UNIVERSITÀ DI TRENTO

Network reconstruction via the minimum description length principle

Author:
Francesco Tanesini

Supervisor:
Dr. Luca Tubiana

A thesis submitted for the Bachelor degree in Physics

in the

Department of Physics

Date

1 Introduction to network science

An important challenge of science lies in understanding complexity. Many systems in nature and modern society exhibit emergent properties: collective behaviors not present at the level of individual constituents, but that arise from the interactions between them. Such systems are formally defined as complex systems. Their study requires an approach that is not reductionist in its essence. While the latter is essential to understand the nature of the single components of a complex system, collective behavior follow emergent laws, that can not be understood by considering the components and their characteristics singularly: a single neuron does not create a brain, a single ant is not an anthill, and one infected person is not an epidemic. These properties emerge only from a group of connected and interacting elements. As Philip Anderson stated in his essay *'More Is Different'* Ref. [1]:

> *"The ability to reduce everything to simple fundamental laws does not imply the ability to start from those laws and reconstruct the universe."*

Complex systems are everywhere in the modern world: societies, markets, insect communities, disease spread, urban traffic, and many more. The interconnected nature of these complex systems suggests modelling them with an appropriate mathematical theory, such as graph theory. A graph is defined as a group of elements, called nodes, some of which are connected by links, called edges. A graph is referred to as a network when it describes/models a real-world system. The study of complex systems through their representation as networks is called network science, an interdisciplinary field that combines not only principles and results of mathematics, computer science, physics, and biology, but also relies on other fields such as sociology, economics, and other social sciences.

Nowadays, network science has reached a remarkable level of development, and its use has led to important results in many contexts. For instance, big tech companies such as Google, Amazon, Apple, and many others base a large part of their business models on networks, and most algorithms for friend suggestions on platforms like Twitter, LinkedIn, and Meta heavily rely on networks.

Moreover, networks also appear in multiple organizational levels of the human body: at a subcellular scale they can describe interactions among genes, proteins, and the environment; at the organ level, neurons and cells communicate throughout the body through physical connections and coordinated signals; and at a societal level, humans interact with each other, forming social, business, and logistical networks Ref. [2].

Another important field that network science has revolutionized is epidemiology. Aside from COVID-19, one of the most important pandemics in modern epidemiology is the H1N1 of 2009. This is because it was the first pandemic whose general evolutionary trend was predicted months before reaching its peak. Even if the models were not able to predict the spread of the disease precisely, they provided a reliable forecast of its overall global trend, demonstrating the potential of network modeling in this field. Before 2000, epidemic models always used the strong assumption that every individual could infect everyone else in the same social or physical compartment (known as the mean field approach). In reality, this assumption is not realistic, and using a network approach to the problem, like in the H1N1 pandemic, helped to improve model accuracy (figure 1).

Nowadays, networks are often used to help understand and contain infectious diseases like Ebola: accurate models of disease spreading are necessary for planning vaccination strategies, especially when no specific therapeutic protocol is available Ref. [3].

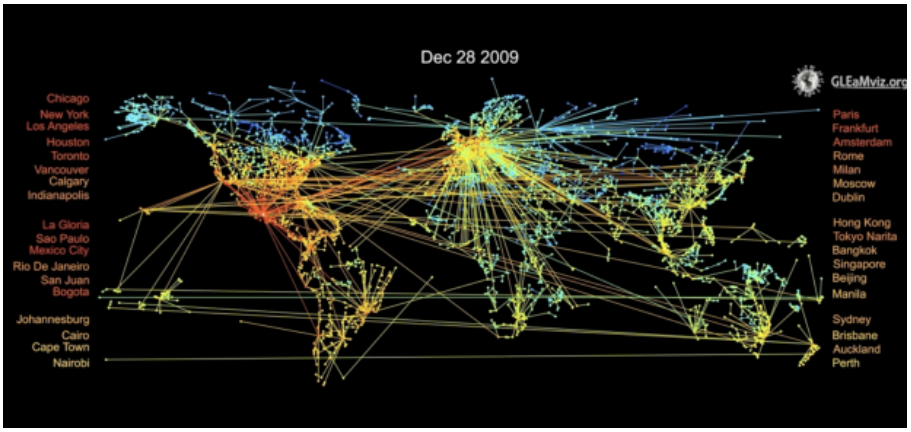


Figure 1: Network of one possible scenario of the H1N1 pandemic, predicted months before its peak of infections

The last example we cite is in the management field: one of the most common causes of inefficiency in a company is the lack of communication between key components, which can lead to poor optimization and consequently lower production and/or income. Mapping the social network of the company exposes such lack of interactions and can help managers diagnose organizational problems. For example, it has been shown how a large company like NASA had

> *"problems such as ineffective leadership, structural integration, communication barriers and practical drift"*

Ref. [4] and how a network analysis is an effective solution to spot and describe these problems.

These are only a few examples of how networks significantly impact many aspects of modern life. A more in-depth explanation of these and other examples can be found in Ref. [5] and Ref. [6].

Nevertheless, the focus of network science is not merely the mathematical theory of networks and graphs, but rather the data, function, and utility: even a well-developed mathematical framework is not sufficient if it does not describe the real world accurately. For this reason, it is important to have a method to predict the structure of a network by only having access to the data about the behavior of its components. We will focus on the prediction of networks, and in particular on the process of inferring the structure of a network only from observations of the system's collective behavior, which is called network reconstruction. To achieve this, the main purpose of this work is to retrace the work presented in T. Peixoto's article "Network reconstruction via the minimum description length principle" Ref. [7], aiming to explicit the steps and reproduce the methods and algorithm he describes.

2 Network basics

First of all, it is essential to emphasize the distinction between a graph and a network. As previously said, a graph is a formal mathematical object, while a network represents its application to a real-world scenario. Consequently, while every network can be viewed as a graph, the opposite is not necessarily true. A clear example illustrating this difference is presented in Figure (2), adapted from Ref. [8]:

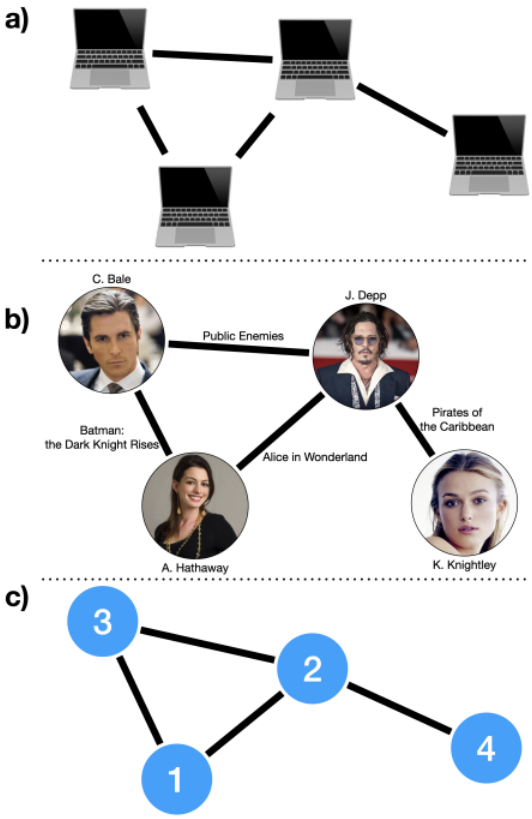


Figure 2: Visual difference between a graph and a network. Panels (a) and (b) describe, respectively, laptops linked and communicating with each other and relationships among a small group of actors. These are examples of network representations. Panel (c), on the other hand, represents the underlying graph of the first two networks.

With this distinction in mind, for the remainder of this work, we will refer to these objects exclusively as networks. For a more in-depth introduction to networks, please refer to Ref. [9], Ref. [10] and Ref. [6].

A network is a mathematical object composed of 2 sets (V, E), in which:

- $V = \{v_1, v_2, ..., v_p\}$ is a set of p *nodes*
- E is a subset of the set of vertex pairs $E \subseteq \{(v_i, v_j),\ i, j = 1, ..., p,\ i \neq j\}$. The pairs of vertices are called *edges*

Figure (3) shows an example of a 7-node and 14-edge network. A network can be *weighted*, which means that a weight is assigned to each edge, and a zero weight is equal to a non-existing edge. Moreover, we define a *directed network* as a weighted network in which the weight ω_{ij}, namely the weight of the edge between the i-th and the j-th node,

is different from ω_{ji}. Clearly, an *undirected* network implies that $\omega_{ij} = \omega_{ji}\ \forall i, j$. In this thesis, we will consider undirected weighted networks.

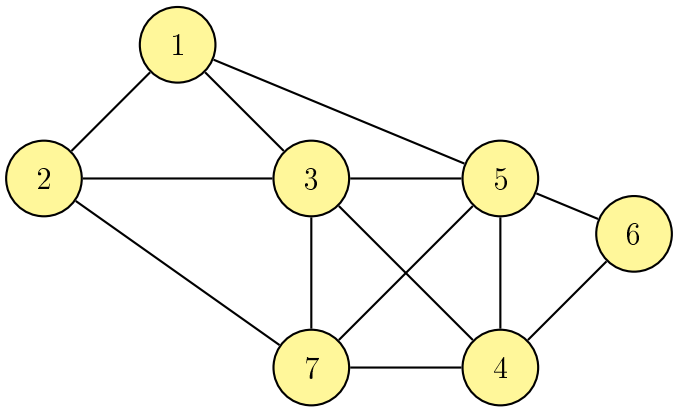


Figure 3: Example of an undirected network with 7 nodes and 14 edges.

To represent a network, we can use a matrix. A weighted network with N nodes can be described by an $N \times N$ matrix $\mathbf{W}$, in which

$$\mathbf{W}_{ij} = \begin{cases} \omega_{ij} & \text{if there is an edge from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}.$$

and where ω_{ij} is the weight of the edge between the i-th and the j-th node. For a non-directed network, the corresponding matrix will be symmetric, since $\mathbf{W}_{ij} = \mathbf{W}_{ji}$. To represent a non-weighted network, we use the so-called *adjacency matrix* $\mathbf{A}$, in which $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ and:

$$\mathbf{A}_{ij} = \begin{cases} 1 & \text{if there is an edge from node } i \text{ to node } j \\ 0 & \text{otherwise} \end{cases}.$$

For both matrices $\mathbf{W}$ and $\mathbf{A}$, the elements on the diagonal are equal to zero, since we assumed that an edge involves two and only two nodes, which means that a node cannot be linked to itself.

An example of an adjacency matrix is the one of the network in Figure (3):

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

3 Limitations of traditional methods in network science and the need for statistical inference

As mentioned above, network science studies complex systems by mapping them as networks, and it can be applied to many areas of everyday life. As L. Peel discusses in his paper "Statistical inference links data and theory in network science" Ref. [11], one big problem in this field is the separation between theory and its application. This means that general methods are often developed without accounting for uncertainties and incompleteness in the available data and, at the same time, methods are sometimes applied out of context because, for instance, they may be incompatible with the specific characteristics of the given data. This happens because scientists and researchers often choose the "shortest" path to determine whether a connection exists between two nodes. For example, when studying brain activity with time series data from different regions, this greedy approach suggests that if two regions show high correlation, then there must be a connection between them. This approach has at least two major problems. The first is purely statistical: correlation does not imply connection. By proceeding in this way, we ignore the fact that two regions may be correlated because they both respond to the activity of a third region, rather than being directly connected to each other. Consider,

