}} \phi_{ij}(x_i, x_j). \tag{3.3}$$

*Then the edge marginal $\mu_{ij}(x_i, x_j) = \sum_{\boldsymbol{x}\setminus x_i, x_j} \mu(\boldsymbol{x})$ and the node marginal $\mu_i(x_i) = \sum_{\boldsymbol{x}\setminus x_i} \mu(\boldsymbol{x})$ can be written in the following form:*

$$\mu_i(x_i) = \prod_{k\in\partial i} \eta_{ik}(x_i) \tag{3.4}$$

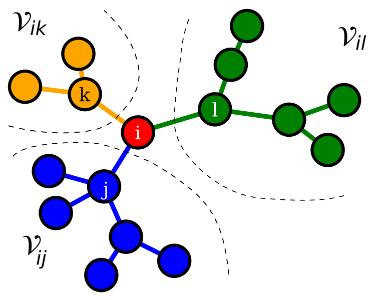


Figure 3.2: Sketch of a tree. The node $i$ in red, while in green, blue and orange the edges and nodes $\mathcal{E}_{ix}, \mathcal{V}_{ix}$ with $x = j, k, l$, respectively. Note that $i$ belongs to $\mathcal{V}_{ik}, \mathcal{V}_{ij}$ and $\mathcal{V}_{i\ell}$.

$$\mu_{ij}(x_i x_j) = \phi_{ij}(x_i, x_j) \prod_{k\in\partial i\setminus j} \eta_{ik}(x_i) \prod_{\ell\in\partial j\setminus i} \eta_{jl}(x_j). \tag{3.5}$$

*The quantities $\eta_{ij}(x_i)$ are defined on the set of directed edges.*

From Equations (3.4, 3.5), exploiting $\mu_i(x_i) = \sum_{x_j} \mu_{ij}(x_i, x_j)$, it is obtained that the *messages* have to satisfy the following fixed point equation

*Message passing*

$$\eta_{ij}(x_i) = \sum_{x_j} \phi_{ij}(x_i, x_j) \prod_{\ell\in\partial j\setminus i} \eta_{jl}(x_j). \tag{3.6}$$

Let us now sketch here the proof of Lemma 3.1 since it is very pedagogical and helpful to understand the essence of cavity method.

*Proof of Lemma 3.1.* Denote with $\mathcal{E}_d$ the set of directed edges of $\mathcal{G}(\mathcal{V}, \mathcal{E})$and consider $(ij) \in \mathcal{E}_\text{d}$. We define $\mathcal{E}_{ij}$ as the set of all edges that can be reached from $i$ only passing through $j$. As a consequence of the fact that on a tree there exists a unique path connecting any two nodes – since there are no cycles –, the two following properties are verified:

$$\forall\, i \in \mathcal{V}, \quad \mathcal{E} = \bigcup_{k\in\partial i} \mathcal{E}_{ik}\,; \tag{3.7}$$

$$\forall\, (ij) \in \mathcal{E}, \quad \mathcal{E} = \{(ij)\} \cup \underbrace{\bigcup_{k\in\partial i\setminus j} \mathcal{E}_{ik}}_{\text{reached from } (ji)} \cup \underbrace{\bigcup_{\ell\in\partial j\setminus i} \mathcal{E}_{jl}}_{\text{reached from } (ij)} . \tag{3.8}$$

Furthermore, note that $\forall\, j \neq k,\ \mathcal{E}_{ij} \cap \mathcal{E}_{ik} = \emptyset$. A pictorial representation of the definition of $\mathcal{E}_{ij}$ is given in Figure 3.2. Exploiting Equation (3.7), $\mu(\boldsymbol{x})$ can then be written as:

$$\mu(\boldsymbol{x}) = \prod_{(ab)\in\mathcal{E}} \phi_{ab}(x_a, x_b) = \prod_{k\in\partial i} \prod_{(ab)\in\mathcal{E}_{ik}} \phi_{ab}(x_a, x_b) := \prod_{k\in\partial i} \psi_{ik}(\boldsymbol{x}_{\mathcal{V}_{ik}}),$$

where $\mathcal{V}_{ik}$ is the set of nodes connected by edges in $\mathcal{E}_{ik}$ ($i$ included) and $\boldsymbol{x}_{\mathcal{V}_{ik}}$ is the variable vector corresponding to those nodes. The node marginal can then be written in the following form

$$\mu_i(x_i) = \sum_{\boldsymbol{x}\backslash x_i} \mu(\boldsymbol{x}) = \sum_{\boldsymbol{x}\backslash x_i} \prod_{k\in\partial i} \psi_{ik}(\boldsymbol{x}_{\mathcal{V}_{ik}}) = \prod_{k\in\partial i} \sum_{\boldsymbol{x}_{\mathcal{V}_{ik}}\backslash x_i} \psi_{ik}(\boldsymbol{x}_{\mathcal{V}_{ik}}).$$

Denoting $\eta_{ik}(x_i) := \sum_{\boldsymbol{x}_{\mathcal{V}_{ik}}\backslash x_i} \psi_{ik}(\boldsymbol{x}_{\mathcal{V}_{ik}})$, we obtain the first equation of Lemma 3.1. Note that $\eta_{ki}(x_i)$ indeed only depends on $x_i$ since the sum is run over all variables $\boldsymbol{x}_{\mathcal{V}_{ik}}$, except $x_i$. Proceeding in a similar way, the expression of the edge marginal is obtained from Equation (3.8).

$$\begin{aligned}
\mu_{ij}(x_i, x_j) &= \sum_{\boldsymbol{x}\backslash x_i x_j} \mu(\boldsymbol{x}) \\
&= \sum_{\boldsymbol{x}\backslash x_i x_j} \phi_{ij}(x_i, x_j) \prod_{k\in\partial i\backslash j} \prod_{(ab)\in\mathcal{E}_{ik}} \phi_{ab}(x_a, x_b) \prod_{\ell\in\partial j\backslash i} \prod_{(cd)\in\mathcal{E}_{k\ell}} \phi_{cd}(x_c, x_d) \\
&= \sum_{\boldsymbol{x}\backslash x_i x_j} \phi_{ij}(x_i, x_j) \prod_{k\in\partial i\backslash j} \psi_{ik}(\boldsymbol{x}_{\mathcal{V}_{ik}}) \prod_{\ell\in\partial j\backslash i} \psi_{j\ell}(\boldsymbol{x}_{\mathcal{V}_{j\ell}}) \\
&= \phi_{ij}(x_i, x_j) \left( \prod_{k\in\partial i\backslash j} \sum_{\boldsymbol{x}_{\mathcal{V}_{ik}}\backslash i} \psi_{ik}(\boldsymbol{x}_{\mathcal{V}_{ik}}) \right) \cdot \left( \prod_{\ell\in\partial j\backslash i} \sum_{\boldsymbol{x}_{\mathcal{V}_{j\ell}}\backslash j} \psi_{j\ell}(\boldsymbol{x}_{\mathcal{V}_{j\ell}}) \right) \\
&= \phi_{ij}(x_i, x_j) \prod_{k\in\partial i\backslash j} \eta_{ik}(x_i) \cdot \prod_{\ell\in\partial j\backslash i} \eta_{j\ell}(x_j).
\end{aligned}$$

□

The essence of the proof of Lemma 3.1 relies on the conditional independence of the node variables on trees. More specifically, to obtain Equation (3.7), one could imagine to remove the node $i$, obtaining $d_i$ (the degree of $i$) disconnected sub-graphs in which variables are independent and hence factorize. Similarly Equation (3.8) is obtained removing the nodes $i$ and $j$ from the graph. We now show that on a tree, Equation (3.2) is verified.

**Lemma 3.2.** *Let $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a tree and let $\mu(\boldsymbol{x})$ be a probability distribution defined on $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that can be written as per Equation (3.3). Then, taking $j, k \in \partial i$ with $j \neq k$ we can write*

$$\mathbb{P}(x_i, x_j, x_k) = \frac{\mathbb{P}(x_i, x_j)\mathbb{P}(x_i, x_k)}{\mathbb{P}(x_i)}$$

*Proof.* For simplicity, we will drop the dependence on the variables $\boldsymbol{x}$. Following the same procedure we used to prove Lemma 3.1, we can easily show that

$$\mathbb{P}(x_i, x_j, x_k) = \phi_{ij}\phi_{ik} \prod_{p\in\partial j\backslash i} \eta_{jp} \prod_{q\in\partial k\backslash i} \eta_{kq} \prod_{r\in\partial i\backslash\{j,k\}} \eta_{ir}.$$

This can be rewritten in the following form

$$\mathbb{P}(x_i, x_j, x_k) = \frac{\left(\phi_{ij} \prod_{p \in \partial j \setminus i} \eta_{jp} \prod_{r \in \partial i \setminus j} \eta_{ir}\right)\left(\phi_{ik} \prod_{q \in \partial k \setminus i} \eta_{kq} \prod_{r \in \partial i \setminus k} \eta_{ir}\right)}{\eta_{ik} \prod_{k \in \partial i \setminus k} \eta_{ir}}$$
$$= \frac{\mathbb{P}_{ij}(x_i, x_j)\mathbb{P}_{ik}(x_i, x_k)}{\mathbb{P}_i(x_i)},$$

where in the last step we used the relations shown in Lemma 3.1. □

Given these results on trees, let us now move to sparse graphs.

### 3.2.1 CAVITY METHOD ON TREE-LIKE GRAPHS

When we consider a graph that is not a tree, the proof we gave above does not generalize because $\mathcal{E}_{ij} \cup \mathcal{E}_{ik} \neq \emptyset$, due to the presence of cycles. In a tree-like graph, however, we know that cycles do not have a short length. When considering two nodes $j, k$ in the neighborhood of a same node, there will be a short path of length 2 connecting them and other very long paths that pass through other nodes. The main intuition we have is that all those long paths are unimportant and the main channel of of relation is the short path connecting them. For this reason, we simply use conditional independence as an *ansatz* that is asymptotically verified on sparse graphs. We can then rewrite the cavity equations on a graph (with cycles) a follows.

The cavity fixed point equations

*Factorizing probabilities on sparse graphs*

$$\eta_{ji}(x_i) = \frac{Z_i}{Z_{ji}} \sum_{x_j} \phi_{ij}(x_i, x_j) \prod_{\ell \in \partial j \setminus i} \eta_{j\ell}(x_j).$$
$$\mu_i(x_i) \approx \frac{1}{Z_i} \prod_{k \in \partial i} \eta_{ik}(x_i)$$
$$\mu_{ij}(x_i, x_j) \approx \frac{1}{Z_{ij}} \phi_{ij}(x_i, x_j) \prod_{k \in \partial i \setminus j} \eta_{ik}(x_i) \prod_{\ell \in \partial j \setminus i} \eta_{j\ell}(x_j).$$

Given these equations, we now introduce the *non-backtracking matrix* or *Hashimoto operator* that naturally comes into play from the cavity method.

## 3.3 THE NON-BACKTRACKING MATRIX

Let us consider the fixed point cavity equation, letting $r_{ij}(x_i) = \log(\eta_{ij}(x_i))$ and $C_{ji} = \log(Z_i) - \log(Z_{ji})$.

$$r_{ji}(x_i) = C_{ji} + \log\left(\sum_{x_j} \exp\left\{\log \phi_{ij}(x_i, x_j) + \sum_{\ell\in\partial j\setminus i} r_{j\ell}(x_j).\right\}\right).$$

Focusing on the sum $\sum_{\ell\in\partial j\setminus i}$, we introduce the non-backtracking matrix.

*The non-backtracking matrix is naturally related to the cavity method*

**The non-backtracking matrix**

Let $\mathcal{E}_\mathrm{d}$ be the set of *directed* edges of a graph $\mathcal{G}$. We define the non-backtracking matrix $B \in [0,1]^{|\mathcal{E}_\mathrm{d}|\times|\mathcal{E}_\mathrm{d}|}$ as

$$B_{(ij),(k\ell)} = \delta_{jk}(1-\delta_{i\ell}), \tag{3.9}$$

for all $(ij), (k\ell) \in \mathcal{E}_\mathrm{d}$. Then, given a vector $\boldsymbol{g} \in \mathbb{R}^{|\mathcal{E}_\mathrm{d}|}$, we have

$$(B\boldsymbol{g})_{(ij)} = \sum_{\ell\in\partial j\setminus i} g_{j\ell}.$$

In simple words, we can say that the non-backtracking matrix $B$ is the linear operator associated to the cavity approximation. To interpret its definition, in essence we can see $B$ as the adjacency matrix of graph in which each node is a directed edge of $\mathcal{G}(\mathcal{V},\mathcal{E})$ and two nodes are neighboring if they are successive and the second one is not the reversed of the first. Unlike the adjacency matrix, the spectral radius of the non-backtracking matrix is "well behaved" in the sparse regime, as stated by the following theorem.

