1 2 UNIVERSIDADE D COIMBRA 9 0

Diogo Robert de Oliveira Rente

Structure Identification of Networked Dynamical Systems under Partial Observability and Colored Noise via Machine Learning tools

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Professor Augusto Santos and co-advised by Professor Jorge Henriques and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE Ð
COIMBRA

Department of Informatics Engineering

Diogo Robert de Oliveira Rente

Structure Identification of Networked Dynamical Systems

under Partial Observability and Colored Noise via Machine Learning tools

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, advised by Prof. Augusto Santos and co-advised by Prof. Jorge Henriques and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

September of 2023

1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Diogo Robert de Oliveira Rente

Identificação da Estrutura de Sistemas Dinâmicos em Rede

sob Observabilidade Parcial e Ruído Colorido através de Ferramentas de Aprendizagem Computacional

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes, orientada pelo Professor Doutor Augusto Santos e co-orientada pelo Professor Doutor Jorge Henriques apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2023

Abstract

Recently, the study of Networked Dynamical Systems (NDS) has become increasingly popular due to its ability to model complex interactions within a wide range of fields, including social networks, biological systems, and engineering applications. Furthermore, understanding the topology of interactions underlying an NDS is essential for predicting and controlling their evolution. The underlying network structure plays a pivotal role in shaping the behavior, dynamics, and emergent properties of the system and can be effective in various different tasks, such as mitigation policies in pandemics, diagnosis of brain diseases, and many others. However, it is hardly the case that we have access to the topology of the network in a transparent manner. On the other hand, time series data is widely available across a variety of distinct domains.

The objective of this Dissertation is the implementation and assessment of a solution that consistently recovers the undirected network of interactions underlying a linear NDS from the available time series. We assume *partial observability*, where we only have access to the time series over a subset of nodes in the network – which is the case in large scale or complex systems. Further, we assume that the system is excited by *colored* noise, i.e., the noise signal is correlated across the nodes in the system. In addition, from this Dissertation rise two major contributions. First, we establish a novel regime of parameters where we guarantee feasibility of the structure identification problem, or *structural consistency* of a particular estimator. Second, we devise a structure identification algorithm that builds on a recently proposed feature based approach. Namely, to each pair of nodes, we assign a feature vector, and under the *feasible* regime of parameters, this set of features is linearly separable. We propose a novel set of features and train Feed Forward Neural Networks (FFNNs) to cluster the features and thus, classify whether a pair is connected or disconnected. The performance assessment shows very significant improvement in accuracy when inferring the network structure regarding other state-of-the-art estimators across distinct regimes of observability, noise correlation and distinct networks, including densely or sparsely connected, small world and real world ones.

Keywords

Structure Identification, Graph Learning, Undirected Graphs, Networked Dynamical Systems, Complex Systems, Colored Noise, Partial Observability, Machine Learning

Resumo

Recentemente, o estudo de Sistemas Dinâmicos em Rede (SDR) tornou-se cada vez mais popular devido à capacidade destes sistemas em modelar interações complexas dentro de uma ampla variedade de domínios, incluindo redes sociais, sistemas biológicos e aplicações de engenharia. Além disso, compreender a topologia de SDR é essencial para prever e controlar a sua evolução temporal. A estrutura subjacente desempenha um papel fundamental no comportamento, na dinâmica e nas propriedades emergentes destes sistemas e pode ser eficaz em várias tarefas, tais como políticas de mitigação em pandemias, diagnóstico de doenças cerebrais e muito mais. Porém, dificilmente temos acesso à topologia da rede de forma transparente. Por outro lado, os dados de séries temporais estão disponíveis numa variedade de domínios distintos.

O objetivo desta Dissertação é a implementação e avaliação de uma solução que recupere consistentemente a rede (não direccionada) de interações subjacente a um SDR linear, a partir das séries temporais disponíveis. Assumimos a *observabilidade parcial*, onde só temos acesso à série temporal em um subconjunto de nós na rede - o que é o caso em sistemas complexos ou de grande escala. Além disso, assumimos que o sistema é excitado por ruído *colorido*, ou seja, o sinal de ruído está correlacionado entre os nós do sistema. Desta Dissertação nascem duas contribuições importantes. Primeiramente, estabelecemos um novo regime de parâmetros onde garantimos a viabilidade do problema da recuperação da estrutura ou a consistência estrutural de um *estimator*. Em segundo, propomos um algoritmo de identificação da estrutura construído sobre uma nova abordagem que utiliza *features*. Nomeadamente, para cada par de nós, atribuímos um *feature-vector* e, sob o regime de parâmetros viáveis, este conjunto de *features* é linearmente separável. A solução proposta é uma FFNN construída sobre um novo conjunto de *features*. A avaliação do desempenho mostra um aumento significativo na performance do modelo a inferir a estrutura da rede, comparativamente a outros *estimators* utilizados na literatura, em diferentes regimes de observabilidade, de correlação do ruído e de redes, incluindo redes esparsas e densas e redes reais.

Palavras-Chave

Identificação da Estrutura, Aprendizagem de Grafos, Grafos não Direcionados, Sistemas Dinâmicos em Rede, Sistemas Complexos, Observabilidade Parcial, Aprendizagem Computacional

Acknowledgements

I am profoundly grateful to all those who have played an instrumental role during this academic stage of my life. This journey would not have been as meaningful or successful without their unwavering support, encouragement, and inspiration. First and foremost, I want to express my gratitude to my advisor Augusto Santos for his invaluable guidance, technical insights, and expertise, which have significantly shaped the direction and content of this Dissertation. I extend my gratefulness to Prof. Jorge Henriques for the research discussions and advising. A special thanks to Prof. José M. F. Moura for his technical feedback and knowledge, and for the incredibly opportunity to pursue this work under the Carnegie Mellon University.

I extend my heartfelt gratitude to my parents and family, whose endless love, sacrifices, and guidance have been the cornerstone of my academic pursuits. A special note of appreciation to my girlfriend, whose unwavering patience, understanding, and encouragement have been a guiding light during the intense periods of research and writing. To my cherished friends, your camaraderie and shared experiences have brought joy and balance to my life throughout this academic journey. I will forever remember the immeasurable amount of fun and good times we've had. Your presence has been a reminder that success is best celebrated when shared with those who matter most.

This work was funded in part by FCT under the project UIDP/00326/2020.

Contents

Acronyms

ANN Artificial Neural Network.

CNN Convolutional Neural Network.

CSD Cross-Spectral Density.

DFT Discrete Fourier Transform.

EEG Electroencephalography.

FC Functional Connectivity.

FFNN Feed Forward Neural Network.

FFT Fast Fourier Transform.

fMRI Functional Magnetic Resonance imaging.

GM Gaussian Mixture.

i.i.d. Independent and Identically Distributed.

MI Mutual Information.

MSC Magnitude Squared Coherence.

NDS Networked Dynamical Systems.

NN Neural Network.

PCA Principal Component Analysis.

PSD Power Spectral Density.

SC Structural Connectivity.

List of Figures

Chapter 1

Introduction

In this chapter, we introduce the *structure identification* problem to be addressed in the Dissertation. We highlight the underlying motivation, Section 1.1, and the proposed research, Section 1.2. In Section 1.3, we outline the remainder of the thesis.

The work produced within the scope of this Dissertation was, in part, submitted for publication [1].

1.1 Motivation and Goals

Discovering the underlying network structure that dictates the evolution of Networked Dynamical Systems (NDS) from data gathered at accessible nodes is a complex problem with widespread applications in various fields. The ultimate goal of solving such inferential issues is to reveal the interaction profile among the network nodes, as the topology of interactions of the system plays a crucial role in its behavior and evolution [2–6].

In this Dissertation, we address the problem of Graph Learning or Structure Identification of NDS. The graph entangles the connections between a set of nodes as well as their individual temporal evolution. In a real scenario, each node may abstract specific information about the system, e.g., concentration of genes or proteins in biological networks, predators and preys in food web networks or even individual users in online social networks. We focus on the regime of *partial observability*, where only some of the nodes are observed. The dynamical law of the NDS is, in general, influenced by *colored* noise, – i.e., noise exhibiting correlations across nodes or units in the system, – which creates an additional layer of spurious correlations between pairs of nodes. The primary goal in this setting is to consistently identify the underlying *undirected* network of links between the nodes comprising the network given the time series data at each node, which ultimately reflects the state evolution of each node and, consequently, the state evolution of the aggregate system over time. Understanding the underlying connectivity pattern is fundamental, as the network structure plays a critical role in the long-term qualitative behavior of these systems, since the nodes in the system may abstract communities of individuals in pandemics, probed regions in the brain, or companies in finance. Furthermore, the edges of the graph conform to the main avenues of interaction or information flow between the nodes, meaning that it is through these peer-to-peer interactions that the *global* state of the system (i.e., the state of all components or nodes) evolves over time. In addition to these benefits, understanding the structure of networked systems can help optimize resource allocation and decision-making, particularly in transportation networks and supply chains.

Even though the underlying connectivity pattern conveys relevant information about these complex systems, as it dictates how nodes influence and interact with each other, and can be further considered in downstream tasks – such as mitigation policies in pandemics [7, 8], diagnosis of diseases in the brain [9, 10], government interventions in economics, and many more – it is hardly the case that we have access (in a transparent manner) to the network. On the other hand, and in the most practical instances of the referred examples, the time series are readily available through the Functional Magnetic Resonance imaging (fMRI) or Electroencephalography (EEG) signals for the brain activity; the report of the number of infected individuals across distinct communities over time in a pandemics; evolution of stock prices in finance. Indeed, A great body of literature is dedicated to the problem of recovering the underlying graph connectivity from the time series reflecting the state evolution of the system [11–16]. All-in-all, Graph Learning aims at developing methods to identify consistently the underlying network structure from the available time series data.

Main goal. In this Dissertation, we study the problem of recovering the network consistently from the available time series data of only a subset of nodes streaming from a linear NDS excited by colored noise. First, we propose a novel feasibility regime on the noise statistical structure (namely, on its covariance matrix) that grants feasibility or well-posedness of the structure identification problem under the challenging regime considered. Then, we propose a feature based algorithm to recover the network structure.

1.2 Proposed Research

We focus on linear NDS, and we aim at creating Machine Learning tools, such as training Neural Network (NN) models, that can reliably extract the structure of the underlying *undirected* graph (i.e., the adjacency matrix). This will be done by analyzing the time series data of the different nodes in the system to track their state evolution over time. Fig. 1.1 presents an overview of the problem: To develop algorithms that can consistently output the underlying network linking a set of nodes, using the time series information of each one of the nodes. We aim at providing guarantees of consistency over partially observed systems, – i.e., when the time series of only a subset of nodes S, can be processed – influenced by *spatially* colored noise.

We remark that the bulk of the related network inference literature focus on scalar based methods: assigning an *affinity* score to each pair of nodes, computed from

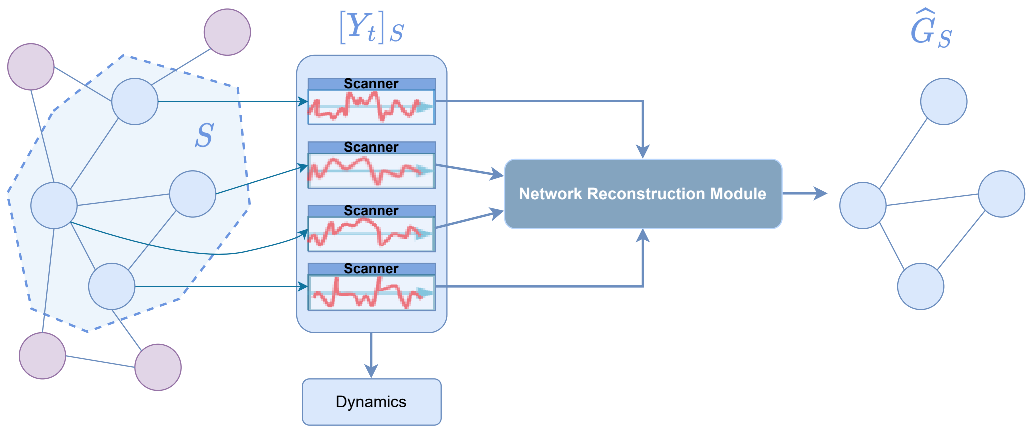


Figure 1.1: From the observed time series $[Y_t]_S$ of a subset of nodes S from the network, we aim at reconstructing $\hat{G}_S$, the observed subset of network.

the observed time series. Examples include, mutual information or conditional mutual information [15, 17, 18], Granger causality [14, 19, 20], correlation [21], Precision matrix or Graphical Lasso [22], etc. For the most part, pairs with higher estimated values are considered connected, otherwise, they are classified as disconnected. Contrarily to the standard approach, we focus on engineering features with good separability properties. These features are computed from the time series stemming from the underlying NDS. We build on the feature-based work developed by Machado et al. [23]. In this work, a feature vector is assigned to each pair of nodes. In other words, the underlying networked dynamical system is embedded in a high-dimensional Euclidean space as illustrated in Fig. 1.2. In order to recover the underlying network graph, we need to consistently cluster these features or find a separation hyperplane to separate the connected and the disconnected pairs. In particular, one needs to find (or learn) the correct surface (or manifold) in feature space that consistently partitions the set of features into features of disconnected pairs and features of connected pairs.

The proposed work extends the feature based causal inference work [23] that is tailored to partially observed linear stochastic NDS excited by diagonal noise (i.e., noise that is independent across nodes). We will compare our method with other popular estimators like Granger, that is tailored to linear NDS and withstand partial observability under diagonal noise [12–14, 20, 24, 25]. It is relevant to remark that these methods contrast with probabilistic graphical models with Independent and Identically Distributed (i.i.d.) samples [26–29].

For the most part the referred literature on linear NDS (including our baseline reference [23]) assumes *whiteness* on the noise input, i.e., independence of the noise across distinct nodes in the system, as it will be explained in Chapter 2. In this work, we pursue the feature based approach to extend the framework of Machado et al. [23] for a broader class of linear NDS, namely, linear NDS excited by colored noise. We propose a new feature-based approach to address the current challenge. Our solution is based on a novel set of features that is consistently linearly separable and stable under the colored noise regime considered,

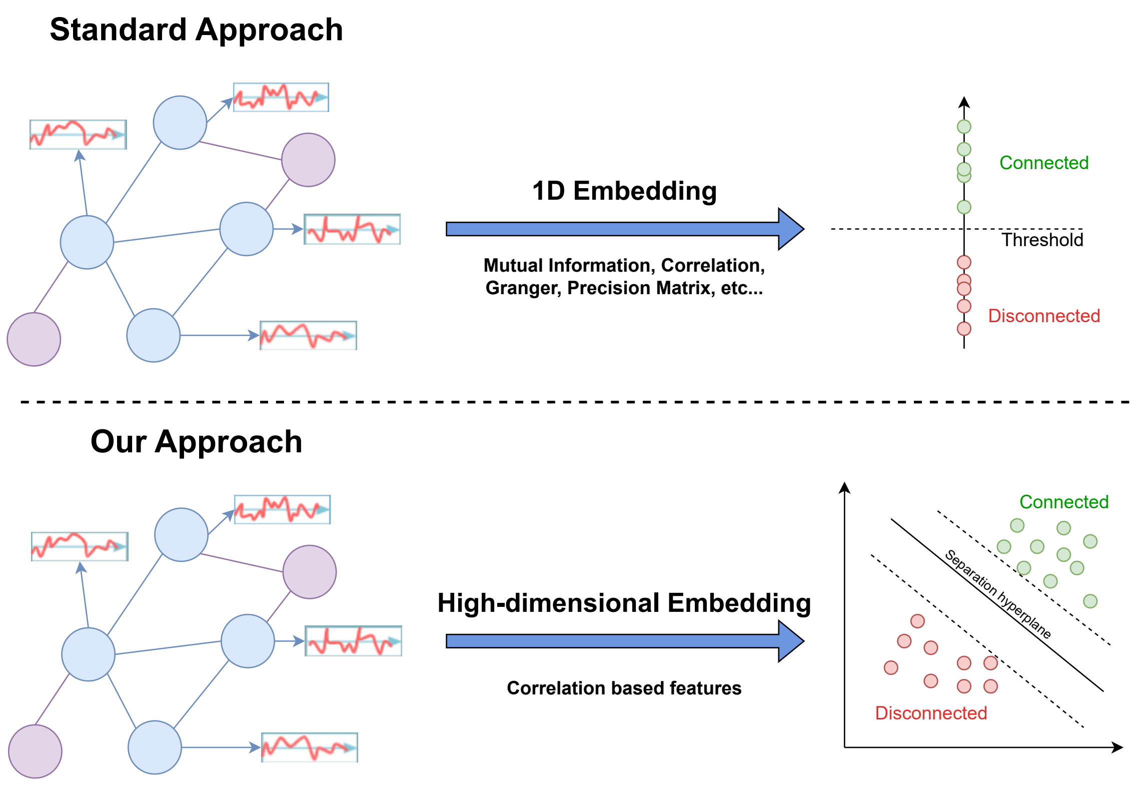

Standard Approach
1D Embedding
Mutual Information, Correlation, Granger, Precision Matrix, etc...
Connected
Threshold
Disconnected
Our Approach
High-dimensional Embedding
Correlation based features
Connected
Separation hyperplane
Disconnected

Figure 1.2: Comparison of the standard approach versus our approach, on the problem of recovering the graph from only the observed time series.

i.e., there exists a specific hyperplane that partitions the set of features into features of connected and disconnected pairs, respectively. Our methodology combines the engineering of new features and the application of a FFNN to achieve a competitive level of performance that is on par with the most popular estimators in the field, across multiple different regimes and on real world networks. This comparison is reported in Section 5.1.

1.3 Report Outline

Below, we outline the content and structure of this Dissertation.

- **Chapter 2** – describes the main theoretical background components of the research that was carried out, the literature review and the problem formulation. We discuss linear NDS; Artificial Neural Network (ANN); FFNN and CNN, subsections of ANN; Graph Learning of NDS.
- **Chapter 3** – entails our technical contribution, namely, our novel technical results and theorems along with the analytical proofs.
- **Chapter 4** – reveals the process of the data generation and the training of the neural network models.

- **Chapter 5** –presents some of the various experiments pursued and the numerical simulations and its results, highlighting the performance of our method compared with state-of-the-art estimators.
- **Chapter 6** – summarizes the research scope and the conclusions from the work developed during this Dissertation. It also highlights the two major contributions derived from the work and identifies possible directions of future work.

Chapter 2

Background and Related Work

This chapter entails some of the essential background concepts in the context of the Dissertation. Section 2.1 discusses the main technical ground for the Dissertation, namely, NDS. Section 2.2 formulates the problem and defines the main equations and notations. The related work is presented in Section 2.3, where we explain some of the most popular solutions to the formulated problem. Section 2.4 briefly discusses Artificial Neural Networks, which will be used in the problem of structure identification.

2.1 Networked Dynamical Systems

An NDS is a system comprised of a set of interconnected nodes that interact with each other over time. The state of the system, i.e., the collective state of the nodes, evolves over time due to these peer-to-peer interactions. These nodes abstract physical objects, such as sensors or actuators in a control system; neurons (or collections of neurons) in the brain; or communities of individuals in a social network. The interactions between the nodes can be direct or indirect, and they can be governed by explicit rules or, at an aggregate level, the dynamical rules can emerge from the collective behavior of the units. NDS can exhibit complex behavior, such as synchronization, oscillations, and chaos, depending on the structure of the network and the nature of the interactions between the nodes [30]; or they may have simpler long-term behavior such as attaining an equilibrium state. The latter is, for instance, common in pandemic models [31, 32].

One key aspect of NDS is their ability to process and transmit information. Information can flow through the system in the form of signals, messages, or activity patterns, and it can be transmitted, transformed, and stored by the nodes in the network. These render NDS useful to model various applications, such as communication networks, distributed control systems [2, 33]; or natural phenomena such as the evolution of pandemics [34, 35], brain activity [36, 37], and social networks. In these cases, the nodes in the network represent the entities in the natural system, and the edges between the nodes capture how the state of these components influence and are influenced by each other. By studying the behavior of the NDS, we can gain insight into the behavior of the natural system and identify patterns and trends that may not be apparent from studying the individual nodes in isolation.

In a more concrete sense, the state of the nodes evolves according to their peer-to-peer interactions constrained by a support network of edges. In particular, the state of a node i is only immediately affected by the state of nodes that are directly connected to i. This causal network is captured by a graph, often a latent structure underlying these systems. It is the goal of Graph Learning to infer this underlying network connectivity from observing the evolution of the graph nodes, i.e., the time series reflecting the state evolution of the system. Problems of this type arise in many domains where knowledge of the underlying topology linking the agents is critical for better inference and control mechanisms, such as: (i) **pandemics**, where the profile of interactions among communities of individuals critically impacts the overall lifetime of a strain of virus [7], (ii) **social networks**, where specific connectivity patterns foster the emergence of inconsistent beliefs or fake news among the involved communities, and also (iii) **brain activity**, where recent evidence shows that the structure of the underlying Functional Connectivity (FC) Matrix conveys relevant signatures about cognitive disorders [9, 10].

2.2 Problem Formulation

The problem can be formulated as follows. Data streams, namely time series data, originating from a network (or a sub network, for the case of partial-observability) are collected, and the goal is to estimate the unknown topology that connects the nodes of this network from the collected data.

Throughout this Dissertation we adopt the following mathematical notation. Given a vector $\mathbf{y} \in \mathbb{R}^M$, $[\mathbf{y}]_S = \left[\mathbf{y}_{m_1}(n)\ \mathbf{y}_{m_2}(n)\ \ldots\ \mathbf{y}_{m_{|S|}}(n)\right]^\top$ is the subvector obtained from $\mathbf{y}$ and indexed by S (subset of observed nodes), with $S = \left\{m_1, m_2, \ldots, m_{|S|}\right\} \subset \{1, 2, \ldots, M\}$ being a subset of indexes with $m_1 < m_2 < \ldots < m_{|S|}$ and $|S| \leq M$. We adopt a similar notation for matrices, namely, given $A \in \mathbb{R}^{M \times M}$, the matrix $A_S \in \mathbb{R}^{|S| \times |S|}$ or $[A]_S \in \mathbb{R}^{|S| \times |S|}$ is defined as the submatrix whose ij^{th} entry is $A_{m_i m_j}$; $\mathsf{Supp}\,(A)$ is the support of the matrix A, i.e., $[\mathsf{Supp}\,(A)]_{ij} = \mathbf{1}_{\left\{A_{ij} \neq 0\right\}}$.

Discrete-time stochastic Linear NDS can be described by the following dynamical law:

$$\mathbf{y}(n+1) = A\mathbf{y}(n) + \mathbf{x}(n+1), \tag{2.1}$$

where $\mathbf{y}(n) = [y_1(n)\ y_2(n)\ \ldots\ y_M(n)]^\top \in \mathbb{R}^M$ represents the state-vector of the M-dimensional NDS at time n that collects the states $y_i(n)$ of each node i at time n; $\mathbf{x}(n) \sim \mathcal{N}\,(0, \Sigma)$ represents the excitation noise associated with the M nodes of the system with covariance matrix $\Sigma \in \mathbb{S}_{++}$, and assumed independent across time n; $A \in \mathbb{R}_{+}^{M\times M}$ refers to the non-negative interaction matrix whose support represents the underlying structure connecting the nodes. A is assumed to be symmetric since we focus on the problem of Graph Learning. Furthermore, the dynamical system is assumed to be stable, i.e., $\rho(A) < 1$, where $\rho(A)$ stands for the spectral radius of A.

We can zoom in further into the equation, to the level of the state of node i:

$$\mathbf{y_i}(n+1) = \sum_{j=1}^{M} a_{ij}\mathbf{y_j}(n) + \mathbf{x_i}(n+1), \tag{2.2}$$

where $\mathbf{y_i}$ represents the time series vector associated with node i and a_{ij} represents how node i is affected by node j. This *local* characterization highlights the network of causal relationships among the nodes in the system: the state of the node j directly affects the state of the node i if and only if $a_{ij} \neq 0$, as shown in Fig. 2.1.

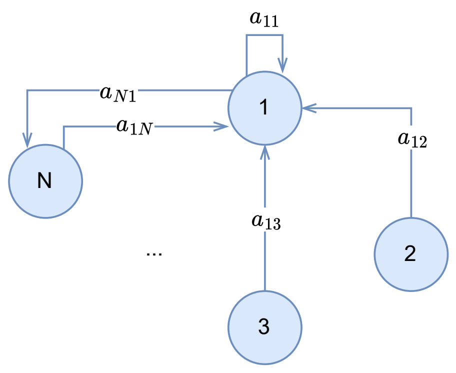


Figure 2.1: Example of the main avenues of interaction affecting node 1 in the network and characterized by an interaction matrix A.