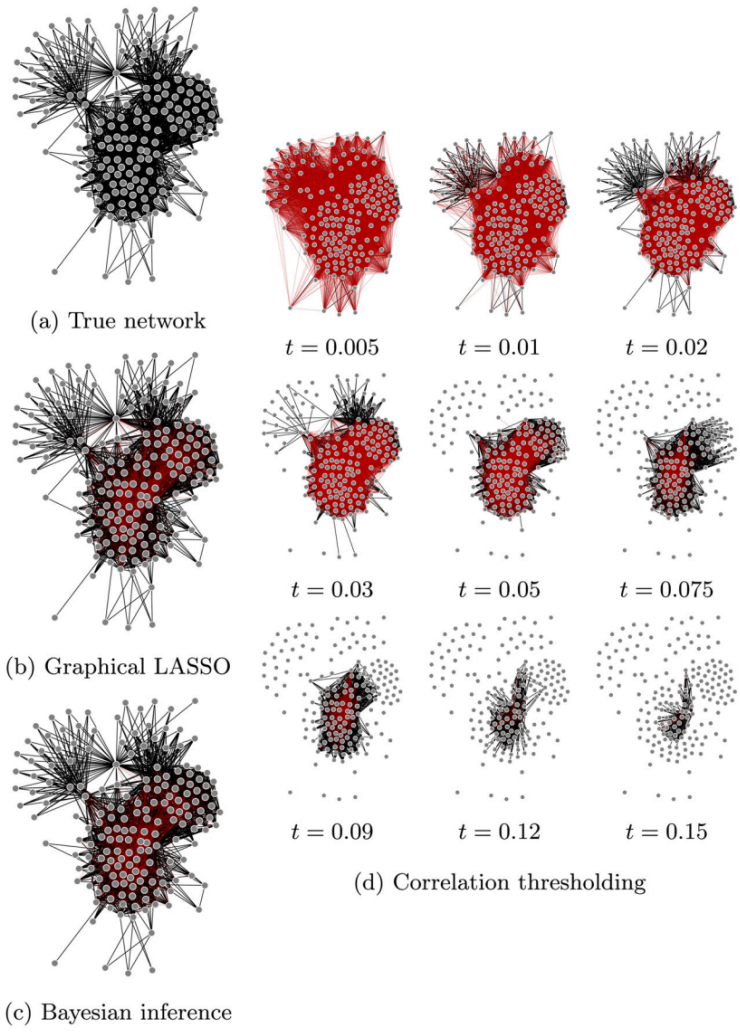


Figure 4: descrizione

for instance, three variables X, Y, and Z such that

$$X(t) = Z(t) + C_1 \tag{1}$$
$$Y(t) = Z(t) + C_2 \tag{2}$$

These two variables are certainly correlated, but not directly connected. A second major problem of this approach is that it requires setting an arbitrary threshold a priori to determine when two regions, or nodes in general, are directly connected. This choice is non-trivial, as it requires either assuming information about the network that is typically unavailable, or performing network reconstruction for many threshold values to identify which one best describes the data. The latter approach, known as cross-validation, is commonly used but is highly computationally expensive. However, this is a minor concern because even with sufficient computational power, the intrinsic problem remains: correlation does not imply connection. As Fig (4) adapted from Ref. [11] shows, the reconstructed networks with different threshold are for the most cases very different from the true network. Choosing the correct threshold requires the model to know more information about the newtork we are trying to reconstruct, such as level of sparsity or the tendency of some nodes to connect more with other nodes, which are information that usually we dont have and, instead, are the type of information that the reconstruction procedure is meant to infer rather than assume.

A second example of the issues with this approach is the case where proximity events are used to map a social network. Suppose we have GPS data showing the movements of a certain group of people. A greedy approach similar to the one described above for brain activity would suggest that when two people are closer than a certain threshold, there is likely a connection between them. We immediately see that the same two problems arise here as well: physical proximity does not imply the existence of a social relationship (two people sitting next to each other on a train are not necessarily relatives or friends), and there is the need to define a threshold a priori, which requires knowing or assuming information about the social network.

Another problem that network science often faces is how data are sampled and treated. Often, the available data are considered true and error-free, when in reality this is an assumption that is almost never valid. There can be various sources of error in network data: incomplete data that can create bias, interpretation errors (people misinterpreting survey questions), or even transcription errors of correct data. Failing to account for these aspects is a major mistake that leads to poor performance in network reconstruction models. The solution that appears the best one in many cases, and has indeed achieved the greatest success to date, is to treat network reconstruction as a statistical inference problem. The data $\mathcal{D}$ are considered sampled from a probability distribution $P(\mathcal{D} \mid \mathbf{W})$, the probability of observing the data $\mathcal{D}$ given the underlying network $\mathbf{W}$, where we can encode the information we know and/or the assumptions we make about the network in the prior $P(\mathbf{W})$, thanks to the posterior probability distribution:

$$P(\mathbf{W} \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \mathbf{W})P(\mathbf{W})}{P(\mathcal{D})} \tag{3}$$

where $P(\mathcal{D})$ is the evidence, representing the probability of observing the data under all possible network configurations. By maximizing the posterior distribution, we can infer an estimate of the true network while better managing the problems explained above. We can do this because with this statistical approach we no longer consider the

data perfectly true; instead, they are considered as generated from a certain distribution that is conditioned on the unseen true network. To reconstruct the network, we need to compute $P(\mathbf{W} \mid \mathcal{D})$ by making assumptions about how the data and the network are generated, and the result will have an uncertainty that depends on how correct our assumptions were and how noisy the data were. This process of network reconstruction via Bayesian inference is shown in Figure 5, adapted again from Ref. [11]. To a brief introduction to Bayesian statistics see next chapter.

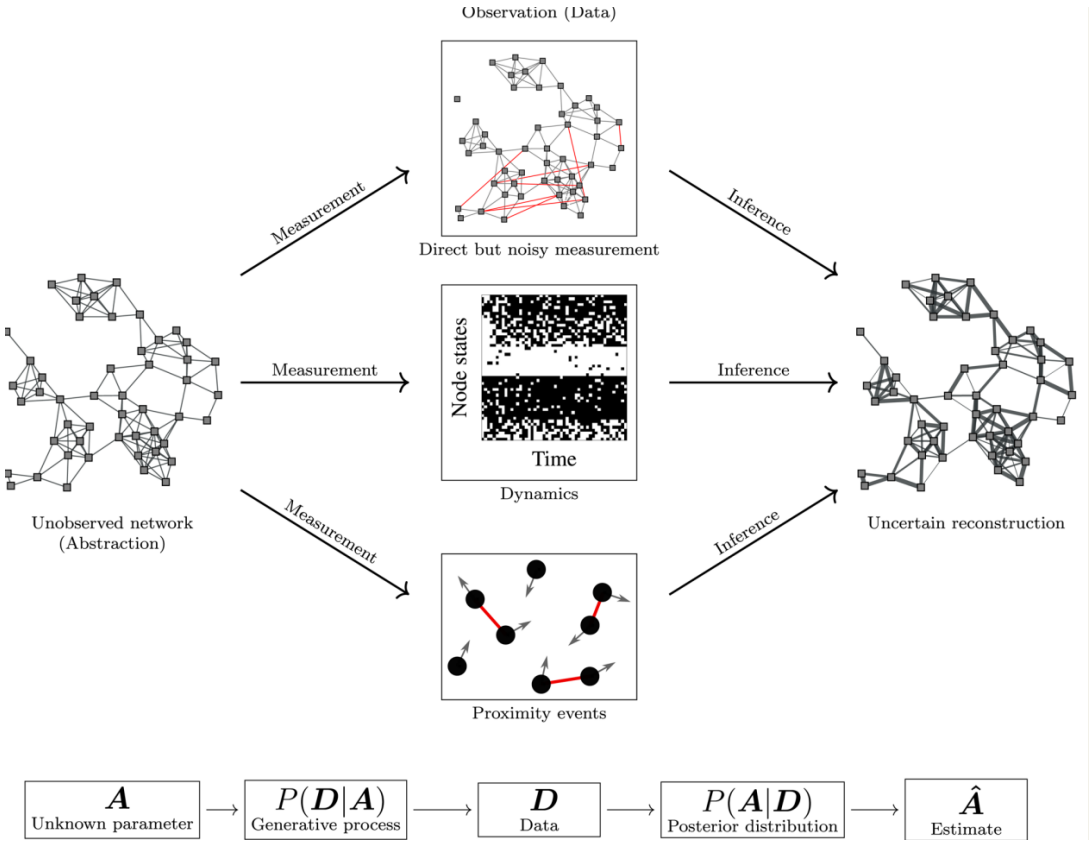


Figure 5

The statistical inference approach does not solve the problems we have explained, but rather provides the necessary tools to address them correctly. The difficulty in network science, and more generally in statistical inference, lies precisely in assigning the correct assumptions to the problem and encoding them in probability distributions. In this thesis, we will examine two methods for statistical network inference. The first uses Lasso regularization, and the second exploits the Minimum Description Length (MDL) principle. We will show how the Lasso (or L1) regularization approach is severely limited by the need to make strong assumptions about the network, and how the MDL principle manages to overcome this limitation.

4 Introduction to bayesian statistics

In network science, and more generally in data science, Bayesian statistics is often used. To properly introduce this framework, we will first recall the fundamental concepts of Kolmogorov's axiomatic theory and frequentist probability.

Kolmogorov's axiomatic theory is a deductive approach based on a triple (Ω, Σ, P), where Ω as the sample space (the set of all possible outcomes), Σ the σ-algebra of events, and P the probability measure. A function P is defined as a probability measure if it satisfies the following three axioms, known as the Kolmogorov axioms:

1. **Non-negativity**: The probability of an event is always a non-negative real number.

$$P(A) \geq 0 \quad \forall A \in \Sigma$$

2. **Normalization**: The probability of the certain event is exactly 1.

$$P(\Omega) = 1$$

3. **Countable Additivity**: For any sequence of mutually exclusive events $\{A_i\}_{i=1}^{\infty}$ (where $A_i \cap A_j = \emptyset$ for $i \neq j$):

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i)$$

The probability measure P has many properties that follow from the axioms, but for this work they are not all essential and therefore some of them will be omitted. For a more detailed and in-depth discussion, please refer to chapter 3 in Ref. [12]. One of these properties is the conditional probability rule (Figure 6):

$$P(A|B) = \frac{P(AB)}{P(B)} \tag{4}$$

We consider two events independent if $P(AB) = P(A)P(B)$, i.e. $P(A|B) = P(A)$. Since $P(AB) = P(BA)$, we have that

$$P(B|A)P(A) = P(BA) = P(AB) = P(A|B)P(B) \tag{5}$$

$$\Rightarrow P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{6}$$

A very important theorem is Bayes' theorem in Kolmogorov's formulation: suppose we have a partition $\{A_1, A_2, \ldots, A_n\}$ of S, i.e. $S = \bigcup_{i=1}^{n} A_i$, so that $P(A_i) > 0 \ \forall i$ and $A_i \cap A_j = \emptyset \ \forall i, j$. Let B be an event such that $P(B) > 0$. Then:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}, \quad \text{where} \quad P(B) = \sum_{j=1}^{n} P(B|A_j)P(A_j) \tag{7}$$

The frequentist probability framework can be applied in this context to associate a probability to an event. This approach is inductive and is fundamentally based on repeated trials. It assumes that for every event there exists a fixed and objective probability

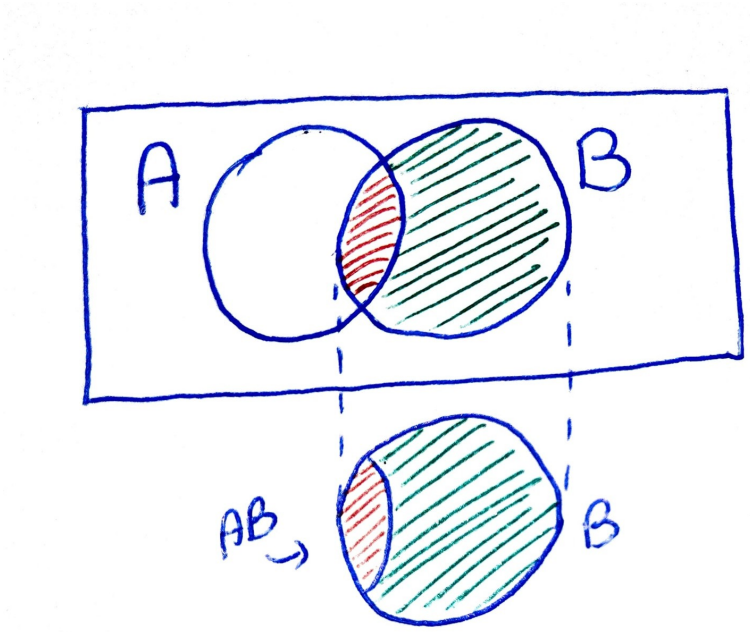


Figure 6: Visual representation of the conditional probability rule

value. To estimate this value, it is necessary to perform N measurements, from which many statistical estimators can be derived, such as mean and standard deviation. The bigger N is, the more this estimators are precise. One important point is that, in this framework, probability is estimated only based on the available data, which are assumed to be drawn from a specific probability distribution whose characteristics are under investigation. While this approach is highly effective in many scenarios, it becomes impractical in others. For instance, if the available dataset is small, or when repeated measurements are impossible, the statistical estimators become significantly imprecise. Furthermore, this approach does not utilize any prior information regarding the situation being analyzed. Finally, a major limitation of the frequentist approach is that it cannot assign probabilities to unique events or cases in which repeated trials are impossible. Consider, for example, the following question: "What is the probability that a specific asteroid will collide with Earth in the next century?" From a frequentist perspective, this question is problematic because we cannot repeat this scenario multiple times to gather statistical data. Each asteroid follows a unique trajectory, making this a one-time event. Moreover, the estimate of this probability can depend heavily on past data known a priori, such as the historical record of asteroid impacts and current astronomical observations of near-Earth objects. It is exactly here that Bayesian statistics becomes necessary.