**Theorem 3.1.** *Consider a symmetric matrix $A \in [0,1]^{n\times n}$ in which the entries are set to $1$ independently (up to symmetry) with probability $\mathbb{P}(A_{ij}=1) = P_{ij}$ and $P_{ij} = O_n(n^{-1})$ for all $i, j$. Then, for all large $n$ with high probability, the spectral radius of the non-backtracking matrix $B$ associated with the adjacency matrix $A$ is*

$$\rho(B) = \rho(P) + o_n(1).$$

This theorem is very general and allows us to consider easily both the ER and the configuration model. For the ER, we have $P = \frac{\langle d\rangle}{n}\mathbf{1}_n\mathbf{1}_n^T$ and $\rho(P) = \langle d\rangle$, the expected average degree. For the configuration model, instead, we can write $P = \frac{1}{2|\mathcal{E}|}\boldsymbol{d}\boldsymbol{d}^T$ and thus $\rho(B) = \frac{\langle d^2\rangle}{\langle d\rangle}$.

Let us now see how the matrix $B$ enters into play when studying the epidemic threshold on a sparse graph with the cavity method.

## 3.4 AN APPLICATION TO EPIDEMICS

Let us now use the cavity method to find the epidemic threshold on a sparse graph, in which the NMF is not appropriate. Let us consider Equation (2.2) that describes the dynamics of the infected state in a *Susceptible-Infected-Recovered model* (SIR) model. We can write

$$\partial_t \mathbb{P}(x_i(t) = I) = \beta \sum_{j\in\mathcal{V}} A_{ij}\mathbb{P}\left(x_i(t) = S, x_j(t) = I\right) - \mu\mathbb{P}(x_i(t) = I).$$

The whole point of going beyond NMF is to realize that $x_i(t), x_j(t)$ are not independent if $A_{ij} = 1$. To move forward, let us lighten a bit the notation. We define $\boldsymbol{p} \in \mathbb{R}^n$ the vector with entries $p_i(t) = \mathbb{P}(x_i(t) = I)$ and with $\boldsymbol{\chi}(t) \in \mathbb{R}^{2|\mathcal{E}|}$ the vector with entries $\chi_{ij}(t) = \mathbb{P}(x_i(t) = S, x_j(t) = I)$. Note that the vector $\boldsymbol{\chi}$ is defined over the set of *directed* edges of the graph and that $\chi_{ij}(t) \neq \chi_{ji}(t)$, in general. With this notation, we can rewrite the state evolution as

$$\partial_t \boldsymbol{p}(t) = \beta T\boldsymbol{\chi}(t) - \mu\boldsymbol{p}(t), \tag{3.10}$$

where we introduced the matrix $T \in \mathbb{R}^{n\times 2|\mathcal{E}|}$, defined as $T_{i,(ab)} = \delta_{ia}A_{ab}$. Now, to proceed, we need to write a state evolution equation for the vector $\boldsymbol{\chi}(t)$ as well. The probability that $i$ and $j$ are respectively susceptible and infected at a given time step implies that they were both susceptible and $j$ got infected (but certainly not from $i$), while $i$ did not or that $j$ was already infected, it did not recover and $i$ did not get infected.

$$\chi_{ij}(t+dt) =$$
$$\mathbb{E}\left[\delta[x_i(t) = S]\delta[x_j(t) = S]\underbrace{\left(1 - \beta dt \sum_{k\in\partial i\setminus j}\delta[x_k(t) = I]\right)}_{i\text{ does not get infected}}\underbrace{\beta dt\left(\sum_{\ell\in\partial j\setminus i}\delta[x_\ell(t) = I]\right)}_{j\text{ gets infected}}\right]$$
$$+\,\mathbb{E}\left[\delta[x_i(t) = S]\delta[x_j(t) = I]\underbrace{\left(1 - \beta dt \sum_{k\in\partial i\setminus j}\delta[x_k(t) = I] - \beta dt\right)}_{i\text{ does not get infected}}\underbrace{(1-\mu dt)}_{j\text{ recovers}}\right].$$

Now, there are two approximations we can perform to simplify the analysis. The first one is just to take the limit for $dt \to 0$ and remove the higher order terms. The second one is done exploiting conditional independence. We first remove the higher order terms in $dt$

$$\chi_{ij}(t+dt) \overset{dt\to 0}{=} \beta dt\left(\sum_{\ell\in\partial j\setminus i}\mathbb{E}\left[\delta[x_i(t) = S]\delta[x_j(t) = S]\delta[x_\ell(t) = I]\right]\right)$$

$$+\chi_{ij}(t)(1-\beta dt-\mu dt)-\beta dt\sum_{k\in\partial i\setminus j}\mathbb{E}\left[\delta[x_i(t)=S]\delta[x_j(t)=I]\delta[x_k(t)=I]\right].$$

We now exploit conditional independence. We denote with $\Omega_{ij}=\mathbb{P}(x_i(t)=S,x_j(t)=S)$ and $s_i=\mathbb{P}(x_i(t)=S)$ and write

$$\mathbb{E}\left[\delta[x_i(t)=S]\delta[x_j(t)=S]\delta[x_\ell(t)=I]\right]=\frac{\Omega_{ij}(t)\cdot\chi_{j\ell}(t)}{s_j(t)}$$
$$\mathbb{E}\left[\delta[x_i(t)=S]\delta[x_j(t)=I]\delta[x_k(t)=I]\right]=\frac{\chi_{ik}(t)\cdot\chi_{ij}(t)}{s_i(t)},$$

thus turning the state evolution equation into

$$\chi_{ij}(t+dt)\overset{dt\to 0}{=}\beta dt\frac{\Omega_{ij}(t)}{s_j(t)}\sum_{\ell\in\partial j\setminus i}\chi_{j\ell}(t)+\chi_{ij}(t)(1-\beta dt-\mu dt)-\beta dt\frac{\chi_{ij}(t)}{s_i(t)}\sum_{k\in\partial i\setminus j}\chi_{ik}(t).$$

To get the epidemic threshold we now want to linearize around the epidemic-free fixed point that is $\Omega_{ij},s_i,s_j\to 1$ and $\chi_{ij}\to 0$ and get

$$\chi_{ij}(t+dt)\overset{dt\to 0}{=}\beta dt\sum_{\ell\in\partial j\setminus i}\chi_{j\ell}(t)+\chi_{ij}(t)(1-\beta dt-\mu dt),$$

that can be written as

$$\partial_t\boldsymbol{\chi}(t)=(\beta B-(\beta+\mu)I_{2|\mathcal{E}|})\boldsymbol{\chi}(t).$$

Injecting this result in Equation (3.10)

$$\begin{pmatrix}\partial_t\boldsymbol{p}(t)\\ \partial_t\boldsymbol{\chi}(t)\end{pmatrix}=\begin{pmatrix}-\mu I_n & \beta T\\ \beta B-(\beta+\mu)I_{2|\mathcal{E}|} & 0\end{pmatrix}\begin{pmatrix}\boldsymbol{p}(t)\\ \boldsymbol{\chi}(t)\end{pmatrix}.$$

From a simple calculation one sees that the stability condition is now obtained on heterogeneous random graphs as

**The reproductive number according to the cavity method**

$$R_0=\frac{\beta(\rho(B)-1)}{\mu}=\frac{\beta}{\mu}\left(\frac{\langle d^2\rangle}{\langle d\rangle}-1\right).$$

If we compare this result with the one obtained from NMF, the main difference is the appearance of the term "$-1$" that accounts the fact that when there is a susceptible-infected pair $(ij)$, certainly $i$ did not infect $j$. This comes "for free", in the sense that we did not have to add this correction manually and it naturally came from the equations. Other than that the results seem quite similar, but we must not forget that $\rho(A)$ is very close to $\rho(B)$ on dense networks but not on sparse ones. In other words, $\frac{\langle d^2\rangle}{\langle d\rangle}$ still is the good quantity to look at, but it is actually not the spectral radius of $A$ in the sparse regime. Using the method explained in Chapter 2, we can derive the $R_0$ of the SIR model in the presences of vaccination. Note that, while with NMF this

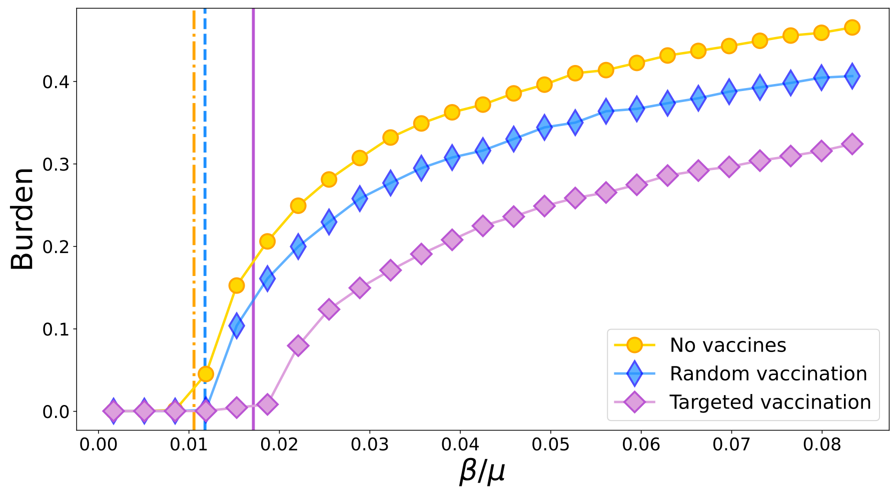


Figure 3.3: **Epidemic burden as a function of $\beta/\mu$ for different vaccination strategies**. The yellow dots are the burden (fraction of non-susceptible people at the end of the simulation) in absence of vaccination; the blue diamonds correspond to the random vaccination; the purple squares are the targeted vaccination in which each person is vaccinated with a probability proportional to the degree. The vertical lines (color coded) correspond to the position of the transition as predicted by the cavity method for the three different scenarios.

result was not rigorous because the effect of vaccination is to sparsify the network, with the cavity method we obtain e precise bound. The simulation shows the goodness of the cavity method in this setting as well as the efficacy of targeted vaccination strategies, as shown in Figure 3.3, we introduce $Q_{i,(ab)} = \delta_{ib}A_{ab}$ and $M_{(ab),(cd)} = A_{ab}A_{cd}\delta_{bc}\delta_{ad}$.

## 3.5 EFFICIENTLY COMPUTING THE SPECTRAL RADIUS OF $B$

We have seen that the non-backtracking matrix naturally appears when adopting the cavity approximation. The $B$ matrix defined in Equation (3.9) however is large (its size scales with the number of edges, not of nodes) and it might not be so straightforward to build. However, we now show that there is a matrix $B_\mathrm{p}$ of size $2n \times 2n$ whose eigenvalues are also eigenvalues of $B$ and it is much more easily built. Similarly to the matrix $T$ introduced earlier $T_{i,(ab)} = \delta_{ia}A_{ab}$, we introduce $Q \in \mathbb{R}^{n\times 2|\mathcal{E}|}$ and $M \in \mathbb{R}^{2|\mathcal{E}|\times 2|\mathcal{E}|}$

$$\begin{aligned} Q_{i,(ab)} &= \delta_{ib}A_{ab} \\ M_{(ab),(cd)} &= \delta_{bc}\delta_{ad}A_{ab}A_{cd}. \end{aligned}$$

Now, the following relations[4] are satisfied:

4 You may try to obtain these relations as an exercise.

$$\boxed{\begin{aligned} Q^T T - M &= B \\ TQ^T &= A \\ QQ^T &= D \\ QM &= T \\ TM &= Q \end{aligned}}$$

With these relations at hand, suppose $\boldsymbol{g}$ is the leading eigenvector of $B$ with eigenvalue $\rho$, then

$$\rho T\boldsymbol{g} = TB\boldsymbol{g} = T(Q^T T - M)\boldsymbol{g} = AT\boldsymbol{g} - Q\boldsymbol{g},$$

and

$$\rho Q\boldsymbol{g} = QB\boldsymbol{g} = Q(Q^T T - M)\boldsymbol{g} = DT\boldsymbol{g} - T\boldsymbol{g}.$$

Denoting $P\boldsymbol{g} = \boldsymbol{x}$ and $Q\boldsymbol{g} = \boldsymbol{y}$ for simplicity, we obtain,

$$\underbrace{\begin{pmatrix} A & -I_n \\ D - I_n & 0 \end{pmatrix}}_{B_{\mathrm{p}}} \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix} = \rho \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{pmatrix}, \tag{3.11}$$

where we introduced the smaller matrix $B_{\mathrm{p}} \in \mathbb{R}^{2n\times 2n}$. This matrix has the same eigenvalues as $B$ (except those equal to $\pm 1$ that have a different degeneracy). This matrix can thus be used to efficiently compute $\rho(B)$.

## 3.6 CONCLUSION

In this section we introduced the cavity method and the closely related non-backtracking matrix. Unlike the NMF approximation, this method is well suited for sparse graphs, being asymptotically exact on sparse random graphs, such as the ER. Consequently, this "second order" approximation (in which we assume independence at the edge, rather than the node level) is more accurate, but we must recall that real-world networks are often sparse but not locally tree-like. Sparse random graphs, in fact, tend to have a much smaller clustering coefficient than a real network with the same average degree. The low clustering coefficient, however, implies the absence of short loops, making the cavity method exact on random, but not on real world graphs, in general. This approximation is a improvement over NMF approach that is appropriate to deal with some of its limitations.

## 3.7 REFERENCES

- Wainwright, Jordan: *Graphical Models, Exponential Families, and Variational Inference*

This review is a milestone for physics methods on graphs and in Chapter 4 it treats the cavity method.