The problem of Graph Learning or Structure Identification can be then more formally stated: Given the time series of each node as input, can we consistently identify the underlying *undirected* network of interactions characterized by the support of the interaction matrix A? That is, can we find a map that consistently outputs the underlying graph structure G (given by the support of the matrix A), given the node-level time series as

$$\mathbf{y}(0), \mathbf{y}(1), ..., \mathbf{y}(N) \xrightarrow{Estimator} \mathsf{Graph}(N) = \mathsf{Supp}(A), \tag{2.3}$$

where $G = \mathsf{Supp}(A)$ is the adjacency matrix, obtained from the interaction matrix A, with entries equal to:

$$G_{ij} = \begin{cases} 1, & \text{if } a_{ij} \neq 0 \\ 0, & \text{if } a_{ij} = 0 \end{cases}, \tag{2.4}$$

where $a_{ij} = a_{ji}$, since we assume undirectness of the network, so consequently, symmetry of the adjacency matrix. The goal of graph recovery under full observability is to consistently infer the support of the interaction matrix A from the time series data. On the other hand, under partial observability, the goal is to recover the support of the sub matrix A_S, that represents the interaction matrix A restricted to a set of observable nodes $S := \{m_1, m_2, ..., m_{|S|}\}$, from the time series of only the observable nodes $\{[y(n)]_S\}_{n=1}^{N}$.

For most standard graphical models, interactions across network nodes are described through a multivariate distribution that characterizes a collection of dependent random variables defined on the nodes [38]. In this graphical model framework, samples are often assumed i.i.d. as opposed to the NDS setting where samples exhibit dependence across time. In other words, over graphical models, the data samples do not arise from a dynamical process governing the time evolution of the outputs of the nodes. In contrast, we will be dealing with NDS, where signals evolve at the nodes and are affected by the evolution of the signals at neighboring nodes as well.

Let us define some important characters for structure estimation in NDS. We define

$$R_0(n) \triangleq E\left[y(n)y(n)^T\right], \tag{2.5}$$

as the correlation (or covariance) matrix associated with the process (or time series) $(\mathbf{y}(n))_{n\in\mathbb{N}}$. Further, we can define the k^{th}-lag correlation matrix as

$$R_k(n) \triangleq E\left[y(n+k)y(n)^T\right]. \tag{2.6}$$

We also define their empirical counterparts, namely

$$\widehat{R}_k(n) = \frac{1}{N-k}\sum_{n=1}^{N-k} y(n+k)y(n)^T. \tag{2.7}$$

The empirical covariance matrices, represented by Equation (2.7), are the ones we can compute from the observed time series in order to approximate the *exact* ones in Equation (2.6). In fact, under certain conditions on the excitation noise $\mathbf{x}(n)$ and on the interaction matrix A, the empirical covariance matrices (or lag-moments) converge (as the number of samples grows to infinite) to the exact covariance matrices [39].

We also define a matrix-valued estimator as any map that takes time series data as input and produces a matrix as output.

$$\begin{array}{cccc} F^{(n)}: & \mathbb{R}^{|S|\times n} & \longrightarrow & \mathbb{R}^{|S|\times|S|} \\ & \{[\mathbf{y}(\ell)]_S\}_{\ell=0}^{n-1} & \longmapsto & \mathcal{F}^{(n)} \end{array}, \tag{2.8}$$

for any given $n \in \mathbb{N}$, where n represents the number of samples of the time series data. The idea is that the ij^{th} entry of the output matrix $\mathcal{F}^{(n)}$ estimates the linking strength of the connection between node i to j from the observed time series. That is, $\mathcal{F}^{(n)}$ is a matrix representing the estimated coupling strength between the pairs of nodes in the system. The estimator $\mathcal{F}^{(n)}$ is *structurally consistent* whenever the estimated strength $\mathcal{F}_{ij}^{(n)}$ of any connected pair (i, j) lies above the estimated strength of any disconnected pair. By applying appropriate thresholding or utilizing clustering algorithms, the underlying network structure can be recovered. The following definition, present in the work of Machado et al. [23], formalizes the previous statement.

Definition 1 (*structural consistency*)**.** A matrix-valued estimator $\mathcal{F}^{(n)}$ is structurally consistent with high probability, whenever there exists a threshold τ so that,

$$\mathbb{P}\left(\mathcal{F}_{ij}^{(n)} > \tau\right) \overset{n\to\infty}{\longrightarrow} 1 \iff i \to j, \tag{2.9}$$

i.e., i links to j if and only if the ijth entry of the estimator matrix $\mathcal{F}^{(n)}$ lies above the threshold τ, provided that there is a large enough number of samples n.

2.3 Related Work

2.3.1 Granger Estimator

The Granger exact matrix-valued estimator, a terminology that arises in the context of Granger Causality, given by $R_1 R_0^{-1}$, conforms to a consistent estimator for the underlying interaction matrix A. In a nutshell, Granger causality refers to the relationships between time series. More concretely, in whether one time series is useful in forecasting another time series. The Granger estimator combines in a suitable way information contained in the correlation matrix, R_0, and in the one-lag correlation matrix, R_1. The analytical proof of the consistency of this estimator can be derived as follows.

Proof. From the linear dynamical law (2.1), we can multiply both members of the equation by $\mathbf{y}(n)^\top$ as follows

$$\mathbf{y}(n+1)\mathbf{y}(n)^\top = (Ay(n) + x(n+1))y(n)^\top,$$

and via applying expectation $\mathbb{E}[\cdot]$ to both sides, we have

$$E\left[\mathbf{y}(n+1)\mathbf{y}(n)^\top\right] = E\left[(Ay(n) + x(n+1))y(n)^\top\right].$$

From equation (2.7), we have that $R_1(n) \overset{\Delta}{=} E\left[y(n+1)y(n)^T\right]$, and so

$$R_1(n) = E\left[\left(Ay(n) + x(n+1)\right)y(n)^T\right],$$

and therefore,

$$R_1(n) = AE\left[y(n)y(n)^T\right] + E\left[x(n+1)y(n)^T\right], \tag{2.10}$$

where $E\left[x(n+1)y(n)^T\right] = 0$, since $x(n)$ and $y(n)$ are assumed independent and $\mathbf{x}(n)$ is a zero mean process. Furthermore, we know that $R_0(n) \overset{\Delta}{=} E\left[y(n)y(n)^T\right]$, defined in equation (2.5), thus we can simplify equation (2.10) down to

$$R_1(n) = AR_0(n). \tag{2.11}$$

Which ultimately yields

$$A = R_1 R_0^{-1}, \tag{2.12}$$

where A represents the interaction matrix. This identity offers a scheme to estimate the underlying interaction matrix A from the node-level time series, namely, via estimating the empirical one-lag and zero-lag covariance matrices.

It is important to remark that the above argument grants consistency of the estimation under full observability, that is, whenever the whole vector of states $\mathbf{y}(n) = (\mathbf{y}_1(n), \mathbf{y}_2(n), \ldots, \mathbf{y}_M(n))$ is observed over time. While the full observability setting is widely explored in the literature, in reality, we can rarely observe the state of all nodes over time in a large-scale NDS. In this setting, only a sub-vector $\mathbf{y}(n) = (\mathbf{y}_1(n), \ldots, \mathbf{y}_S(n))$, with $S < M$, is feasibly observable over time. In this partial observability framework, the Granger estimator looses the consistency to precisely estimate the interaction matrix A as in this case we have

$$A_S = \left[R_1 R_0^{-1}\right]_S \neq [R_1]_S \left([R_0]_S\right)^{-1} =: \widehat{A}_S,$$

where $\widehat{A}_S$ represents the Granger estimator applied to the observed time series, i.e., ignoring the latent part of the system. Nevertheless, under certain conditions on the underlying network of interactions, this estimator is structurally consistent [12, 13], i.e.,

$$\min_{ij\,:\,A_{ij}\neq 0} \left[\widehat{A}_S\right]_{ij} > \max_{ij\,:\,A_{ij}=0} \left[\widehat{A}_S\right]_{ij}.$$

In other words, the smallest entry of the Granger estimator (under partial observability) $\widehat{A}_S$ across connected pairs is higher than the greatest entry of $\widehat{A}_S$ across disconnected pairs. This means that the network structure can be recovered from the Granger estimator (even under partial observability) via properly thresholding the entries of $\widehat{A}_S$.

Indeed, we are mostly interested in the framework of Graph Learning under partial observability, namely, in developing methods with technical guarantees of structural consistency.

2.3.2 One-Lag Estimator

The one-lag estimator is a simple estimator from the one-lag correlation matrix. As we previously explored in Equation (2.10)

$$R_1(n) = AR_0(n),$$

R_1 is given by R_0, the correlation matrix, and A, the interaction matrix. From this, we can expand this series as follows:

$$R_1 = AR_0 = \sigma^2(A + A^3 + A^5 + ...), \tag{2.13}$$

where σ^2 represents the variance of the input process. The proof for Eq. (2.13) is presented below. Applying Eq. (2.13) only to the subset of observed nodes S yields:

$$[R_1]_S = \sigma^2 \left(A_S + \left[A^3\right]_S + \left[A^5\right]_S + ...\right).$$

It is evident that this estimator depends on the variance of the input process σ^2 contrarily to the Granger estimator, as in the Granger estimator the one-lag correlation matrix multiplies the precision matrix, the inverse of the correlation matrix, the effect of σ^2 disappears. Although this is a disadvantage against the Granger estimator, the latter seems to be more sensitive to the level of observability, whereas it tends to be more performing when one approaches the regime of full observability [25], since the calculation of the inverse of the correlation matrix requires observation of all entries of the matrix contrarily to just calculating the correlation matrix, which only requires the observation of pairwise correlations. More precisely, in order to calculate the ij^{th} entry of the correlation matrix, it is only required access to the time series data of nodes i and j. The same is not applicable to the inverse of the correlation matrix.

Proof. First, from the dynamical law (2.1) and assuming that $y(0) = x(0)$ for simplification, we have the first instant vector:

$$\mathbf{y}(1) = A\mathbf{y}(0) + \mathbf{x}(1) = A\mathbf{x}(0) + \mathbf{x}(1),$$

and the second instant vector:

$$\begin{aligned}\mathbf{y}(2) &= A\mathbf{y}(1) + \mathbf{x}(2) = A(A\mathbf{y}(0) + \mathbf{x}(1)) + x(2) \\ &= A(A\mathbf{x}(0) + \mathbf{x}(1)) + \mathbf{x}(2) = A^2\mathbf{x}(0) + A\mathbf{x}(1) + \mathbf{x}(2).\end{aligned}$$

It is evident that we can generalize for the n instant vector as follows

$$\mathbf{y}(n) = \sum_{i=0}^{n} A^{n-i}\mathbf{x}(i), \tag{2.14}$$

where n denotes the number of total samples observed.

From the correlation matrix, defined in equation (2.5), and equation (2.14), and assuming A is symmetric, we have

$$\begin{aligned}R_0(n) &\overset{\Delta}{=} \mathbb{E}\left[\mathbf{y}(n)\mathbf{y}(n)^T\right] \\ &= \textstyle\sum_{i=0}^{n}\sum_{j=0}^{n} \mathbb{E}\left[A^{n-i}\mathbf{x}(i)\mathbf{x}(j)^T A^{n-j}\right] \\ &= \textstyle\sum_{i=0}^{n}\sum_{j=0}^{n} A^{n-i}\mathbb{E}\left[\mathbf{x}(i)\mathbf{x}(j)^T\right] A^{n-j} \\ &\overset{(a)}{=} \textstyle\sum_{i=0}^{n} A^{n-i}\mathbb{E}\left[\mathbf{x}(i)\mathbf{x}(i)^T\right] A^{n-i} \\ &= \textstyle\sum_{i=0}^{n} A^{n-i}\Sigma_x A^{n-i},\end{aligned}$$

where the identity (a) holds in view of the temporal independence and zero mean nature of the noise process $(\mathbf{x}(n))$, i.e., $\mathbb{E}\left[\mathbf{x}(i)\mathbf{x}^\top(j)\right] = \mathbb{E}\left[\mathbf{x}(i)\right]\mathbb{E}\left[\mathbf{x}(i)\right]^\top = 0$ for any $i \neq j$, and Σ_x is the covariance matrix of the noise process. Therefore, if we assume diagonal noise, i.e., $\Sigma_x = \sigma^2 I$, we have that the limiting correlation matrix is given by

$$\begin{aligned}
R_0 &= \lim_{n\to\infty} A^N\Sigma_x A^N + A^{N-1}\Sigma_x A^{N-1} + ... + A\Sigma_x A + \Sigma_x \\
&= \lim_{n\to\infty} A^N\sigma_x^2 I A^N + A^{N-1}\sigma_x^2 I A^{N-1} + ... + A\sigma_x^2 I A + \sigma_x^2 I \\
&= \lim_{n\to\infty} \sigma_x^2 \left(A^N A^N + A^{N-1}A^{N-1} + ... + AA + I\right) \\
&= \lim_{n\to\infty} \sigma_x^2 \left(A^{2N} + A^{2(N-1)} + ... + A^2 + I\right) \\
&= \sigma_x^2 \left(I + A^2 + A^4 + ...\right).
\end{aligned} \tag{2.15}$$

Taking equation (2.11) and the previous expansion of R_0, we have

$$\begin{aligned}
R_1 &= AR_0 \\
R_1 &= A\sigma_x^2 \left(I + A^2 + A^4 + ...\right) \\
R_1 &= \sigma_x^2 \left(A + A^3 + A^5 + ...\right),
\end{aligned}$$

and thus, we prove equation (2.13), under the assumption that the noise is *white*.

2.3.3 Estimator R_1 - R_3

Chen et al. [40] present a recently proposed matrix-valued estimator for Graph Learning linear NDS under partial observability, given by $R_1 - R_3$. This estimator assumes that the input noise is diagonal. Namely, its covariance matrix is given by $\Sigma_x = \sigma^2 I$, where I is the identity matrix.

Proof. Recall equation (2.6)

$$R_k(n) = \mathbb{E}\left[\mathbf{y}(n+k)\mathbf{y}(n)\right],$$

as the k^{th}-lag covariance matrix of the process $(\mathbf{y}(n))_{n\in\mathbb{N}}$ and remark that

$$R_k(n) = A^k R_0(n). \tag{2.16}$$

Let $\mathbf{y}(0) = \mathbf{x}(0)$, we have shown that

$$\mathbf{y}(n) = \sum_{i=0}^{N} A^{N-i}\mathbf{x}(i),$$

which further implied

$$R_0 = \mathbb{E}\left[\mathbf{y}(n)\mathbf{y}(n)^\top\right] = \sum_{i=0}^{N} A^i \Sigma_x A^i,$$

where Σ_x is the covariance matrix of the process $(\mathbf{x}(n))$. In other words,

$$R_0 := \lim_{n\to\infty} R_0(n) = \sum_{i=0}^{\infty} A^i \Sigma A^i. \tag{2.17}$$

In the work of Chen et al. [40], the authors considered the following matrix-estimator: $\widehat{A}(n) = \widehat{R}_1(n) - \widehat{R}_3(n)$, where $\widehat{R}_1$ and $\widehat{R}_3$ are the empirical 1-lag and 3-lag covariance matrices. The authors also assumed that the covariance matrix of the process $(\mathbf{x}(n))$ was the identity (up to a multiplicative positive scalar), i.e., $\Sigma_x = \sigma_x^2 I$. This estimator was shown to be a consistent estimator and the main idea is the following. The exact covariance matrices obey

$$R_1 - R_3 = AR_0 - A^3R_0 = \sigma_x^2 \left(A \left(\sum_{i=0}^{\infty} A^{2i} \right) - A^3 \left(\sum_{i=0}^{\infty} A^{2i} \right) \right) = \sigma_x^2 A. \quad (2.18)$$

This is easily proven. From equation (2.16), and recalling the R_0 expansion in (2.15), we have

$$\begin{cases} R_1 &= AR_0 = \sigma_x^2 A \left(I + A^2 + A^4 + ... \right) = \sigma_x^2 \left(A + A^3 + A^5 + ... \right) \\ R_3 &= A^3 R_0 = \sigma_x^2 A^3 \left(I + A^2 + A^4 + ... \right) = \sigma_x^2 \left(A^3 + A^5 + ... \right), \end{cases}$$

which leads to

$$\begin{aligned} R_1 - R_3 &= \sigma_x^2 \left(A + A^3 + A^5 + ... \right) - \sigma_x^2 \left(A^3 + A^5 + ... \right) \\ R_1 - R_3 &= \sigma_x^2 \left(A + A^3 - A^3 + A^5 - A^5 + ... \right) \\ R_1 - R_3 &= \sigma_x^2 A. \end{aligned}$$

Since this estimator is given by the linear combination of covariance matrices, this estimator can consistently recover the underlying interaction matrix up to a multiplicative term also under partial observability

$$[R_1]_{\mathcal{S}} - [R_3]_{\mathcal{S}} = \sigma_x^2 [A]_{\mathcal{S}}.$$

When the covariance matrix is not a multiple of the identity – e.g., the process $(\mathbf{x}(n))$ is not spatially independent or white, – then, the last identity in equation (2.18) no longer holds as we loose the *telescopic* property of the involved series and rather, we have

$$R_1 - R_3 = AR_0 - A^3R_0 = A \left(\sum_{i=0}^{\infty} A^i \Sigma A^i \right) - A^3 \left(\sum_{i=0}^{\infty} A^i \Sigma A^i \right).$$

2.3.4 Precision Matrix

The precision matrix represents the inverse of the covariance matrix and encodes the partial correlations between the nodes of the graph. In the context of learning the connectivity structure of a graph from time series data, the precision matrix serves as a powerful tool for estimating the relationships between different nodes. The precision matrix is defined as

$$[R_0]^{-1} \triangleq \mathbb{E} \left[\mathbf{y}(n)\mathbf{y}(n)^\top \right]^{-1}.$$

Proof. Remarking Eq. (2.15), we can define R_0 as

$$R_0 = \sigma_x^2(I + A^2 + A^4 + \ldots),$$

which can be writen as (assuming $0 < \rho < 1$, where ρ is the spectral radius of A)

$$R_0 = \sigma_x^2(I - A^2)^{-1},$$

because, when $0 < \rho < 1$, the series $(I + A^2 + A^4 + \ldots)$ converges and can be computed as the sum of a geometric series.

Now, remark that the precision matrix is R_0^{-1}, so we have

$$R_0^{-1} = \frac{1}{\sigma_x^2}(I - A^2). \tag{2.19}$$

which conveys relevant information about the underlying support graph of A through A^2.

2.3.5 Feature-Vector approach

Machado et al. [23] proposed a novel approach based on a feature-vector to infer the topology of the graph. Define the set of features $\mathcal{F}^M(n) = \left\{\mathcal{F}_{ij}^M(n)\right\}_{ij}$

$$\mathcal{F}_{ij}^M(n) := \left(\left[\widehat{R}_{-\frac{M}{2}+1}(n)\right]_{ij}, \ldots, \left[\widehat{R}_0(n)\right]_{ij}, \ldots, \left[\widehat{R}_{\frac{M}{2}}(n)\right]_{ij}\right),$$

built on the observed part of the lag moments. The feature vector has M different lag moments, with negative and positive lags. This feature-vector is linearly separable as long as $\frac{M}{2} \geq 3$, or more concretely, as long as the R_1 and the R_3 lag moments are present in the feature-vector.

The feature-vector is then normalized by the maximum value

$$\overline{\mathcal{F}}_{ij}(n) = \frac{\mathcal{F}_{ij}(n)}{\max\left(\mathcal{F}_{ij}(n)\right)}, \tag{2.20}$$

to ensure generalization and robustness across structurally distinct graphs. The authors set the number of features to 200, meaning that the set of features $\mathcal{F}_{ij}(n)$ consists of $\widehat{R}_k$, with $k \in \{-99, \ldots, 100\}$.

From these features, a CNN is trained iteratively from each feature-vector associated with a pair of nodes, meaning the input to the CNN is a vector of 200 features for each pair. Each pair is associated with a target value (1 if the pair is connected and 2 if disconnected). The performance of this approach is guaranteed by **Theorem 1**, defined in the work of Machado et al. [23], since $\mathcal{F}_{ij}(n)$ contains the 1-lag and the 3-lag correlation matrix. In view of **Remark 2**, defined in the work of Machado et al. [23], this approach has the advantage of being pairwise, meaning that to infer the connectivity between node i and node j, we only need the time series of i and j. This *locality* property is not present in most estimators. For example, to reconstruct the ij^{th} entry of the Precision Matrix (R_0^{-1}), the whole correlation matrix (R_0), or most of it, must be available.

2.3.6 Recovery of Polytrees

An approach for recovering the structure of the network of dynamical polytrees with unobserved nodes, meaning *partial observability*, is proposed by Materassi and Salapaka [41]. This approach restricts the topology of the graph to be a polytree and acyclic, and assumes the knowledge of the cross-spectral densities of the observed processes $\phi_{x_i x_j}(e^{iw})$, which are critical assumptions that we do not hold. They use the Log-Coherence Distance to measure the distance between nodes x_i and x_j. The proposed algorithm is a generalization of the Recursive Regrouping Algorithm, an algorithm that consistently reconstructs minimal rooted trees, to the case of polytrees of dynamical systems.

Using this algorithm, the recovery of the network is guaranteed under some conditions and assumptions on the degrees of the nodes, the structure of the graphs and the knowledge of the spectral densities between signals. Namely, the method offers technical guarantees over polytrees, but the performance degrades for denser networks. Furthermore, the technical guarantee proposed does not hold when the system is excited by *colored* noise, as they assume that the noise signal acting on each node is unrelated to other nodes.

2.3.7 Wiener Filter approach

In the work of Materassi and Salapaka [42], the authors propose to infer the topology of self-kin networks using the Wiener Filter. Examples of self-kin networks are rooted trees and ring topologies. The authors provide guarantees of exact reconstruction of the graph if the topology of the original graph is described by a self-kin network. They also provide a more general guarantee, for the case where the network is not self-kin, where the developed procedure reconstructs the topology of the smallest self-kin network embraced in the true network. However, this approach assumes *full observability*, and the Granger estimator, described in Section 2.3.1 already provides technical guarantees of exact recovery of a broader spectrum of networks for the *full observability* regime.

2.3.8 Inverse of the Power Spectral Density Matrix (IPSDM)

Veedu and Salapaka [43] and Veedu et al. [44] consider linear dynamic influence models (LDIMs), defined in the frequency domain, with correlated noise and develop a Wiener filter based approach tailored for sparse networks - leveraging on a sparse plus low rank decomposition method. These works make two important limiting assumptions on the noise correlation: (i) it results from affine interactions and (ii) the excitation noise is assumed independent across latent nodes. In the work proposed by Veedu and Salapaka [43], it is also assumed that the knowledge of the correlation graph is known and all that the nodes were excited by uncorrelated noise.

Veedu et al. [44] propose a new method for rebuilding the structure of linear NDS that have latent nodes, which relies on the breaking down of the inverse of the power spectral density matrix (IPSDM) that is derived from the observed nodes into a sum of sparse and low-rank matrices. This approach allows the graph to have directed loops and bidirected edges. The authors have outlined conditions and techniques for breaking down the IPSDM into sparse and low-rank components. The sparse component produces the moral graph linked to the observed nodes. The authors also assumed the knowledge of the perfect IPSDM in order to reconstruct the topology of the network in the performed simulations and results.

2.4 Artificial Neural Networks

In this section, we present an introduction to ANNs, which are outstandingly efficient models for learning complicated data patterns.

2.4.1 Definition

ANNs, or simply NNs, describe a subset of machine learning algorithms designed to acquire their own knowledge of a specific problem by extracting functional patterns from data and learning by analyzing examples (training data). NNs are function approximators or estimators, mapping inputs to outputs, composed of multiple intertwined units (neurons). This architecture is inspired by the human brain, mimicking how biological neurons interact (i) neurons receive signals from the dendrites (ii) the signal is processed, and an output signal might be sent through the axons. Similarly, an ANN takes inputs from the input layer, weighs them individually, and passes them through an activation function in order to produce an output.

Each artificial neuron comprises inputs, weights, activation functions, and an output.