Bayesian statistics represent uncertainty about parameters by assigning them a probability distribution. Even if the parameters may be fixed and unknown, bayesian statistics uses probability distributions as a measure of how much we know, or we don't know, about their values, and the data is considered fixed after being sampled from a probability distribution. Hence, rather than producing a single "true" estimate, Bayesian inference aims to characterize the full range of plausible parameter values with their associated plausibility, answering the question: "Given these observations and what I already know, what is the probability distribution of the parameter?". With reference to the asteroid example, Bayesian statistics allows us to incorporate considerations such as: the historical record of asteroid impacts, current astronomical surveys and tracking data, our understanding

of orbital mechanics, and the estimated population of near-Earth objects. This approach is particularly powerful because it remains effective even with limited data or unique events, where frequentist methods would fail.

A common criticism of this approach is its subjectivity due to the requirement to define a prior distribution. The main challenge here is that it is difficult to formalize subjectivity. In this context, the concept of probability is redefined as a "degree of plausibility" $\beta(A)$. Unlike the Kolmogorov approach, based on the frequency of events, the Bayesian approach is built on propositions: $\beta(A)$ represents the degree of plausibility of the proposition A being true. To be a good statistical theory, the plausibility measures must be mapped into real numbers in order to allow consistent comparison. Furthermore, the degree of plausibility must be updatable as new evidence is acquired. Using the notation $\beta(A|B)$ to represent the plausibility of A given the information B, this means that, in general, $\beta(A|B) \neq \beta(A|BC) \neq \beta(A|BCD)$, and so forth. Finally, it is essential for the theory to be consistent, and this requires three fundamental principles: first, if a result can be derived in multiple ways, each of them must lead to the same conclusion. Second, the framework must account for all available evidence in order not to discard relevant information. Third, equivalent states of evidence must correspond to equivalent degrees of plausibility. To transform these logical principles into a usable mathematical tool, we map any function $\beta(A)$, representing a degree of plausibility, onto the real interval $[0, 1]$. The value 0 represents a proposition that is certainly false, while 1 represents one that is certainly true. By doing so, we can treat the degree of plausibility as probabilities with all their rules. To a more in depth explanation see the first chapter of Ref. [13].

Bayes' theorem is central to many machine learning and statistical inference methods, as it provides a principled way to update our beliefs after we observed data. In this context, it is often written as

$$P(\theta \mid \mathcal{D}, \mathcal{H}) = \frac{P(\mathcal{D} \mid \theta, \mathcal{H})\, P(\theta \mid \mathcal{H})}{P(\mathcal{D} \mid \mathcal{H})}. \tag{8}$$

where

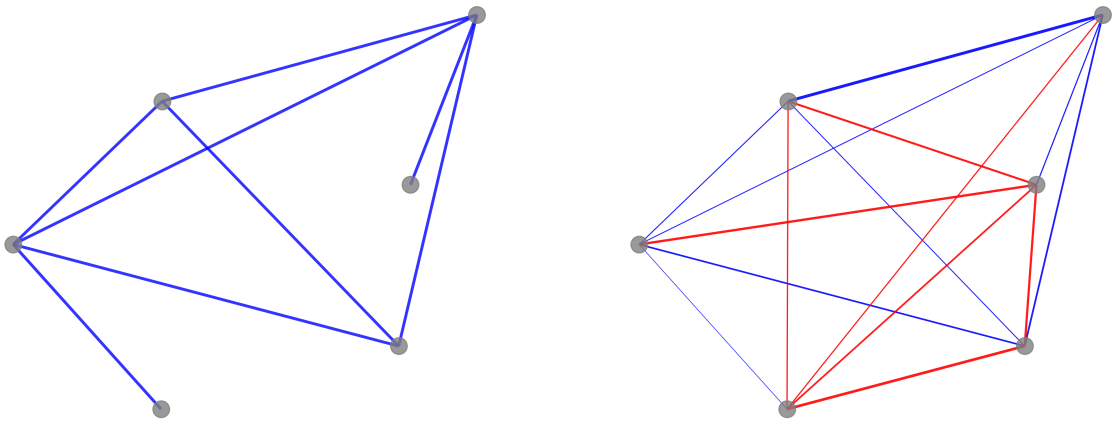
- $\mathcal{D}$ is the observed dataset.
- $\mathcal{H}$ is the hypothesis: it identifies all that we know about the system.
- θ is the set of parameters of a function $f(\theta)$ which generates the data $\mathcal{D}$.
- $P(\theta|\mathcal{D}\mathcal{H})$ is the **posterior distribution**. It is the probability distribution of θ being the parameters that generated the data $\mathcal{D}$, given the model $\mathcal{H}$.
- $P(\mathcal{D}|\theta, \mathcal{H})$ is the **likelihood function**. It describes how well the parameters θ explain the observed data $\mathcal{D}$.
- $P(\theta|\mathcal{H})$ is the **prior distribution**. This is where we put all the information, knowledge, and assumptions about the parameters before any data is collected and/or analyzed.
- $P(\mathcal{D}|\mathcal{H})$ is the **evidence**. It represents the probability of observing the data $\mathcal{D}$ assuming that the model $\mathcal{H}$ is correct. In practice, however, it is often used only as a normalization constant.

Here, θ denotes an unknown parameter (or a collection of parameters). The key idea of Bayesian statistics is that we can associate a probability distribution to one or more unknown parameters, while in Kolmogorov's framework this makes no sense because probability is associated to events. Importantly, this means that the probability distribution of θ reflects our incomplete knowledge about its value. As more informative data (or stronger prior knowledge) become available, the posterior distribution typically becomes more concentrated, giving more precise estimations and uncertainty quantifications.

5 Network reconstruction: L1 regularization

Network reconstruction is the process of determining the unseen interactions between elements of a complex system when only data describing their observed behavior are available Ref. [7]. In other words, given a collection of interacting units, we can observe their activity (e.g., time series, states, or responses), but the underlying interaction structure is not directly accessible. The goal is to infer *who influences whom*, and possibly *how strongly*, from the observed behavior alone. A simple example is provided by social media. Suppose that we only have access to behavioral data recording users actions, such as likes and comments over time. From these observations, we may aim to reconstruct an underlying network of influence: which users tend to affect the activity of others, and with what intensity, even if the explicit social graph, for instance follower or friendship relations, is not available.

Network reconstruction is a complex task that has a fundamental problem associated with, which is determining the right model complexity in order to prevent overfitting and to have a statistically justifiable number of edges with justifiable weights. We say that a model overfits when it learns the training data too closely, capturing not only the underlying signal but also random noise. In the context of network reconstruction, overfitting occurs when the inferred network explains the observed dataset extremely well by introducing spurious edges and/or unrealistic edge weights. As a consequence, the reconstructed network generalizes poorly and have worse performance on unseen data. Indeed, the main problem with a classic MLE (Maximum Likelihood Estimation) approach, and also with an L_2 or *Ridge* regularization, is that we cannot distinguish a low-weight edge with a non existing one: this approaches produces a massive overfitting because we do not have any sort of threshold which tells us when a weight has to be considered non existent. At the end the model will produce a network that has nearly all the possible edges, i.e. a number of that ike $O(N^2)$. For instance, figure (7) shows how this approach produces overfitting. As we can see, even with a small number of nodes (6) all possible edges are inferred.



(a) True network

(b) inferred network with L2 regularization

Figure 7: In the inferred newtork, blue edges correspond to correctly inferred connections and red edges to incorrectly inferred connections. The width of the edges correspond to the weight magnitude.

For this reason a L_1 or *Lasso* regularization is generally used: it prevents overfitting by setting the smaller weights exactly to zero, creating what is called *weight shrinkage*, and promoting sparsity in the network. It consist in forcing a constraint on the weight vector/matrix to have the 1-norm equal or smaller than a given constant:

$$\|\mathbf{W}\|_1 = \sum_{i<j} |\mathbf{W}_{ij}| \leq c \tag{9}$$

This constraint leads to the final equation

$$\mathbf{W}_{lasso} = \arg \min_{\|\mathbf{W}\|_1 \leq c} [E_{in}(\mathbf{W}) + \lambda \|\mathbf{W}\|_1] \tag{10}$$

The strength of the regularization depends on the hyperparameter λ. A big value corresponds to a bigger weight shrinkage, which means that more small weights are set to zero but also that big weights are shriked more (figure 8). The network will have more sparsity and the model will have a bias caused by the distortion of the weights. On the other side, a small value of λ will cause a smaller shrinkage i.e. more edges and less sparsity or, in other words, more overfitting.

To be optimal the L_1 regularization needs to improve the right level of sparsity, controlled by the hyperparameter λ. Sparsity thus become an input value; this is a big problem to face, because in the majority of cases we want to have it as an output parameter. Thus, since λ is not knownw a priori, it must be selected by exploring a range of many candidate values. An attempt to solve this problem is by using L_1 regularization together with cross validation. Even if this approach improves a bit the predicted network, it significantly increases the computational cost and still needs the value of λ to be chosen between a set of arbitrary values.

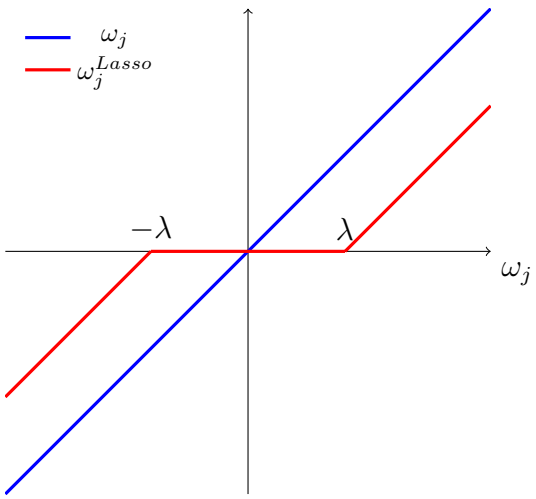


Figure 8: Visual representation of weight shrinkage for the linear regressor

There are many other regularization approaches, but these always requires cross validation and impose strong assumptions on the weight distribution. To put it briefly, there is no classic regularization approach that is simultaneously: principled, effective at preventing overfitting, algorithmically efficient and non-parametric, in particular without needing to have the level of sparsity to be known in advance.

Following T. Peixoto's paper in Ref. [7], in this thesis we will discuss the the minimum description length (MDL) approach to the network reconstruction problem. This

approach consist in fining the right level of sparsity so that the data can be represented in the most compact way possible, through quantization of weights. As will be discussed in section 6, the MDL approach has all the features we are looking for: it is principled, it requires only one fit to produce an output (no cross validation required), it gives the level of sparsity as an output parameter and it effectively prevents overfitting.

First of all we assume that the data, represented by a matrix $\mathbf{X}$, are originated from a distribution with likelihood:

$$P(\mathbf{X} \mid \mathbf{W}) \tag{11}$$

with $\mathbf{W}$ the $N \times N$ matrix of weights. The element W_{ij} correspond to the weight of the edge between the nodes v_i and v_j, and because we using a non-directional network $\mathbf{W}$ is symmetric. We consider the matrix $\mathbf{X}$ as an $N \times M$ matrix: the N rows corresponds to the N nodes and the M columns corresponds to M independent samples. The element X_{nm} correspond to the m-th sample of the n-th node. Since the samples are independent we can write the likelihood as

$$P(\mathbf{X} \mid \mathbf{W}) = \prod_{m=1}^{M} P(X_m \mid \mathbf{W}) \tag{12}$$

where X_m is the m-th column of $\mathbf{X}$.

In our case we proceed by using a posterior distribution:

$$P(\mathbf{W} \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid \mathbf{W})P(\mathbf{W})}{P(\mathbf{X})} \tag{13}$$

It's importat to notice that this approach works with any generative model of the data with any type of likelihood. By proceeding with the Maximum A Posteriori we finally obtain:

$$\begin{aligned} \widehat{\mathbf{W}} &= \arg\max_{W} \left[P(\mathbf{W}|\mathbf{X})\right] \\ &= \arg\max_{W} \left[\log P(\mathbf{X}|\mathbf{W}) + \log P(\mathbf{W})\right] \end{aligned} \tag{14}$$

since the natural logarithm is an increasing monotonic function. This correspond to the most likely weight distribution that generates the given data.

As said, the first attempt to solve the problem is by using an L_1 regularization. This is equivalent to using a Laplace prior in the MAP approach

$$P(\mathbf{W}|\lambda) = \prod_{i<j} \frac{1}{2} \lambda e^{-\lambda |W_{ij}|} \tag{15}$$

As common we compute the logarithm

$$\begin{aligned} \log\left(P(\mathbf{W}|\lambda)\right) &= \sum_{i<j} \left[\log(\frac{\lambda}{2}) - \lambda |W_{ij}|\right] \\ &= \binom{N}{2} \log(\frac{\lambda}{2}) - \lambda \sum_{i<j} |W_{ij}| \end{aligned} \tag{16}$$

in which $\binom{N}{2}$ comes from the fact that the sum on $i < j$ means that we are summing over the upper-diagonal elements of the matrix, which are $\frac{N^2 - N}{2} = \binom{N}{2}$ (all the elements minus the diagonal over 2).