- Mezard, Montanari: *Information, Physics and Computation.*
  This is another relevant reference. The cavity method is discussed in Chapter 14.

4

# GRAPH FOURIER TRANSFORM

4.1 The Fourier transform . . . . . . . . . . . . . . . . . . . . . . . 37
  4.1.1 Deriving the Fourier transform . . . . . . . . . . . . 38
4.2 The graph Fourier transform . . . . . . . . . . . . . . . . . . . 38
  4.2.1 Heat diffusion on graphs . . . . . . . . . . . . . . . . 38
  4.2.2 Decompose a signal on the graph Fourier modes . . 39
  4.2.3 Frequencies on graphs . . . . . . . . . . . . . . . . . . 40
  4.2.4 Some basic spectral properties of the graph Laplacian 42
4.3 Graph signal processing . . . . . . . . . . . . . . . . . . . . . . 44
  4.3.1 Tikhonov regularization . . . . . . . . . . . . . . . . . 44
  4.3.2 Filtering . . . . . . . . . . . . . . . . . . . . . . . . . . 45
4.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 48
4.5 References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49

## 4.1 THE FOURIER TRANSFORM

The Fourier transform is a fundamental tool in mathematical analysis that was introduced by Joseph Fourier in to solve the heat equation that in the one dimensional case reads

$$\partial_t u(x,t) = \alpha\, \partial_x^2 u(x,t), \tag{4.1}$$

*The heat equation*

where $\alpha$ is the thermal diffusivity. The Fourier transform was introduced with the scope of solving this differential equation and it consists of moving to the "space of frequencies" that, in some cases, like this one, can be very convenient. This powerful tool of signal processing, in fact, allows one to define simple operations such as filtering and convolutions in the frequency domain that would otherwise be very hard to perform in the original one. Given its power, we want to extend its definition to non-Euclidean domains and in particular to functions that are defined on the vertices of a graph.

As a first step, we here re-derive the definition of Fourier transform, starting from Equation 4.1 to then attempt to generalize it to the graph domain.

### 4.1.1 DERIVING THE FOURIER TRANSFORM

In Equation 4.1, the second derivative can be seen as an operator that acts on the function $u(x,t)$. The simplest way to derive the Fourier transform is to see it as a decomposition of $u(x,t)$ on the eigenfunctions of that operator. We start from these two basic properties

*The eigenfunctions of the second derivative*

$$\partial_x^2 \, e^{ikx} = -k^2 e^{ikx}, \tag{4.2}$$

$$\frac{1}{2\pi} \int_{\mathbb{R}} dk \, e^{ikx} = \delta(x). \tag{4.3}$$

These two equations imply that $e^{ikx}$ are the eigenfunctions of the second derivative and they form an orthogonal basis. For a given function $f(x)$ we can thus write

*The Fourier transform*

$$\begin{aligned} f(x) &= \int_{\mathbb{R}} dy \, f(y)\delta(x-y) \\ &\overset{(a)}{=} \frac{1}{2\pi} \int_{\mathbb{R}} dy \, f(y) \int_{\mathbb{R}} dk \, e^{ik(x-y)} \\ &\overset{(b)}{=} \frac{1}{2\pi} \int_{\mathbb{R}} dk \, e^{ikx} \underbrace{\int_{\mathbb{R}} dy \, f(y) e^{-iky}}_{\hat{f}(k)} \\ &\overset{(c)}{=} \frac{1}{2\pi} \int_{\mathbb{R}} dk \, \hat{f}(k) e^{ikx}, \end{aligned}$$

where in $(a)$ we used the definition of the $\delta$ function given in Equation (4.3), in $(b)$ we inverted the order of the two integrals and in $(c)$ we introduced the Fourier transform definition as $\hat{f}_k$. The function $f(x)$ is then expressed as combination of the eigenfunctions of the second derivative – Laplacian, in higher dimensions – operator. Here, the $\hat{f}(k)$ plays the role of the "weight coefficient" of each eigenfunction.

Repeating these steps, we now derive a the graph Fourier transform .

## 4.2 THE GRAPH FOURIER TRANSFORM

We here derive the *Graph Fourier transform* (GFT) definition, first writing the heat equation on a graph, then decomposing a signal on the basis of the new Laplacian operator as done before.

### 4.2.1 HEAT DIFFUSION ON GRAPHS

Consider a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ and let $u_i(t)$ be a signal defined on node $i \in \mathcal{V}$ at time $t$. We suppose that this signal evolves with a diffusive process on the

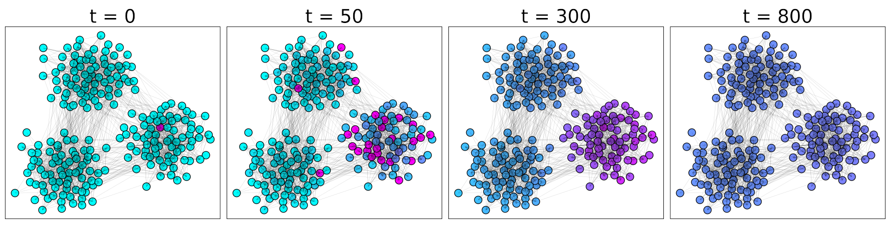


Figure 4.1: **Diffusion on a graph with communities**. In color code we have the value $u_i(t)$ and the title indicates the four different time-steps.

graph and that, at each time-step, a $i$ exchanges a fraction $\alpha$ of information $u_i(t)$ with each of its neighbors. In equations, this becomes

$$u_i(t+dt) = u_i(t) + \alpha \sum_{j\in\mathcal{V}} A_{ij}\big(\underbrace{u_j(t)}_{j\to i} - \underbrace{u_i(t)}_{i\to j}\big).$$

This allows us to rewrite the diffusion equation in vector form as

*Heat diffusion on a graph*

$$\partial_t \boldsymbol{u}(t) = -\alpha(D-A)\boldsymbol{u}(t) := -\alpha L\boldsymbol{u}(t),$$

where we introduced the graph Laplacian matrix $L$. In Figure 4.1we show a simple example of diffusion on a graph with three groups of nodes – called communities – that are more densely connected among themselves than with other. We initialize $u_i(t) = 0$ for all nodes except and perform the simulation. The result reported at three successive time-steps evidences a diffusion process that follows a concept of proximity on the graph: first the signal is propagated in the node in the same community and to few of the other communities ($t = 50$), then it progressively tends to a homogeneous distribution. Let us now formally define the graph Laplacian matrix

**The graph Laplacian matrix**

*The graph Laplacian matrix*

Given an undirected graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with $A \in [0,1]^{n\times n}$ as adjacency matrix and $D = \mathrm{diag}(A\mathbf{1}_n)$ the diagonal degree matrix, the graph Laplacian associated to $\mathcal{G}(\mathcal{V},\mathcal{E})$ is

$$L = D - A \tag{4.4}$$

Given this matrix, we now use it to decompose the signal $\boldsymbol{u}$ on its basis and show that it how we can relate this decomposition to a "frequency domain".

### 4.2.2 DECOMPOSE A SIGNAL ON THE GRAPH FOURIER MODES

Following the same procedure as above, to define the graph Fourier transform we decompose the signal on the orthonormal basis defined by the eigenvectors of $L$. Since $L$ is a Hermitian matrix, these eigenvectors form and orthonormal basis and they play the same role as $e^{ikx}$ in the classical Fourier transform. We denote with $\boldsymbol{x}_k$ the eigenvector $k$ associated with $\lambda_k$, the $k$ smallest eigenvalue of $L$ *i.e.* $L\boldsymbol{x}_k = \lambda_k \boldsymbol{x}_k$, then

*The eigenvectors of L form an orthonormal basis*

$$I_n = \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{x}_k^T.$$

For any vector $\boldsymbol{u} \in \mathbb{R}^n$ defined on the vertices of $\mathcal{G}(\mathcal{V}, \mathcal{E})$, we can write

*Projecting $\boldsymbol{u}$ on the basis of L*

$$\boldsymbol{u} = I_n \boldsymbol{u} = \sum_{k=1}^{n} \boldsymbol{x}_k \underbrace{\boldsymbol{x}_k^T \boldsymbol{u}}_{\hat{u}_k} = \sum_{k=1}^{n} \boldsymbol{x}_k \hat{u}_k.$$

By analogy with the classical Fourier transform, we define $\hat{\boldsymbol{u}} = X^T \boldsymbol{u}$ is the GFT, having denoted with $X \in \mathbb{R}^{n \times n}$ the matrix with the eigenvectors of $L$ in its columns.

**The graph Fourier transform**

*The graph Fourier transform*

Consider an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $L$ the graph Laplacian matrix of Equation (4.4), and a signal $\boldsymbol{u} \in \mathbb{R}^n$ defined on the set $\mathcal{V}$. Let $L = X \Lambda X^T$ be the eigenvector decomposition of $L$, with $X \in \mathbb{R}^{n \times n}$ the matrix with the eigenvectors of $L$ in its columns. We define the GFT $\hat{\boldsymbol{u}}$ and its inverse as

$$\hat{\boldsymbol{u}} = X^T \boldsymbol{u} \tag{4.5}$$
$$\boldsymbol{u} = X \hat{\boldsymbol{u}}. \tag{4.6}$$

The expression of the inverse GFT easily comes from the property $XX^T = I_n$. Now that we introduced a definition for the GFT we want to see how it actually relates to some concept of frequency of the signal on the graph.

## 4.2.3 FREQUENCIES ON GRAPHS

The best way to understand the relation between the graph Laplacian and the concept of frequency is through this lemma that we will then prove.

**Lemma 4.1.** *Let $\boldsymbol{u} \in \mathbb{R}^n$ be a vector defined on the vertices of a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with graph Laplacian matrix L. then*

$$\boldsymbol{u}^T L \boldsymbol{u} = \frac{1}{2} \sum_{i,j \in \mathcal{V}} A_{ij} (u_i - u_j)^2. \tag{4.7}$$

*Proof.*

$$\begin{aligned} \boldsymbol{u}^T L \boldsymbol{u} &= \sum_{i,j \in \mathcal{V}} u_i L_{ij} u_j \\ &\overset{(a)}{=} \sum_{i,j \in V} u_i D_{ij} u_j - \sum_{i,j \in \mathcal{V}} u_i A_{ij} u_j \end{aligned}$$

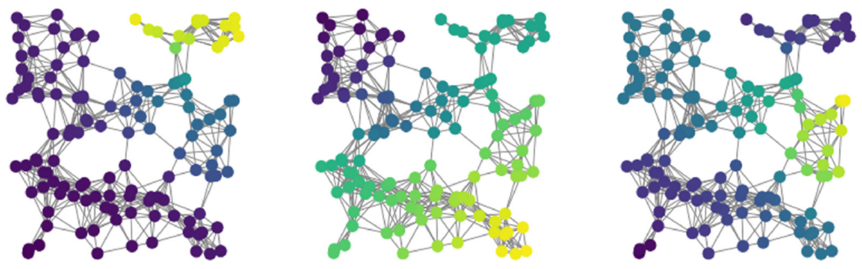


Figure 4.2: **Some graph Fourier modes on a graph**. From left to right: the eigenvector associated to the second, third and fourth smallest eigenvalue of $L$. Color-map: positive values in yellow, negative ones in blue. Picture taken from 10.1016/j.crhy.2019.08.003.

$$
\begin{aligned}
&= \sum_{i,j\in V} u_i d_i \delta_{ij} u_j - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&= \sum_{i\in V} u_i^2 d_i - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&\overset{(b)}{=} \sum_{i,j\in V} u_i^2 A_{ij} - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&\overset{(c)}{=} \frac{1}{2}\left(\sum_{i,j\in V} u_i^2 A_{ij} + \sum_{i,j\in V} u_j^2 A_{ji}\right) - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&\overset{(d)}{=} \frac{1}{2}\left(\sum_{i,j\in V} u_i^2 A_{ij} + \sum_{i,j\in V} u_j^2 A_{ij}\right) - \sum_{i,j\in\mathcal{V}} u_i A_{ij} u_j \\
&= \frac{1}{2}\sum_{i,j\in V} A_{ij}(u_i^2 + u_j^2 - 2u_i u_j) \\
&= \frac{1}{2}\sum_{i,j\in V} A_{ij}(u_i - u_j)^2,
\end{aligned}
$$

where in $(a)$ we used the definition of graph Laplacian $L = D - A$, in $(b)$ we rewrote the degree $d_i = \sum_{j\in\mathcal{V}} A_{ij}$, in $(c)$ we exploited the fact that $i, j$ are dummy variables that can be inverted and in $(d)$ we used $A_{ij} = A_{ji}$. □

The consequence of Lemma 4.1 is that the eigenvalues of $L$ can be interpreted as a “frequency” of the corresponding eigenvector, in the sense that they quantify how fast $\boldsymbol{x}_k$ changes on the graph.

$$
\lambda_k = \boldsymbol{x}_k^T L \boldsymbol{x}_k = \frac{1}{2}\sum_{i,j\in\mathcal{V}} (x_{k,i} - x_{k,j})^2,
$$

so if the eigenvector $\boldsymbol{x}_k$ changes *smoothly* on the graph, *i.e.* is has similar value for neighboring nodes, the corresponding eigenvalue will be small. Figure 4.2 shows in color code the frequency of the second, third and fourth eigenvectors of $L$ on a graph.