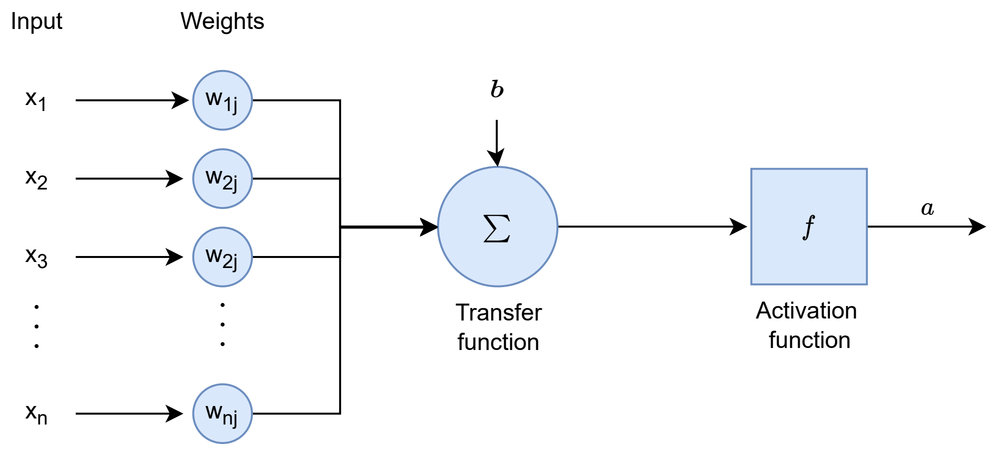


Figure 2.2: Structure of a single artificial neuron.

The neuron is the basic processing element in a NN. The structure of the neuron, shown in Fig. 2.2, dictates how the information is processed. Its structure consists of the inputs x_j connected to the neuron, the weights w_{ij} that linearly combine the inputs, the activation function f, the bias b, and the output a_j. The activation function defines how the weighted sum of the input is transformed into output and is chosen based on the modeling problem.

The output of a neuron j (a_j) is given by:

$$a_j = f_j(Wx_j + b_j),$$

where f_j is the activation function, b_j is the bias, W is the weight matrix and x_j is all the inputs of the neuron j.

Another vital aspect of NNs is the learning rule. It is a systematic procedure to modify a neural network's weights and biases to work as we need and perform its tasks accurately. This procedure is what is known as training the NN. First, the neural network processes these inputs and calculates their output, and an error is calculated by comparing the output and the target. Finally, the NN updates its weights and biases using the backpropagation algorithm. Learning is complete when examining additional observations does not usefully reduce the error rate. The learning rate defines the size of the corrective steps that the model takes to adjust for errors in each observation.

The backpropagation algorithm: In each iteration k, with input x_k, and for layer m (collection of neurons)

$$w_{ij}^{m}(k+1) = w_{ij}^{m}(k) - \alpha \frac{\partial F}{\partial w_{ij}^{m}}(k),$$

where α is the learning rate (the same applies to the biases). In a straightforward way, to compute this expression, we need the error value, the weights, the bias, the activation functions' derivatives, and the input. A forward pass is needed to obtain all these values: given input and set of values for the weights and bias, we compute all the intermediate values in the network until the end. Then we can compute the error. Then the backpropagation is made considering the intermediate values computed in the forward pass, and the weights and bias are updated with the gradients. Finally, the process is repeated for all the following inputs.

Ultimately, these models aim to learn complex non-linear functions by replicating how information travels in the human brain. From the wide range of ANNs configurations, we cover two of them: FFNNs and CNNs.

2.4.2 Feed Forward Neural Networks

A FFNN is a type of ANNs where information flows in one direction, from the input layer through one or more hidden layers to the output layer. This type of network is known as "feed-forward" because there are no loops or feedback connections between the layers. FFNNs are commonly used for tasks such as classification, regression, and pattern recognition. The structure of a FFNN consists of three main types of layers: the **input layer**, the **hidden layer** and the **output**

layer. The **input layer** is where the network receives the initial data or input features. Each neuron in the input layer corresponds to a specific input data feature. Following the input layer, we have the **hidden layer**. There can be one or more hidden layers between the input and output layers. These layers consist of interconnected neurons, each performing computations on the input it receives. The term "hidden" refers to the fact that the data is not directly observed but is transformed through these intermediate layers. The neurons in the hidden layers apply weights to the input data and pass the weighted sum through an activation function. The last layer is the **output layer**, where the output result is produced. Depending on the nature of the problem, the output layer might have one neuron for Binary Classification or Regression tasks, or multiple neurons for Multi-class Classification.

The process of information flow through a FFNN follows a linear sequence. The input values are passed to the neurons in the input layer. Each input value is multiplied by a corresponding weight, and these weighted inputs are then summed up. The sum of the weighted inputs is passed through an activation function. This function introduces non-linearity into the network, allowing it to learn complex relationships in the data. There are several activation functions, the most common being the *sigmoid*, *tanh*, and *ReLU* (Rectified Linear Unit). The processed data then flows through the hidden layers similarly. The final hidden layer's outputs are then passed to the output layer, where the final results are computed based on the specific problem the network is designed to solve. An example of the architecture of a FFNN is shown in Fig. 2.3.

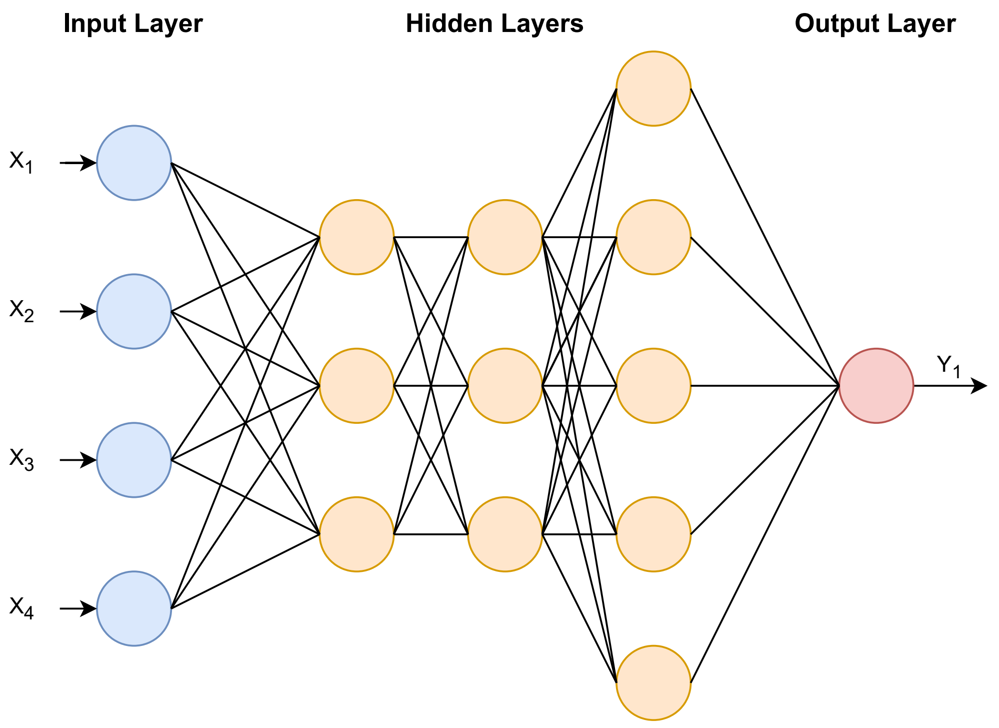


Figure 2.3: Example of the structure of a FFNN with 3 hidden layers and 1 single output value.

2.4.3 Convolutional Neural Networks

CNNs are a type of ANNs specifically designed for processing data with a grid-like topology, such as an image. CNNs are composed of multiple interconnected layers of artificial neurons that process and transform the input data as it passes through the network. These layers may not be fully connected (this reduces the number of weights to be learned). CNNs are inspired by our visual cortex and are incredibly robust for image analysis. The input layer takes raw data, such as an image, and the output layer produces the final prediction or classification. The hidden layers are responsible for extracting features and patterns from the input data and transforming it into a more helpful representation for the output layer. Although image analysis has been the most widespread use of CNNs, it can also be used for other data analysis or classification problems. Most generally, a CNN is an artificial neural network with some specialization for picking out or detecting patterns and making sense of them.

The convolutional layer is the core of the structure of CNNs. It applies filters to the input data to detect specific patterns and features. These filters slide across the data, performing element-wise multiplications and summing the results to produce a new set of features. Unlike the fully connected layers present in Feed Forward Neural Networks, which are unpractical for large inputs such as high-resolution images because of the large number of parameters to be learned, convolutional layers downsample the input, reducing the number of learnable parameters and allow the network to be deeper, in the following manner: Given an input with width W_{in} and height H_{in}, a filter of dimensions K x K, a stride S and padding P, we have

$$W_{out} = \frac{W_{in} - K + 2P}{S} + 1,$$

$$H_{out} = \frac{H_{in} - K + 2P}{S} + 1,$$

where W_{out} represents the width of the output and H_{out} represents the height of the output. The filter is usually a two-dimensional (2-D) array of weights, which represents part of the image

In each step, the Hadamard multiplication is applied ($Wx + b = a$) to a region of the entire input, where W represents the filter, x is a section of the input data and b is the bias.

In Fig. 2.4, the Hadamard multiplication for the first step of the convolutional layer is as follows:

$$9 \times 0 + 4 \times 2 + 1 \times 1 + 1 \times 4 + 1 \times 1 + 1 \times 0 + 1 \times 1 + 2 \times 0 + 1 \times 1 = 16 \quad (2.21)$$

The max and average pooling layers are helpful in order to downsample the convoluted layer, reducing the number of features and lowering the number of parameters to be learned in the following layers and in order to make the network more robust to small translations and distortions in the input data. They divide their input into rectangular pooling regions and compute each region's maximum or average (respectfully), taking stride (the moving step) as a parameter.

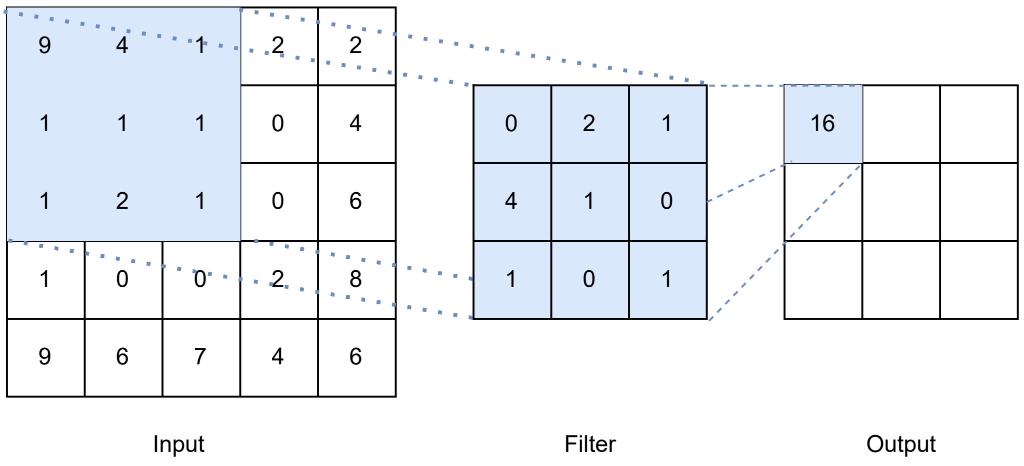


Figure 2.4: An example of a single step of a convolutional layer.

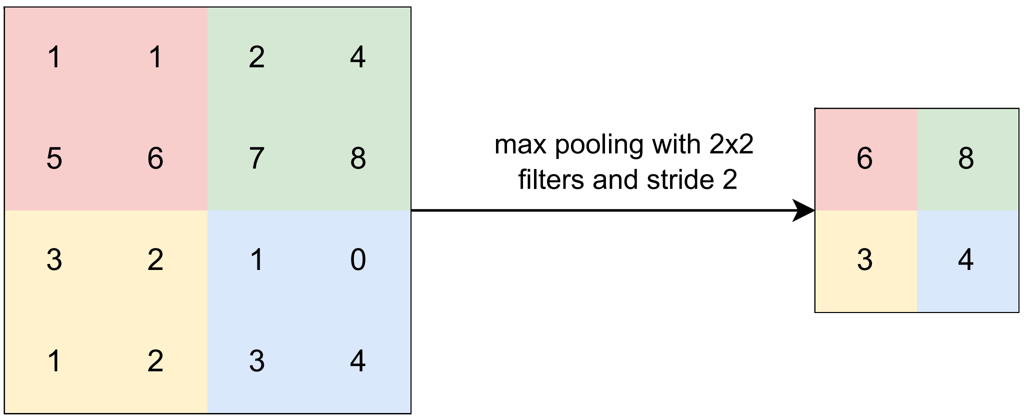


Figure 2.5: An example of a max-pooling layer.

The combination of convolutional and pooling layers allows CNNs to learn hierarchical representations of the input data, starting with simple patterns and features in the lower layers and building up to more complex ones in the higher layers. This hierarchical structure is what makes CNNs well-suited for tasks such as image classification, object detection, and segmentation.

The ReLU layer applies a threshold operation to the convoluted output, by which any value less than zero is set to zero. The ReLU has fewer vanishing gradient problems compared to sigmoid and tanh. As CNNs are also commonly used in natural language processing, brain-computer interfaces, and time series analysis, there are several variations of this layer:

$$\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases} \tag{2.22}$$

$$\sigma(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{if } x \leq 0 \end{cases} \tag{2.23}$$

$$\sigma(x) = \begin{cases} max & \text{if } x > max \\ x & \text{if } 0 < x \leq max \\ 0 & \text{if } x \leq 0 \end{cases} . \tag{2.24}$$

Normalization can favor the training of the CNNs. Usually, the inputs to the CNN are normalized, the most common way is to normalize the data to zero mean and unit variance (whitening). However, this normalization is lost after all the calculations in the previous layers. So it can be convenient to normalize the intermediate layers again to zero mean and unit variance. A batch normalization layer does this.

The fully connected layers follow the convolution and pooling layers. All of its neurons connect to all the neurons in the previous layer. In the case of classification problems, the output size of the last fully connected layer is equal to the number of different classes.

A softmax layer and a classification layer follow the last fully connected layer (for classification problems).

Chapter 3

Technical Part: Feasibility Condition

In this section, we offer a novel condition on the noise structure (namely on its covariance matrix) that grants feasibility of the structure identification problem under partial observability and colored noise. The technical results presented, namely, the theorems, proofs, some discussion and definitions have been submitted for publication [1].

3.1 Assumptions

Our work builds on three main assumptions:

Assumption 1. *(Stability)* The interaction matrix A is assumed symmetric, non negative and stable, i.e., $\rho(A) < 1$, where $\rho(A)$ is the spectral radius of A.

Assumption 2. *(Homogeneity)* We assume that $\sigma^2 = \mathbb{E}\left[\mathbf{x}_i^2\right]$ for all i.

Under Assumption 2, observe that $\sigma^2 \geq \mathbb{E}\left[\mathbf{x}_i\mathbf{x}_j\right] \ \forall\, i,j$. Indeed,

$$\begin{aligned} 0 \leq \mathbb{E}[(\mathbf{x}_i - \mathbf{x}_k)^2] &= \mathbb{E}[\mathbf{x}_i^2] + \mathbb{E}[\mathbf{x}_k^2] - 2\mathbb{E}[\mathbf{x}_i\mathbf{x}_k] \\ &= 2\sigma^2 - 2\mathbb{E}[\mathbf{x}_i\mathbf{x}_k]. \end{aligned} \tag{3.1}$$

and the above inequality implies the inequality $\sigma^2 \geq \mathbb{E}\left[\mathbf{x}_i\mathbf{x}_j\right] \ \forall\, i,j$.

Assumption 3. *(Distinguishability)* We assume that $\sigma^2 > \mathbb{E}\left[\mathbf{x}_i\mathbf{x}_j\right] = [\Sigma_x]_{ij}$ for all $i \neq j$. That is, the off-diagonal entries of the noise covariance matrix are strictly smaller than the diagonal.

We remark that under Assumption 2, the latter Assumption 3 is a mild assumption. Indeed, in view of inequality (3.1), $\sigma^2 = \mathbb{E}\left[\mathbf{x}_i\mathbf{x}_j\right]$ if and only if $\mathbb{E}\left[\left(\mathbf{x}_i - \mathbf{x}_j\right)^2\right] = 0$, where this latter identity is equivalent to $\mathbf{x}_i \overset{a.s.}{=} \mathbf{x}_j$. In other words, strict inequality holds whenever there is no two nodes with the same noise applied (almost surely).

Under Assumptions 2 and 3, we refer to σ_{gap}^2 as the gap between the diagonal and the off-diagonal entries of the covariance matrix Σ_x, defined as

$$\sigma^2_{gap} \triangleq \sigma^2 - max_{i \neq j}\mathbb{E}[x_i x_j] > 0.$$

3.2 Identifiability

A matrix valued estimator $\widehat{A}_S^{(n)}$ summarizes the interaction strengths across distinct pairs of nodes (within the observed set) and can be characterized as

$$\widehat{A}_S^{(n)} = A_S + \mathcal{E}_S^{(n)}, \tag{3.2}$$

where A_S is the ground-truth interaction matrix linking the subset of observed nodes S and $\mathcal{E}_S^{(n)}$ is the error matrix associated with the estimation obtained with n time series samples.

The estimator is structurally consistent whenever any entry ij of $\widehat{A}_S^{(n)}$ associated with a connected pair is greater than any other entry of a disconnected pair – please, refer to Definition 1, defined in the work of Machado et al. [23]. This is equivalent to requiring that the entries of $\mathcal{E}_S^{(n)}$ are not too distinct. More concretely, define the oscillation of a matrix as

$$\mathsf{Osc}\,(\mathcal{E}) \triangleq \mathcal{E}_{\max} - \mathcal{E}_{\min},$$

where $\mathcal{E}_{\min}$ and $\mathcal{E}_{\max}$ are the minimum and maximum of the off-diagonals entries of the error matrix $\mathcal{E}$. Let $A^{+}_{\min}$ be the smallest entry of A different than zero. We can say that if

$$\mathsf{Osc}\,(\mathcal{E}) \leq \frac{A_{\min}}{2}, \tag{3.3}$$

then this means that if ij represents the pair of connected nodes i and j and kl is the pair of disconnected nodes k and l, then $\widehat{A}_{ij} > \widehat{A}_{kl}$ under constraint (3.3), meaning that the matrix-valued estimator $\widehat{A}_S$ is structurally consistent, and the structure of the network can be recovered via proper thresholding or clustering of the off-diagonals of $\widehat{A}_S$. Evidently, the error matrix $\mathcal{E}$ determines whether the structural information of the networks is lost or preserved in the time series. If $\mathcal{E}$ is *flat* enough ($\mathcal{E}_{\max}$ - $\mathcal{E}_{\min}$ is *small*) then an estimator $\widehat{A}_S$ is structurally consistent. Furthermore, when the error matrix is *flat* ($\mathcal{E} \propto \beta 1 1^T$) then the entries of the estimator $\widehat{A}$ are the entries of A shifted by β. This reveals the importance of the characterization of the error term and the analysis of the *flatness* of the colored noise in the estimation of the structure of the network. It also highlights the difficulty in handling the colored noise regime.

All-in-all, two important steps shall be in general accounted when establishing the structural consistency of an estimator. First, its error characterization. Then, the analysis of its entries variability. For example, this is done by Matta et al. [13] to establish the structural consistency of Granger under partial observability and diagonal noise.

3.3 Error characterization

It is widely understood, when conducting statistical inference in high-dimensional settings, that the ability to perform inference tasks is fundamentally dependent upon the various sets of parameters and noise excitation levels. In particular, the regime of parameters, observability or noise level determine whether or not the information about the structure of inference is entailed in the observed data or fundamentally lost, rendering inference impossible in the latter case. The following theorem characterizes the error term in the $R_1 - R_3$ estimator.

Theorem 1. Let $\Sigma_x \in \mathbb{S}_+{}^{M\times M}$ be the covariance of the noise process[1] $(\mathbf{x}(n))$, satisfying assumptions 2 and 3. Define

$$\Sigma_x := \sigma^2_{\text{gap}} I + \beta 1 1^T + \overline{\Sigma}_x, \tag{3.4}$$

where $\sigma^2_{\text{gap}} I$ is a diagonal matrix, $\beta 1 1^T$ is the average offset matrix and $\overline{\Sigma}_x$ contains the variability of the off-diagonals entries of Σ_x. Then

$$R_1 - R_3 = \sigma^2_{\text{gap}} A + \mathcal{E}, \tag{3.5}$$

with

$$\mathcal{E} \overset{\Delta}{=} \beta\rho 1 1^T + \left(I - A^2\right)\left(\sum_{i=0}^{\infty} A^{i+1}\overline{\Sigma}_x A^i\right)$$

where we refer to $\overline{\Sigma}$ as the *residual* covariance matrix of the process $(\mathbf{x}(n))$ and to $\mathcal{E}$ as the error term in the matrix-estimation problem. We can further consider the partial observability case

$$\widehat{A}_{\mathcal{S}} = [R_1]_{\mathcal{S}} - [R_3]_{\mathcal{S}} = \sigma^2 A_{\mathcal{S}} + \mathcal{E}_{\mathcal{S}},$$

where we have defined

$$[R_k]_{\mathcal{S}} \overset{\Delta}{=} \mathbb{E}\left[[\mathbf{y}(n+k)]_{\mathcal{S}}\,[\mathbf{y}(n)]_{\mathcal{S}}^{\top}\right],$$

representing the k-lag covariance matrix built upon the observed samples, i.e., ignoring the latent part of the network.

From this theorem, we characterize the limiting error term $\mathcal{E}$ of this estimator under various conditions of full and partial observability and spatially correlated noise. We establish the requirements for the covariance matrix of the noise signal Σ_x, which ensures that the structural information is preserved and can be recovered from the observed time series data.

Proof of theorem 1. Remarking that

$$R_0 \overset{\Delta}{=} \lim_{n\to\infty} R_0(n) = \sum_{i=0}^{\infty} A^i \Sigma_x A^i,$$

[1]Where $\mathbb{S}_+{}^{M\times M}$ is the set of symmetric and positive semi-definite matrices. A matrix C is considered positive semi-definite when $x^\top C x \geq 0, \forall x \in \mathbb{R}^M$. The covariance matrix Σ obeys this property. In particular, $v^\top \Sigma v = v^\top E[xx^\top]v = E[(x^\top v)^2] \geq 0, \forall v$.

and that

$$R_k = A^k R_0, \forall k \in \mathbb{N},$$

we can represent R_1 and R_3 as

$$\begin{cases} R_1 &= AR_0 = \sigma^2_{\text{gap}} A \sum_{i=0}^{\infty} A^i \Sigma_x A^i \\ R_3 &= A^3 R_0 = \sigma^2_{\text{gap}} A^3 \sum_{i=0}^{\infty} A^i \Sigma_x A^i. \end{cases}$$

Considering the definition of Σ_x present in (3.4), we have

$$\begin{cases} R_1 &= \sigma^2 A \sum_{i=1}^{\infty} A^{2i} + \beta A \sum_{i=0}^{\infty} A^i 11^T A^i + A \sum_{i=0}^{\infty} A^i \overline{\Sigma} A^i \\ R_3 &= \sigma^2 A^3 \sum_{i=1}^{\infty} A^{2i} + \beta A^3 \sum_{i=0}^{\infty} A^i 11^T A^i + A^3 \sum_{i=0}^{\infty} A^i \overline{\Sigma} A^i. \end{cases}$$

We can split $R_1 - R_3$ into 3 different terms, as follows:

$$\begin{aligned} R_1 - R_3 &= \left(\sigma^2 A \sum_{i=1}^{\infty} A^{2i} - \sigma^2 A^3 \sum_{i=1}^{\infty} A^{2i}\right) \\ &+ \left(\beta A \sum_{i=0}^{\infty} A^i 11^T A^i - \beta A^3 \sum_{i=0}^{\infty} A^i 11^T A^i\right) \\ &+ \left(A \sum_{i=0}^{\infty} A^i \overline{\Sigma} A^i - A^3 \sum_{i=0}^{\infty} A^i \overline{\Sigma} A^i\right). \end{aligned} \tag{3.6}$$

Now, it is trivial that the first term can be simplified down to

$$\sigma^2_{\text{gap}} A \sum_{i=0}^{\infty} A^{2i} - \sigma^2_{\text{gap}} A^3 \sum_{i=0}^{\infty} A^{2i} = \sigma^2_{\text{gap}} A,$$

since

$$\begin{aligned} &\sigma^2_{\text{gap}} A \sum_{i=0}^{\infty} A^{2i} - \sigma^2_{\text{gap}} A^3 \sum_{i=0}^{\infty} A^{2i} && \Leftrightarrow \\ &\sigma^2_{\text{gap}} \left(A + A^3 + A^5 + ...\right) - \sigma^2_{\text{gap}} \left(A^3 + A^5 + ...\right) && \Leftrightarrow \\ &\sigma^2_{\text{gap}} \left(A + A^3 - A^3 + A^5 - A^5 + ...\right) && \Leftrightarrow \sigma^2_{\text{gap}} A. \end{aligned}$$

From this simplification of the first term of the estimator, our interaction matrix A arises. On the other hand, the error term $\mathcal{E}$ stems from the simplification of the second term (remark that $A1 = \rho 1$)

$$\begin{aligned} &\beta A \sum_{i=0}^{\infty} A^i 11^T A^i - \beta A^3 \sum_{i=0}^{\infty} A^i 11^T A^i && \Leftrightarrow \\ &\beta A \left(\sum_{i=0}^{\infty} A^i 11^T A^i - A^2 \sum_{i=0}^{\infty} A^i 11^T A^i\right) && \Leftrightarrow \\ &\beta \left(I - A^2\right) A \sum_{i=0}^{\infty} A^i 11^T A^i && \Leftrightarrow \\ &\beta \left(I - A^2\right) A 11^T \sum_{i=0}^{\infty} \rho^{2i} && \Leftrightarrow \\ &\tfrac{\beta\rho(1-\rho^2)}{1-\rho^2} 11^T && \Leftrightarrow \beta\rho 11^T, \end{aligned}$$

and the third term

$$\begin{aligned}
&A \textstyle\sum_{i=0}^{\infty} A^i \overline{\Sigma}_x A^i - A^3 \sum_{i=0}^{\infty} A^i \overline{\Sigma}_x A^i \quad \Leftrightarrow \\
&A \textstyle\sum_{i=0}^{\infty} A^i \overline{\Sigma}_x A^i - A^2 \sum_{i=0}^{\infty} A^{i+1} \overline{\Sigma}_x A^i \quad \Leftrightarrow \\
&(I - A^2) \textstyle\sum_{i=0}^{\infty} A^{i+1} \overline{\Sigma}_x A^i.
\end{aligned}$$

These three simplified terms together yield

$$R_1 - R_3 = \sigma_{\text{gap}}^2 A + \underbrace{\beta\rho 1 1^T + \left(I - A^2\right)\left(\sum_{i=0}^{\infty} A^{i+1} \overline{\Sigma}_x A^i\right)}_{\mathcal{E}},$$

which was to be demonstrated.