This approach has many pros like the convexity of $\log\left(P(\mathbf{W}|\lambda)\right)$ which means that it exist one single minimum and it is computational efficient to find it numerically, and the fact that promotes sparsity even if it introduces bias. However, there are two big problems here: first of all we need to give the level of sparsity as an input hyperparameter, while as already said, in the majority of times we want to have it as an output value. Moreover the laplacian prior in eq (15) in unbounded, and it goes to $+\infty$ with $\lambda \to \infty$ and $|W_{ij}| = 0$, causing the model to be not trustable when it happens.

The commonly used solution to this is using k-fold methos. It consists in divinding the dataset $\mathbf{X}$ in k subsets so that $\mathbf{X} = \{\mathbf{X_1}, ..., \mathbf{X_k}\}$ and for each k (usually k=5) we train the model:

$$\widehat{\mathbf{W}}(\lambda) = \arg\max_{W} \left[\log\ P(\mathbf{X} \setminus \mathbf{X_k}|\mathbf{W}) + \log\ P(\mathbf{W}|\lambda)\right] \tag{17}$$

after that we compute the log-likelihood on the non trained data (we could use any kind of loss function)

$$\mathcal{L}_k(\lambda) = \log\ P(\mathbf{X_k}|\widehat{\mathbf{W}}(\lambda)) \tag{18}$$

and finally we can find the best value of λ with

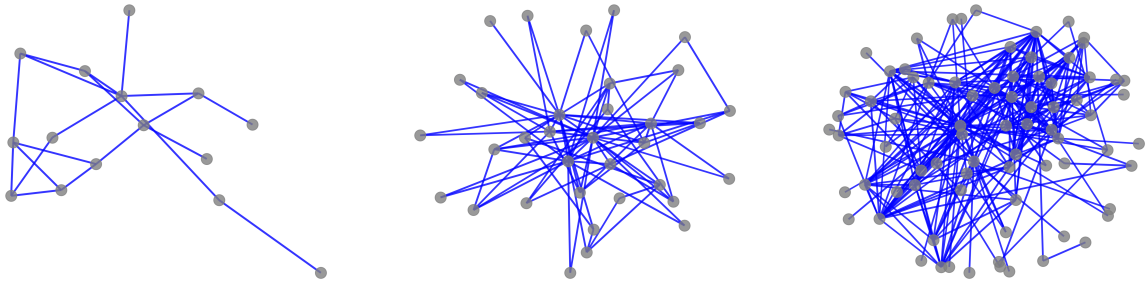
$$\overline{\lambda} = \arg\max_{\lambda} \sum_k \mathcal{L}_k(\lambda) \tag{19}$$

This approach can prevent overfitting in many cases but it has an enormous computational cost: it is needed to train the model $K \times L$ times, where L is the number of values of λ that we want to try. Even if it works in theory, Peixoto shows in his paper that this procedure does not sufficently prevent the model to overfit. To show this we are now going to use the L_1 regularization method to reconstruct a weight matrix $\widehat{\mathbf{W}}$ from a matrix of data $\mathbf{X}$ generated from a true weight matrix $\mathbf{W}$. The idea is to take a pre-existing network and assign weights to its edges, so that we have a true matrix that generates the data matrix $\mathbf{X}$. In particular, to generate the data we will use the kinetic Ising model, which we explain below. The purpose of this step is to have a way to evaluate the performance of the reconstruction model: once the data are generated, we 'forget' the true weight matrix $\widehat{W}$ that generated them, and proceed to reconstruct the matrix $\mathbf{W}$ using only the data. Finally, we use the original true matrix to evaluate how well we have reconstructed the network. To confront the two matrices we will use the Jaccard similarity $s\left(\mathbf{W}, \widehat{\mathbf{W}}\right)$:

$$s\left(\mathbf{W}, \widehat{\mathbf{W}}\right) = 1 - \frac{\sum_{i<j} \left|W_{ij} - \widehat{W}_{ij}\right|}{\sum_{i<j} |W_{ij}| + |\widehat{W}_{ij}|} \tag{20}$$

where $s \in [0, 1]$; $s = 1$ means that the two matrices are equal. In particular we will also use this comparison with the binarized matrices of $\widehat{\mathbf{W}}$ and $\mathbf{W}$, which are simply

$$A_{ij}\left(\mathbf{W}\right) = 1\ if\ \ |W_{ij}| > 0,\ else\ 0 \tag{21}$$



(a) Florentine Families (b) Zachary's karate club (c) Les Misérables character

Figure 9: Visual representation of the three used networks

We use three different python built-in networks as topology of the network we want to reconstruct: the Zachary's karate club network with $N = 34$ nodes and $E = 78$ edges; the Les Misérables character network with $N = 77$ nodes and $E = 254$ edges; the Florentine Families graph with $N = 15$ nodes and $E = 20$ edges. Figure (9) shows these three networks. To each edge of every network is assigned a new weight sampled from a normal distribution $\mathcal{N}(N/2E, 0.01)$:

$$\mathcal{N}(0.22, 0.01) \quad \text{for the Zachary's karate club network} \tag{22}$$

$$\mathcal{N}(0.15, 0.01) \quad \text{for the Les Misérables character network} \tag{23}$$

$$\mathcal{N}(0.37, 0.01) \quad \text{for the Florentine Families graph} \tag{24}$$

$$\tag{25}$$

The matrix of data $\mathbf{X}$ is generated using the kinetic Ising model as generative model. The kinetic Ising model describes the time evolution of a lattice of spin 1/2 particles. The hamiltonian of the i-th particle is:

$$H_i(t) = \sigma_i h_i(t) = J\sigma_i \sum_{j \ near} \sigma_j(t) + \sigma_i H_{ext} \tag{26}$$

where:

$$h_i(t) = J \sum_{j \ near i} \sigma_j(t) + H_{ext} \tag{27}$$

where J is the coupling coefficient, the sum runs over all the neighbors of particle i, $\sigma_j(t) \in \{-1, 1\}$ is the spin state of the particle j at time t and H_{ext} is an external costant field. At every time step, we consider the probability that a particle is found in the state $\sigma_i(t+1) = +1$ as $P(\sigma_i(t+1) = +1|\sigma(t))$ and the probability that it is found in the state $\sigma_i(t+1) = -1$ as $P(\sigma_i(t+1) = -1|\sigma(t))$. This probabilities must satisfy the detailed balance condition, i.e.:

$$\frac{P(\sigma_i(t+1) = +1|\sigma(t))}{P(\sigma_i(t+1) = -1|\sigma(t))} = e^{-\beta \Delta E} \tag{28}$$

where $\Delta E = E(\sigma_i = +1) - E(\sigma_i = -1) = -2h_i$ because $E(\sigma_i) = \sigma_i h_i$. If we call $P(\sigma_i(t+1) = +1|\sigma(t)) = p$ then $P(\sigma_i(t+1) = -1|\sigma(t)) = 1 - p$, and we can write:

$$\frac{p}{1-p} = e^{2\beta h_i(t)} \quad \Rightarrow \quad p = \frac{1}{1 + e^{-2\beta h_i(t)}} = \frac{e^{\beta h_i(t)}}{2\cosh(\beta h_i(t))} \tag{29}$$

In our case, we initialize each node with a random value in $\{-1, 1\}$, and then we simulate $M = 1500$ transitions. We consider as neighbors all nodes that are connected to the given node with a non-zero weight. In particular, we consider the field on acting on each node as:

$$h_i(t) = \theta_i + \sum_{j=1}^{N} W_{ij} S_j(t) \tag{30}$$

where θ_i is the external field, which we set to zero, W_{ij} is the weight between nodes i and j, and $S_j(t)$ is the value of the node j at time t. At each time step t and for each node, we compute the value of $p = P(S_i(t+1) = +1|S_j(t))$ as derived above and generate a random number between 0 and 1: if it is less than p, the we set $S_i(t+1) = +1$, otherwise we set $S_i(t+1) = -1$. At the end we obtain a matrix $\mathbf{X}$ in which each of the N rows correspond to vectors of the time evolution of the value of a single node, or in other words the M columns correspond to vector of the value of all nodes at a set time.

We have seen that, for one node i at a given time t we have

$$P(S_i(t+1) \mid S_i(t)) = \frac{e^{\beta S_i(t+1) h_i(t)}}{2\cosh(\beta h_i(t))} \tag{31}$$

so that, if we include every node

$$P(S(t+1) \mid S(t)) = \prod_{i=1}^{N} \frac{e^{\beta S_i(t+1) h_i(t)}}{2\cosh(\beta h_i(t))} \tag{32}$$

and by including all the M independent samples

$$P(\mathbf{X} \mid \mathbf{W}) = \prod_{t=0}^{M-1} \prod_{i=1}^{N} \frac{e^{\beta S_i(t+1) h_i(t)}}{2\cosh(\beta h_i(t))} \tag{33}$$

thus, we can obtain the log-likelihood by compute the logarithm:

$$\log P(\mathbf{X} \mid \mathbf{W}) = \sum_{t=0}^{M-1} \sum_{i=1}^{N} \left[\beta S_i(t+1) h_i(t) - \log(2\cosh(\beta h_i(t)))\right] \tag{34}$$

later, in section (6), we will also need two more things. First, derivative of the log-likelihood:

$$\frac{\partial \log P(\mathbf{X} \mid \mathbf{W})}{\partial W_{ij}} = \beta \sum_{t=0}^{M-1} S_j(t) \left[S_i(t) - \tanh(\beta h_i(t))\right] \tag{35}$$

where we used $\partial h_i / \partial W_{ij} = S_j$. After that it is also helpful to calculate how much does the log likelihood change when a single weight $W'_{ij} \to W_{ij}$ change

$$h'_i(t) = h_i(t) + (W'_{ij} - W_{ij}) S_j(t) \Rightarrow \Delta h_i(t) = (W'_{ij} - W_{ij}) S_j(t) \tag{36}$$

$$\Rightarrow \Delta \log P(\mathbf{X} \mid \mathbf{W}) = \sum_{t=0}^{M-1} \left[\beta S_i(t+1) \Delta h_i(t) - \log\left(\frac{\cosh(\beta h'_i(t))}{\cosh(\beta h_i(t))}\right)\right] \tag{37}$$

After doing the data generation for the three networks, we apply the L_1 regularization to infer the weight matrix. We do this by using an L_1 regularized logistic regressor. The i-th row of the inferred weight matrix $\widehat{\mathbf{W}}$ is obtained by fitting a logistic regressor where

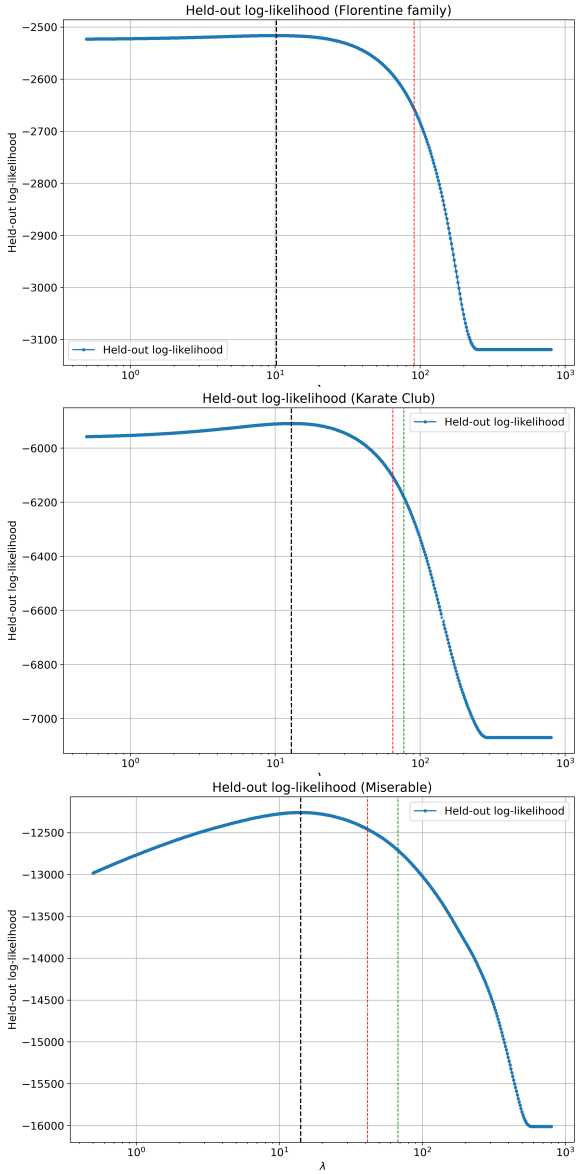
the features are the values of all the nodes at time t and the target is the value of the node i at time $t+1$, and this for every t. We performed this procedure for 500 values of the hyperparameter λ for each network, and for every value of λ we carried out a 5-fold cross-validation and computed the corresponding held-out log-likelihood. The results can be seen in figure (10), figure (11) and summarized in table (1). Figure (12) shows the three reconstructed networks.