Using the GFT notation we can write, for a generic vector $\boldsymbol{u}$

$$\boldsymbol{u}^T L \boldsymbol{u} = \boldsymbol{u} \left( \sum_{k=1}^{n} \lambda_k \boldsymbol{x}_k \boldsymbol{x}_k^T \right) \boldsymbol{u} = \sum_{k=1}^{n} \lambda_k \hat{u}_k^2.$$

The term $\hat{u}_k$ is a weight that accounts for how much $\boldsymbol{u}$ is aligned with $\boldsymbol{x}_k$ and $\lambda_k$ is the corresponding frequency. The scalar $\boldsymbol{u}^T L \boldsymbol{u}$ can hence be used to quantify how fast the signal $\boldsymbol{u}$ changes on the graph.

To summarize, the eigenvectors of the graph Laplacian matrix are the Fourier eigenmodes and by projecting a signal over them we are in the Fourier space of frequencies that are the eigenvalues of the same matrix. We now detail some basic but relevant property of the eigenvalues of the graph Laplacian matrix that is fundamental to understand the GFT.

### 4.2.4 SOME BASIC SPECTRAL PROPERTIES OF THE GRAPH LAPLACIAN

We here list and prove three basic facts about the graph Laplacian eigenvalues. We first formally state that $L$ does not have any negative eigenvalues.

**Corollary 4.1.** *The graph Laplacian matrix L is positive semi-definite,* i.e. *it does not have negative eigenvalues. The all one vector $\mathbf{1}_n$ is an eigenvector of L with eigenvalue* 0*.*

*Proof.* This is a corollary to Lemma 4.1. Let $\boldsymbol{x}_k$ be an eigenvector of $L$ with eigenvalue $\lambda_k$, then

$$\lambda_k = \boldsymbol{x}_k^T L \boldsymbol{x}_k = \frac{1}{2} \sum_{i,j \in \mathcal{V}} (x_{k,i} - x_{k,j})^2 \geq 0.$$

The equality is reached for $\boldsymbol{x}_1 = \mathbf{1}_n$ that is an eigenvector because $D\mathbf{1}_n = \boldsymbol{d}$ and $A\mathbf{1}_n = \boldsymbol{d}$, where $\boldsymbol{d}$ denotes the degree vector. □

The second property concerns the multiplicity of the 0 eigenvalue and its relation to the connectedness of the graph.

**Corollary 4.2.** *The multiplicity of the* 0 *eigenvalue of graph Laplacian matrix L equals the number of connected components of the graph.*

*Proof.* Also in this case the proof is a straightforward consequence of Lemma 4.1. Consider a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $c$ connected components $\{\mathcal{V}_a\}_{a=1,\ldots,c}$ so that

$$\cup_{a=1,\ldots,c} \mathcal{V}_a = \mathcal{V}$$
$$\forall\, a \neq b \quad \mathcal{V}_a \cap \mathcal{V}_b = \emptyset.$$

Denote with $\boldsymbol{y}^{(a)}$ the vector with entries $y_i^{(a)} = 1$ if $i \in \mathcal{V}_a$ and 0 otherwise. Then

$$\begin{aligned}
(L\boldsymbol{y}^{(a)})_i &= \sum_{j\in\mathcal{V}} (D-A)_{ij} y_j^{(a)} \\
&= \sum_{b=1}^{c} \sum_{j\in\mathcal{V}_b} (D-A)_{ij} y_j^{(a)} \\
&\overset{(a)}{=} \sum_{j\in\mathcal{V}_a} (D-A)_{ij} \\
&\overset{(b)}{=} y_i^{(a)} \cdot (d_i - d_i) = 0,
\end{aligned}$$

where in $(a)$ we used the fact that $y_j^{(a)} = 0$ for all $j \notin \mathcal{V}_a$, while in $(b)$ that all the connections each node has are within the same connected component. Given their definition, the indicator vectors are orthogonal, *i.e.* $(\boldsymbol{y}^{(a)})^T \boldsymbol{y}^{(b)} = \delta_{ab}$ and thus are different eigenvectors of $L$, concluding the proof. □

Finally, we provide a result on the *largest* eigenvalue of $L$ that will be useful for the next sections.

**Lemma 4.2.** *Letting $d_{\max}$ be the largest degree of a graph, The spectral radius of L satisfies the following inequality.*

$$\rho(L) \leq 2d_{\max}.$$

*Proof.* Let $\boldsymbol{x}$ be the eigenvector associated with the largest eigenvalue of $L$, so that $L\boldsymbol{x} = \rho(L)\boldsymbol{x}$. Consider the index $i$ satisfying $|x_i| \geq |x_j|$ for all $j$. Then we can write

$$\begin{aligned}
|\rho(L)x_i| &= |(L\boldsymbol{x})_i| \\
&= \left| \sum_{j\in\mathcal{V}} L_{ij} x_j \right| \\
&\overset{(a)}{\leq} \sum_{j\in\mathcal{V}} |L_{ij}||x_j| \\
&\overset{(b)}{\leq} |x_i| \sum_{j\in\mathcal{V}} |L_{ij}| \\
&= |x_i| 2d_i \\
&\leq |x_i| 2d_{\max},
\end{aligned}$$

where in $(a)$ we used the triangle inequality and in $(b)$ we exploited the property of the index $i$. □

As one can see from the proof, this is not a very tight bound and better results exist. Yet, this is a simple bound we can find without the need to explicitly compute the largest eigenvalue. We now proceed in our discussion with some more practical applications of the GFT to graph signal processing.

## 4.3 GRAPH SIGNAL PROCESSING

### 4.3.1 TIKHONOV REGULARIZATION

We formulate here a semi-supervised learning problem of on a graph that we solve exploiting the concept of GFT. We suppose that there is a signal $\boldsymbol{u} \in \mathbb{R}^n$ defined on the nodes of a graph but only some entries of this signal are known. In particular $\mathcal{Q}$ is the set of measured nodes meaning that we know $u_i$ for all $i \in \mathcal{Q}$ and our problem is to guess $u_i$ for all $i \in \mathcal{V} \setminus \mathcal{Q}$. We attempt to reconstruct the signal on the whole graph by identifying a vector $\boldsymbol{v} \in \mathbb{R}^n$ that is at the same time close to $\boldsymbol{u}$ for all $i \in \mathcal{Q}$ and that is smooth on the graph.[1] For all $i \notin \mathcal{Q}$ we hence make a sort of interpolation with the known signal values. We let $\tilde{\boldsymbol{v}}, \tilde{\boldsymbol{u}} \in \mathbb{R}^{|\mathcal{Q}|}$ be two vectors defined only on the set of labeled vertices of $\mathcal{G}(\mathcal{V},\mathcal{E})$and we define the following loss function

$$\mathcal{L}(\boldsymbol{v},\boldsymbol{u}) = \|\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}\|^2 + \gamma \boldsymbol{v}^T L \boldsymbol{v}.$$

The first term simply imposes that the loss is minimized when $\boldsymbol{v}$ is equal to $\boldsymbol{u}$ for all $i \in \mathcal{V}$. The second term instead represents instead the "frequency" of $\boldsymbol{v}$ that we want to minimize to ensure the signal changes smoothly over the graph. The factor $\gamma$ is a weight that balances these two terms: large $\gamma$ will enforce high regularization, while small $\gamma$ will tend to force a matching result on the labeled nodes. To explicitly formulate the optimization problem, we let $\mathcal{Q} \in \mathbb{R}^{n\times n}$ be a matrix defined as follows

$$Q_{ij} = \delta_{ij}\mathbb{1}_{i\in\mathcal{Q}}. \tag{4.8}$$

The reconstructed signal is then $\boldsymbol{u}^*$, the solution to Tikhonov regularization.

*Tikhonov regularization*

**Tikhonov regularization**

Consider a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ with Laplacian matrix $L$; a matrix $Q$ as per Equation 4.8; a signal $\boldsymbol{u} \in \mathbb{R}^n$ defined on $\mathcal{V}$; and a positive scalar $\gamma$. We define $\boldsymbol{u}^*$ the solution of Tikhonov regularization as

$$\boldsymbol{u}^* = \underset{\boldsymbol{v}\in\mathbb{R}^n}{\arg\min}\ (\boldsymbol{u}-\boldsymbol{v})^T Q(\boldsymbol{u}-\boldsymbol{v}) + \gamma \boldsymbol{v}^T L \boldsymbol{v}. \tag{4.9}$$

1 For instance, the values $u_i$ can be the temperatures measured by weather stations at different locations. If we want to have a guess of the temperature in places where no station is available, we may solve this problem on a spatial graph in which each node corresponds to a point on the Earth and edges connect nodes with a distance below a given threshold.

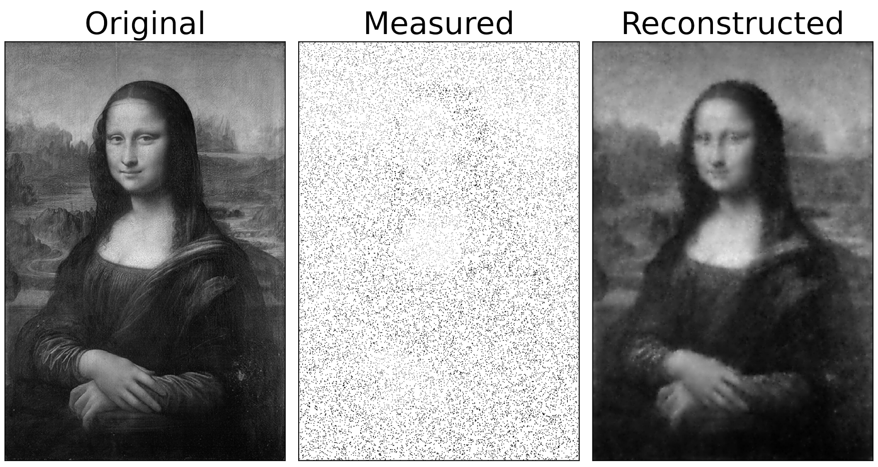


Figure 4.3: **Tikhonov regularization for image reconstruction**. Left: the original black and white image. Center: the measured image in which only 10% of the pixels are known. Right: The reconstructed image using Tikhonov regularization for $\gamma = 0.1$.

The optimization of Equation (4.9) can be solved analytically, in fact,

$$\nabla \boldsymbol{v} \mathcal{L}(\boldsymbol{u}, \boldsymbol{v}) = -2Q(\boldsymbol{u} - \boldsymbol{v}) + 2\gamma L \boldsymbol{v}.$$

Setting $\nabla_{\boldsymbol{v}} = 0$, we get[2]

$$\boldsymbol{u}^* = (Q + \gamma L)^{-1} Q \boldsymbol{u}.$$

In Figure 4.3 we show an example of application of this algorithm to reconstruct an image given that only some pixels are known. We naturally define a grid graph that connects each pixel to its neighbors and show the input image against the reconstructed one. This example clearly shows the power of Tikhonov regularization.

## 4.3.2 FILTERING

We now address a closely related problem to the one above, that is filtering. Suppose that the signal $\boldsymbol{u} \in \mathbb{R}^n$ is observed on all the graph vertices but that there is some noise. Our goal is to obtain a cleaner version of the signal, removing the noise by exploiting the fact that the signal changes smoothly over the graph, while noise does not. Equation (4.9) can still be used to achieve this task letting $Q = I_n$. Let us look more closely at the solution of the smoothed vector $\boldsymbol{u}^*$

$$\begin{aligned} \boldsymbol{u}^* &= (I + \gamma L)^{-1} \boldsymbol{u} \\ &= \sum_{k=1}^{n} \frac{1}{1 + \gamma \lambda_k} \boldsymbol{x}_k \boldsymbol{x}_k^T \boldsymbol{u} \end{aligned}$$

2 Note that inverting the matrix $Q + \gamma L$ is not computationally efficient nor necessary and the problem can be solved in a faster way finding the solution to $(Q + \gamma L)\boldsymbol{u}^* = Q\boldsymbol{u}$ with an appropriate solver.

$$= \sum_{k=1}^{n} \underbrace{\boldsymbol{x}_k}_{\text{anti}-\text{transform}} \underbrace{\frac{1}{1+\gamma\lambda_k}}_{\text{filter in frequency demain}} \underbrace{\hat{u}_k}_{\text{Fourier transform}} .$$

The order in which we should see the operations is from right to left. First we project the signal $\boldsymbol{u}$ on the Laplacian eigenvectors, thus moving to the Fourier space. Here we re-weight every mode with the function $f(x) = (1+\gamma x)^{-1}$. This is a low-pass filter because the smallest eigenvalue $\lambda_1 = 0$ gets a weight equal to one and it corresponds to the eigen-mode with smallest frequency. As we consider larger values of $k$ (hence larger frequencies), $f(\lambda_k)$ decreases, thus filtering out the contribution of high frequency states. Finally, we make the anti-transform and get back to the original space.