3.4 Feasibility Condition

Theorem 2. Let $A = p\overline{A}$, where $\overline{A}$ is a stochastic matrix with $0 < p < 1$. Under assumption 2 and 3, if

$$\frac{\text{Osc}\left(\text{Off}(\Sigma_x)\right)}{\sigma_{\text{gap}}^2} \leq \frac{A_{\min}(1-\rho^2)}{2\rho(\rho^2+1)}, \tag{3.7}$$

where $\sigma_{\text{gap}}^2 \overset{\Delta}{=} \mathbb{E}\left[\mathbf{x}_i^2\right] - \max_{i \neq j} \mathbb{E}\left[\mathbf{x}_i \mathbf{x}_j\right] > 0$, $\text{Off}(\Sigma_\text{x})$ are the off-diagonal entries of Σ_x, and $A_{\min}$ is the smallest nonzero entry of the interaction matrix A, then features $\left\{\mathcal{F}_{ij}(n)\right\}_{i \neq j}$ are linearly separable and stable, where

$$\mathcal{F}_{ij} := \left(\left[\widehat{R}_D(n)\right]_{ij}, \left[\widehat{R}_{D+1}(n)\right]_{ij}, \ldots, \left[\widehat{R}_M(n)\right]_{ij}\right), \tag{3.8}$$

as long as $D \leq 1$ and $M \geq 3$. This result asserts a nontrivial condition on the covariance matrix of the process $(\mathbf{x}(n))$ to grant linear separability of these features. The idea is to apply machine learning methods to cluster the referred features under the assumed conditions.

Proof of theorem 2. Recalling that $A = \rho\overline{A}$, with $0 < \rho < 1$, that $\text{Osc}(\text{A}) = \text{Osc}(-\text{A})$, and the properties of the function $\text{Osc}(\cdot)$, defined in Appendix D, we have that

$$\begin{aligned}
\mathsf{Osc}(\mathcal{E}) &= \mathsf{Osc}\left(A(I-A^2)\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&= \mathsf{Osc}\left(\rho\overline{A}(I-A^2)\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&\overset{P.2}{=} \rho\mathsf{Osc}\left(\overline{A}(I-A^2)\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&\overset{P.1}{\leq} \rho\mathsf{Osc}\left((I-A^2)\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&= \rho\mathsf{Osc}\left(\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i - A^2\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&\overset{P.3,\, P.4}{\leq} \rho\mathsf{Osc}\left(\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) + \rho^3\mathsf{Osc}\left(\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&= \rho(1+\rho^2)\mathsf{Osc}\left(\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right),
\end{aligned}$$

which ultimately results in

$$\mathsf{Osc}(\mathcal{E}) \leq \rho(1+\rho^2)\mathsf{Osc}\left(\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right). \tag{3.9}$$

Now, observe that

$$\begin{aligned}
\textstyle\sum_{i=0}^{\infty} A^i\left(\overline{\Sigma}_{\max}11^T\right)A^i &= \overline{\Sigma}_{\max}\textstyle\sum_{i=0}^{\infty} A^i 11^T A^i \\
&= \overline{\Sigma}_{\max}\textstyle\sum_{i=0}^{\infty} \rho^{2i}\overline{A}^i 11^T \overline{A}^i \\
&= \overline{\Sigma}_{\max}\left(\textstyle\sum_{i=0}^{\infty}\rho^{2i}\right)11^T \\
&= \tfrac{\overline{\Sigma}_{\max}}{1-\rho^2}11^T,
\end{aligned}$$

where $\overline{\Sigma}_{\max}$ represents the maximum entry of the $\overline{\Sigma}$. The same simplification applies to the minimum entry $\overline{\Sigma}_{\min}$. From this, we can affirm that $\sum_{i=0}^{\infty} A^i\overline{\Sigma}A^i$ is bounded above and below as follows

$$\frac{\overline{\Sigma}_{\min}}{1-\rho^2}11^T \leq \sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i \leq \frac{\overline{\Sigma}_{\max}}{1-\rho^2}11^T,$$

and thus

$$\mathsf{Osc}\left(\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \leq \frac{\overline{\Sigma}_{\max}-\overline{\Sigma}_{\min}}{1-\rho^2}. \tag{3.10}$$

From inequation (3.9) and (3.10), we have

$$\begin{aligned}
\mathsf{Osc}(\mathcal{E}) &= \mathsf{Osc}\left(A(I-A^2)\textstyle\sum_{i=o}^{\infty} A^i\overline{\Sigma}_x A^i\right) \\
&\leq \tfrac{\rho(1+\rho^2)}{1-\rho^2}\left(\overline{\Sigma}_{\max}-\overline{\Sigma}_{\min}\right).
\end{aligned} \tag{3.11}$$

In view, of inequality (3.3), we have structural consistency (or features separability) whenever

$$\mathsf{Osc}(\mathcal{E}) \leq \sigma^2_{\text{gap}} \frac{A_{\min}}{2},$$

which leads to Eq. (3.7),

$$\frac{\mathsf{Osc}\left(\mathsf{Off}(\Sigma)\right)}{\sigma^2_{\text{gap}}} \leq \frac{A_{\min}(1-\rho^2)}{2\rho(1+\rho^2)},$$

which was to be demonstrated.

3.5 Exogenous Interventions

An exogenous intervention refers to the application of external inputs or influences to an NDS. Such interventions commonly involve the utilization of control signals to regulate and manipulate the system's behavior, ultimately leading to the stabilization of the system.

Assume that the time series $(\mathbf{y}(n))$ stem from the following linear NDS with exogenous interventions

$$\mathbf{y}(n+1) = A\mathbf{y}(n) + \mathbf{x}(n+1) + \gamma\xi(n+1), \tag{3.12}$$

where we assume Σ and $\sigma^2_{\xi} I$ as the covariance matrices of the excitation noise $(\mathbf{x}(n))$ and the exogenous intervention process $(\xi(n))$. The parameter γ controls the strength of the signal ξ and, consequently, the level of the exogenous interventions.

Remark that under exogenous interventions to the NDS following dynamical law referenced in Eq. (3.12), then Eq. (3.7) becomes

$$\frac{\mathsf{Osc}\left(\mathsf{Off}(\Sigma_x)\right)}{\sigma^2_{\text{gap}} + \sigma^2_{\xi}} \leq \frac{A_{\min}(1-\rho^2)}{2\rho(\rho^2+1)}. \tag{3.13}$$

The above corollary states that if the *level* of the intervention (captured by the variance σ^2_{ξ}) is high enough, then the features are always linearly separable and thus, Machine Learning can be applied. Which implies, in particular, that regardless of the covariance matrix of the input process $(\mathbf{x}(n))$, the feature vector, represented as

$$\mathcal{F}(n) := \left(\left[\widehat{R}_D(n)\right], \left[\widehat{R}_{D+1}(n)\right], \ldots, \left[\widehat{R}_M(n)\right]\right), \tag{3.14}$$

is linearly separable with high probability as soon as the level (or variance) σ^2_{ξ} of the external intervention $(\xi(n))$ is high enough.

3.6 Impact of Colored Noise

While the affine separability property depends on the level of oscillation in the error matrix term $\mathcal{E}$, or it is possibly recovered under exogenous intervention, the separating hyperplane is shifted away from the origin by the level of correlation in the noise captured by the parameter $\beta > 0$, in view of the *flatness* property of the noise structure. Figures 3.1 and 3.2 summarize the effect of the correlation structure of the noise process $(\mathbf{x}(n))_{n\in\mathbb{N}}$ in the set $\{\mathcal{F}_{ij}(n)\}_{i\neq j}$: i) the average off-diagonal of Σ_x yields a drift of the features away from the origin; ii) the oscillation of the off-diagonal entries $\mathsf{Osc}(\mathsf{Off}(\Sigma_x))$ contributes to the *spread* of the features, possibly compromising separability.

Due to the sensitivity of the separating hyperplane on the color of the noise, the method deployed by Machado et al. [23] degrades as the average of the absolute values of the off-diagonal entries of Σ_x increase (captured by β). To mitigate the shift of the separating hyperplane and allow the deployment of supervised methods, we renormalize the features applying *standard scaling*.

Consider the following equation for the NDS, as an expansion of Eq. (2.1) for *colored* noise,

$$\mathbf{y}(n+1) = A\mathbf{y}(n) + \alpha\mathbf{x}(n+1) + \frac{\beta}{\sqrt{M}}\mathbf{1}\mathbf{1}^T\mathbf{x_2}(n+1), \tag{3.15}$$

where signal $\mathbf{x_2}$ introduces colored noise into the system, and β controls the strength of the colored noise. Colored noise is a type of noise where the correlation structure between different nodes is non-zero, meaning that the noise is spatially dependent across nodes.

3.6.1 Impact of Colored Noise on the Networked Dynamical System

When β is set to 0, as mentioned, there is no colored noise. In other words, only the diagonal noise, controlled by the parameter α, affects the system's evolution. Diagonal noise implies that the noise values at different nodes are independent. On the other hand, when β is non-zero, the term $\frac{\beta}{\sqrt{M}}11^T\mathbf{x_2}(n+1)$ introduces colored noise into the system, where the noise at each node is determined by the corresponding entry of the vectors $\mathbf{x}(n+1)$ and $\mathbf{x_2}(n+1)$. Increasing the value of β amplifies the strength of the signal $\mathbf{x_2}(n+1)$ and, consequently, the effect of the colored noise on the system. Higher β values indicate stronger colored noise, which can lead to more significant deviations or fluctuations in the network dynamics.

In summary, β controls the presence and strength of colored noise in the NDS. Increasing β amplifies the effect of the colored noise and its correlation structure, potentially leading to more complex and interconnected dynamics among the nodes.

3.6.2 Impact of Colored Noise on the feature vector

Increasing the value of β can impact the magnitude of the cross-correlation values in the feature vectors. Higher β values indicate a stronger colored noise component, which can introduce additional fluctuations and variations in the time series of the nodes. These fluctuations can influence the cross-correlation values and potentially increase their magnitudes. All the plots derive from a generated graph with 20 nodes, 50% of connectivity and 100000 samples.

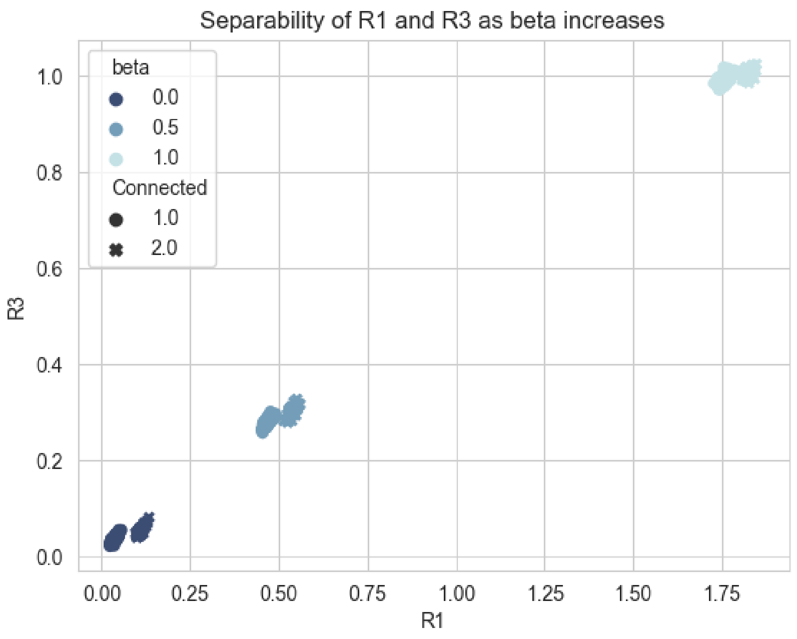


Figure 3.1: Shift of the R_1 and R_3 features as β increases. The features drift from the origin. Separability of the features is somewhat compromised.

Figures 3.1 and 3.2 indicate that, indeed, there is an increase in the magnitudes of the features, causing a shift of the centroid of the cluster.

In the work of Machado et al. [23], the feature-vector was normalized by the maximum value of the features, as defined in Eq. (2.20). Figure 3.3 shows the impact that normalizing by the max has on the features.

Although this normalization somewhat brings the features to the same feature space, it does not seem to improve separability. So we decided that in order to prevent the increase in the magnitude of the features, we need to shift the features back. To do this, we utilize the Standard Scaler (also called z-normalization), which standardizes the features by removing the mean and scaling to unit variance:

$$z = \frac{(x-\mu)}{\sigma},$$

where x stands for the features, μ and σ correspond to the mean and standard deviation, respectively, of the features. Finally, z represents the new shifted and

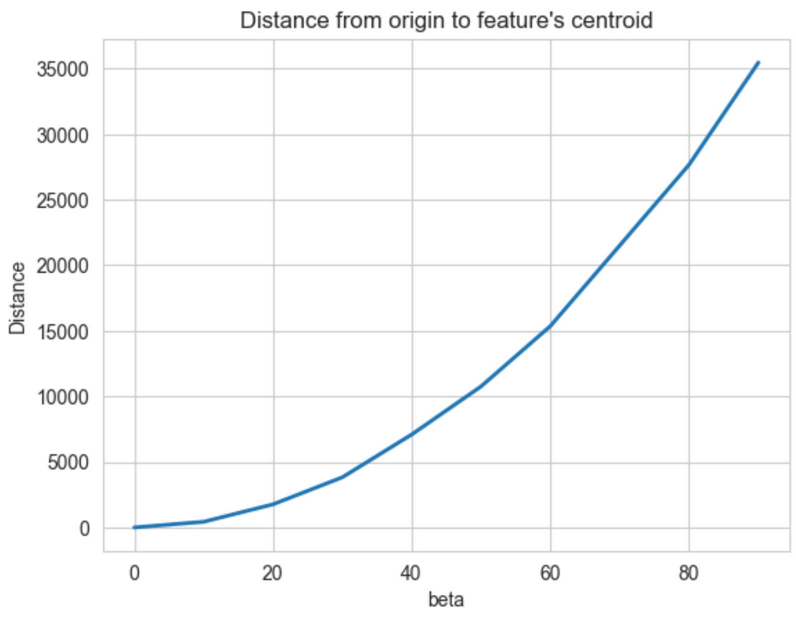


Figure 3.2: Adimensional distance in the feature space, measured from the origin to the feature's centroid.

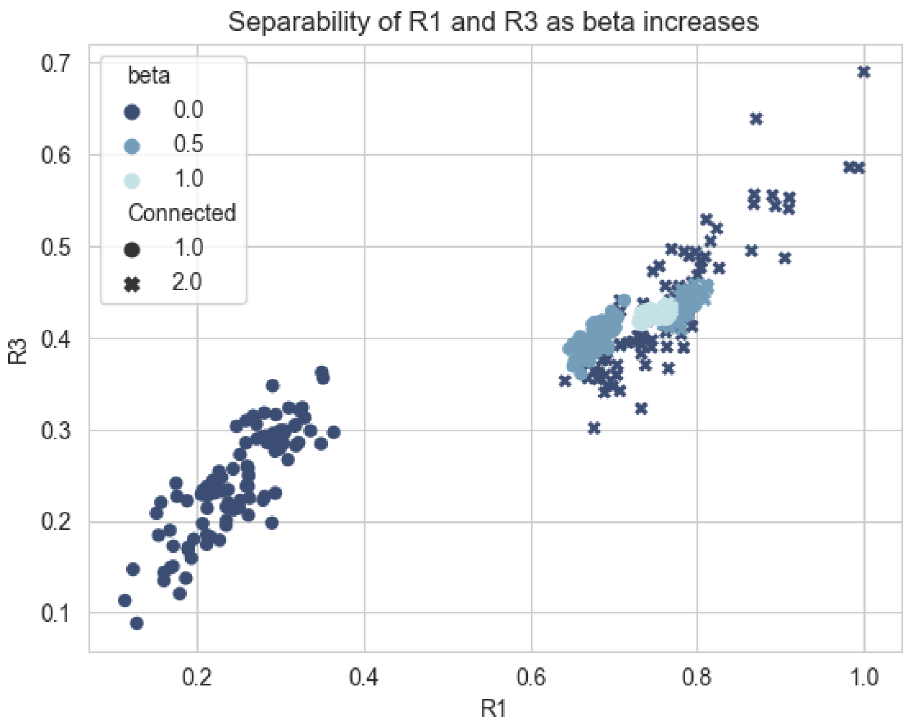


Figure 3.3: Separability of R_1 and R_3 as β increases, when the feature vectors are normalized by their maximum value.

scaled features. It is important to note that the centering and scaling happen independently on each feature by computing the relevant statistics on the samples. Figure 3.4 shows the impact of the *Standard Scaler* on the features, namely, the improved separability of the *Standard Scaler* over the max normalization and no scaling.

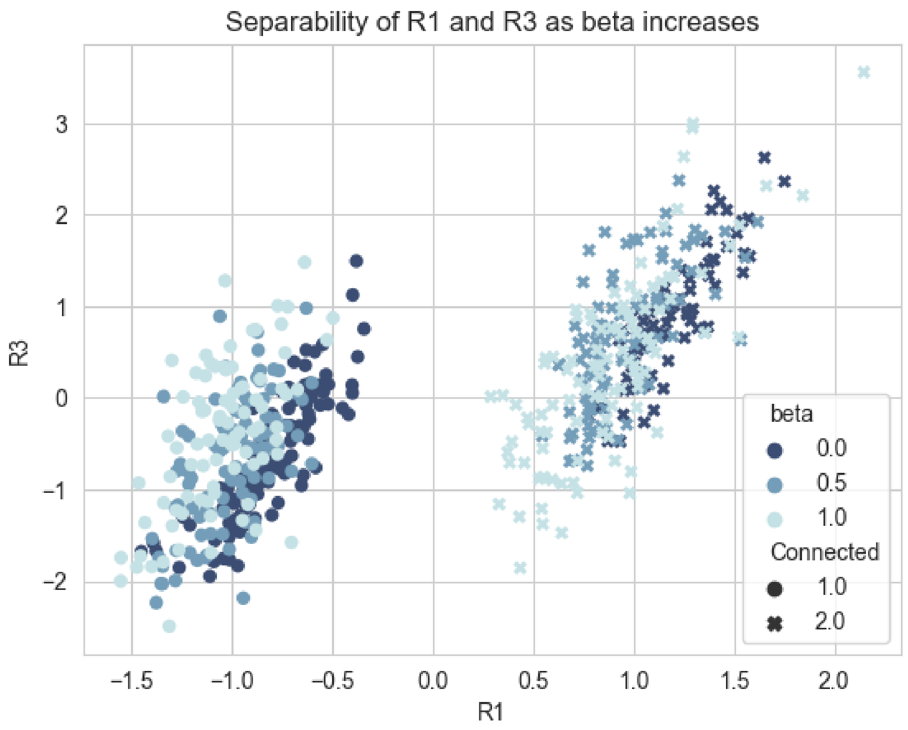


Figure 3.4: Separability of R_1 and R_3 as β increases, when the feature vectors are standardized by the *Standard Scaler*.

The parameter β also affects the statistical significance of the cross-correlation values in the feature vectors. As β increases, the colored noise component becomes more prominent, and the overall noise level in the system rises, making it more challenging to distinguish between significant cross-correlation values and random fluctuations, which means that the underlying patterns or relationships between the nodes become obscure. Therefore, higher β values can potentially reduce the statistical significance of the cross-correlation values since additional fluctuations, driven by the noise, can overlap or interfere with the true correlations present in the system.

The presence of high levels of noise, particularly colored noise, can decrease the signal-to-noise ratio in the system, meaning that the signal that contains the information about the structure, $\mathbf{x}(n+1)$, becomes dominated by the colored noise signal, $\mathbf{x_2}(n+1)$. This ratio represents the strength of the meaningful signal relative to the background noise. As β increases, the noise component becomes more dominant, making it harder to distinguish the relevant signal from the noise. This can lead to reduced separability between the features derived from the cross-correlation of the time series. Figure 3.5 evidences the increased difficulty in separating the features as β increases. Higher beta values may lead to an impossibility in distinguishing features from connected and disconnected pairs.

It is evident that the features become less separable as β increases, making it more difficult to separate between connected and disconnected clusters accurately. This may also be caused by the increase in sample complexity as β in-

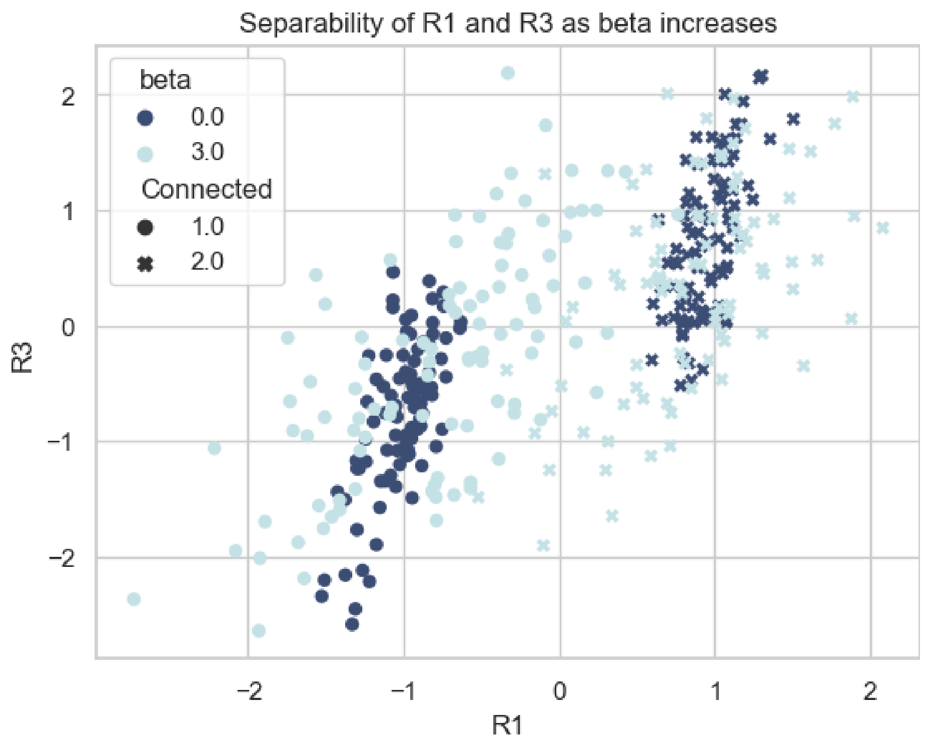


Figure 3.5: Influence of a higher β in the separability of R_1 and R_3.

creases.