Network	N	E	λ^W	λ^A	λ^{LL}	s^W_{max}, s^A_{max}	$s^W_{\lambda_{LL}}, s^A_{\lambda_{LL}}$	$\mathbf{W}^{mean}$	$\widehat{\mathbf{W}}^{mean}$	t
Florentine Family	15	20	91.0	91.0	10.2	0.96, 1.00	0.70, 0.43	0.37	0.37	50 s
karate Club	34	78	64.8	77.4	12.9	0.89, 0.98	0.64, 0.42	0.22	0.17	3min 45 s
Les Misérables	77	254	41.0	65.8	14.1	0.70, 0.78	0.55, 0.34	0.15	0.08	32 min

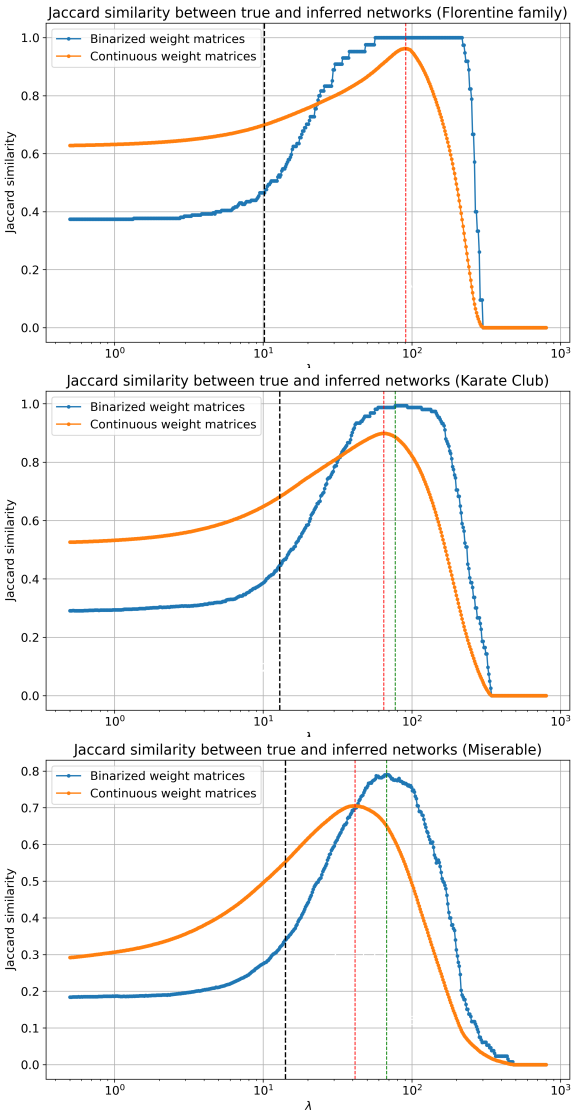
Table 1: Values of the three reconstructed networks. The colums $\mathbf{W}^{mean}$ and $\widehat{\mathbf{W}}^{mean}$ refers to the mean value of the non zero weights for the weight matrix that maximize the continuous Jaccard similarity. The column t is the time that the algorithm needed to reconstruct the inferred weight matrix. Note that this times are a mean of more reconstruction of a computer with an Intel(R) Core(TM) i5-14600KF (3.50 GHz) CPU

From the plots in figure (10) and (11) it is evident that, in all three networks, the increase of the regularization hyperparameter λ simultaneously sparsifies the inferred network and shrinks the magnitude of the inferred weights. As anticipated, it is clear that having a priori the knowledge of the true level of sparsity is essential in order to obtain an accurate reconstruction. For instance, the highest value of the log-likelyhood of the Les Misérables network is reah with a value of $\lambda^{LL} = 12.9$, for an associated network with $E = 1132$. The best value of the hyperparameter that maximize the binarized Jaccard similarity is $\lambda^A = 65.8$, for an associated network with $E = 262$. The continuous and binarized Jaccard similarity are 0.55 and 0.34 for λ^{LL}, 0.66 and 0.79 for λ^A. This means, as expected, that the model is massively overfitting. Indeed, we can see that the reconstructed networks shown in figure (12), in confront the true ones, have a large amount of small edges that are not correct, an istantly recognizable sign of overfitting.

Another relevant observation is that we can see that the bigger are the number of true nodes and edges, the narrower is the interval of values of λ in which the number of inferred edges is reasonably accurate. In other words, the more complex the true underlying network is, the more crucial it becomes to know its true sparsity level. Nevertheless, even when the correct value of λ is known, weight shrinkage remains a problem (figure (11)): expecially in complex networks, to the value of best λ corresponds a very strong weight shrinkage of inferred weights. For example, for the karate club network the maximum of binarized Jaccard similarity is achieved for a value $\lambda^A = 77.4$, for a network with $E = 79$ non zero weights, which mean is 0.17 with respect to the true value 0.22 (23% loss). This is more evident for the les Misérables network, which has the maximum of of binarized Jaccard similarity with $\lambda^A = 65.8$, for a network with $E = 262$ non zero weights, which have a mean of 0.08 with respect to the true value 0.15 (47% loss).This results clearly highlights the trade-off between edges recovery and weight magnited preservation, mostly controlled by the choice of the hyperparameter λ. It is interesting to note that the value of λ that maximizes the binarized Jaccard similarity tipically gives a slightly more accurate number of inferred edges, expecially for larger and more complex networks, and it is always bigger than the value obtained from the continuous one. We can also see that the value of best λ strongly depends on the topology and size (nodes and edges) of the network. Finally we can say, by looking in table (1), that the time needed to reconstruct

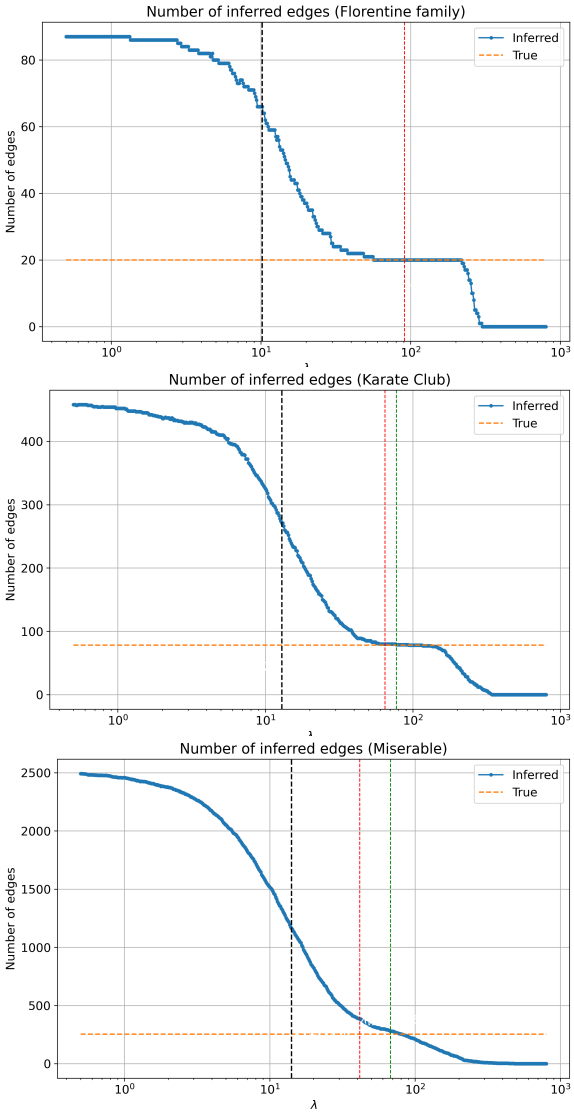


Held-out log-likelihood vs regularization strength for the three networks.

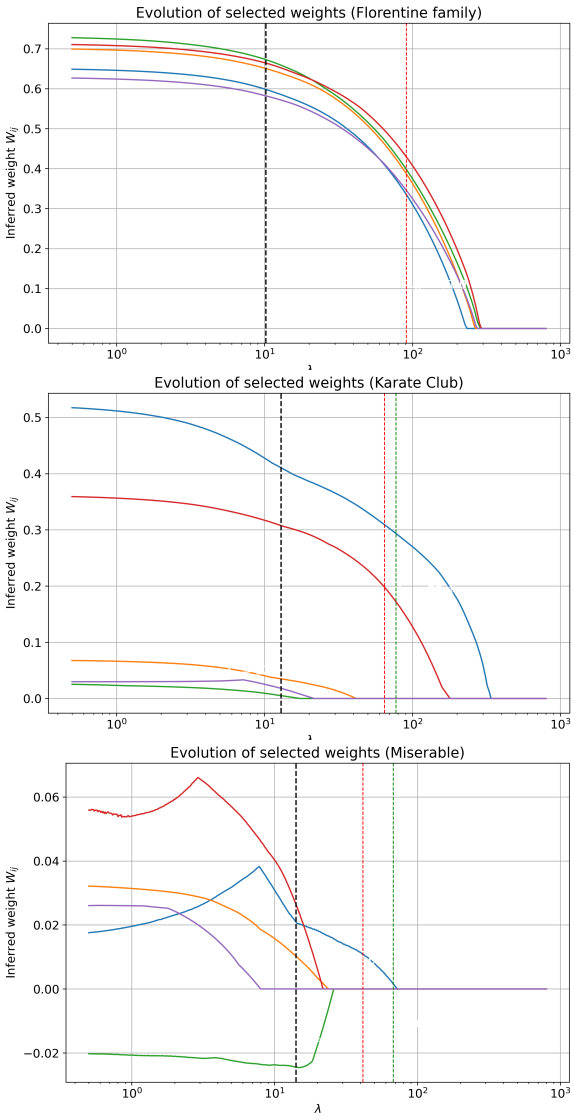


Jaccard similarity vs regularization strength

Figure 10: Results of the L_1 reconstruction. For each network, the black line represent the value of hyperparameter λ^{LL} that maximizes the held-out log-likelihood, obtained from a 5-folds cross validation on each value of λ. In red is marked the best value of λ^W that maximizes the continuous Jaccard similarity, while in green the best value of λ^A that maximizes the binarized Jaccard similarity.

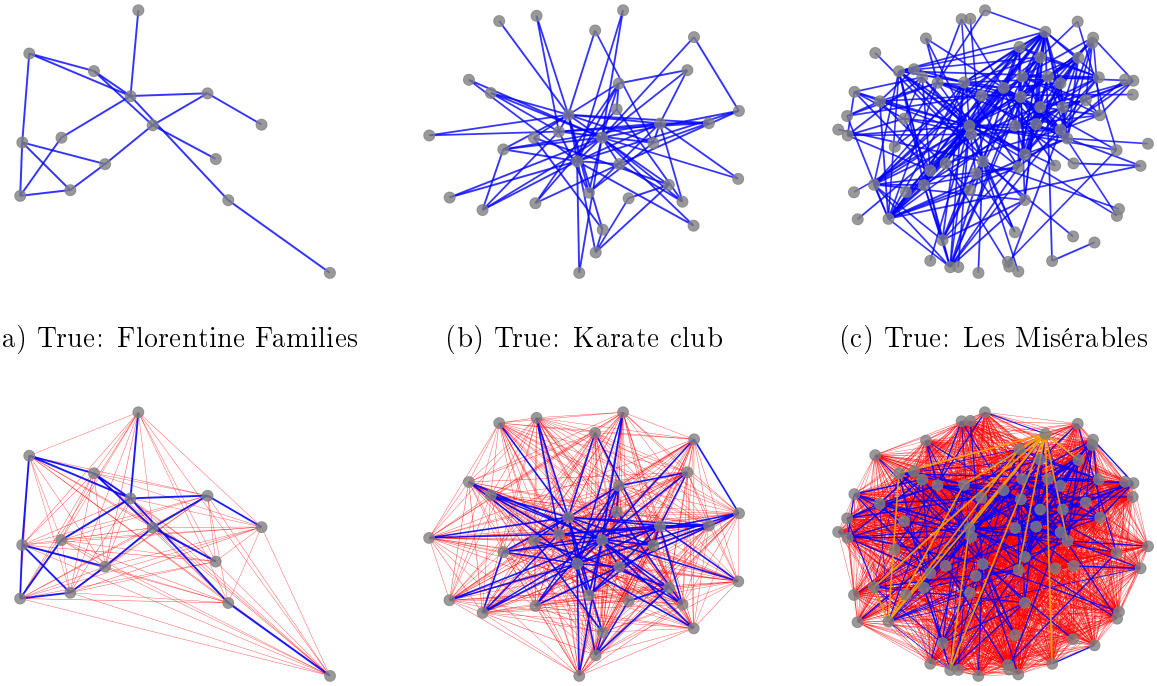


Number of inferred edges vs regularization strength



Weighs magnitude vs regularization strength

Figure 11: Results of the L_1 reconstruction. For each network, the black line represent the value of hyperparameter λ^{LL} that maximizes the held-out log-likelihood, obtained from a 5-folds cross validation on each value of λ. In red is marked the best value of λ^W that maximizes the continuous Jaccard similarity, while in green the best value of λ^A that maximizes the binarized Jaccard similarity.