Now that we have introduced the concept of filtering, we can design any filter that acts in the frequency domain so to get a good result. Considering a general filtering function $f$, we can write

*Filtering in the Fourier space*

$$\boldsymbol{u}^* = \sum_{k=1}^{n} \boldsymbol{x}_k f(\lambda_k)\hat{u}_k. \tag{4.10}$$

A relevant problem is that to solve exactly this problem, we must compute *all* the eigenvalues of $L$ which requires $\mathcal{O}(n^3)$ operations and is unfeasible for large networks. If we use a polynomial filter (or the polynomial approximation of the filtering function), however, we can greatly simplify things. Suppose that

*Polynomial filter*

$$f(x) = \sum_{a=0}^{p} \alpha_a x^a,$$

then we can rewrite Equation (4.10) as

$$\begin{aligned}\boldsymbol{u}^* &= \sum_{k=1}^{n}\sum_{a=1}^{p} \boldsymbol{x}_k a_p \lambda_k^p \hat{u}_k \\ &\overset{(a)}{=} \sum_{k=1}^{n}\sum_{a=1}^{p} a_p L^p \boldsymbol{x}_k \hat{u}_k \\ &\overset{(b)}{=} \sum_{k=1}^{n}\sum_{a=1}^{p} a_p L^p \boldsymbol{x}_k \boldsymbol{x}_k^T \boldsymbol{u} \\ &\overset{(c)}{=} \sum_{a=1}^{p} a_p L^p \boldsymbol{u},\end{aligned}$$

where in $(a)$ we exploited the fact that $\boldsymbol{x}_k$ is an eigenvector of $L^p$ with eigenvalue $\lambda_k^p$ and in $(b)$ we explicitly rewrote $\hat{u}_k = \boldsymbol{x}_k^T \boldsymbol{u}$ and in $(c)$ we used $I_n = \sum_{k=1}^{n} \boldsymbol{x}_k \boldsymbol{x}_k^T$. From the last equation we can see that the smoothed function can be computed using a polynomial of $L$, without the need of diagonalizing the matrix. One can still argue, however, that for large $p$, the matrix $L^p$ is quite dense and thus the computational complexity to perform this operation is still very high. What has to be noticed is that we do not need to compute $L^p$ but actually only $L^p\boldsymbol{x}$ and this can be done efficiently.

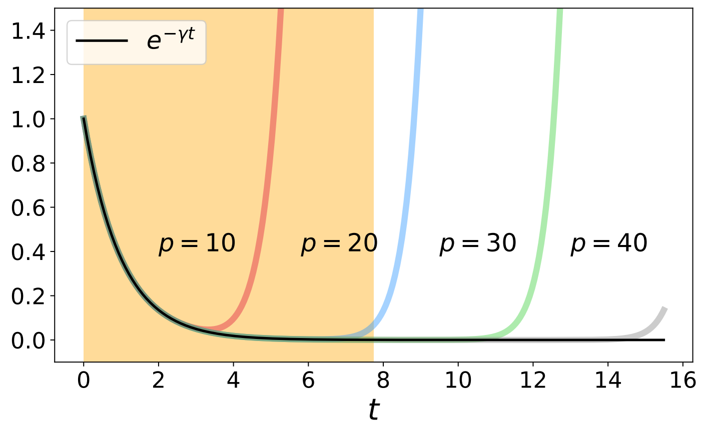


Figure 4.4: **Polinomial approximation of the exponential decay**. The exponential decay function is shown in black for $\gamma = 1$, while in color we have the $p$ order polynomial approximation for different values of $p$. The orange background indicates the values of $t$ for which the approximation must not diverge and it spans from 0 to $\gamma\rho(L)$.

In fact, let $y_p = L^p \boldsymbol{x}$, then $L^{p+1}\boldsymbol{x} = L\boldsymbol{y}_p$. This is the product between a (presumably sparse) matrix and e vector. Iterating this for all $p$, we see that the calculation can be performed very efficiently.

We now conclude with an example. We consider the NYC taxi dataset containing several information about taxi drives, including the region (of NYC) of origin and destination and fare of the ride. These data can be found here. The city is divided in approximately 300 regions and we compute the average fare of the destination for each origin. This signal signal is strongly auto-correlated, meaning that nearby regions have similar average fares to be paid. We now introduce some noise in this dataset, taking a random sample including 25% of the total number of regions and randomly reassigning them the average fare amount. Our goal is to reconstruct the original signal by filtering out the high frequency components introduced by noise and exploiting geometrical proximity to smooth the signal.

From the dataset we can extract the coordinates of the centroid of each region and use that to build a graph. We generate it using a fast $k$ nearest neighbors algorithm that connects each node $i$ to its $k$ closest nodes. Note that the matrix we get in this way is not symmetric: Palermo and Roma are (probably) the closest provinces to Sassari, but the opposite is not true. We then must first symmetrize the adjacency matrix to get the correct representation of our graph. We then design an exponential filter $e^{-\gamma x}$, exploiting its polynomial approximation, *i.e.*

$$f(x) = \sum_{a=1}^{p} \frac{(-\gamma x)^a}{a!},$$

for some $p$. As we know, the modulus of a polynomial function tends to infinity for $x \to \infty$ and is thus not a good approximation of the exponential function. Yet, the argument of the function is bounded by $\gamma\rho(L)$ and

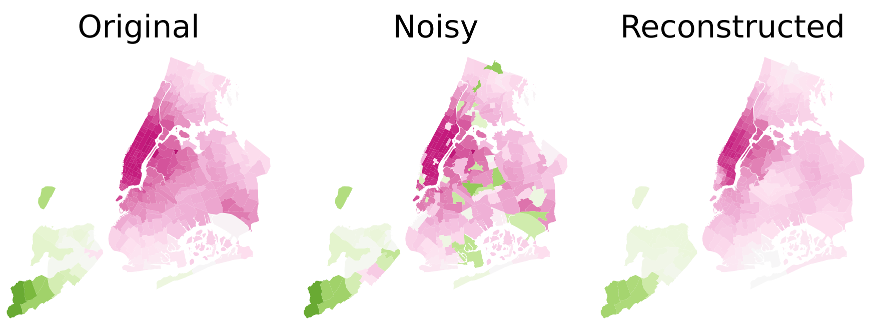


Figure 4.5: **Signal filtering of taxi fares in New York**. Left panel: original signal of the average fare of the region of destination. Center panel: noisy version of the signal. Right panel: smoothed signal using the exponential filter and $p = 30$.

we must choose the order of the polynomial $p$ so that is well approximates the exponential function for all $x \in [0, \gamma\rho(L)]$. Note that, from 4.2 we do not need to explicitly compute $\rho(L)$ as we have a bound as per Lemma 4.2. Figure 4.4 shows the function $f(x)$ for different values of $p$: the colored background denotes the region in which the filter must well approximate the exponential, hence $p = 30$ is a reasonable choice to proceed. Figure 4.5 shows in color code the result of this experiment.

## 4.4 CONCLUSION

In this chapter we introduced the concept of Fourier transform on graphs and showcased to example os application to signal reconstruction and denoising. The GFT is an important tool because it allows one to introduce the concept of "frequency" of a signal on a graph *i.e.*, of how fast it changes of the neighbors. Even if we did not discuss it, the relevance of the GFT goes well beyond these two examples and is at the basis of the convolutional graph neural networks. Note that the convolution of two function $f, g$ can be written exploiting their Fourier transforms $\hat{f}, \hat{g}$ as

$$(f * g)(x) = \frac{1}{2\pi} \int_{\mathbb{R}} dk\, \hat{f}(k)\hat{g}(k)e^{ikx},$$

*i.e.* the Fourier transform of the convolution of two functions if the product of their Fourier transforms. Consequently, this allows one to define the convolution on a graph exploiting the definition of GFT and, in fact, the graph Laplacian matrix is a fundamental building block of convolutional neural networks. As a final remark, we would like to stress that, even though we only mentioned the relation between the matrix $L$ and the GFT, other matrices can similarly be used to define the concept of Fourier transform on graphs. One of the most commonly adopted ones is the *normalized Laplacian*

$$L_{\rm n} = I_n - D^{-1/2}AD^{-1/2}. \tag{4.11}$$

## 4.5 REFERENCES

- Ricaud, Borgnat, Tremblay, Gonçalves, Vandergheynst: *Fourier could be a data scientist: From graph Fourier transform to signal processing on graphs*
  This is a quite pedagogical review on GFT.
- Tremblay, Gonçalves, Borgnat: *Design of graph filters and filterbanks.*
  This is a more advanced reference with the focus on graph filters.

# 5 COMMUNITY DETECTION

5.1 Community detection . . . . . . . . . . . . . . . . . . . . . . . 51
5.1.1 Defining communities . . . . . . . . . . . . . . . . . . 52
5.1.2 The number of communities . . . . . . . . . . . . . . 53
5.1.3 Comparing partitions . . . . . . . . . . . . . . . . . . 53
5.1.4 Computational complexity . . . . . . . . . . . . . . . 54
5.2 Optimization approaches . . . . . . . . . . . . . . . . . . . 55
5.2.1 Some definitions . . . . . . . . . . . . . . . . . . . . . 55
5.2.2 Louvain algorithm . . . . . . . . . . . . . . . . . . . . 56
5.2.3 Pitfalls of optimization approaches . . . . . . . . . . 58
5.3 Inference in the DCSBM . . . . . . . . . . . . . . . . . . . 59
5.3.1 The degree corrected stochastic block model . . . . . 60
5.3.2 Information theoretic limits in the DCSBM . . . . . . 60
5.3.3 DCSBM Bayesian inference . . . . . . . . . . . . . . . 61
5.3.4 Optimization vs model based: a statistical physics perspective . . . . . . . . . . . . . . . . . . . . . . . . 62
5.4 Spectral clustering . . . . . . . . . . . . . . . . . . . . . . . 64
5.4.1 Vanilla spectral clustering . . . . . . . . . . . . . . . 65
5.4.2 The relation with optimization approaches . . . . . . 66
5.4.3 A random matrix perspective . . . . . . . . . . . . . 67
5.4.4 Spectral clustering in sparse graphs . . . . . . . . . . 69
5.4.5 Final remarks on spectral clustering . . . . . . . . . 75
5.5 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
5.6 References . . . . . . . . . . . . . . . . . . . . . . . . . . . 76

## 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

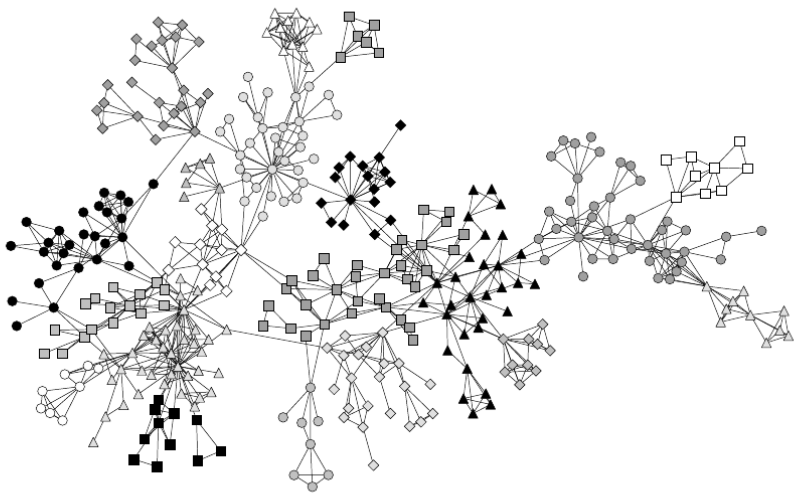


Figure 5.1: **A graph with communities**. Each community is marker coded. Picture taken from *Newman, Netwoks*.

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

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

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.

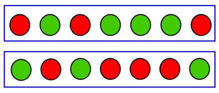
## 5.1.2 THE NUMBER OF COMMUNITIES

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: $\mathrm{AMI} = 1$ if the partitions are equivalent; $\mathrm{AMI} = 0$ 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.

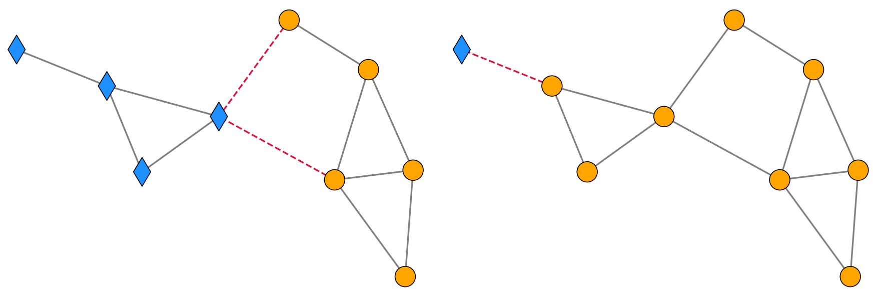


Figure 5.2: **Graph cut evaluation**. Two partitions of the same graph: the one on the left has a graph cut equal to 2, while the one on th right has a graph cut equal to 1.

## 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*

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

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 $\mathscr{Q}_A^{\mathrm{Cut}}$, 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).