In summary, the parameter β in the NDS equation can impact the magnitude, statistical significance, and interpretation of the cross-correlation values in the constructed feature vectors. Higher β values can lead to increased magnitudes and more complex interpretations while potentially reducing the statistical significance of the cross-correlation values.

Chapter 4

Methodology and Algorithms

This chapter describes the underlying generative process associated with the synthetic dataset. It describes, in detail, the generation of the adjacency matrix, a binary matrix of connections, of the network, the generation of the interaction matrix A, a weighted adjacency matrix, and the generation of the time series data. It also outlines the details of the feature-vector and how it is obtained. Lastly, it describes the training setup and architecture of the NNs used in the numerical simulation in Chapter 5. We use *Python* as the programming language, the *Keras* package for the neural networks and *NumPy*, *SciPy* and *scikit-learn* for data processing and calculations.

4.1 Data generation

The generative process of the data and the features follows a linear flow. Figure 4.1 shows a simple overview of the generative process. First of all, the adja-

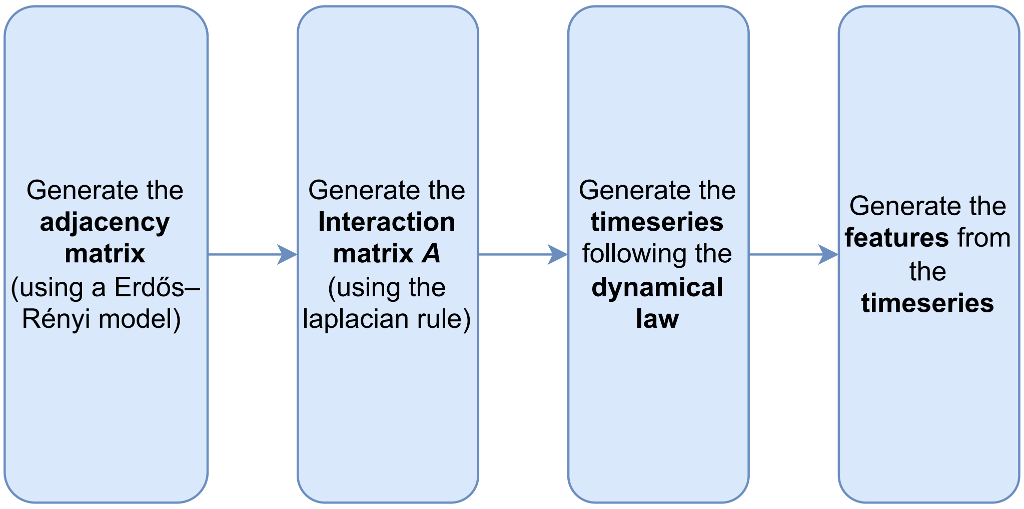

Generate the **adjacency matrix** (using a Erdős–Rényi model)
Generate the **Interaction matrix *A*** (using the laplacian rule)
Generate the **timeseries** following the **dynamical law**
Generate the **features** from the **timeseries**

Figure 4.1: Flow graph of the feature generation process.

cency matrix is generated using an Erdős–Rényi random graph model [45]. An example of a graph generated from this model is shown in Fig. 4.2. It is important to remark that we assure symmetry of the adjacency matrix, since we focus on the structure recovery of the network and not in causal inference. A link between nodes in the network is randomly drawn, with all possible connections between the nodes being equally likely, which means that each edge has a fixed probability p of being present (or $1 - p$ of being absent), independently of the other edges. The probability of each edge being connected is set by the parameter p, which ultimately reflects the sparsity of the network, i.e., the smaller the parameter p, the sparser the network. After generating the adjacency matrix, we

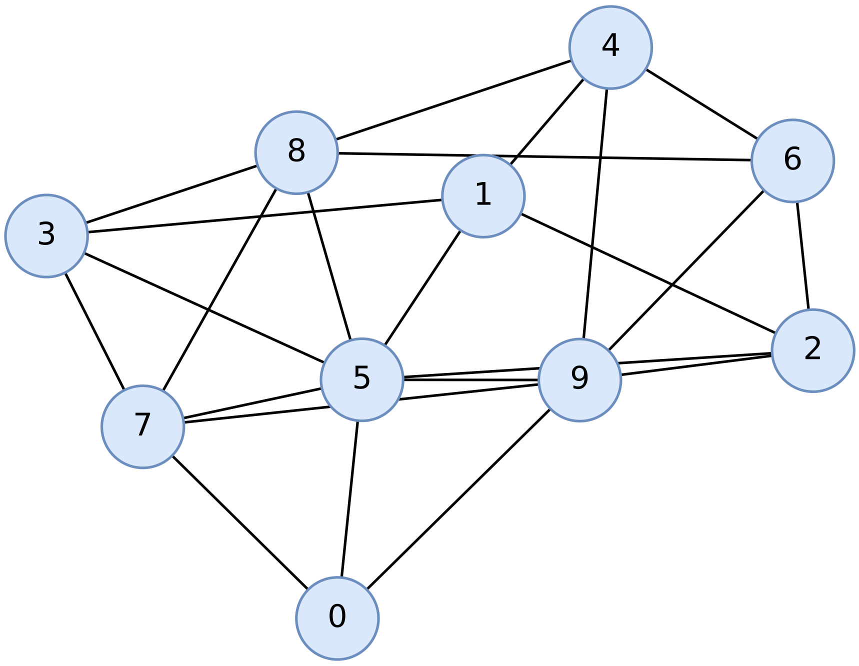


Figure 4.2: Example of a realization of an Erdős-Rényi random graph on $M = 10$ nodes and with probability of $p = 0.5$ of existing an edge between each pair of nodes.

build the underlying interaction matrix A with weights based on the Laplacian rule [46]. Firstly, we need to generate a Laplacian Matrix L from the adjacency matrix as

$$L_{i,j} = \begin{cases} deg(i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } i \text{ is adjacent to } j \\ 0 & \text{otherwise} \end{cases},$$

where $deg(i)$ is the degree of node i, i.e., its number of connections. Then, using the Laplacian Matrix, we can generate a stable matrix of interactions A as

$$A = \rho \times \left(I - \frac{c}{n_{\max}} L \right),$$

where $n_{\max}$ is the maximum in-flow degree of the underlying graph of the adjacency matrix and ρ and c are constants such that $0 < \rho \leq c < 1$. This weighting of the graph yields a stable matrix A (and hence, a stable linear NDS), meaning that $p(A) < 1$, where $p(A)$ is the spectral radius of A, with support graph given by the generated Erdős-Rényi. After generating the interaction matrix A, we generate the synthetic time series data following the dynamical law

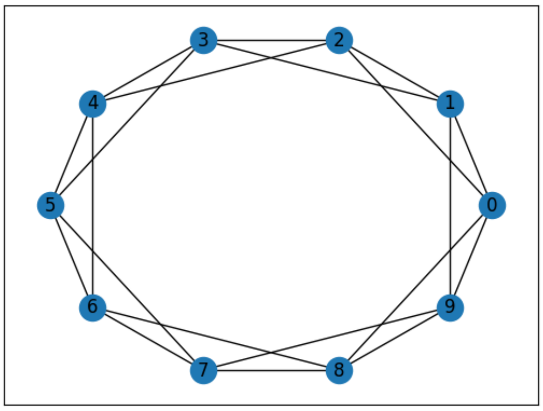
$$\mathbf{y}(n+1) = A\mathbf{y}(n) + \alpha\mathbf{x}(n+1) + \frac{\beta}{\sqrt{M}}\mathbf{1}\mathbf{1}^T\mathbf{x_2}(n+1), \tag{4.1}$$

where A is the interaction matrix, α and β are the noise standard deviation (α for the diagonal noise and β for the colored noise) and $\mathbf{1}\mathbf{1}^T$ is simply a matrix with all entries equal to one. If we want the noise to be diagonal, we set the parameter β to 0. Lastly, from the time series obtained by using equation (4.1), we generate the feature-vector.

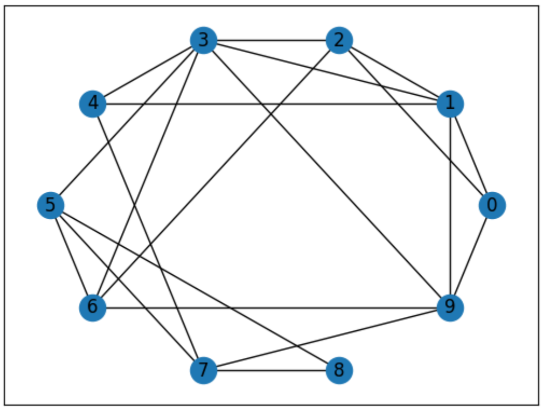
4.1.1 Small-World Networks

A small-world network is a mathematical graph where most nodes are not directly connected, but the neighbors of any given node are often connected. This means that most nodes can be reached from any other node with only a few steps. In particular, a small-world network is characterized by the typical distance L between two randomly selected nodes (measured in steps required) growing proportionally to the logarithm of the number of nodes M in the network.

The Watts-Strogatz model [3] generates random graphs with small-world properties, such as short average path lengths and high clustering. Given parameters M, k and p, the method works as follows: Starting with a ring of M vertices connected to k closest neighbors by undirected edges, we randomly decide to rewire edges, with a probability p, to a vertex chosen uniformly at random. This process is repeated until all edges have been considered once. It is shown that, for intermediate values of p, more concretely for $0 < p < 1$, the graph becomes a small-world network, highly clustered like a regular graph yet with a small characteristic path length like a random graph. Figures 4.3a and 4.3b expose the differences between the effect of p. In Fig. 4.3a the graph has a perfect ring topology, caused by $p = 0$, meaning that no edges were rewired. On the other hand, Fig. 4.3b reveals the effect of rewiring the edges, where in this case $p = 0.3$.



(a) Watts-Strogatz graph with $p = 0$.



(b) Watts-Strogatz graph with $p = 0.3$.

4.2 Feature-Vector

In this section, we introduce our new set of proposed features $\mathcal{K}^M(n)$. The data generation process is the same as explained in 4.1.

Define the set of features $\mathcal{T}^M(n) = \left\{\mathcal{T}^M_{ij}(n)\right\}_{ij}$

$$\mathcal{T}^M_{ij}(n) = \left(\left[\left(\left[\widehat{R}_{-\frac{M}{2}+1}(n)\right]_S\right)^{-1}\right]_{ij}, \dots, \left[\left(\left[\widehat{R}_{\frac{M}{2}}(n)\right]_S\right)^{-1}\right]_{ij}\right),$$

built on the inverse of the observed part of the lag moments. We have empirically observed that these features exhibit nontrivial separability and stability across a broad range of connectivity regimes and noise correlation levels. Fig. 4.4 reports the separability of some of the lag moments of this feature vector, namely, the inverse of the zero-lag, the one-lag and the third-lag correlation matrix.

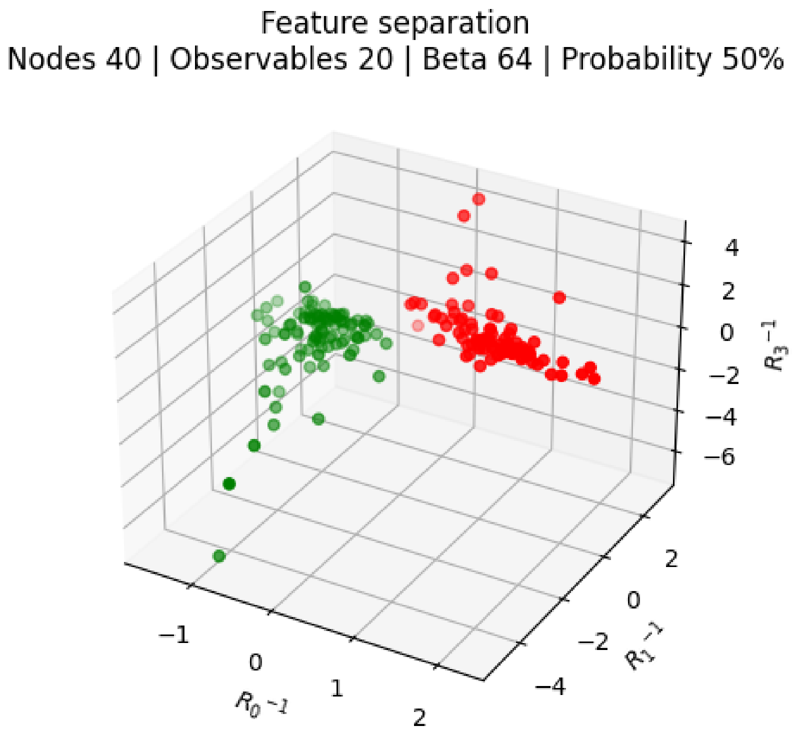


Figure 4.4: Separation of a subset of the feature-vector $\mathcal{T}(n)$.

Further, define the set of features $\mathcal{F}^M(n) = \left\{\mathcal{F}^M_{ij}(n)\right\}_{ij}$

$$\mathcal{F}_{ij}(n) = \left(\left[R_{-\frac{M}{2}+1}(n)\right]_{ij}, \dots, [R_0(n)]_{ij}, \dots, \left[R_{\frac{M}{2}}(n)\right]_{ij}\right), \tag{4.2}$$

built on the various lag moments and that was previously defined in Section 3 in Eq. 3.8. Our proposed feature-vector is defined as $\mathcal{K}^M(n)$, where $\mathcal{K}^M(n) \triangleq \mathcal{T}^M(n) \times \mathcal{F}^M(n)$. This represents the cartesian product between the two set of features previously defined, i.e., $\mathcal{K}^M_{ij}(n) = \left(\mathcal{F}^M_{ij}(n), \mathcal{T}^M_{ij}(n)\right)$ for each ij.

Ultimately, $\mathcal{K}^M(n)$ is represented as:

$$\begin{aligned}\mathcal{K}_{ij}^M(n) = \Bigg(&\left[\left[R_{-\frac{M}{2}+1}(n)\right]_S\right]_{ij}, \left[\left(\left[R_{-\frac{M}{2}+1}\right]_S\right)^{-1}\right]_{ij}, \dots, \\ &\left[\left(\left[R_{\frac{M}{2}}(n)\right]_S\right)\right]_{ij}, \left[\left(\left[R_{\frac{M}{2}}(n)\right]_S\right)^{-1}\right]_{ij}\Bigg),\end{aligned} \tag{4.3}$$

and is built upon the observed part of the lag moments and the inverse of the observed part of the lag moments, with negative and positive lags. M refers to the dimension of the feature-vector.

The *identifiability gap* represents the distance from the lowest prediction of a connected pair to the highest prediction of a disconnected pair, and allows us to assess if a model can separate the connected pairs from the disconnected pairs with a great margin for error, where the bigger the distance, the better the model. In a more formal manner, and similarly to how it is defined in the work done by Matta et al. [25], given a matrix-valued estimator $\mathcal{F}$, the *identifiability gap* of $\mathcal{F}$ is defined as

$$\mathcal{T}\left(\mathcal{F}\right) = \min_{ij:A_{ij}\neq 0} \mathcal{F}_{ij} - \max_{ij:A_{ij}=0} \mathcal{F}_{ij}. \tag{4.4}$$

An estimator $\mathcal{F}$ is structurally consistent if and only if $\mathcal{T}\left(\mathcal{F}\right) > 0$, meaning that, if and only if connected pairs are separated from disconnected pairs, in view of the entries of the matrix $\mathcal{F}$. In view of Lemma 2, presented by Machado et al. [23], the identifiability gap is larger for the features $\mathcal{K}^M(n)$ than the features $\mathcal{T}^M(n)$ and $\mathcal{F}^M(n)$ with high probability.

This tends to render the features $\mathcal{K}^M(n)$ more amenable to generalization when deploying supervised methods to cluster them. The overarching idea is that the proposed features exhibit the proper identifiability properties once either the variability of the off-diagonal entries of Σ_x is not too large (Section 3.4) or the exogeneous intervention is large enough (in view of inequality (3.13)). We demonstrate the competitive performance of FFNN trained with a properly normalized version of our features $\mathcal{K}^M(n)$ in Section 5.1, via numerical experiments for distinct connectivity regimes and noise correlation levels.

4.3 Training of the Neural Networks

This section provides information on the training and architecture of the NNs used in Chapter 5. Firstly, we explain the architecture of the CNN used in a variety of experiments, more concretely, in the simulations performed on Subsections 5.2.4, 5.2.5 and 5.2.6. Then, we explain the training and architecture of our proposed approach, used in Section 5.1.

4.3.1 Several Numerical Simulations

On several numerical simulations, in Section 5, we used a CNN as our model, with the feature vector consisting of the set of features proposed by Machado et al. [23] and defined in (4.2), or similar features and several adaptations (depending on the experiment). The architecture of the CNN, proposed by Machado [47], is shown in Fig 4.5.

For the training of the CNN model, we generate an Erdős-Rényi graph with $p = 0.5$ and generate time series with $\beta = 0$ and with 500000 samples (number of instants). We train under full-observability and with a training and test split of 80% and 20%, respectively, meaning that 20% of all edges in the graph (pair of nodes) are used for testing the model. From the training set, we use 10% for validation and, to avoid over-fitting, we stop training if the validation loss of the model does not decrease in 7 epochs. We repeat the training to obtain 10 different models. The best performing model in the test set is chosen. The best model is chosen based on the *accuracy* and *identifiability gap*, defined in Eq. (4.4), obtained in the test set. Although the training is done on a single graph and under full observability, the model generalizes well for the case of partial observability. Furthermore, the model is only trained with $p = 0.5$, where p is the probability of connection between each pair of nodes, but generalizes well for sparse and dense networks.

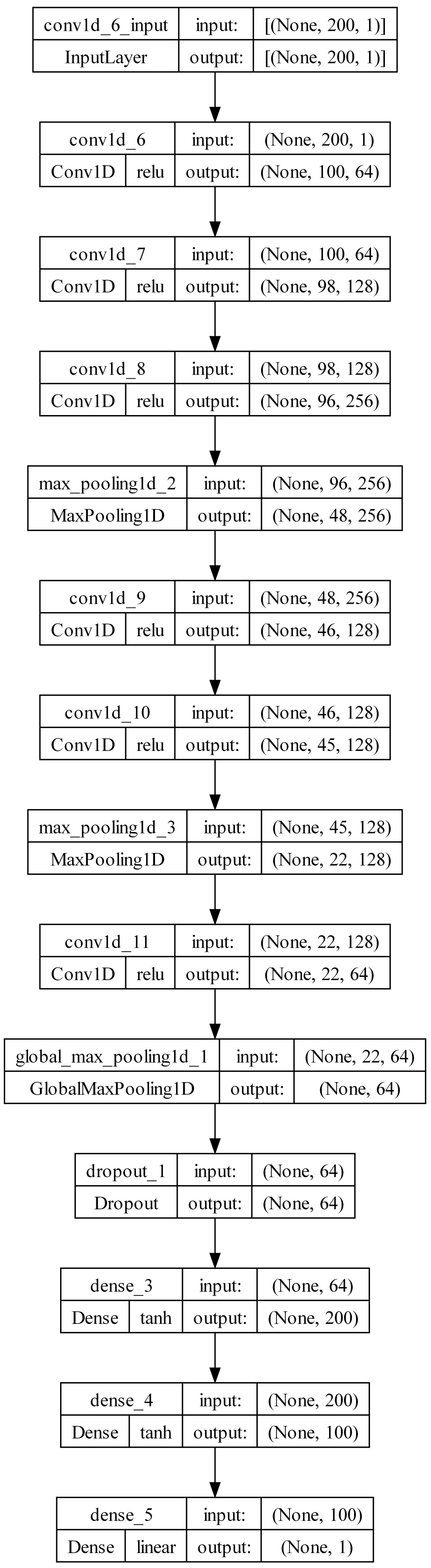

conv1d_6_input | InputLayer | input: [(None, 200, 1)] | output: [(None, 200, 1)]
conv1d_6 | Conv1D | relu | input: (None, 200, 1) | output: (None, 100, 64)
conv1d_7 | Conv1D | relu | input: (None, 100, 64) | output: (None, 98, 128)
conv1d_8 | Conv1D | relu | input: (None, 98, 128) | output: (None, 96, 256)
max_pooling1d_2 | MaxPooling1D | input: (None, 96, 256) | output: (None, 48, 256)
conv1d_9 | Conv1D | relu | input: (None, 48, 256) | output: (None, 46, 128)
conv1d_10 | Conv1D | relu | input: (None, 46, 128) | output: (None, 45, 128)
max_pooling1d_3 | MaxPooling1D | input: (None, 45, 128) | output: (None, 22, 128)
conv1d_11 | Conv1D | relu | input: (None, 22, 128) | output: (None, 22, 64)
global_max_pooling1d_1 | GlobalMaxPooling1D | input: (None, 22, 64) | output: (None, 64)
dropout_1 | Dropout | input: (None, 64) | output: (None, 64)
dense_3 | Dense | tanh | input: (None, 64) | output: (None, 200)
dense_4 | Dense | tanh | input: (None, 200) | output: (None, 100)
dense_5 | Dense | linear | input: (None, 100) | output: (None, 1)

Figure 4.5: Architecture of the CNN model proposed by Machado [47].

4.3.2 Our model

The model used in our proposed approach is a FFNN and the architecture of the model is represented in Fig. 4.6. We did not use the CNN model proposed by Machado [47] as the CNN would not learn with the use of the *Standard Scaler* on the features, for reasons we could not understand. Since the *Standard Scaler* was crucial in mitigating the shift of the features, we opted for relying on a FFNN model, instead of using a CNN. The model receives a feature vector, containing 200 features, as input. Next, follows two fully connected layers with 512 and 256 neurons respectively, with a *ReLu* activation function. We then insert a dropout layer in order to prevent overfitting of the model. Then, two more fully connected layers follow, with 200 and 100, with a *tanh* activation function. Lastly, the last layer consists of a single neuron, linearly outputting a single float value. The loss function used for training the model is the mean-squared error, defined by:

$$MSE = \frac{1}{n}\sum (y - \hat{y})^2, \tag{4.5}$$

where y represents the true values and $\hat{y}$ represents the predicted values. The chosen model optimizer is the $ADAM$ optimizer, which is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments. Using a loss function meant for regression allows the post processing of the values and the clustering of the predictions into 2 classes (connected and disconnected pairs). In the proposed solution, a K-Means algorithm is used to cluster the output of the FFNN model.

For the training of the model, we generate multiple graphs with different graph sizes (number of nodes) and for each graph, we generate time series with a different β value and number of samples. More concretely, the values of β range from 0 to 50 and the size of the training graphs range from 30 to 80 nodes. We then concatenate all *datasets* into a single one and train the model. We refer to *dataset* as the set of all the feature vectors associated with all the pairs of nodes in a graph. We use a validation split of 10% and stop training if the validation loss does not decrease in 7 epochs. We train several models and the best performing model in a test *dataset* is chosen. This test is generated from a graph with 40 nodes with only 20 of them being observable, from where 100000 samples are generated, with $\beta = 2$ and $\alpha = 1$. The models are trained with *datasets* generated from graphs and time series with different parameters. The best model is chosen based on the *accuracy* and *identifiability gap*, defined in Eq. (4.4), obtained in the test set. Although the training is performed under full observability and the test set is generated from a single graph, the chosen model shows good performance and generalizes well for the case of partial observability, for different regimes of connectivity and with values of β outside the training data.