(a) True: Florentine Families (b) True: Karate club (c) True: Les Misérables

(d) L1: Florentine Families (e) L1: Zachary's karate club (f) L1: Les Misérables character

Figure 12: Visual representation of the three reconstructed networks with L1 regularization (bottom row) and the corresponding true networks (top row). Blue edges correspond to correctly inferred connections, red edges to incorrectly inferred connections, and orange edges to true connections that were missed by the reconstruction.

the network increase more or less linearly with $N \cdot E$. i.e. the size of the network.

To conclude this section, we have seen that the classical L_1 regularization approach has two main problems: the need to know the sparsity level a priori, which strongly depends from size and topology of the network, and the unavoidable trade-off between sparsity and weight shrinkage. Moreover, this approach can become computational expensive for large networks. For this reasons, it is necessary to develop an inference algorithm that can effectively handle this issues.

6 The minimum description length principle

Bayesian model comparison can be viewed from an information-theory perspective by shifting from probabilities to length in bit of messages. In this framework, instead of focusing on the probability of an event, we focus on the length in bits of a message needed to communicate that event without information loss (Ref. [14], chap. 28.3). The relation between probabily $P(x)$ and message length $L(x)$ is given by:

$$P(x) = 2^{-L(x)} \iff L(x) = -log_2 P(x) \tag{38}$$

This is the basic concept of the minimum description length (MDL) principle: the more short is the necessary message to communicate the evenet, the more probable is the event i.e. models that can communicate data with the smallest possible number of bits should be preferred.

We want to implement a regularization principle that is principled, promotes sparsity, does not need a cross validation and does not rely on weight shrinkage. The idea is to find a proper prior $P(\mathbf{W})$ that can satisfy our needs through the MAP estimation

$$\widehat{\mathbf{W}} = \arg\max_{W} \left[\log P(\mathbf{X}|\mathbf{W}) + \log P(\mathbf{W})\right] \tag{39}$$

and connecting these equation in terms of the MDL principle:

$$\Sigma(\mathbf{X}, \mathbf{W}) = -\log P(\mathbf{X}|\mathbf{W}) - \log P(\mathbf{W}) \tag{40}$$

where $\Sigma(\mathbf{X}, \mathbf{W})$ is the description length of the model, $-logP(\mathbf{X}|\mathbf{W})$ represent the information required to encode the data $\mathbf{X}$ when the parameters $\mathbf{W}$ are known i.e. the fit error and $-logP(\mathbf{W})$ represent the amount of information needed to encode the parameters $\mathbf{W}$. Note that we should use base-2 logarithms, but since we only need to minimize this quantity and since log and $\log_2$ differ only by the constant factor $\log 2$, we can use the natural logarithm anyway. In our scenario, the prior $P(\mathbf{W})$ is a sort of penalty to the likelihood that can prevent over-complex models from being inferred and thus can prevent overfitting. For this reason, the optimization of Eq. 39 should give a model with the ideal balance between quality of fit and model complexity.

We notice that if the weights that constitute $\mathbf{W}$ are considered continuous with infinite precision, the prior $P(\mathbf{W})$ become a probability density function and therefore the term $-\log P(\mathbf{W})$ cannot be interpreted as information. This is the reason why we cannot determine the ideal value of the hyperparameter λ in Eq. 15 without cross validation: the ideal value is not automatically the most compressive one. For this reason, we have to determine a weight precision and this means that the possible weights need to be discretely distributed. This is advantageous because with discrete weights, every non-zero weight, having more precision and adding more edges, costs bits and hence it penalizes the likelihood, promoting sparsity and preventing overfitting.

Our purpose now is to find an expression for the prior $P(\mathbf{W})$. We do this by steps.

First of all, we only consider the topology of the network using the binarized adjacency matrix $\mathbf{A}$, which tells us which edges of the weight matrix $\mathbf{W}$ are non-zero. We assume that $\mathbf{A}$ is sampled from a uniform distribution with a given number of edges E. Given the number of elements N, the possible edges are $\binom{N}{2}$, and the possibilities of sampling E edges from these $\binom{N}{2}$ are $\binom{\binom{N}{2}}{E}$:

$$P(\mathbf{A}|E) = \frac{\delta_{\left(\sum_{i<j} A_{ij}\right), E}}{\binom{\binom{N}{2}}{E}} \tag{41}$$

where the numerator stands only for meaning that E must be the number of non-zero edges $\sum_{i<j} A_{ij}$. We sample the value E from a uniform prior

$$P(E) = \frac{1}{\binom{N}{2} + 1} \tag{42}$$

where the $+1$ stands for the possible zero value of E. Now that we have $P(\mathbf{A}|E)$, we sample the weights as:

$$P(\mathbf{W}|\mathbf{A}) = \prod_{i<j} P(W_{ij})^{A_{ij}}\, \delta_{W_{ij},0}^{1-A_{ij}} \tag{43}$$

In this case $P(W_{ij})$ is the probability of a non-zero weight W_{ij}, equal for all (i, j): if $A_{ij} = 0$, then we have $P(\mathbf{W}|\mathbf{A}) = \delta_{W_{ij},0}$ i.e. $W_{ij} = 0$, and if $A_{ij} = 1$ then $P(\mathbf{W}|\mathbf{A}) = P(W_{ij})$. In particular, $P(W_{ij})$ needs to be a discrete probability mass function, and we can consider the weight as a result of this quantization

$$W_{ij} = \Delta \left\lfloor \frac{Y_{ij}}{\Delta} \right\rceil \tag{44}$$

with Y_{ij} an auxiliary value, $\lfloor \cdot \rceil$ the round operation and Δ the precision of the weights. The weight W_{ij} will be a multiple of Δ, and the probability mass will be $P(W_{ij}|Y_{ij}, \Delta) = \delta_{W_{ij}, \left\lfloor \frac{Y_{ij}}{\Delta} \right\rceil}$.

We can use a Laplace distribution for the $\mathbf{Y}$ values

$$P(Y_{ij}|\lambda) = \frac{1}{2}\lambda e^{-\lambda|Y_{ij}|} \tag{45}$$

and thus we can find that the probability mass:

$$\begin{aligned} P(W_{ij} \mid \lambda, \Delta) &= \frac{\int P(W_{ij} \mid Y_{ij}, \Delta)\, P(Y_{ij} \mid \lambda)\, dY_{ij}}{1 - \int P(0 \mid Y_{ij}, \Delta)\, P(Y_{ij} \mid \lambda)\, dY_{ij}} \\ &= \begin{cases} \frac{1}{2} e^{-\lambda|W_{ij}|}\, (e^{\lambda\Delta} - 1), & \text{if } W_{ij} = \Delta \left\lfloor \frac{W_{ij}}{\Delta} \right\rceil \\ & \qquad W_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{46}$$

where the denominator is there to exclude the value zero for W_{ij}. To do that integral, we start from the numerator:

$$\int P(W \mid, \Delta)\, P(Y \mid \lambda)\, dY = \int \delta_{W, \Delta \left\lfloor \frac{Y}{\Delta} \right\rceil} \cdot \frac{\lambda}{2} e^{-\lambda|Y|} \tag{47}$$

and since

$$\delta_{W, \Delta \left\lfloor \frac{Y}{\Delta} \right\rceil} = \begin{cases} 1, & \text{if } Y \in [W - \frac{\Delta}{2}, W + \frac{\Delta}{2}] \\ 0, & \text{otherwise} \end{cases} \tag{48}$$

then

$$\int P(W\mid,\Delta)\,P(Y\mid\lambda)\,dY = \frac{\lambda}{2}\int_{W-\frac{\Delta}{2}}^{W+\frac{\Delta}{2}} e^{-\lambda|Y|} = \frac{\lambda}{2}\left[-\frac{1}{\lambda}e^{-\lambda Y}\right]_{W-\frac{\Delta}{2}}^{W+\frac{\Delta}{2}} \tag{49}$$

$$= \frac{1}{2}\left[e^{-\lambda(W-\frac{\Delta}{2})} - e^{-\lambda(W+\frac{\Delta}{2})}\right] \tag{50}$$

$$= \frac{1}{2}e^{-\lambda W}\left(e^{\lambda\Delta/2} - e^{-\lambda\Delta/2}\right) \tag{51}$$

$$= \frac{1}{2}e^{-\lambda W}e^{-\lambda\Delta/2}\left(e^{\lambda\Delta} - 1\right) \tag{52}$$

The we proceed with the denominator the same way

$$\int P(0\mid Y,\Delta)\,P(Y\mid\lambda)\,dY = \int_{-\Delta/2}^{\Delta/2}\frac{\lambda}{2}\,e^{-\lambda|Y|}\,dY = 1 - e^{-\lambda\Delta/2} \tag{53}$$

$$\Rightarrow 1 - \int P(0\mid Y,\Delta)\,P(Y\mid\lambda)\,dY = e^{-\lambda\Delta/2} \tag{54}$$

and we find our result

$$P(W\mid\lambda,\Delta) = \frac{1}{2}\frac{e^{-\lambda W}e^{-\lambda\Delta/2}\left(e^{\lambda\Delta}-1\right)}{e^{-\lambda\Delta/2}} = \frac{1}{2}\,e^{-\lambda|W|}\left(e^{\lambda\Delta}-1\right) \tag{55}$$

At this point we have already solved two main issues: we can quantify the description length thanks to the quantization procedure, and thanks to that we can promote sparsity independently from weight magnitude. The problem is that by using this prior we are still relying on $L1$ regularization and thus on weight shrinkage. To solve this problem we need to be able to learn the weight distribution directly from data and not assume that their probability follows the Laplace distribution. Instead we assume at first that our weights are sampled from an arbitrary discrete distribution

$$P(W_{ij}\mid\mathbf{p},\mathbf{z}) = \sum_{k=1}^{K}\delta_{W_{ij},z_k}\,p_k, \tag{56}$$

in which $\mathbf{z} = \{z_k\}$ is a set of K real values for the weights and $\mathbf{p} = \{p_k\}$ their corresponding probabilities. Obviously we have $\sum_k p_k = 1$. We can then obtain

$$P(\mathbf{W}\mid\mathbf{p},\mathbf{z},\mathbf{A}) = \prod_k p_k^{m_k}\times\prod_{i<j}\delta_{W_{ij},0}^{1-A_{ij}} \tag{57}$$

where $m_k = \sum_{i<j}A_{ij}\delta_{Wij,z_k}$ is the dimension of the k-th category. From this we can derive the marginal distribution

$$P(\mathbf{W}\mid\mathbf{z},\mathbf{A}) = \int P(\mathbf{W}\mid\mathbf{p},\mathbf{z},\mathbf{A})\,P(\mathbf{p})\,d\mathbf{p} \tag{58}$$

by assuming a uniform prior density $P(\mathbf{p}) = (K-1)!$ we have

$$\int P(\mathbf{W} \mid \mathbf{p}, \mathbf{z}, \mathbf{A})\, P(\mathbf{p})\, d\mathbf{p} = (K-1)! \int \prod_k p_k^{m_k} \times \prod_{i<j} \delta_{W_{ij},0}^{1-A_{ij}} d\mathbf{p} \tag{59}$$

$$= (K-1)! \prod_{i<j} \delta_{W_{ij},0}^{1-A_{ij}} \int \prod_k p_k^{m_k} d\mathbf{p} \tag{60}$$

$$= (K-1)! \prod_{i<j} \delta_{W_{ij},0}^{1-A_{ij}} \frac{\prod_k m_k!}{(E+K-1)!} \cdot \frac{E!}{E!} \tag{61}$$

$$= \frac{\prod_k m_k!}{E!} \binom{K+E-1}{E}^{-1} \prod_{i<j} \delta_{W_{ij},0}^{1-A_{ij}} \tag{62}$$

where we used Ref. [14], chap. 23.5

$$\int \prod_k p_k^{m_k} d\mathbf{p} = \frac{\prod_k \Gamma[m_k+1]}{\Gamma\left[\sum_k (m_k+1)\right]} = \frac{\prod_k m_k!}{\Gamma\left[E+K\right]} = \frac{\prod_k m_k!}{(E+K-1)!}, \text{ where } \begin{cases} \sum_{k=1}^K \mathbf{p}_k = 1 \\ \sum_k m_k = E \end{cases} \tag{63}$$

the only problem here is that we are allowing all the categories z_k with $m_k = 0$, which means that we need to label nonexistent categories. The term that needs to be changed is $\binom{K+E-1}{E}$. This can be interpreted as how many solutions of $m_1 + \ldots + m_k = E$ there are, and in analogy we now want the number of solutions of $m_1 + \ldots + m_k = E$ with $m_i \geq 1\ \forall i$. To find this we just substitute $m_k = n_k + 1$ so that our problem becomes finding the number of solutions of

$$(n_1 + 1) + \ldots + (n_k + 1) = E \iff n_1 + \ldots + n_k = E - K \tag{64}$$

which is $\binom{E-1}{K-1}$. Finally we have

$$P(\mathbf{W} \mid \mathbf{z}, \mathbf{A}) = \frac{\prod_k m_k!}{E!} \binom{E-1}{K-1}^{-1} \prod_{i<j} \delta_{W_{ij},0}^{1-A_{ij}} \tag{65}$$

We now need to find a probability mass for the z_k weight categories. Since we are trying to minimize the description length, we expect $K \ll E$, thus, this term is relatively unimportant compared to the others. For this reason we can use the Laplace distribution again, so that

$$P(z_k \mid \lambda, \Delta) = \begin{cases} \frac{1}{2} e^{-\lambda |z_k|} \left(e^{\lambda \Delta} - 1\right), & \text{if } z_k = \Delta \left[\frac{z_k}{\Delta}\right] \\ & z_k \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{66}$$

and with

$$P(\mathbf{z} \mid \lambda, \Delta, K) = \prod_{k=1}^K P(z_k \mid \lambda, \Delta) = \frac{1}{2^K} e^{-\lambda \sum_k |z_k|} \left(e^{\lambda \Delta} - 1\right)^K \tag{67}$$

This choice will lead to weight shrinkage, but as just said, this term is not very important and for this reason, in most cases, the weight shrinkage will be almost unnoticeable.