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

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*

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

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

*Modularity*

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

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 $\mathcal{Q}_A^{\text{Mod}}(\boldsymbol{\ell})$. 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:

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

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

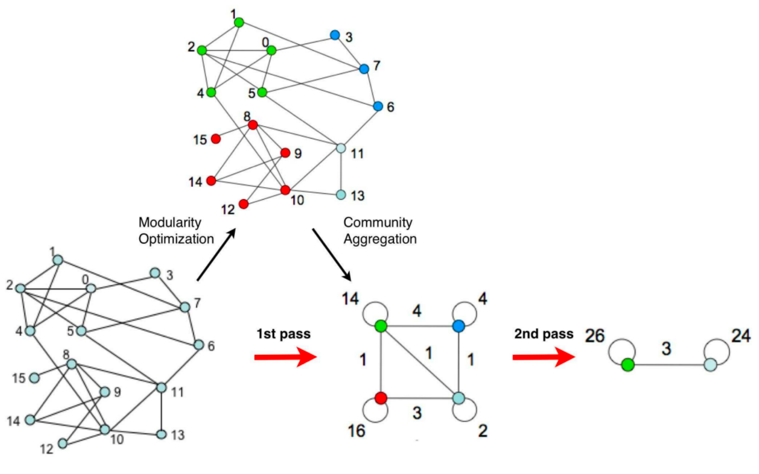


Figure 5.3: **Schematic representation of Louvain algorithm**. Each pass is made of two phases: one where modularity is optimized by allowing only local changes of communities; one where the found communities are aggregated in order to build a new network of communities. The passes are repeated iteratively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

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

*Rewriting the modularity in a simpler form*

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.

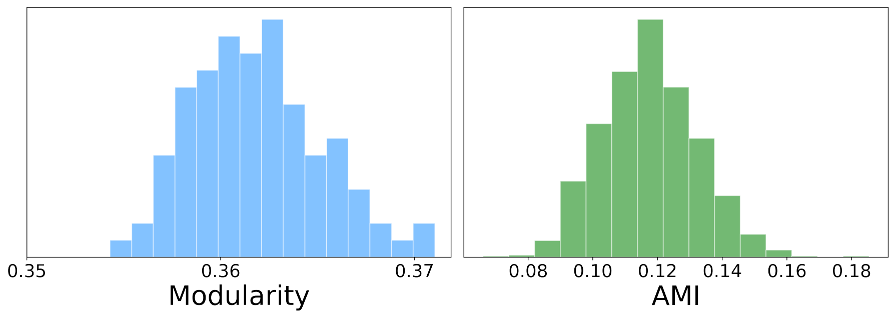


Figure 5.4: **Evaluation of Louvain algorithm**. The Louvain algorithm was run 100 times on a random graph with communities. The left plot reports the histogram of the modularity obtained after each run. In the right plot we selected 1000 random pairs and computed for all of them the AMI and plotted 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 $\Delta\mathscr{Q}_A^{\rm Mod}(\boldsymbol{\ell})$ 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.

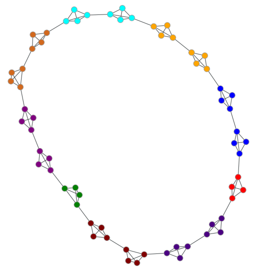
- Optimization problems such as NCut, RCut and the modularity maximization are NP hard and only approximate solutions can be obtained by efficient algorithms. For this reason, we must inderline that the output of the Louvain algorithm (or any other modularity maximization technique) is **not** guaranteed 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 $\gamma$:

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

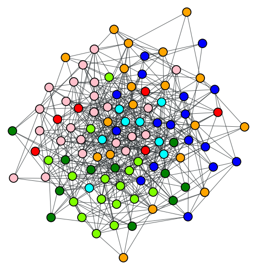
  Tuning the value $\gamma$ it is possible to identify communities at different length scales, but this requires an *ad hoc* solution, depending, in general, on the underlying graph.

*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

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

*The Bayes formula*

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 $\boldsymbol{\ell} \in \{1, \ldots, k\}^n$ 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 $\nu$, satisfying $\mathbb{E}[\theta] = 1, \mathbb{E}[\theta^2] = \Phi = O_n(1)$. For each node, $\theta_i$ is drawn independently at random from $\nu$.*

*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 $\theta_i$, *i.e.* $\mathbb{E}[d_i] \propto \theta_i$. Consequently the vector $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$ 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 $\pi_a$ 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.

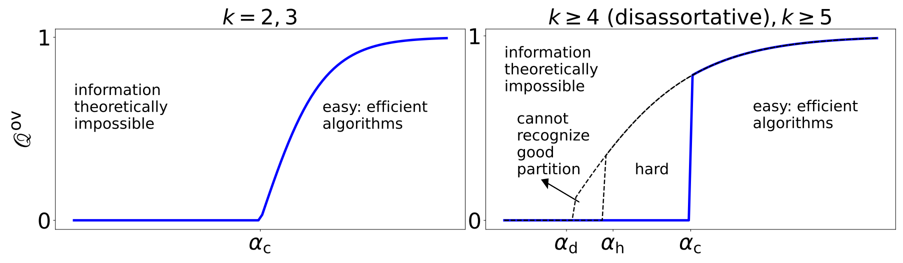


Figure 5.5: Schematic representation of the phase transition in the sparse DCSBM. The $y$ axis represents the performance of reconstruction. Picture adapted from *Moore, The Computer Science and Physics of Community Detection.*

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 $c_{\text{in}}$, while the off-diagonal are $c_{\text{out}}$ 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 $c_{\text{in}}$ and the off-diagonal ones be $c_{\text{out}}$ and denote with $c = (c_{\text{in}} + c_{\text{out}})/2$ the expected average degree. Let the control parameter $\alpha$ 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

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

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

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

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

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

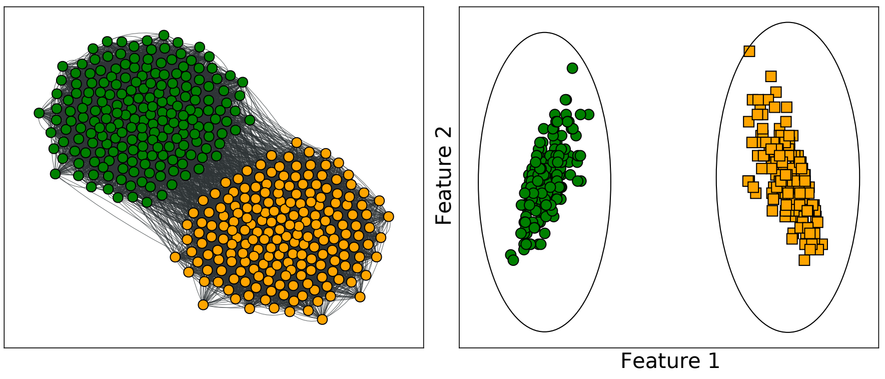


Figure 5.6: **Left**: a graph with two communities, highlighted with a different color code. **Right**: a 2 dimensional node embedding of the graph on the left. The embedding dimensions are here denoted with as *features*.

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 $\mathscr{Q}_A^\bullet$, 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 : $\ell \in \{1, \ldots, k\}^n$ 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*

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

*Proof.* Exploiting Lemma 4.1, we can write

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

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*

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

□

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 $H^T H = I_k$, then the relaxation of the *RatioCut* optimization problem is obtained by solving

*Relaxed optimization*

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

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 $L^{\mathrm{rw}}$ the matrix $L^{\mathrm{sym}} = I_n - D^{-1/2}AD^{-1/2}$ is preferred. This matrix has the same eigenvalues of $L^{\mathrm{rw}}$ 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 $c_{\mathrm{out}}$ 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 $\sigma_i = -1$

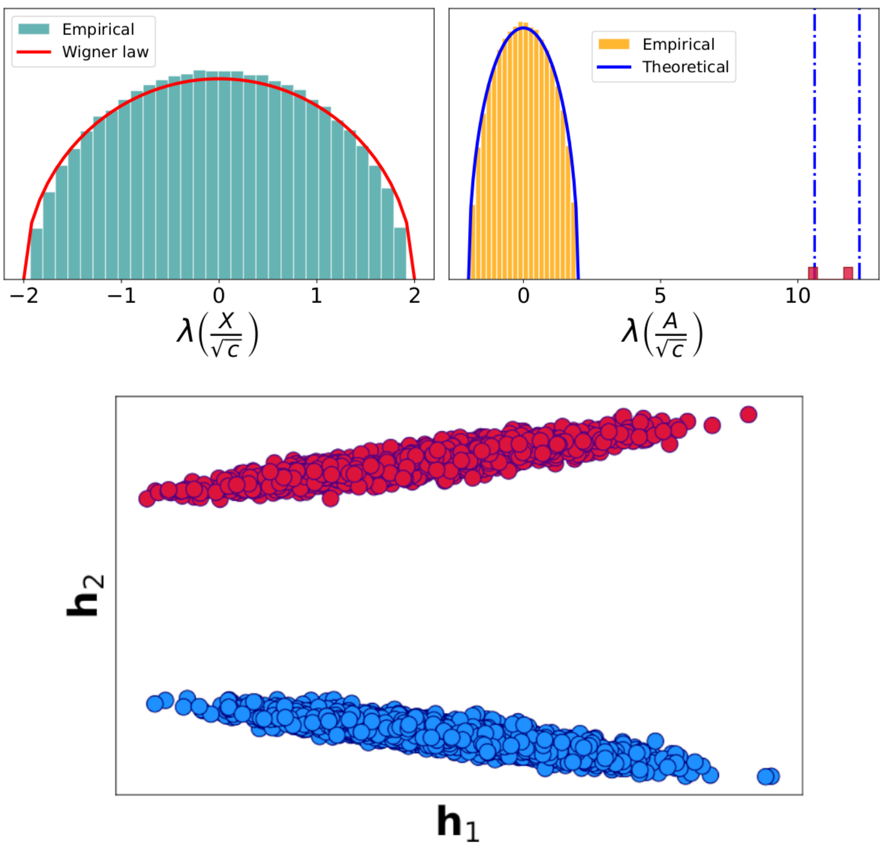


Figure 5.7: **Visualization of spectral properties of the adjacency matrix**. *Top left panel*: spectrum of the centered adjacency matrix $X$. *Top right panel*: spectrum of the adjacency matrix $A$ with a zoom inset on the largest isolated eigenvalues. *Bottom panel*: two dimensional embedding obtained 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 $-2\alpha$ and $2\alpha$, 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 $\mu$ be an eigenvalue of $\tilde{A} + X$, then there exists an eigenvalue $\lambda$ 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 $\sigma$. 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

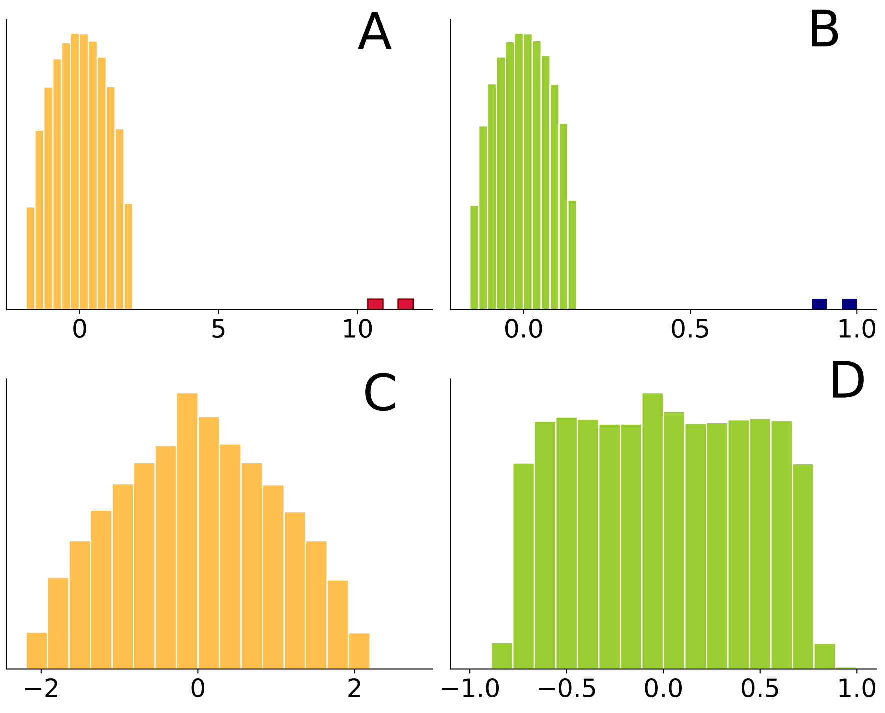


Figure 5.8: **Spectral behavior of graph matrix representations in the sparse and dense regimes**. We consider two graphs with two communities with a large (panels *A* and *B*) and a small (panels *C* and *D*) average degree. In the panels *A* and *C* we show the spectrum of the adjacency matrix $A$ 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