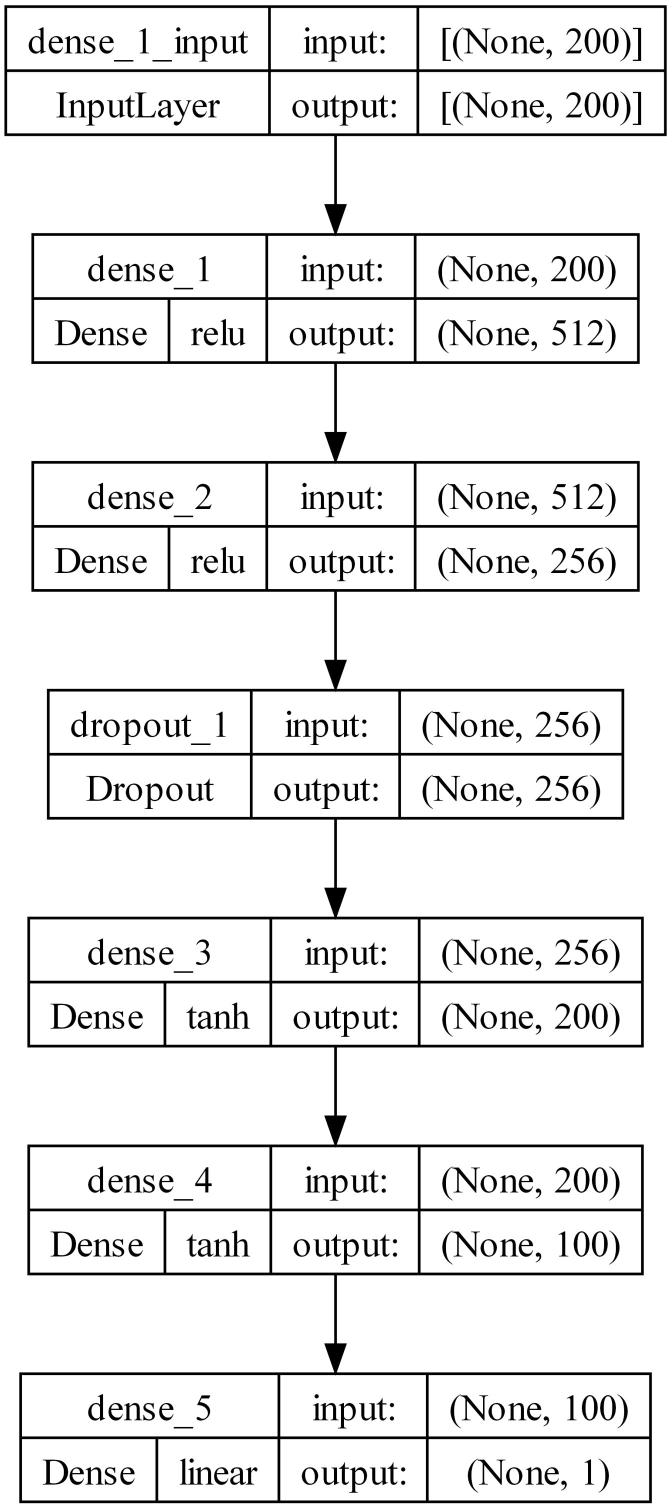


Figure 4.6: Architecture of our FFNN model.

Chapter 5

Numerical Simulations and Results

This chapter entails some of the numerical simulations performed, namely the most interesting experiments. Some of these simulations have been submitted for publication [1].

Firstly, we assess the performance of our approach in Section 5.1. On the other hand, in Section 5.2, we show a variety of different experiments and simulations.

5.1 Proposed approach performance assessment

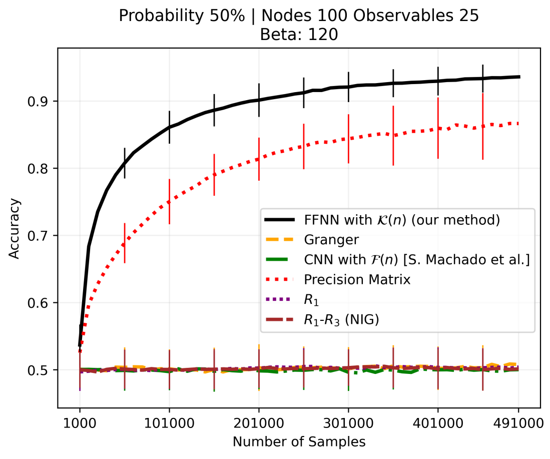
In this section, we assess the performance of a Feed Forward Neural Network (FFNN) model trained with the novel set of features. The generative process of the synthetic data is the same as explained in Section 4.1 and the training of the FFNN is explained in Subsection 4.3.2.

The experimental setup is as follows. Firstly, we generate an undirected adjacency matrix using the Erdős-Rényi model [45] for the synthetic graphs. For real world networks we skip this step as we already have an adjacency matrix. Secondly, from the adjacency matrix we use the Laplacian rule [46] to generate an interaction matrix. Thirdly, we generate the time series from the dynamical law defined in Eq. (3.15) and, lastly, we generate the feature vector, defined in Eq. (4.3), with $M = 200$ number of features. All the hyper-parameters in the generation of the Erdős-Rényi model and in Eq. (3.15) vary in every simulation, except for the value of α, which is set to 1. The brain Structural Connectivity network is the only case where we set $\alpha = 2$. For the Laplacian rule, we set $\rho = 0.75$ and $c = 0.9$.

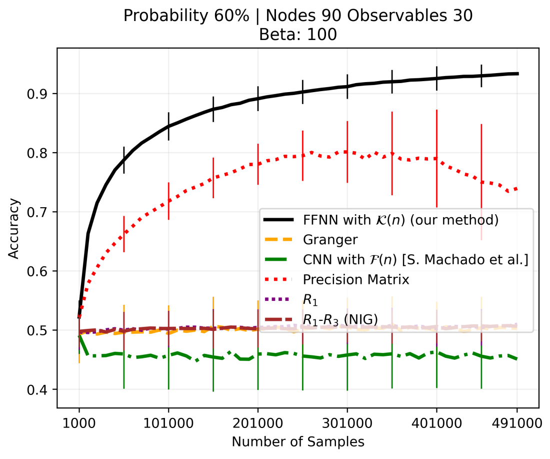
5.1.1 Results

We compare our method with some popular estimators that are tailored to structure identification of linear systems under partial observability: **i)** the Granger estimator (see Subsection 2.3.1); **ii)** the one lag R_1 estimator (see Subsection 2.3.2); **iii)** the precision matrix or inverse of the covariance (see Subsection 2.3.4); **iv)** the $R_1 - R_3$ estimator (see Subsection 2.3.3); **v)** the feature-based approach (see

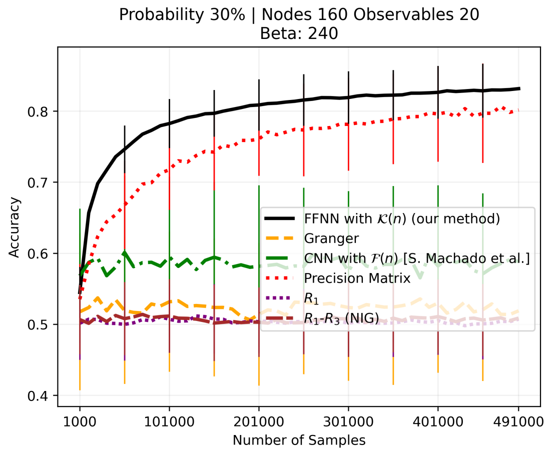
Subsection 2.3.5). We apply Gaussian Mixture (GM) at the matrix valued estimators to classify the pairs as connected or disconnected. Figure 5.1 shows the different estimators' accuracy as a function of the number of time series samples over undirected graphs (Erdös-Rényi, see Section 4.1) with distinct connectivity regimes and noise correlation levels. The black curve depicts the performance of our proposed method. Approximately 200 runs were computed for each and the average of the accuracy over all runs is plotted. Accuracy is a widely used metric in machine learning that measures the correctness of predictions made by a model. It represents the ratio of correct predictions to the total number of predictions made. We gain insights into how well a model can classify or predict outcomes accurately by assessing the accuracy. In our case, the accuracy is calculated between the ground truth adjacency matrix and the resulting binary matrix of the clustering of the predicted matrix $\hat{A}$.



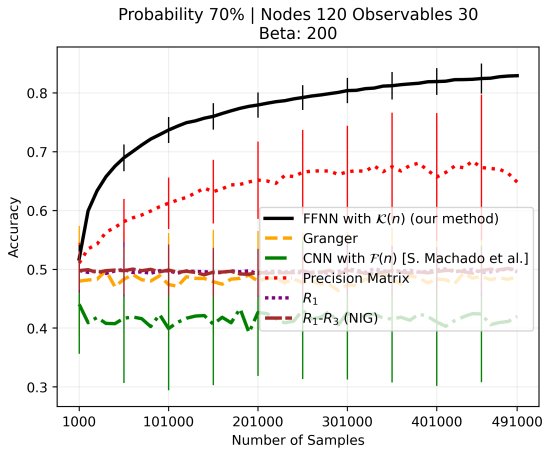
(a) Estimators' accuracy in a regime of 50% probability of connectivity, a graph consisting of 100 nodes with only 25 of them being observable. β is equal to 120.



(b) Estimators' accuracy in a regime of 60% probability of connectivity, a graph consisting of 90 nodes with only 30 of them being observable. β is equal to 100.



(c) Estimators' accuracy in a regime of 30% probability of connectivity, a graph consisting of 160 nodes with only 20 of them being observable. β is equal to 240.



(d) Estimators' accuracy in a regime of 70% probability of connectivity, a graph consisting of 120 nodes with only 30 of them being observable. β is equal to 200.

Figure 5.1: Comparing the performance of our approach against state-of-the-art solutions in different setups.

From Fig. 5.1, we observe that under the colored noise regime with partial observability, our proposed approach displays competitive performance against the referred estimators, beating them by a large margin in all the regimes considered. Remarkably, the FFNNs are trained under full observability, but generalize well for partial observability. From the plotted results, we can confirm our approach's robustness across distinct regimes of *partial observability* and *colored* noise, with different values of β. In addition, our proposed method is clearly more stable and regular, as the standard deviation of the performance is rather low when comparing to the other estimators. Furthermore, all the other estimators (except the precision matrix) seem to completely collapse under the regime of *colored noise*, and the predicted network seems to be random (with an accuracy of $\approx 50\%$).

We also compared the performance of the estimators in a real world network. This network represents the Brain Structural Connectivity matrix of a human [48]. The connectivity regime calculated was $\approx 58\%$. The network is composed of 90 nodes representing cortical regions of interest and we set $\alpha = 2$, in Eq. (3.15). The adjacency matrix [1], shown in Fig. 5.2, is calculated by performing a threshold at 0.0005, where all the values below are set to 0 and set to 1 otherwise. The performance of the estimators is reflected in Fig. 5.3. Although the precision

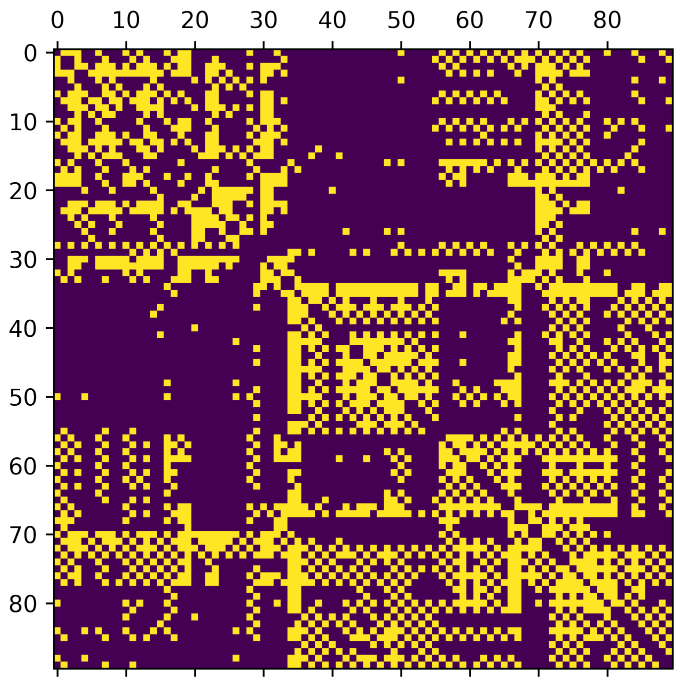


Figure 5.2: Brain Structural Connectivity matrix [48], with 90 cortical ROIs.

matrix estimator start off well, with a performance similar to our approach, our method outperforms it in the long run. From Fig. 5.3, we also depict the stability of our network reconstruction approach against other the approaches.

[1]For the sake of a Structure Identification problem, we force the matrix to be symmetric, by copying the upper triangle into the lower triangle of the connectivity matrix.

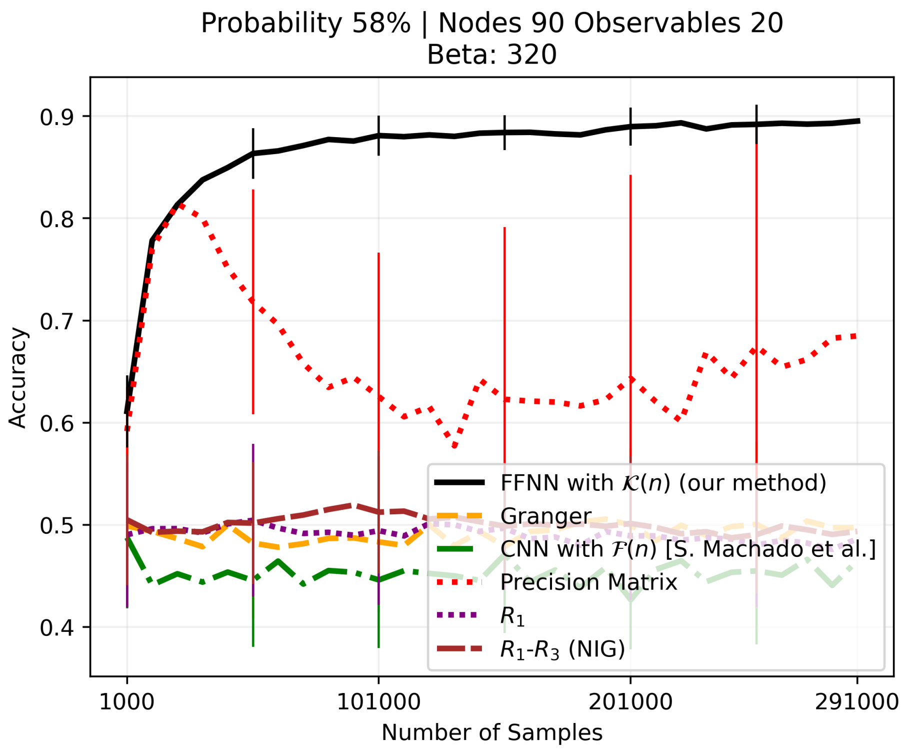


Figure 5.3: Estimators' accuracy in a real world network. The network is a Brain Structural Connectivity matrix [48].

Furthermore, we simulated in a small-world network (see Section 4.1.1), using the Waltz-Strogatz model, with $M = 60$ nodes, $k = 40$ neighbors and $p = 0.5$ of probability of rewiring. Figure 5.4 shows the network we tested on, and Fig 5.5 shows the overall performance of the estimators.

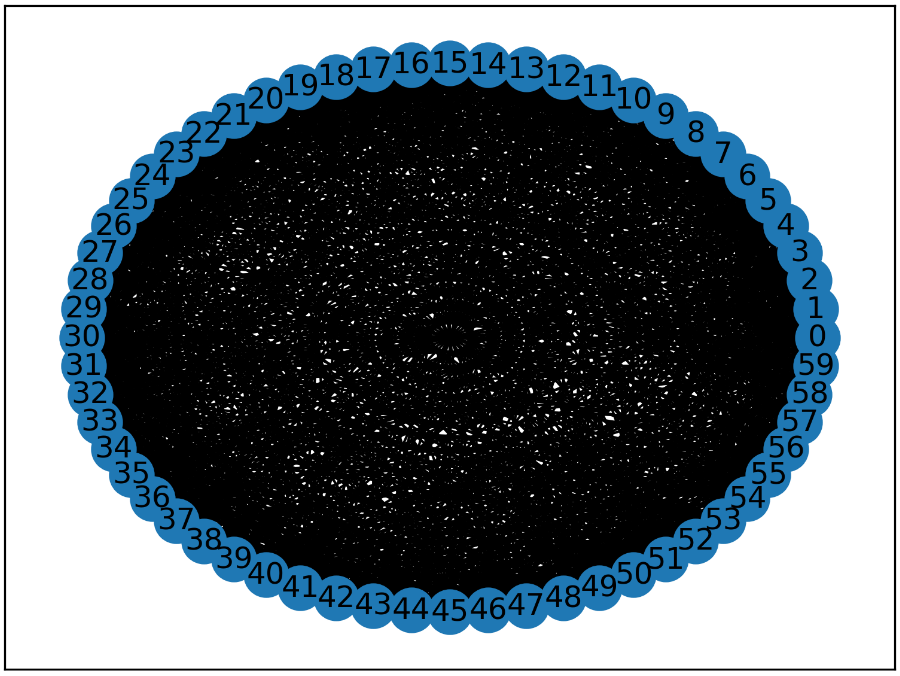


Figure 5.4: Example of the small-world network with $N = 60$, $k = 40$ and $p = 0.5$.

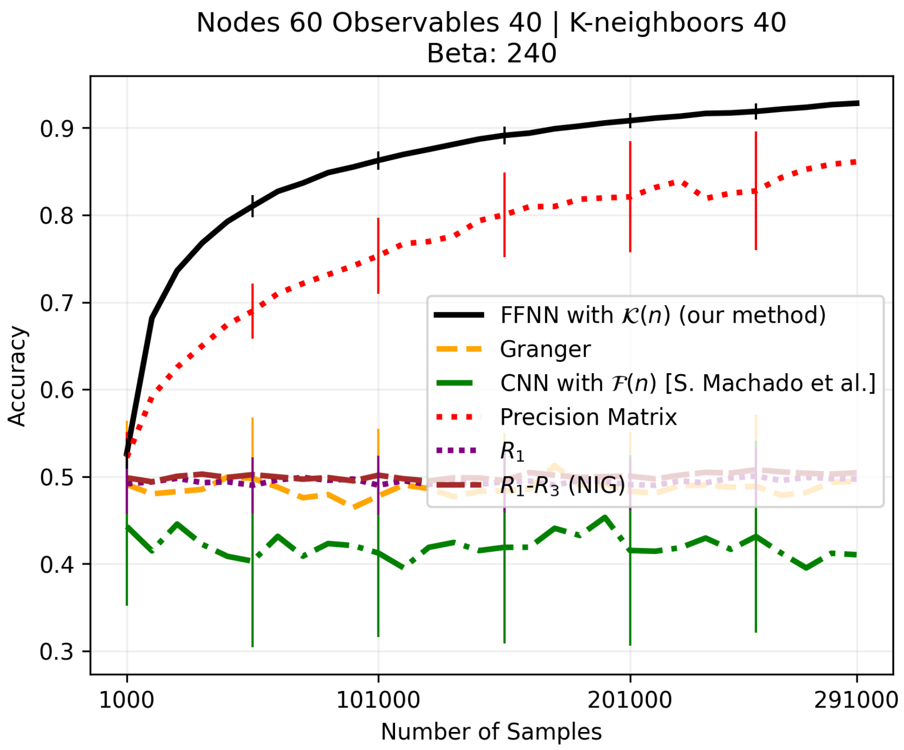


Figure 5.5: Estimators' accuracy in a small-world network using the Waltz-Strogatz model.

In addition, we also compared the performance of all estimators in a real-world food web network [49], from the island of St. Martin. It contains 44 different species (number of nodes). We assume indirectness of the network [1]. The connectivity regime is $\approx 20\%$.

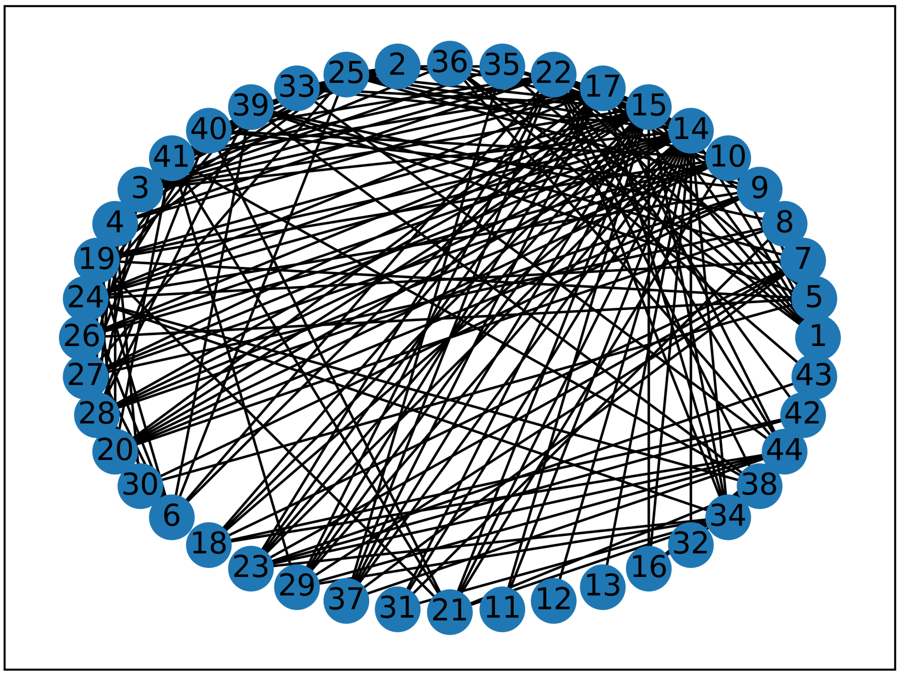


Figure 5.6: Graph of the food web predator-prey network.

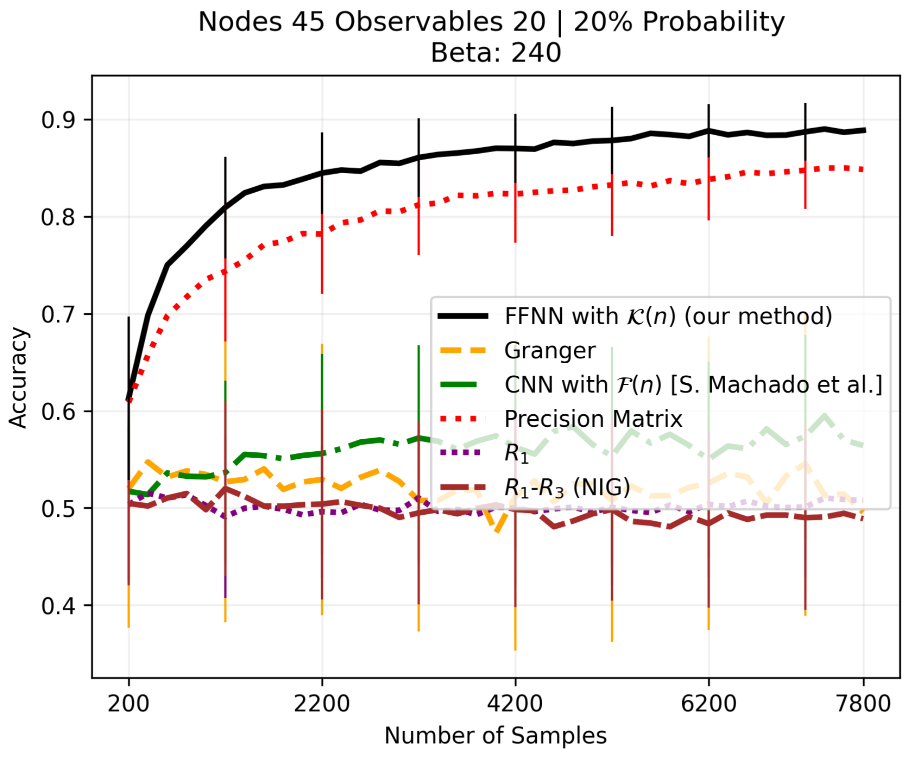


Figure 5.7: Estimators' accuracy in a food web predator-prey real network.

Our approach shows better performance, even in the case of a sparser network ($\approx$ 20% of connectivity). Although the precision matrix estimator is close, the margin between our approach and the precision matrix estimators' accuracy is still $\approx$ 5% .

For a more complete performance assessment, we compared the estimators performance as β increases, as p increases and as the number of observable nodes increases. Figures 5.8, 5.9 and 5.10 illustrates these results.

Figure 5.8 shows that our method outperforms all other state-of-the-art estimators for $\beta \gtrapprox 2$. For $\beta \leq 2$, our approach performs worse than almost all the other estimators. This may be a consequence of the model's generalization for a large variety of β. In Fig. 5.9, it is evident that for sparser regimes ($\lessapprox$ 30%) the precision matrix has a better performance. However for every other regime, our approach outperforms all estimators. Figure 5.10 exposes that our algorithm is robust under various regimes of *partial observability*.