To conclude thi part, we only need the prior for the number of weight categories K, which we again assume to be uniform:

$$P(K \mid \mathbf{A}) = \frac{1 - \delta_{E,0}}{E} + \delta_{E,0}\delta_{K,0}. \tag{68}$$

this means that, if $E = 0$ there are no edges and $P(K \mid \mathbf{A}) = \delta_{K,0}$ i.e. K must be zero, while if $E > 0$ we have $P(K \mid \mathbf{A}) = 1/E$. Putting all the terms all together we obtain our prior

$$P(\mathbf{W} \mid \lambda, \Delta) = \sum_{z,K,\mathbf{A},E} P(\mathbf{W} \mid \mathbf{z}, \mathbf{A})\, P(\mathbf{z} \mid \lambda, \Delta, K)\, P(K \mid \mathbf{A})\, P(\mathbf{A} \mid E)\, P(E) \tag{69}$$

$$= \frac{\prod_k m_k! \times e^{-\lambda \sum_k |z_k|} \left(e^{\lambda\Delta} - 1\right)^K}{E! \binom{E-1}{K-1} 2^K \max(E,1) \binom{\binom{N}{2}}{E} \left[\binom{N}{2} + 1\right]}, \tag{70}$$

where the term $\max(E, 1)$ only needs to handle the case $E = 0$.

We note that, in case of $m_k \simeq 1\ \forall k \iff K \simeq E$ the combinatory part $\prod_k m_k! \simeq 1$ does not shrink the prior and thus i have a model where every weight is coded singularly, like a $L1$ regularization. For this reason, this prior works well only if $K \ll E$, which is not a drawback since the MDL principle intrinsically promotes a small number of categories.

It is important to note that the vlaue of Δ is not the precision of the weights, but the precision of the weight categories. In particular, W_{ij} has a value z_k and Δ tells how much distance there is between z_k and $z_{k\pm1}$. This is very important because if we approach this problem using MDL principle together with $L1$ regularization, as in Eq. (46), each weight pays for its own value: a large weight costs more bit and thus weight shrinkage is inevitable. Instead, using a Laplace prior only for the weight categories, and not for each weight, shifts the penalty on the complexity of the set $\{z_k\}$ rather than to each single weight, which then become a label of an existing value z_k.

Our model now has two hyperparameters, λ and Δ. Surely they can be both optimized, but as we will see in the following section it is not necessary. We can set Δ to a small value such $\Delta = 10^{-8}$, so that we can treat the weight values as continuous, and set $\lambda = 1$, optimizing it only if nrcrssary, since as previously mentioned, the part of the prior that depends on λ is less significant.

It is also important to notice that this regularization is, as we wanted, non-parametric. This is not because it has no parameters, but because the distribution and the number of parameters adapt to the data and what is consequently statistically justifiable. That means that the number of edges E, the number of weight categories K, and the value of the categories z_k change their value during the reconstruction in order to compress the data at maximum, letting the not to overfit. This procedure is exactly what we wanted: it is non-parametric, it prevents overfitting and weight shrinkage, and promotes sparsity. There is only one downside, and it is that now the prior $-\log P(\mathbf{W} \mid \lambda, \Delta)$ is not convex. Thus, the minimization approach requires a specific algorithm, which will be shown in the following section.

```python
def log_prior(state: W_state):
    W = state.W
    tol = state.tol
    Delta = state.Delta
    N = W.shape[0]
    lam = state.lam
    # Cerco gli archi diversi da zero
    iup, jup = np.triu_indices(N, k=1)
    w = W[iup, jup]

    active_mask = np.abs(w) > tol
    w_active = w[active_mask]
    E = int(w_active.size)

    B = N * (N - 1) // 2 # Comodo da avere: binom(N 2)

    # Considero separatamente il caso E == 0
    if E == 0:
        # In questo caso il numeratore e' pari a 1, e al denominatore e' tutto pari a 1 tranne (N 2) + 1 -> rimane solo -log(B + 1)
        return -math.log(B + 1.0)


    # Ora mi prendo le categorie
    q = np.rint(w_active / Delta).astype(int)
    q = q[np.abs(q) > 0]
    uq, counts = np.unique(q, return_counts=True)
    counts = counts.astype(int)
    K = int(uq.size)
    z_all = uq.astype(float) * Delta

    # Calcolo il numeratore
    log_num = 0

    log_num += float(np.sum([log_factorial(int(mk)) for mk in counts])) # Π_k m_k!
    log_num += - lam * float(np.sum(np.abs(z_all))) # exp(-λ Σ_k |z_k|)
    log_num += K * math.log(math.expm1(Delta*lam)) #  (e^{Δλ}-1)^K

    # Calcolo il denominatore
    log_den = 0

    log_den += log_factorial(E)  # E!
    log_den += log_binom(E - 1, K - 1) # binom(E-1  K-1)
    log_den += K * math.log(2.0) # 2^K
    log_den += math.log(E) # E' il termine max(E,1), che qua e' sempre E perche abbiamo gia gestito E == 0
    log_den += log_binom(B, E)   # binom(binom(N 2) E)
    log_den += math.log(B + 1.0) # binom(N 2) + 1

    # Unisco i due pezzi
    logP = log_num - log_den
    return logP
```

Figure 13: The figure shows how the the function that calculate the log prior has been implemented

7 Inference Algorithm

To minimize a non-convex function like the prior in Eq. (70) the classical methods like gradient descent cannot be used. For this reason we have to implement a new algorithm that can find the minimum of the description length, i.e. the minimum of

$$\Sigma(\mathbf{X}, \mathbf{W}) = -\log P(\mathbf{X}|\mathbf{W}) - \log P(\mathbf{W}) \tag{71}$$

where $-\log P(\mathbf{X}|\mathbf{W})$ is the log-likelihood of the data and $-\log P(\mathbf{W})$ is the log-prior of Eq. (70).

The algorithm consists of many different types of updates, which will be accepted only if they improve the description length. At first, we start with an empty weight matrix $\mathbf{W}$ and an empty set of categories $\mathbf{z}$, and then the following types of updates will be tried sequentially until the description length no longer improves:

Edge update: We first find the N best zero weights as update candidates, finding the ones that miximize

$$\pi_{ij} = \max[\log P(\mathbf{W}^+), \log P(\mathbf{W}^-)] \tag{72}$$

where $\mathbf{W}^+$ is the starting matrix in which the weight W_{ij} has been replaced by the smallest positive value of the set of weights category $\mathbf{z}$, and $\mathbf{W}^-$ is the same but with the smallest magnitude negative value. If $\mathbf{z}$ is currently empty, the gradient will be used as score:

$$\pi_{ij} = \left| \frac{\partial \ln P(\mathbf{W}' \mid X)}{\partial W'_{ij}} \right|_{\mathbf{W}'=\mathbf{W}} \tag{73}$$

We create the set $\mathcal{E}$ as the union of the candidates found and the current active weights. After that, we visit every entry (i, j) in $\mathcal{E}$ and try to update them $W_{ij} \to W'_{ij}$ singularly one by one, accepting the update only if it improves the description length (DL), choosing randomly from these three options:

1. $W'_{ij} \in \mathbf{z}$ is chosen by doing a random bisection search over the existing values of $\mathbf{z}$ minimizing the DL

2. $W'_{ij} \in \mathbb{R}$ and $W'_{ij} \notin \mathbf{z}$ is chosen by doing a random bisection search over the allowed values for the weights. This option will produce a new weight category with $\mathbf{z}' = \mathbf{z} \cup \{W'_{ij}\}$

3. $W'_{ij} \in 0$ if $W'_{ij} > 0$

Edge moves: Given the set $\mathcal{E}$ from above, we iterate over each node i that has at least one connection. For each such node, we randomly pick one of its existing edges (i, j) and one potential edge (i, u) chosen from $\mathcal{E}$ where currently $W_{iu} = 0$. We swap these weights, transferring the weight from (i, j) to (i, u) while putting to zero (i, j).
Edge swaps: We chosse randomly two non-zero edges in $\mathbf{W}$, (i, j) and (u, v) with $i \neq j \neq u \neq v$, and swap their endpoints: $(W_{ij}, W_{uv}) \to (W_{iv}, W_{ju})$.
Weight category values: For each category $z_k \in \mathbf{z}$ we try to update the value $z'_k \to z_k$ where z'_k is the result of a random bisection search over the allowed values.
Weight category distribution: We execute the following moves in order, and each of them will be accecpted if they improve the DL:

1. *Merge*: Two categories z_s and z_t are removed and one category z_{new} is created. All the edges that were in the categories z_s and z_t will now have the value z_{new}, which is the result of a random bisection search over the allowed values that improves the DL. In this case the number of categories decrease by one.

2. *Split*: One category z_k is splitted into two categories z_s and z_t. The weights that were in the removed category are splitted into the new two, and the values z_s and z_t are searched via random bisection search. The number of categories increass by one.

3. *Merge-split*: The two steps above are apllied in successione, so that some weights are redistributed into two new categories which potentially improve the DL. The number of categories remains the same.

Every time a new weight is proposed, it needs to be quantized to a multiple of Δ before it is tested for the change of DL.

The results of this algorithm can be seen in Figure 14, and the reconstructed networks are shown in figure (15). As we can see, for all three networks the MDL regularization is capable of reconstructing the correct weights when sufficient data is available. The results are summarized in table 2. While $L1$ regularization with cross-validation gives

very low Jaccard similarity between the inferred and true networks (see Table 1), MDL regularization produces Jaccard similarities approaching 1 for all three networks. As expected, more complex networks such as Les Misérables require larger datasets to converge compared to simpler ones like Family. Another very important achievement is that the mean weight values are now correctly inferred: we can see in Table 2 that the mean and standard deviation of the inferred edges are significantly closer than the one we obtained via $L1$ regularization, confirming what we predicted theoretically i.e. the MDL principle with weight quantization no longer penalizes large weights in favor of smaller ones, or more precisely, it does so only minimally and negligibly in most cases, allowing the algorithm to accurately infer the true weight values. Similarly, this also applies to the