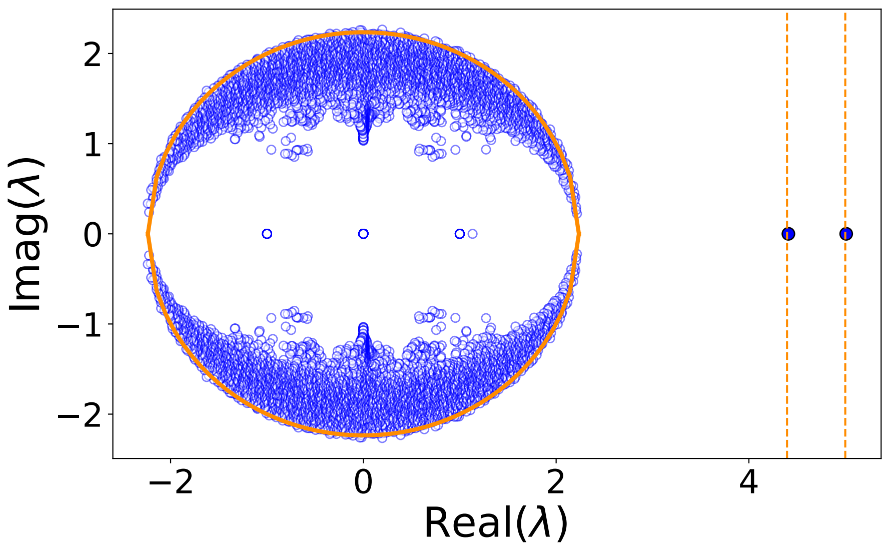


Figure 5.9: **Non-backtracking matrix spectrum in the complex plane**. Scatter plot of the real vs imaginary part of the eigenvalues of $B$ for a graph obained from the DCSBM model with $k = 2$ communities in the sparse regime. The blue dots are the eigenvalues of the matrix, while the orange lines are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

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

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, $\lambda_1(C\Pi) = c = \frac{c_{\rm in}+c_{\rm out}}{2}$ and $\lambda_2(C\Pi) = c = \frac{c_{\rm in}-c_{\rm out}}{2}$. The eigenvalue $\lambda_2(B)$ is isolated if

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

that implies

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

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 $\boldsymbol{g}^{\rm in} = T\boldsymbol{g}$. By construction the vector $\boldsymbol{g}^{\rm in} \in \mathbb{R}^n$ and it can be used to define a SC algorithm with the eigenvectors of $B$. Moreover, still referring to Chapter 2, we recall that $\boldsymbol{g}^{\rm in}$ can be extracted by the first $n$ entries of the matrix $B_{\rm p}$ defined in Equation (3.11) that has size $2n \times 2n$ and it can thus be efficiently computed on large graphs.

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

## *The Bethe-Hessian matrix*

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

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

that leads to

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

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.

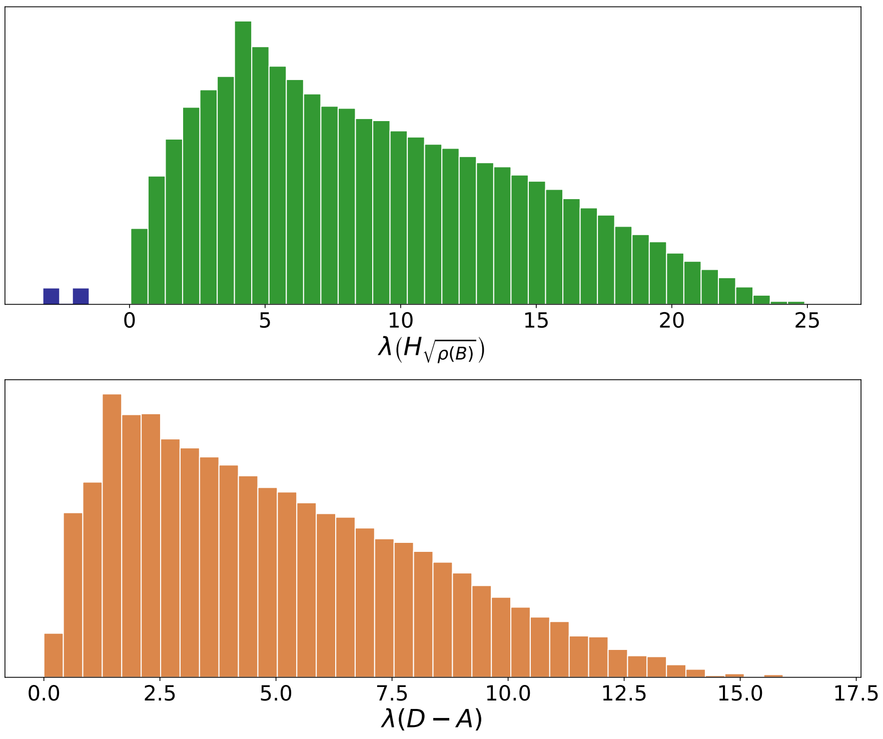
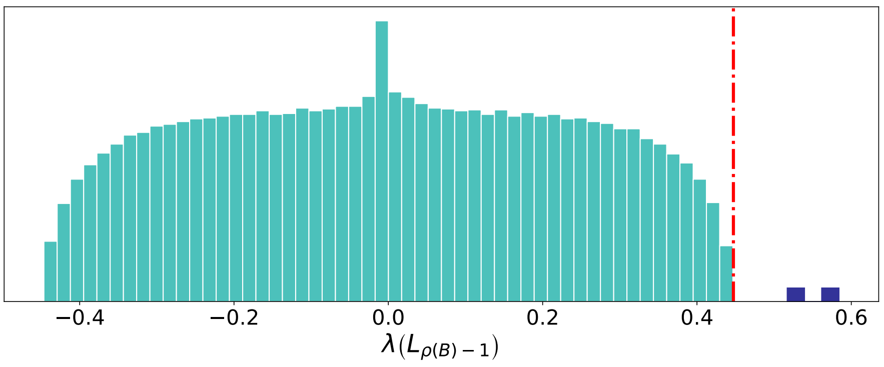


Figure 5.10: **Spectra of the Bethe-Hessian and Laplacian matrices on a sparse graph with communities**. The top row shows the spectrum of $H_{\sqrt{B}}$ for a graph with $k = 2$ communities generated from the DCSBM and expected average degree equal to 5. In blue we evidence the two isolated *negative*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

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

The matrix $H_\gamma$ is the *Bethe-Hessian* matrix and $\boldsymbol{g}^{\mathrm{in}}$ is an eigenvector of $H_\gamma$ is $\gamma$ 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 $H_\gamma$ and it has been shown that for some choices of $\gamma$, the algorithmic threshold of a SC algorithm based on $H_\gamma$ 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 $H_\gamma$ 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.



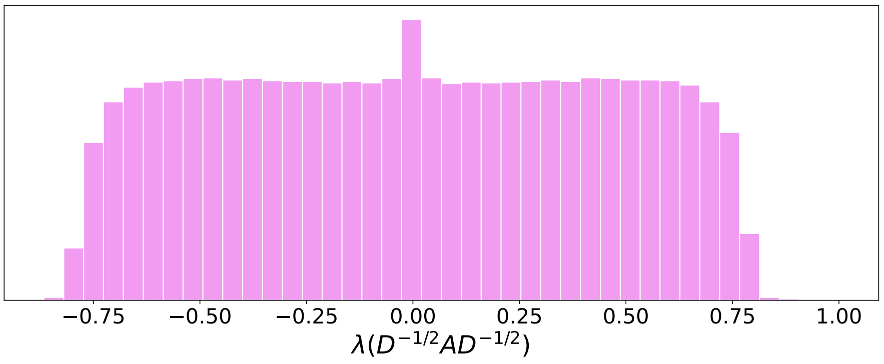


Figure 5.11: **Spectra of the regularized and non-regularized symmetric Laplacian matrices on a sparse graph with communities**. The top row shows the spectrum of $L^{\mathrm{sym}}_{\sqrt{B}}$ for a graph with $k = 2$ communities generated from the DCSBM and expected average degree equal to 5. In blue 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 $L^{\mathrm{sym}} = D^{-1/2}AD^{-1/2} = L^{\mathrm{sym}}_1$.

## *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

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

where we introduce the normalized Laplacian matrix $L^{\mathrm{sym}}_\gamma = D^{-1/2}_{\gamma^2-1} A D^{-1/2}_{\gamma^2-1}$. 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 $L^{\text{sym}}_{1}$ 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.

# 6 GRAPH EMBEDDINGS

6.1 Graph embeddings . . . 77
6.2 Word2Vec . . . 78
6.2.1 Skip-Gram . . . 78
6.2.2 Definition of a loss function . . . 79
6.2.3 Training the model parameters . . . 79
6.3 Node2Vec . . . 81
6.4 Conclusion . . . 82
6.5 References . . . 83

## 6.1 GRAPH EMBEDDINGS

As we have seen all along this course, graphs are a very powerful data representation tool, capable of properly modeling complex interaction patterns among the items of a dataset. However, this complexity makes any operation on graphs intimately hard to even define. Think for instance for instance of the problem of defining the distance between two nodes, or two graphs or a concept of continuity on graph: all these problem have been approached and have solutions, but they are not unique and they need to be defined. This is because graphs live in a high dimensional non-Euclidean space in which most mathematical operations are not easily defined. Consequently, a powerful method to deal with graph is to project them into an Euclidean space. This operation is called *embedding* and *Spectral clustering* (SC) actually exploits an embedding that leverages on an appropriate graph matrix representation. Now, there is not a unique way to embed graphs and the embedding itself should be defined so that it preserves some relevant graph properties. Nonetheless, once it is defined, we move to a space in which several mathematical operations and algorithms (such as clustering) can be deployed. We can formally define a node embedding as follows

*Embeddings allow one to more simply represent graphs*

Node embeddings are at the basis of graph neural networks and allow one to provide meaningful representations of complex objects. In the remainder we describe the `Node2Vec` algorithm, that is one of the most popular algorithms to obtain non-linear node embeddings. Note that this embedding method can also be used to perform *Community detection* (CD) as well, by performing clustering on the embedded space, equivalently to SC.

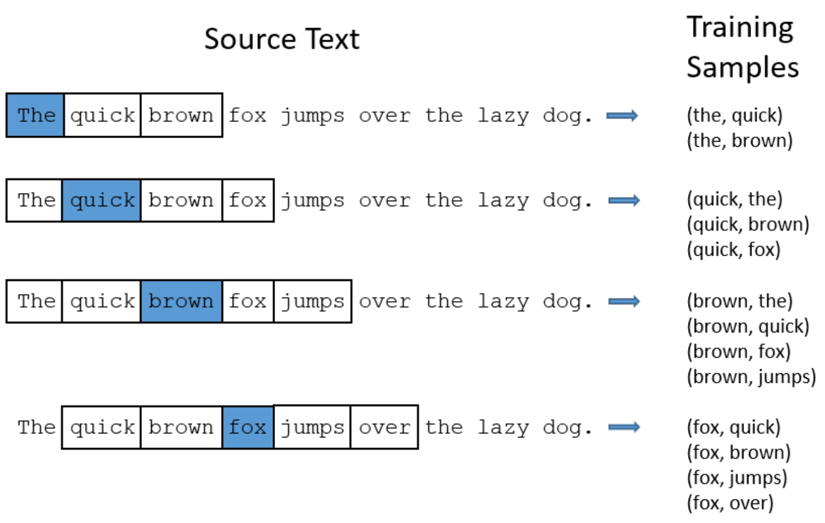


Figure 6.1: **Visual representation of the Skip-Gram algorithm**. The central word is highlighted in blue, while the context words are surrounded by white boxes. In this example the window size is set equal to 2. On the right we have all the pairs (central, context) that are obtained scanning through the document that constitute the output of the Skip-Gram algorithm. Picture taken from mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/.

## 6.2 WORD2VEC

The `Node2Vec` algorithm builds on another (even more popular) algorithm, called `Word2Vec`. This algorithm was introduced to embed the words appearing in a text and capturing the semantic similarity between them. For instance, if $i =$ milk, $j =$ cow, $k =$ gun, for the corresponding embedding vectors, we expect that $\boldsymbol{x}_i^T\boldsymbol{x}_j$ is large (milk is related to cow), while $\boldsymbol{x}_i^T\boldsymbol{x}_k$ is small. Let us now detail the method to obtain this desiderata.

### 6.2.1 SKIP-GRAM

*Defining context words*

If our goal is to provide similar representations to words that appear in similar contexts, we must first define what contexts are and we must do so in a simple, content-agnostic way. To do so, the skip-gram algorithm takes an arbitrary word of the text and considers the surrounding ones (within a certain distance, called *window size*) as the context of that word. To give an increasing weight to the words that are closer to the central one, one may draw for each word the window size from a uniform distribution between 1 and a maximal value. The Skip-Gram algorithm then takes a text as input and a value of the (maximal) window size and outputs a list of pairs $(\mathrm{center}, \mathrm{context})$ that relate every words appearing in the text with the surrounding ones. Figure 6.1 depicts the Skip-Gram procedure. If two words $a, b$ are closely related – such as *gold* and *crown* –, one expects the pair $(a, b)$ to appear several times. However, thinking of the case of synonyms, one can expect them to rarely appear in the same context, even if they have the same meaning. The Skip-Gram algorithm, however, can properly deal also with this type of similarity because synonyms will be surrounded by similar context words, thus allowing one to recover their similarity.