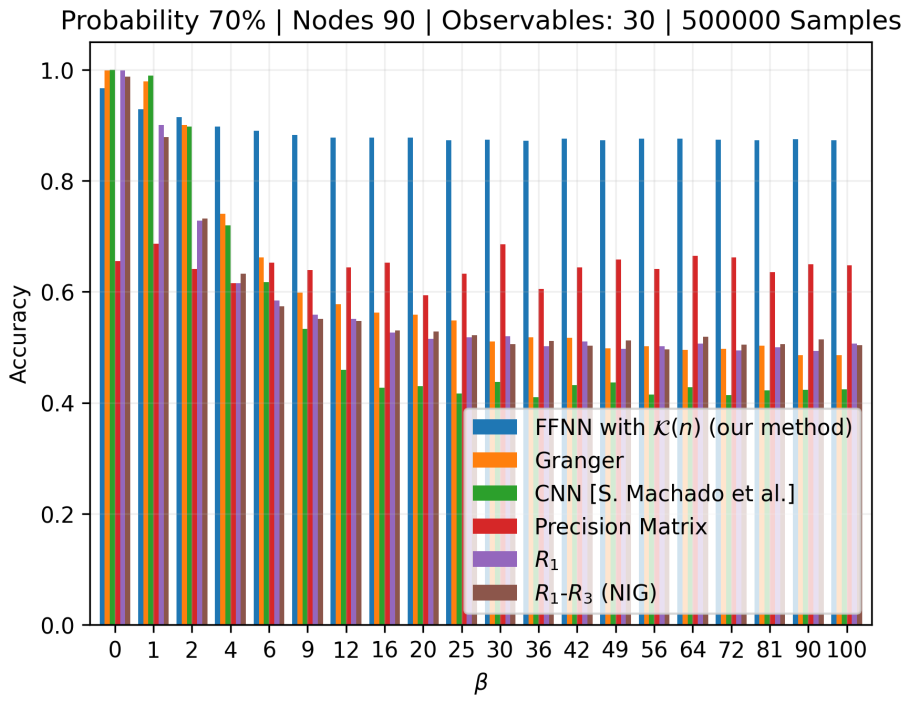


Figure 5.8: Estimators' accuracy as β changes.

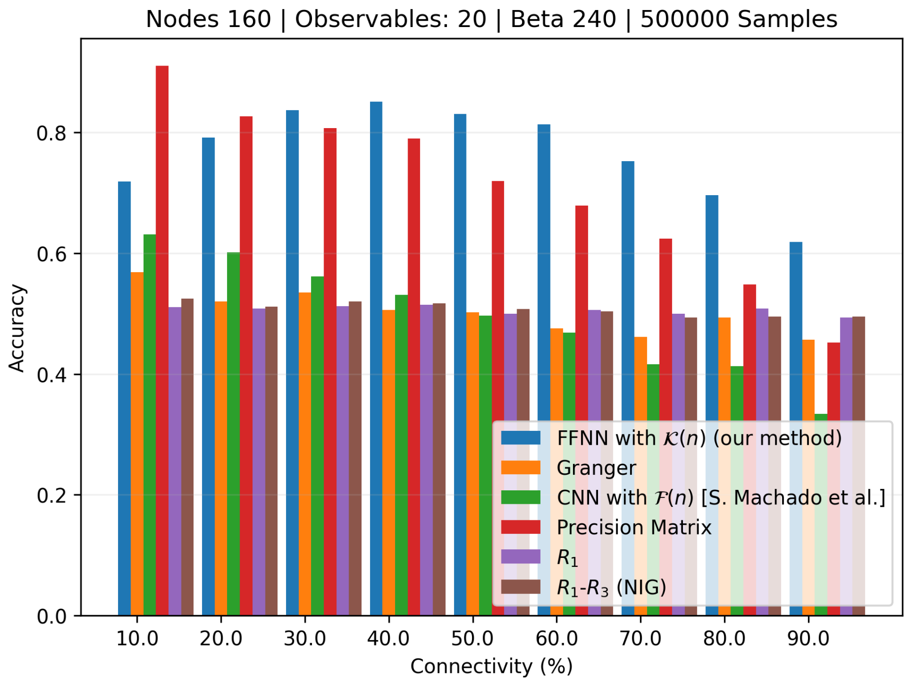


Figure 5.9: Estimators' accuracy as the probability of connectivity (p) changes.

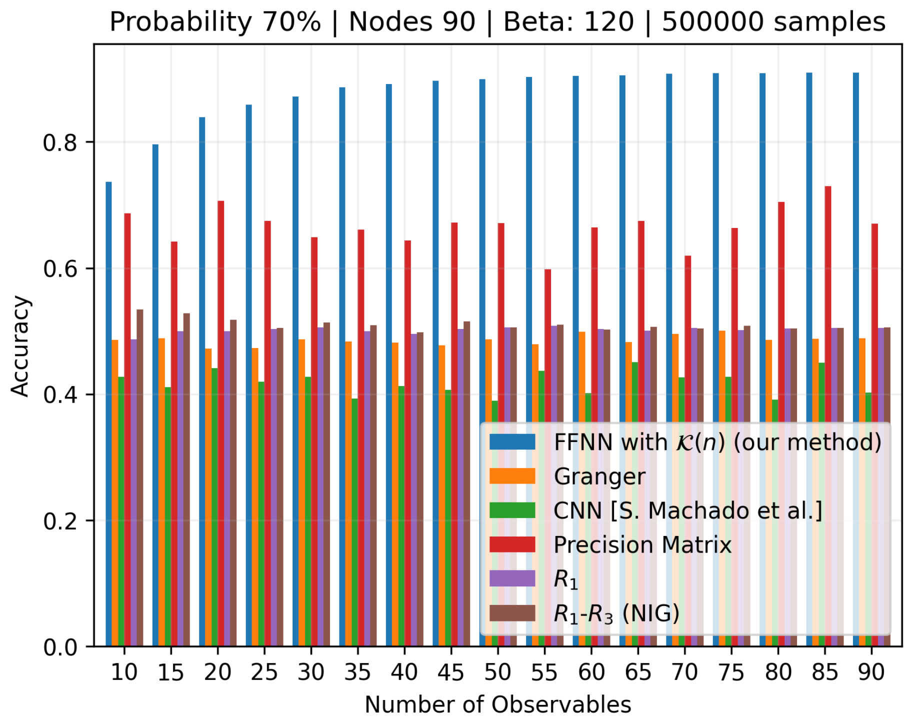


Figure 5.10: Estimators' accuracy as the number of observable nodes increases from 10 to 90 nodes.

Lastly, we assess the impact that the introduction of exogenous intervention (see Section 3.5) has on controlling the noise signal and stabilizing the system. For this, we compared the accuracy of all the estimators across various runs and plotted the average of all runs, as the level of the exogenous interventions increases, i.e., as the value of γ increases (see Eq. (3.12)). This is shown in Fig. 5.11.

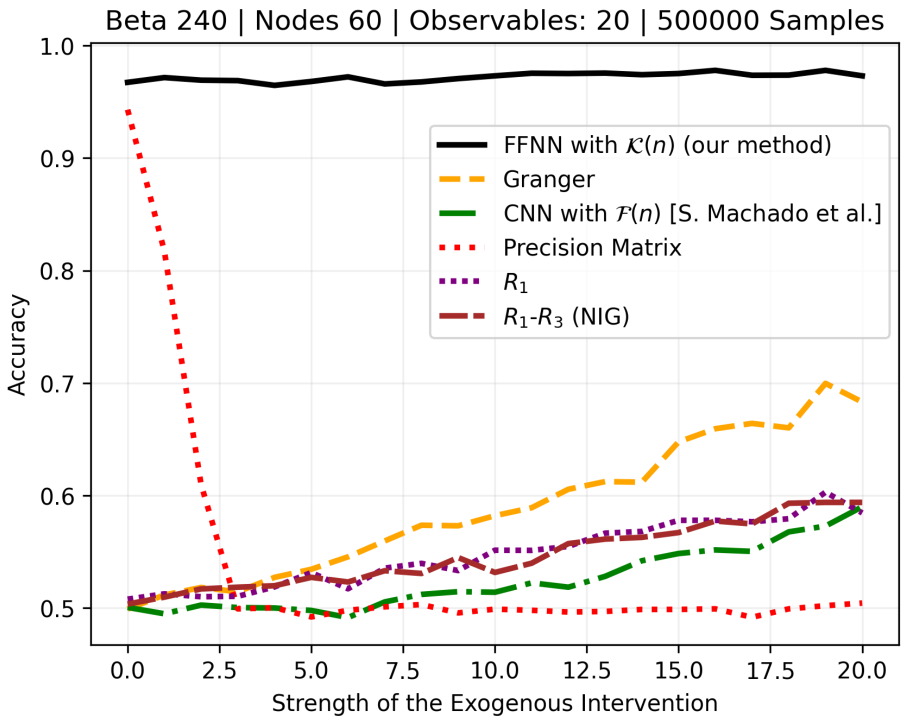


Figure 5.11: Estimators' accuracy as the strength γ of the exogenous intervention increases.

It is clear that, from the analysis of Fig. 5.11, the exogenous interventions improve the performance of most estimators in recovering the structure of the graph. However, the precision matrix performance drops drastically with exogenous interventions and becomes a random estimator when the strength of the exogenous interventions is $\gtrapprox 2.5$. As the level of exogenous interventions increases, the value of σ^2 (diagonal values of the covariance matrix of the signal) also increases, which yields the precision matrix, defined as R_0^{-1}, less separable, since $R_0^{-1} = \frac{1}{\sigma^2}(I - A^2)$. This is a clear disadvantage of the precision matrix estimator.

5.2 Other Simulations

In this particular section, we present a compilation of interesting experiments that were simulated throughout the semester. These experiments followed a particular line of thought and are, in most part, related to one another.

In Sections 5.2.1 and 5.2.2, we attempt to directly cluster the features, under colored noise, with known clustering algorithms or reducing the dimensionality of the feature-vector. In Section 5.2.3, we apply Mutual Information (MI) between each lag-moment of the correlation and the ground-truth structure. In Sections 5.2.4, 5.2.5 and 5.2.6, we tackle the regime of diagonal noise, in order to find the best version of the Convolutional Neural Network (CNN), used in the work of Machado et al. [23]. We try to optimize the dimension of the feature-vector, we discuss the influence that the training regime has on the models and, from these, we attempt to build a new approach that outperforms the default CNN model.

Section 5.2.7 reveals the consistency of the precision matrix as an estimator for recovering the structure of the network of a Networked Dynamical Systems (NDS).

In Sections 5.2.8 and 5.2.9 we present different approaches for recovery of the network and possible features to be introduce in the feature-vector.

5.2.1 PCA and K-Means

This experiment consisted of applying Principal Component Analysis (PCA), explained in appendix C, directly to the features. In summary, PCA is a widely used statistical technique that effectively reduces the dimensionality of datasets while preserving the maximum amount of information. We then use the K-Means algorithm, described in appendix A, to group the reduced dimensionality features from the PCA into clusters in order to develop an estimator.

Firstly, we observed the projection of the feature-vector, generated from synthetic data (see Section 4.1) with $\alpha = 0.1$, with diagonal ($\beta = 0$) and colored noise ($\beta = 0.2$) into a single component of the PCA.

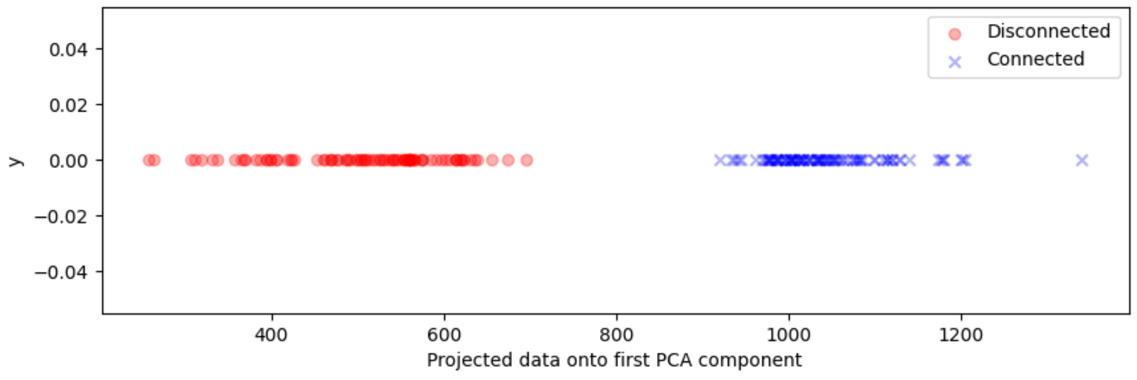


Figure 5.12: PCA projection of data with diagonal noise.

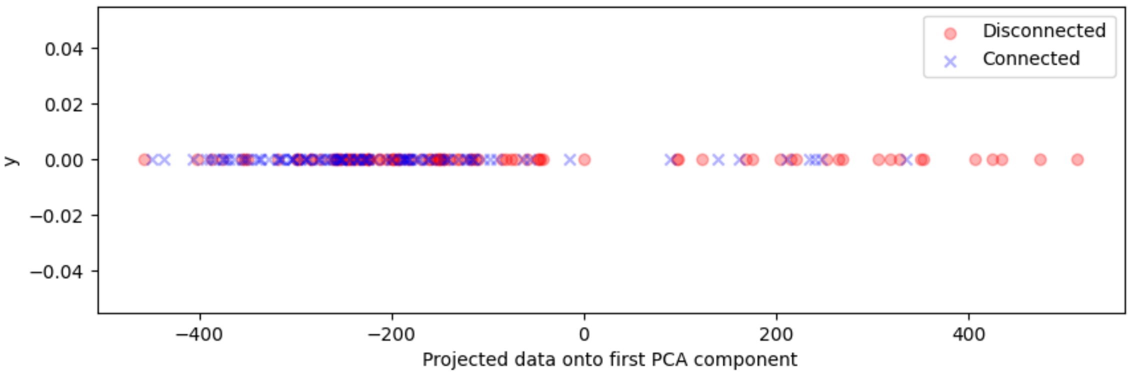


Figure 5.13: PCA projection of data with colored noise.

As it is observed in Figs. 5.12 and 5.13, the introduction of colored noise mixes up the features in a 1D space, leading to an estimator that is not capable of consistently separating the features. However, the metrics of this estimator were still measured as the number of samples increased, as a last concluding experiment, shown in Fig. 5.14.

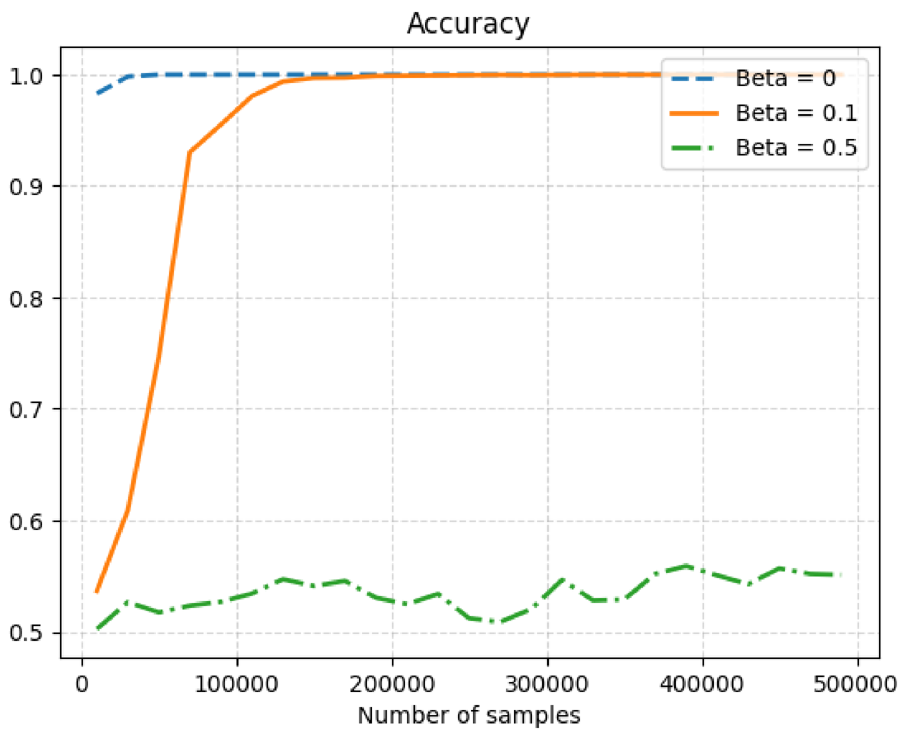


Figure 5.14: PCA and K-Means accuracy as the number of samples increases, with different β values.

It is clear that this estimator does not perform well as β increases, and when $\beta = 0.5$ it seems to lean towards a random estimator, that does not increase performance metrics as the number of samples increases. Using PCA to reduce to the feature-vector to one dimension is clearly not a good approach, as the features may not be able to hold separability in a one dimensional space. This results shows the increased difficulty of the introduction of *colored* noise.

5.2.2 Clustering the features

This experiment consisted of applying a clustering algorithm, namely the *K-Means* algorithm or the GM algorithm, directly to the 200 features. Both these algorithms are described in detail in appendix A and B respectively. A major problem with this approach was that the 2 centroids found would sometimes flip in order, which meant that features that were from class 1 sometimes were classified as class 1 and sometimes as class 2, depending only on the initialization of the centroids. This problem was fixed by sorting the centroids by their norms. However, as β increases, it is not clear that the higher norm belongs to the class of connected pairs and further experiments must be realized.

The experiment was repeated 20 times and the accuracy shown in Fig. 5.15 is the average of the 20 runs. In each run, a different graph was generated (with 20 nodes and a probability of an edge between each node pair of 50%) and the time series length varies from 10000 to 500000 with a step of 10000. We set the value of α to 1.

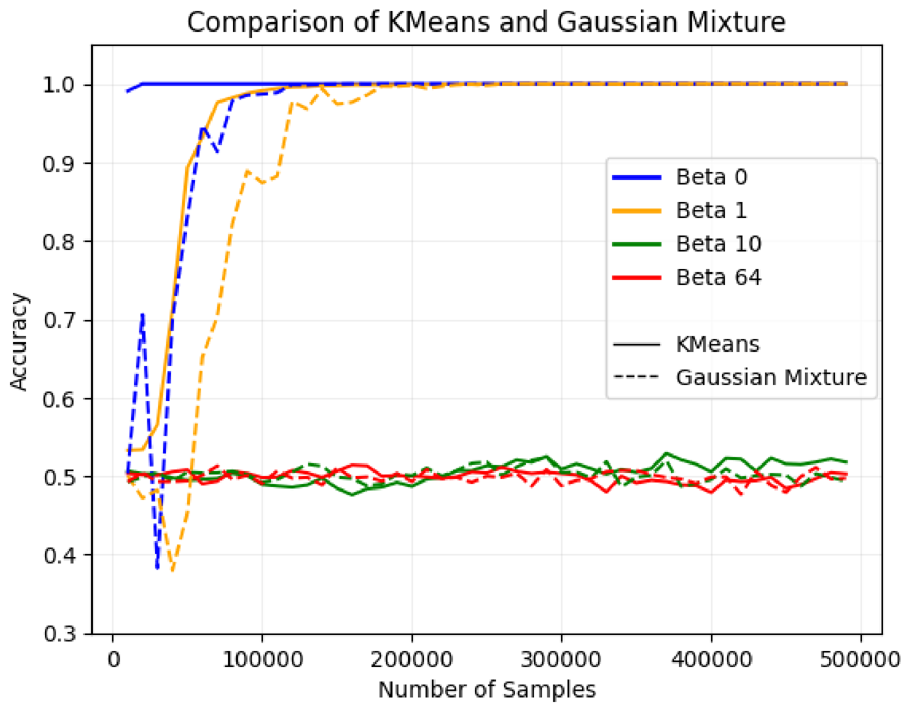


Figure 5.15: K-Means vs GM with $\beta \in \{0, 1, 10, 64\}$ as the number of samples increases.

Although this approach leads to good results with low values of $\beta = 0$ and $\beta = 1$, with both clustering algorithms managing to achieve 100% accuracy with a fairly low amount of samples, as β increases, the performance degrades and eventually stagnates at around 50% (random predictions). Furthermore, this experiment may indicate that the *K-Means* algorithm is better suited to this specific problem than the GM algorithm. This may be explained from the fact that the *K-Means* algorithm is more suitable for linearly separable features than the GM algorithm, which assumes that the features are explained by a finite set of gaussian distribuitions. This may also explain why the GM Model is more unstable as the number of samples increases, when compared to the *K-Means*.

It is also evident, in this experiment, the problem of sample complexity, as the features become increasingly separable as the number of samples increases, which is to be expected.

5.2.3 Mutual Information based formulation

MI measures how much information two random variables share. It expresses the dependency between two variables and in other words, MI can be used to quantify the amount of information one random variable contains about another.

The MI is expressed as follows:

$$\mathrm{MI}(X,Y) = H(X) + H(Y) - H(X,Y)$$

where $H(X)$ is the entropy of X, $H(Y)$ is the entropy of Y and $H(X,Y)$ is the joint entropy of X and Y.

The entropy of a random variable X and the joint entropy of two random variables X and Y can, respectively, be expressed as:

$$\begin{aligned} H(X) &= -\textstyle\sum_{x\in X} p(x)\log p(x) \\ H(X,Y) &= -\textstyle\sum_{x\in X}\sum_{y\in Y} P(x,y)\log P(x,y), \end{aligned}$$

where $P(x,y)$ is the joint probability of these values occurring together. In Fig. 5.16, we depict the MI between matrix valued estimators and the underlying ground-truth graph structure.

For the matrix-valued estimators, we considered the Granger and covariance matrices $\widehat{R}_k(n)$ with distinct lag-moments k. The abscissa indexes the distinct lags considered except for the first entry which was reserved for the Granger estimator. This experiment lies consistent with a recent observation in the work done by Machado [47]. In particular, for the most part, information about the underlying graph structure lived in the low-lag moment estimators.

When we compute the MI score with randomly generated graphs (i.e., the underlying graph structure is wrong with high probability), then we observe much lower values of MI as illustrated in Fig. 5.17. We also remark that the width of the MI curve depends on the sparsity of the network. The sparser the network, the wider the base of the bell-shaped MI graph, i.e., more structural information lies in higher order lag-moments. Fig. 5.18 shows an experiment where the underlying true graph was generated as an Erdős-Rényi realization for distinct p, i.e., probability of edge drawing at each pair (see Section 4.1). The smaller the p, the sparser the network.

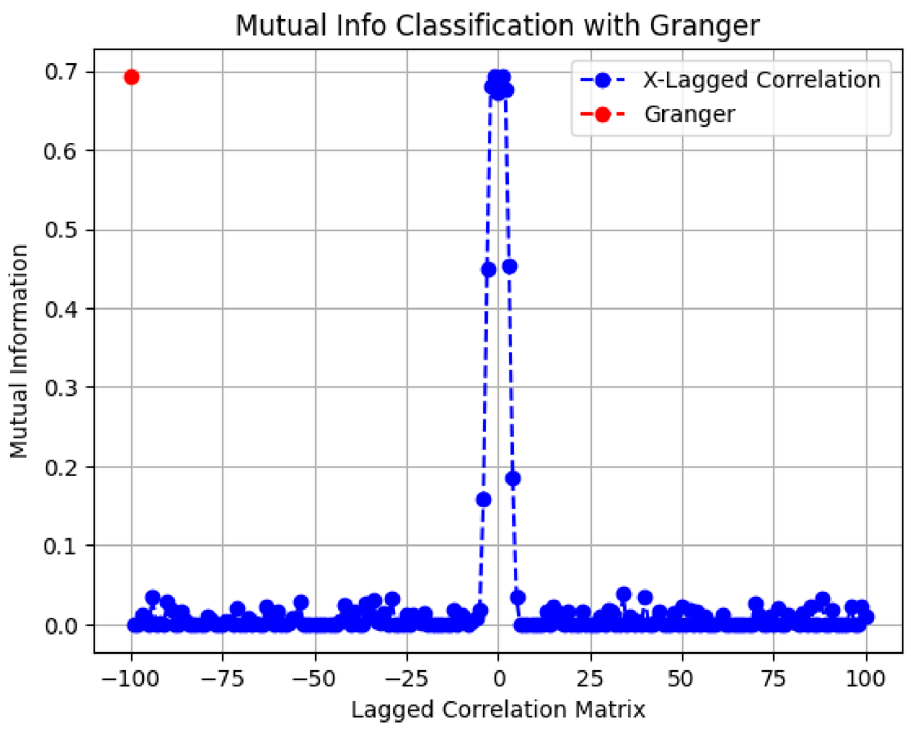


Figure 5.16: MI is maximal at the Granger estimator and low-lag estimators.