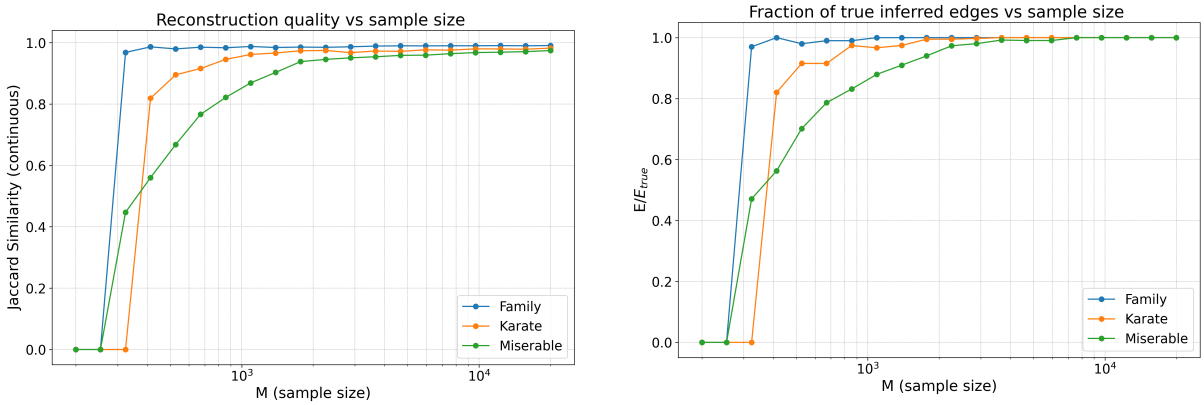


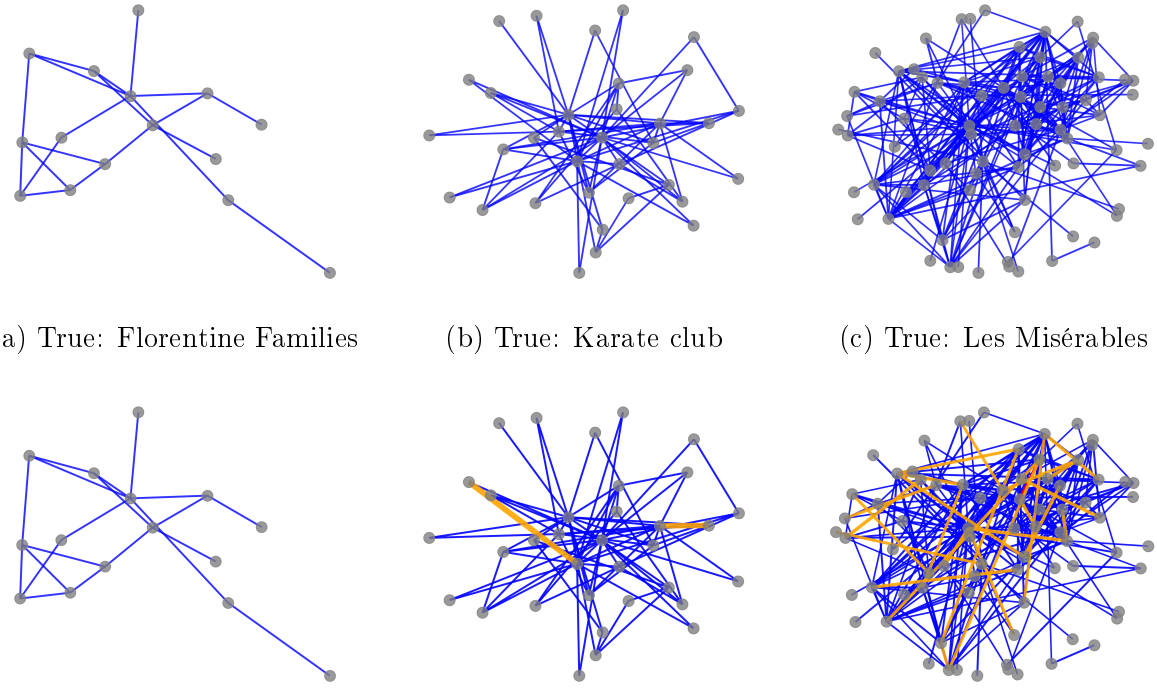
Figure 14: This figure shows that MDL regularization tends to infer the network correctly. Each point represents the average over five independent reconstructions.

number of inferred edges: the MDL principle, by penalizing unnecessary edges, tends to infer the correct number of edges, both in quantity and position within the network. Indeed, we can see that the reconstruction performance is consistently high: accuracy and precision are close to 1, indicating a very low false-positive rate, while the recall is also close to 1, meaning that most true edges are recovered. The only noticeable exception is the *Les Misérables* network in Table 2, whose recall is lower because the experiments were performed with a small sample size ($M = 1000$). When increasing to $M = 2000$, the recall improves, reaching about 0.93, and 0.97 when we use $M = 3000$.

Network	JS MDL	JS L1	mean±std	$\#\mathbf{z}$	E_{true}	TP, FP, FN	accuracy, precision, recall	t
Florentine Family	0.98	0.70	0.38±0.01	2	20	20, 0, 0	1.00, 1.00, 1.00	30s
Karate Club	0.96	0.64	0.22±0.01	9	78	76, 0, 2	0.99, 1.00, 0.97	40s
Les Misérables	0.86	0.55	0.16±0.02	13	254	228, 1, 26	0.99, 0.99, 0.89	1min 20s

Table 2: Values of the three reconstructed networks with MDL regularization with a sample of $M = 1000$. The values TP, FP, and FN, and consequently accuracy, precision and recall, refer to the binarized adjacency matrix of the inferred weight matrix, i.e., they indicate how many correct edges the algorithm has inferred, but they do not consider the weight magnitude. The column t represents the time required by the algorithm to reconstruct the inferred weight matrix. Note that these times are averaged over multiple reconstructions on a computer with an Intel(R) Core(TM) i5-14600KF (3.50 GHz) CPU.

Notably, the number of weight categories remains consistently small relative to the total number of edges, indicating that the algorithm appropriately penalizes insufficiently justified additions, thus maintaining $K \ll E$.



(a) True: Florentine Families (b) True: Karate club (c) True: Les Misérables

(d) MDL: Florentine Families (e) MDL: Karate club (f) MDL: Les Misérables

Figure 15: Visual representation of the three reconstructed networks with MDL regularization (bottom row) and the corresponding true networks (top row). Blue edges correspond to correctly inferred connections, red edges to incorrectly inferred connections, and orange edges to true connections that were missed by the reconstruction.

Another key advantage of this algorithm is that it can infer the network structure much faster than $L1$ regularization with cross-validation. Given the complexity and number of updates involved, the algorithm takes 5 to 10 times longer than a single $L1$ run, but it is significantly faster than performing a full cross-validation over the values of λ (see Table 1), as shown in Table 2.

As mentioned above, this is a non-parametric regularization in the sense that the number of parameters changes during the inference process in the most statistically justifiable way: the values chosen for the two hyperparameters Δ and λ should not significantly affect the inference outcome. In fact, as we can see in Figure 16, even with large variations in both hyperparameters, the performance of the MDL regularization remains remarkably stable. The maximum difference observed is between Jaccard similarities of $J_{\min} = 0.901$ and $J_{\max} = 0.926$, representing a variation of only 2%, despite hyperparameter ranges spanning $\lambda_{\min} = 0.03$ to $\lambda_{\max} = 0.25$ (a factor of $\lambda_{\max}/\lambda_{\min} \simeq 8$) and $\Delta_{\min} = 10^{-3}$ to $\Delta_{\max} = 9 \times 10^{-6}$ (a factor of $\Delta_{\max}/\Delta_{\min} \simeq 10^{-2}$).

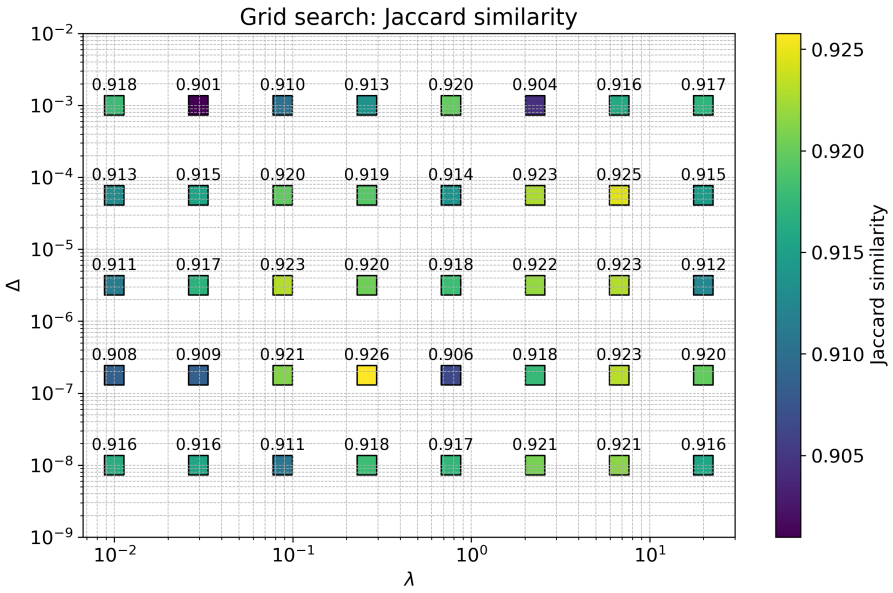


Figure 16: The figure shows how the algorithm performance changes very little even with order of magnitude variations in the hyperparameters. All reconstructions were performed on the Karate Club network with a sample size of $M = 2000$.

8 Conclusions

In this work, we have seen the importance of network science, and in particular the ability to reconstruct network structure from observed dynamics, in the modern world. There exists many network reconstruction methods but several of them can be unreliable in practice, especially when they lack a solid theoretical grounding such as the "threshold" approach. However, even approaches with a well defined theoretical basis may exhibit poor performance in realistic settings. A clear example discussed in this thesis is L1 regularization: although it is well motivated theoretically and should promotes sparsity, it can be computationally demanding, relies on weight shrinkage, and requires the level of sparsity as regularization strength to be specified. When this quantity is treated as a hyperparameter and selected from data, we observed that the reconstruction quality can degrade substantially: the model tends to include too many edges, accepting many spurious connections and leading to severe overfitting.

The solution proposed by Peixoto in his paper, and which we adopted and reproduced in this thesis, is MDL regularization with weight quantization. In this framework, adding an edge, changing a weight, or rewiring one or more edges is penalized in terms of description length. Since the MDL principle aims to minimize the total description length, a proposed modification of the inferred network is accepted only if it yields a sufficient increase in the data likelihood $P(\mathbf{X} \mid \mathbf{W})$ to compensate for the additional complexity. As a consequence, the model naturally promotes sparsity by accepting only edges and weight values that are statistically supported by the data, reducing overfitting. Moreover, weight quantization makes the penalty on large weights substantially weaker than in L1 regularization: the cost is primarily associated with the single weight categories, rather than penalizing each individual weight magnitude. As a result, the method relies less on weight shrinkage. Although the model now has two hyperparameters, the parameter λ, which controls the distribution of weight categories, and the quantization precision Δ,

we observed that performance is only weakly sensitive to their exact values over a large range.

Another major advantage of MDL regularization is computational: it only requires a single fit to infer the network, while L1 regularization generally requires cross-validation over the regularization strength, forcing the reconstruction to be repeated many times. For this reason, as shown in our experiments, MDL regularization can reconstruct the network in significantly less time than L1 regularization while achieving significantly better reconstruction quality.

References

[1] Philip W Anderson. "More is different: broken symmetry and the nature of the hierarchical structure of science." In: *Science* 177.4047 (1972), pp. 393–396.

[2] Marko Gosak et al. "Network science of biological systems at different scales: a review". In: *Physics of Life Reviews* 24 (2018), pp. 118–135. URL: https://www.matjazperc.com/publications/PhysLifeRev_24_118.pdf.

[3] Alessandro Rizzo, Biagio Pedalino, and Maurizio Porfiri. "A network model for Ebola spreading". In: *Journal of Theoretical Biology* 394 (2016), pp. 212–222. ISSN: 0022-5193. URL: https://www.sciencedirect.com/science/article/pii/S0022519316000485.

[4] Craig Schreiber and Kathleen M Carley. *Ineffective Organizational Practices at NASA: A Dynamic Network Analysis*. Technical Report. Carnegie Mellon University, 2005. URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2726789.

[5] Albert-László Barabási. "Network science". In: vol. 371. 1987. The Royal Society Publishing, 2013. Chap. 1, p. 20120375. URL: https://networksciencebook.com/chapter/1#vulnerability.

[6] Federico Battiston et al. "Networks beyond pairwise interactions: Structure and dynamics". In: *Physics reports* 874 (2020), pp. 1–92.

[7] Tiago P Peixoto. "Network reconstruction via the minimum description length principle". In: *Physical Review X* 15.1 (2025), p. 011065. URL: https://journals.aps.org/prx/abstract/10.1103/PhysRevX.15.011065.

[8] Gianmarco Zanardi. "Agent-based models of dynamical formation and emergence of memory on networks". PhD Thesis. Trento, Italy: University of Trento, 2025.

[9] Jonathan L Gross and Jay Yellen. *Handbook of graph theory*. CRC press, 2003.

[10] CBP Unitn. *Tutorial 10: Graph Theory and Network Reconstruction*. https://cbp-unitn.gitlab.io/QCB/tutorial10_graph. Accessed: 2025-12-17. 2025.

[11] Leto Peel, Tiago P Peixoto, and Manlio De Domenico. "Statistical inference links data and theory in network science". In: *Nature Communications* 13.1 (2022), p. 6794.

[12] Maurizio Loreti. "Teoria degli errori e fondamenti di statistica". In: *Decibel, Zanichelli* (2006), p. 121.

[13] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

[14] David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.