### 6.2.2 DEFINITION OF A LOSS FUNCTION

Now that we have identified context words, we want to define a loss function of the embedding vectors that promoted the alignment of for the pairs (central, context). Let $i$ be an index running over all words in the text and let $\pi(i)$ be a function mapping word $i$ to its position in the dictionary. Then, letting $\mathcal{C}_i$ be the context of word $i$, we write

$$\mathcal{L} = -\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{C}_i} \log\sigma\left(\boldsymbol{x}_{\pi(i)}^T\boldsymbol{x}_{\pi(j)}\right),$$

where $\mathcal{T}$ is the set of words appearing in the text, $\sigma(\cdot)$ is the sigmoid function[1] and $\boldsymbol{x}_a \in \mathbb{R}^d$ is the embedding of the word $a$. Now, minimizing this loss function we promote the alignment between central and context words. This loss function actually has a trivial minimum that is obtained for $\boldsymbol{x}_a = \mathbf{1}_d$ for all words $a$. This because it lack an *adversarial* term, like the one appearing in the modularity cost function.

We add this term with a technique called *negative sampling*. For each word $i \in \mathcal{T}$, we sample a set of $\mathcal{R}_i$ of random words sampled from the text and write the following loss function

*Negative sampling*

$$\mathcal{L} = -\sum_{i\in\mathcal{T}}\left[\sum_{j\in\mathcal{C}_i}\log\sigma\left(\boldsymbol{x}_{\pi(i)}^T\boldsymbol{x}_{\pi(j)}\right) + \sum_{j\in\mathcal{R}_i}\log\sigma\left(-\boldsymbol{x}_{\pi(i)}^T\boldsymbol{x}_{\pi(j)}\right)\right]. \quad (6.1)$$

This newly added term takes random pairs of words and gives a gain in the loss function when they are misaligned, thus preventing the trivial minimum. We now detail the strategy to optimize this cost function.

### 6.2.3 TRAINING THE MODEL PARAMETERS

The cost function is optimized with stochastic gradient descent and backpropagation. This is a modified version of gradient descent that is a method to optimize a multivariate function. Given a random argument of the function to optimize, the idea is to move in the direction of the negative gradient of the loss function, as depicted in Figure 6.2. This means

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} - \eta\nabla\mathcal{L}(\boldsymbol{x}_{\text{old}}),$$

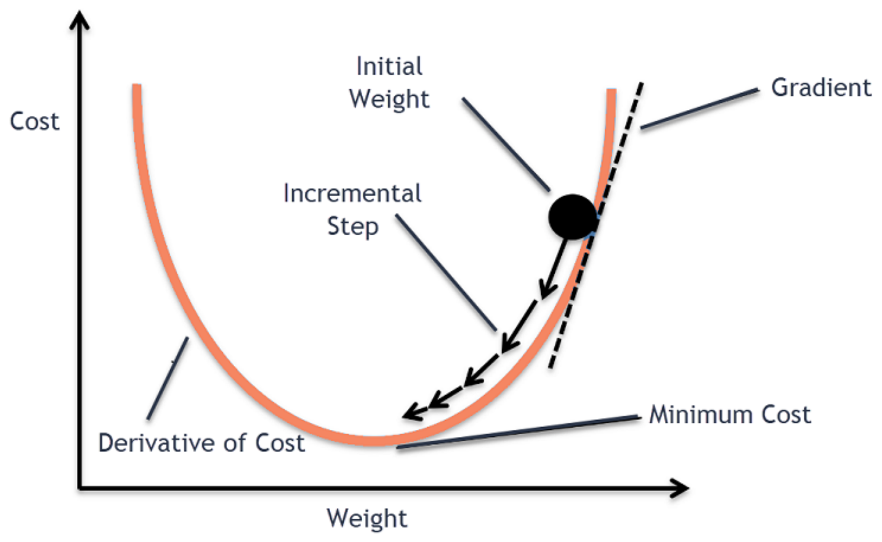


Figure 6.2: **Visualization of gradient descent**. Picture taken from https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/.

where $\eta > 0$ is the learning parameter. Now, computing the gradient may be unpractical. To simplify things we let $\mathcal{L} = \sum_{i \in \mathcal{T}} \mathcal{L}_i$ and we update the weight $\boldsymbol{x}_i$ with the value of the gradient of $\mathcal{L}_i$ (and not of the whole function $\mathcal{L}$) with respect to $\boldsymbol{x}_i$.

$$\frac{\partial \mathcal{L}_i}{\partial x_{\pi(i)a}} = -\left[\sum_{j \in \mathcal{C}_i} \frac{\sigma'\left(\boldsymbol{x}_{\pi(i)}^T \boldsymbol{x}_{\pi(j)}\right)}{\sigma\left(\boldsymbol{x}_{\pi(i)}^T \boldsymbol{x}_{\pi(j)}\right)} x_{\pi(j)a} - \sum_{j \in \mathcal{R}_i} \frac{\sigma'\left(-\boldsymbol{x}_{\pi(i)}^T \boldsymbol{x}_{\pi(j)}\right)}{\sigma\left(-\boldsymbol{x}_{\pi(i)}^T \boldsymbol{x}_{\pi(j)}\right)} x_{\pi(j)a}\right], \tag{6.2}$$

where $\sigma'$ denotes the derivative of the sigmoid function. One can easily verify that the following relation holds

$$\sigma'(x) = \sigma(x)\sigma(-x).$$

Plugging these relations in Equation (6.2) we obtain

*Stochastic gradient descent*

$$\begin{aligned} g_{\pi(i)a} &= \frac{\partial \mathcal{L}}{\partial x_{\pi(i)a}} \\ &= -\left[\sum_{j \in \mathcal{C}_i} \sigma\left(-\boldsymbol{x}_{\pi(i)}^T \boldsymbol{x}_{\pi(j)}\right) x_{\pi(i)a} + \sum_{j \in \mathcal{R}_i} \sigma\left(\boldsymbol{x}_{\pi(i)}^T \boldsymbol{x}_{\pi(j)}\right) x_{\pi(j)a}\right]. \end{aligned} \tag{6.3}$$

By iteratively updating the weights, we find an approximation of a minimum of the loss function $\mathcal{L}$ that provides us with a good representation of the words. Algorithm 6.1 summarizes the `Word2Vec` algorithm.

1 The sigmoid function is $\sigma(x) = (1 + e^{-x})^{-1}$. This is an increasing function, bounded between 0 and 1.

**Algorithm 6.1 :** `Word2Vec`

```
Input : Text $\mathcal{T}$ with $N$ words, dictionary with $n$ words, embedding
        dimension $d$, number of training epochs $\mathrm{n_{epochs}}$, learning
        rate $\eta$, window size $\omega$, number of negative samples $m$
Output : $\{x_a\}_{a\in[n]}$ word embedding vectors in $\mathbb{R}^d$
begin
    Randomly intialize $x_a$ for all $a \in [n]$ ;
    for epoch $= 1, \dots, \mathrm{n_{epochs}}$ do
        for $i \in \mathcal{T}$ do
            Draw $w$ uniformy at random $w \leftarrow \mathcal{U}(1, \omega)$;
            Get $\mathcal{C}_i$ for window width $w$ with Skip-Gram;
            Get $\mathcal{R}_i$ selecting $m$ random words from the text;
            Compute $g_{\pi(i)}$ as per Equation (6.2);
            Update $x_{\pi(i)} \leftarrow x_{\pi(i)} - \eta g_{\pi(i)}$
        end
    end
    return $\{x_a\}_{a\in[n]}$
end
```

## 6.3 NODE2VEC

Let us now go back to graphs. The `Node2Vec` algorithm uses the `Word2Vec` algorithm to obtain a node embedding by first "translating" the graph into a text and then embedding its words, corresponding to the graph nodes. The text is obtained performing random walks on the graph. Random walks are are paths made of sequences of adjacent nodes of the type $(v_1, v_2, v_3, \dots, v_T)$, where $T$ here denotes the walk length. Neighboring nodes are in contact and, in some sense, belong to the same context. Random walks are hence use to probe the network structure and to extract information out of it. The strategy is to define random walks with memory, with the definition of two parameters.

*Translating a graph into a text*

Suppose that the walker moves from $i$ to $j$ then it is at distance 1 from $i$. By performing a further step there are three possibilities: moving to a node that is at distance 2 from $i$, moving to a node that is at distance 1 from $i$ or going back to $i$ itself. These three options are taken with different probabilities that are proportional to $1/p$, $1/q$ and 1 respectively. The values of $p$ and $q$ are an input of the algorithm. For $p = q = 1$ we have a simple random walk without memory. The walking strategy is displayed in Figure 6.3. How to choose the parameters $p$ and $q$? The value of $p$ determines the probability of immediately returning to node the walker came from. Choosing large values of $p$ thus prevents the walker from bouncing back and forth between the same nodes and instead it encourages faster exploration of the network. This is exactly what happens with non-backtracking random walks and it is particularly useful for sparse graphs in which, given that each node has very

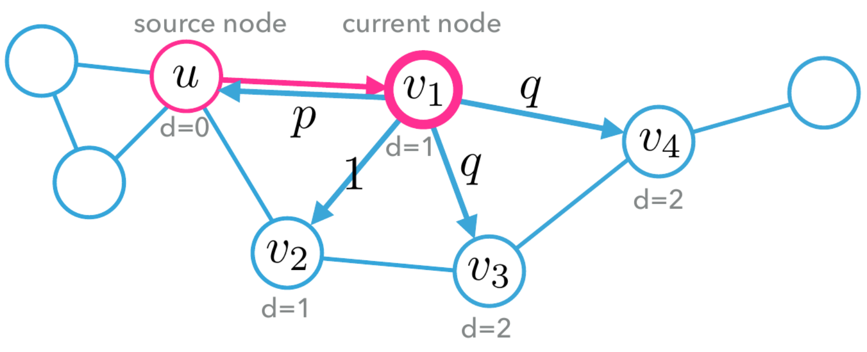


Figure 6.3: **Visualization of the random walk strategy of Node2Vec**. In the last step the random walker moved from $u$ to $v_1$. In the next step it will move: i) back to $u$ with a probability proportional to $1/p$; ii) to $v_3$ or $v_4$ with a probability proportional to $1/q$, since they are at a distance 2 from $u$; iii) to $v_2$ with a probability proportional to 1 since $v_1$ is at a distance 1 from both $u$ and $v_1$.

few neighbors, it is likely to move back to the node of origin. The value of $q$ determines to what extent the random walker is inclined to visit nodes that are further away from the original one. In particular, if $q > 1$ the walker is more inclined to remain close to $u$. This kind of transition accounts for the triangles that may be present in the network and attributes a higher chance to visit nodes that are tightly connected among them. On the other hand, small values of $q$ will lead to avoid these paths and to more rapidly explore the rest of the network. This has a direct impact on the type of information stored in the embedding, as shown in Figure 6.4. This plot shows the result of clustering based on the embedding obtained on the same graph for $p = 1$ and $q = 0.5, 2$, respectively. For $q = 0.5$ the algorithm is more prone to find tightly connected communities, while for $q = 2$ it groups the node according to structural equivalence.

## 6.4 CONCLUSION

Graph embedding methods are very powerful to analyze and represent relational data. In this chapter we introduced the `Node2Vec` algorithm that is one of the most influential methods developed in the past 10 years. Given its dependence on `Word2Vec`, it is rather fast and its complexity scales as $\mathcal{O}(|\mathcal{E}|md\omega)$. Moreover, note that, similarly to `Node2Vec`, a wealth of algorithms have explored similar strategies to exploit the `Word2Vec` algorithm and provide meaningful representations to complex mathematical objects. You should then see this as a relevant example of a widely used pipeline to define representation learning algorithms.

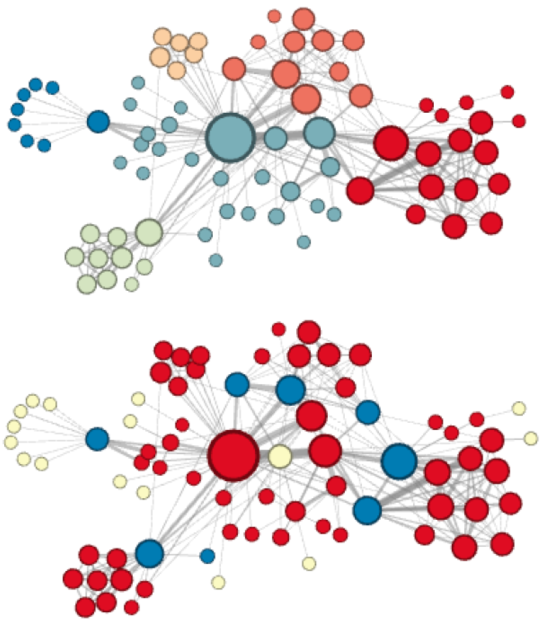


Figure 6.4: **Color coding according to the embedding obtained with two different values of $q$ of the Node2Vec algorithm**. The input graph shows the co-appearances of characters in *Les Misérables*. For the top plot $q = 0.5$, while for the bottom plot $q = 2$. Picture taken from *Grover, Leskovec, node2vec: Scalable Feature Learning for Networks.*

## 6.5 REFERENCES

- Mikolov, Sutskever, Chen, Corrado, Dean: *Distributed representations of words and phrases and their compositionality*
  This is the paper in which `Word2Vec` was introduced
- Grover, Leskovec: *node2vec: Scalable feature learning for networks*
  This is the paper in which `Node2Vec` was introduced