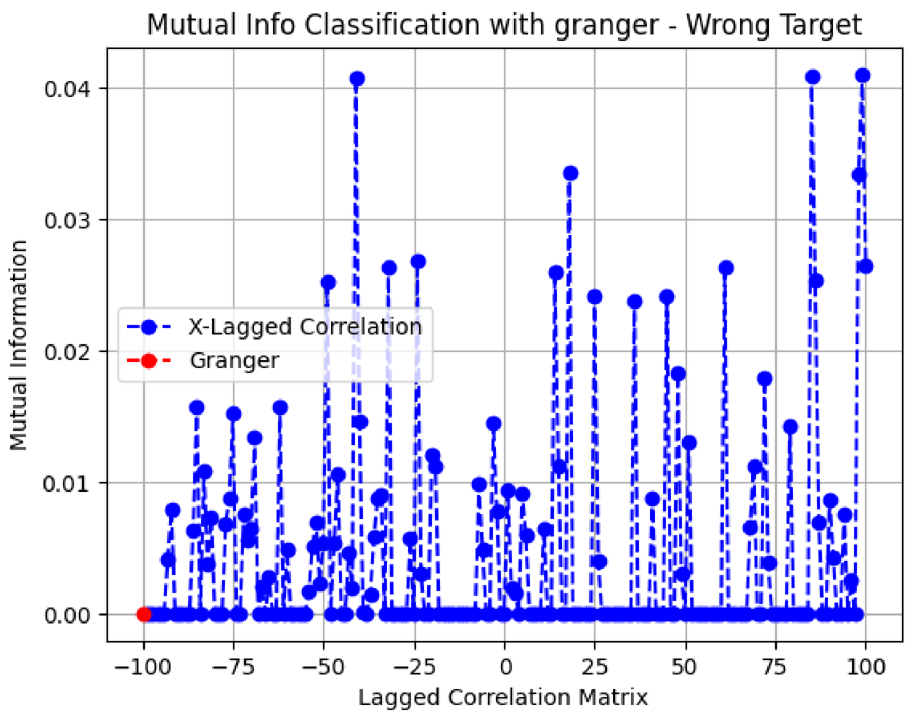


Figure 5.17: MI is low when the underlying structure does not match the correct one.

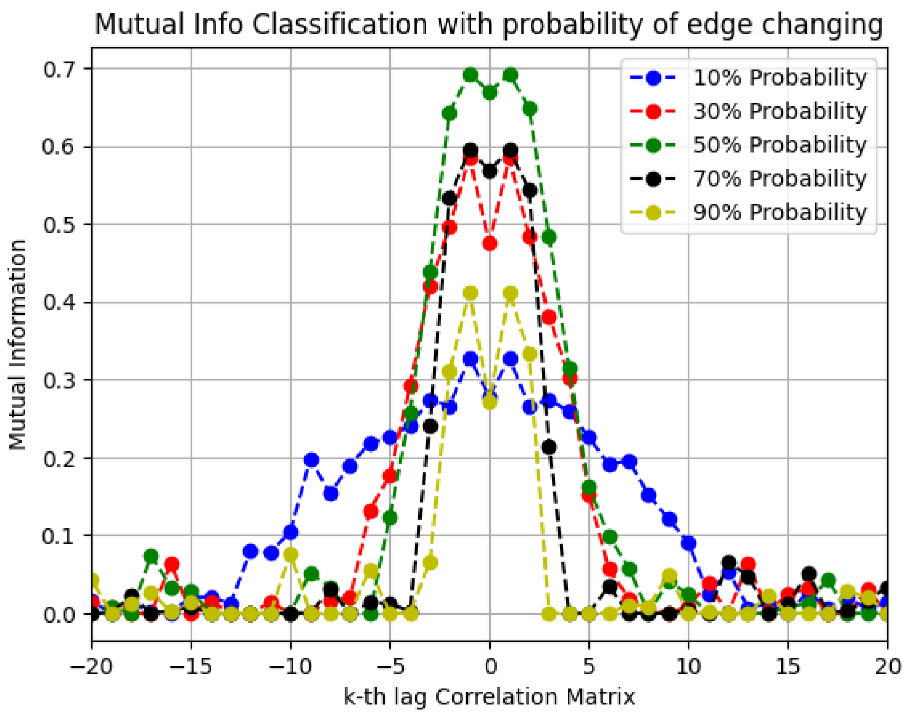


Figure 5.18: MI exhibits wider base for sparser networks. Higher order lag-moments play a role in the estimation.

5.2.4 Feature-vector dimension

The observations discussed in Section 5.2.3 serve as motivation for this section. We focus on exploring if the knowledge of the sparsity of the (unknown) network can aid on the search for the right separating hyperplane (or manifold) as we might need to weigh higher order lag moments accordingly. It is also relevant in finding the optimal dimension for the feature-vector. As we previously observed, the number of optimal lags to introduce as features is highly dependent on the connectivity regime. From the mutual information, discussed in Section 5.2.3, higher density regime (higher p value) tends to need less lags to unveil the structural connectivity. In order to discover the correct dimension of the feature-vector we conducted the following experiment:

We trained 4 different CNN with varying number of features: 500, 300, 200 and 100 features, where the feature vector is defined as:

$$\mathcal{F}_{ij}^{(n)} \triangleq \left(\left[\widehat{R}_{-\frac{M}{2}+1}(n) \right]_{ij}, \ldots, \left[\widehat{R}_{\frac{M}{2}}(n) \right]_{ij} \right),$$

where M represents the dimension of the feature vector.

We then tested the different CNN in a graph of 40 nodes, with only 20 of them being observed, with a probability of connection of 50%. The time series range from 1000 to 100000 samples, with a step of 2000.

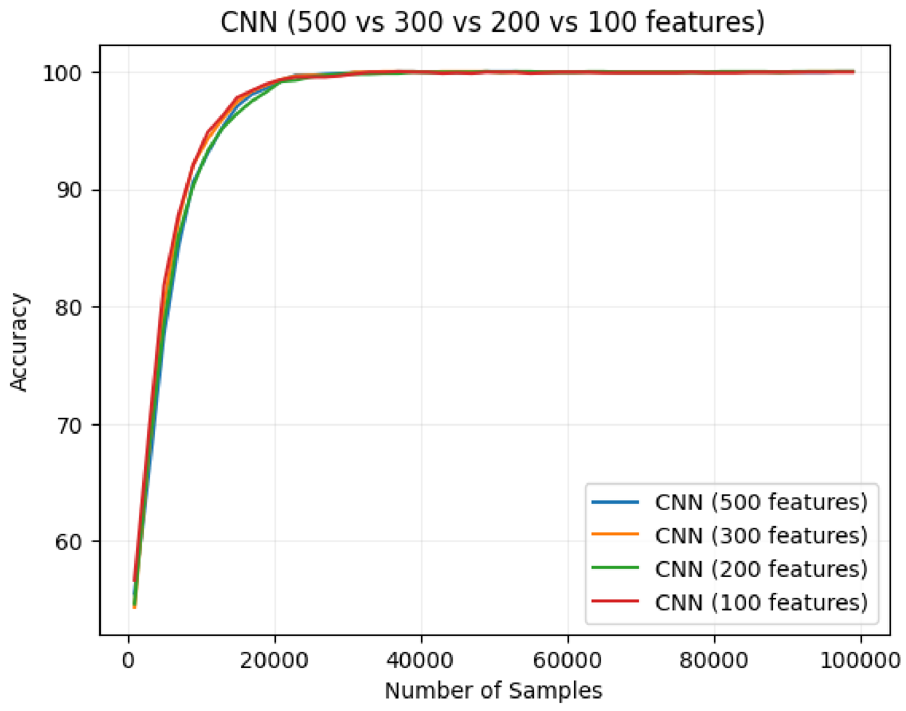


Figure 5.19: Accuracy of CNN with varying number of features.

It is to be noted that the best models for each test regime are circled in white, to facilitate the reading of the figure. The CNN with a feature vector of size 100 seems to have a slightly better performance than the others CNN, but no clear pattern between the dimension of the feature-vector and the performance of the model is observed.

From this, we decided to perform a more detailed and comprehensive simulation. Different models were trained in different regimes of connectivity with different number of features. All the trained models were then compared across all regimes.

As plotting the accuracy as the number of samples increased would result in a confusing cluster of lines, as we trained in 10 different regimes and with 10 different number of features, resulting in 100 different models, we decided to assess the number of samples needed to reach a certain level of accuracy in order to compare the different models.

Figures 5.20, 5.21 and 5.22 represent the test in the regime of 30%, 50% and 70% probability of connectivity, respectively.

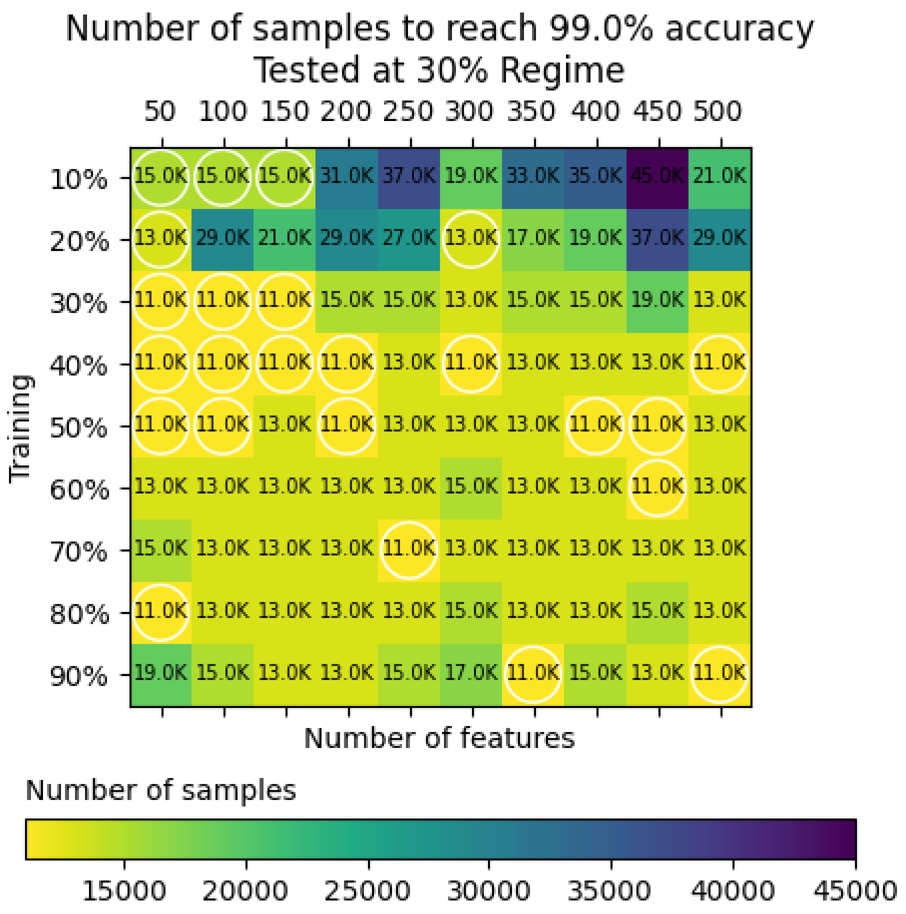


Figure 5.20: Number of samples necessary to reach 99% accuracy, on the test regime of 30% while varying the number of features and the node connectivity probability of the training setup.

It is clear that denser networks are harder to correctly infer their structure than sparser networks, shown by a general increase, across all the training regimes and the different number of features, of the number of samples necessary to reach an accuracy of 99%. It seems that, from the performed simulations, there is no per-

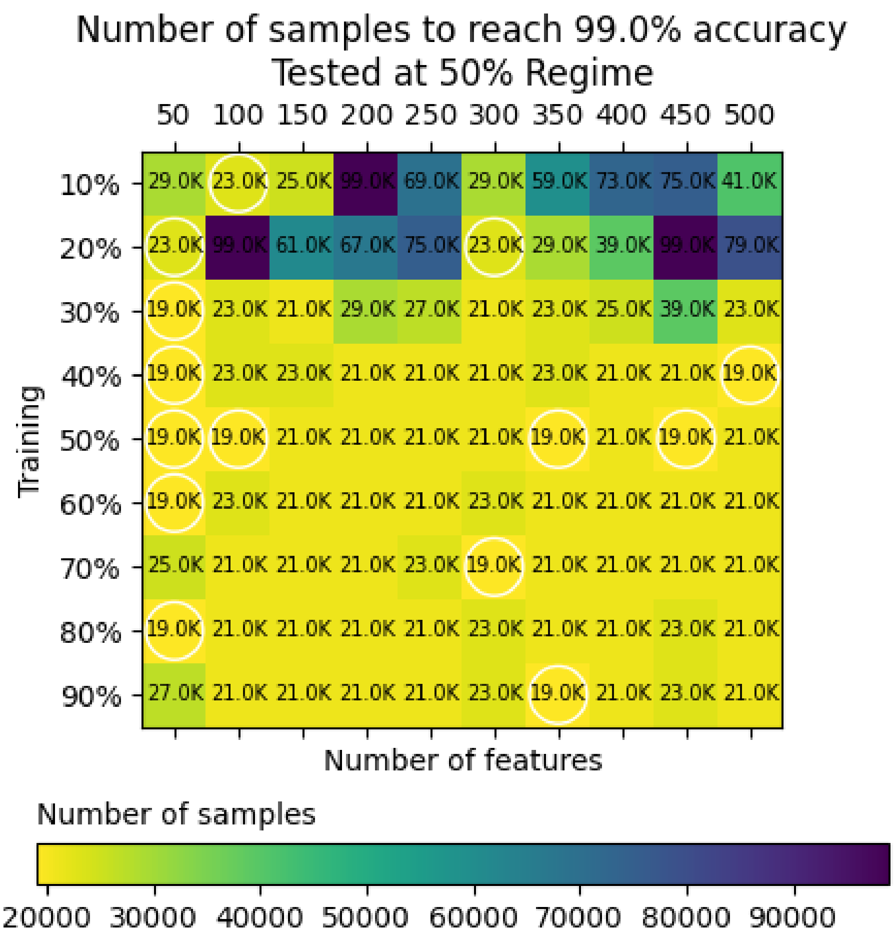


Figure 5.21: Number of samples necessary to reach 99% accuracy, on the test regime of 50% while varying the number of features and the node connectivity probability of the training setup.

fect dimension for the feature-vector, as the results greatly depend on sparsity of the training regime and the test regime. However, the low number of features models, seemingly, tend to perform better or similarly to models with more features. In general, we can say that the best perfoming models for a specific test regime are the models trained in similar regimes of connectivity. From this last remark, the different training regimes are explored in the next section.

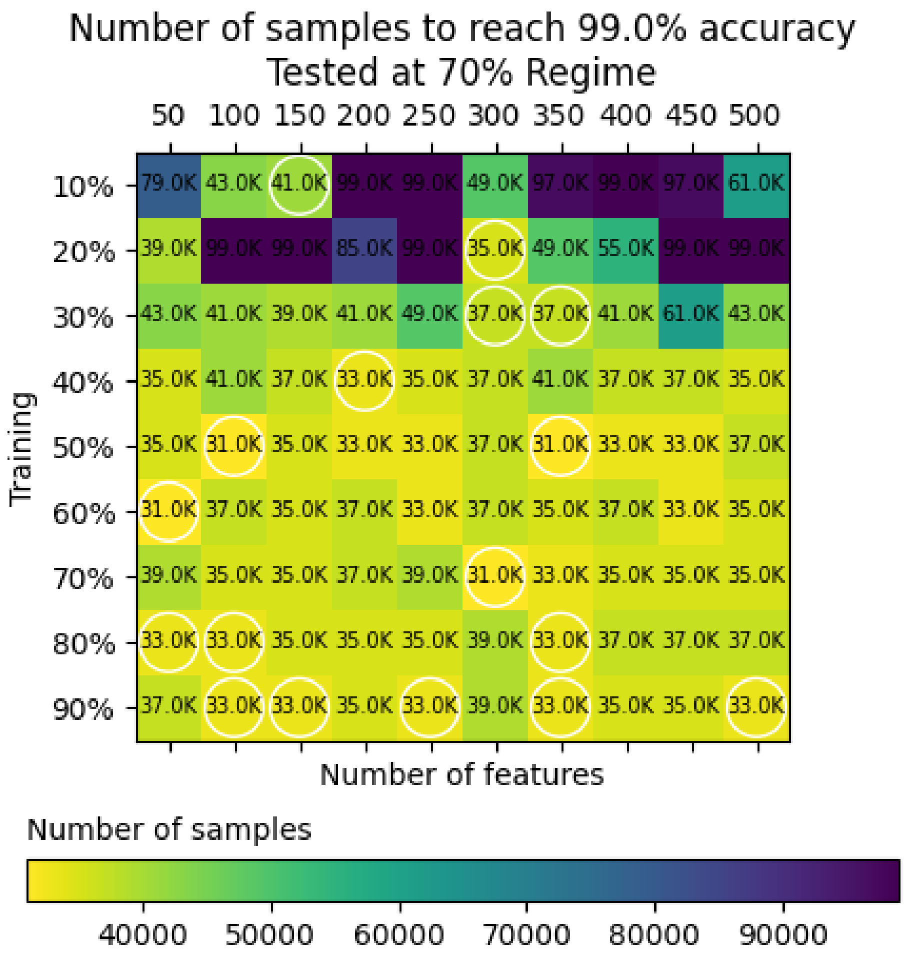


Figure 5.22: Number of samples necessary to reach 99% accuracy, on the test regime of 70% while varying the number of features and the node connectivity probability of the training setup.

5.2.5 Influence of the training regime

Discovering the perfect training regime for a model is not a trivial problem. There are many hyper-parameters that influence the training and consequently the performance of a model, such as the probability of connectivity, the number of features to include, the number of nodes of the generated networks, whether to train in full-observability or in partial-observability, the amount of samples to generate, and so much more.

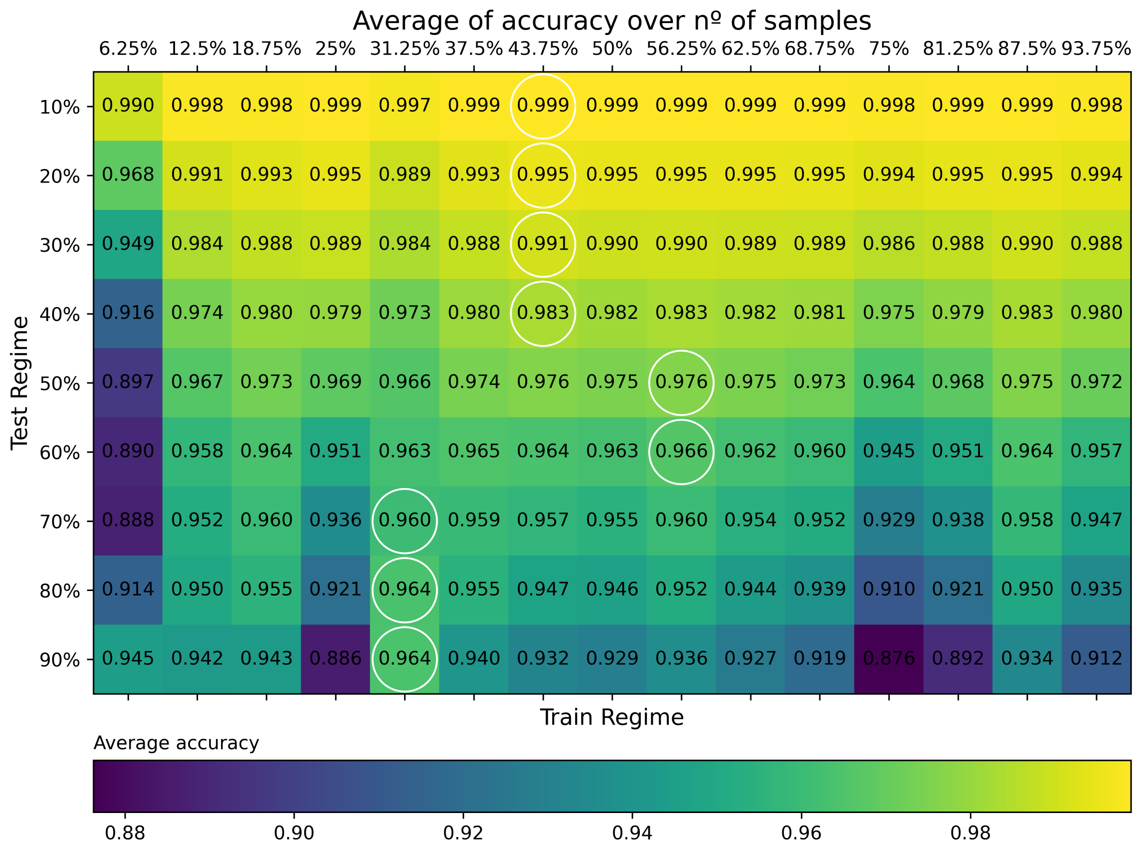
This section delves into the nuanced variations in training models across different connectivity regimes, in concrete, in training models with datasets built with different p, where p stands for the probability of each edge in the graph representing a connection between two nodes on an Erdős–Rényi random graph model, which ultimately reflects the sparsity of the network. Models were trained in 14 different regimes of connectivity and all the models were tested in regimes that range from a probability of 10 to 90%, in order to find a pattern.

In our study, for each training regime, we trained ten different models to identify the best-performing model for the task. To evaluate the performance of these models, we employed two key metrics: *accuracy* and *identifiability gap* (see Section 4.2).

We assessed the *accuracy* and *identifiability gap* for each trained model to determine the best model. Our objective was to identify the model with the highest *accuracy* and *identifiability gap* in a test dataset.

We then tested the 14 best models, each model trained in a different connectivity regime, in 10 different test regimes. As plotting the *accuracy* as the number of samples increased would result in a confusing cluster of lines, as we trained and tested in multiple regimes, we instead computed the general *accuracy* of the models over the number of samples, represented in Figure 5.23. It is to be noted that the best models for each test regime are circled in white, to facilitate the reading of the figure.

The average *accuracy* of a model's performance can be misleading, as a model that starts off poorly but then proceeds to outperform other models can have its average *accuracy* dragged down from the initial performance. For example, a model that is consistently at 97% *accuracy* as the number of samples increases but never reaches 100% *accuracy* can be wrongly considered a better model than a model that starts off badly but reaches the 100% *accuracy* at a certain number of samples. Furthermore, the *accuracy* fails as a metric for imbalanced datasets, which becomes the case when the test regimes largely deviate from the 50% connectivity. For this reason, we also decided to verify the number of samples needed to reach a certain *accuracy* threshold, in order to compare the different models. This is shown in Figure 5.24, where the lower the samples needed to reach 99% *accuracy*, the better the model. It is to be noted that the brighter the square the better the model performance is. Setting the *accuracy* threshold to 99% guarantees that the imbalance issues do not influence high performance models. For example, a model achieving 90% *accuracy* on a test regime of 90% may indicate that its predicting all the pair of nodes to be connected).



	6.25%	12.5%	18.75%	25%	31.25%	37.5%	43.75%	50%	56.25%	62.5%	68.75%	75%	81.25%	87.5%	93.75%
10%	0.990	0.998	0.998	0.999	0.997	0.999	0.999	0.999	0.999	0.999	0.999	0.998	0.999	0.999	0.998
20%	0.968	0.991	0.993	0.995	0.989	0.993	0.995	0.995	0.995	0.995	0.995	0.994	0.995	0.995	0.994
30%	0.949	0.984	0.988	0.989	0.984	0.988	0.991	0.990	0.990	0.989	0.989	0.986	0.988	0.990	0.988
40%	0.916	0.974	0.980	0.979	0.973	0.980	0.983	0.982	0.983	0.982	0.981	0.975	0.979	0.983	0.980
50%	0.897	0.967	0.973	0.969	0.966	0.974	0.976	0.975	0.976	0.975	0.973	0.964	0.968	0.975	0.972
60%	0.890	0.958	0.964	0.951	0.963	0.965	0.964	0.963	0.966	0.962	0.960	0.945	0.951	0.964	0.957
70%	0.888	0.952	0.960	0.936	0.960	0.959	0.957	0.955	0.960	0.954	0.952	0.929	0.938	0.958	0.947
80%	0.914	0.950	0.955	0.921	0.964	0.955	0.947	0.946	0.952	0.944	0.939	0.910	0.921	0.950	0.935
90%	0.945	0.942	0.943	0.886	0.964	0.940	0.932	0.929	0.936	0.927	0.919	0.876	0.892	0.934	0.912

Figure 5.23: Average *accuracy* over the number of samples, representing the area under the curve of each model's *accuracy* over the number of samples.

Figures 5.23 and 5.24 show the increased difficulty of denser regimes, which is backed up by the literature, where most estimators only work or are meant for sparser networks. The best training regimes for each test regime are circled in white. It is clear that the best models in Fig. 5.23 differ from the best models in Fig. 5.24. For the reasons previously mentioned, we will abstain from discussing the results shown in Fig. 5.23 and will focus on the results exposed in Fig. 5.24. In Fig. 5.24, there are some visible patterns that we further analyze: models trained in a specific regime have better performance in similar connectivity regimes of the training regime than other models. Although there are clear outliers (valley drop in performance of models trained in the 31.25% and 75% connectivity regime), and the pattern is not perfectly clear, it served as basis for our next experiment: a tree of CNN.

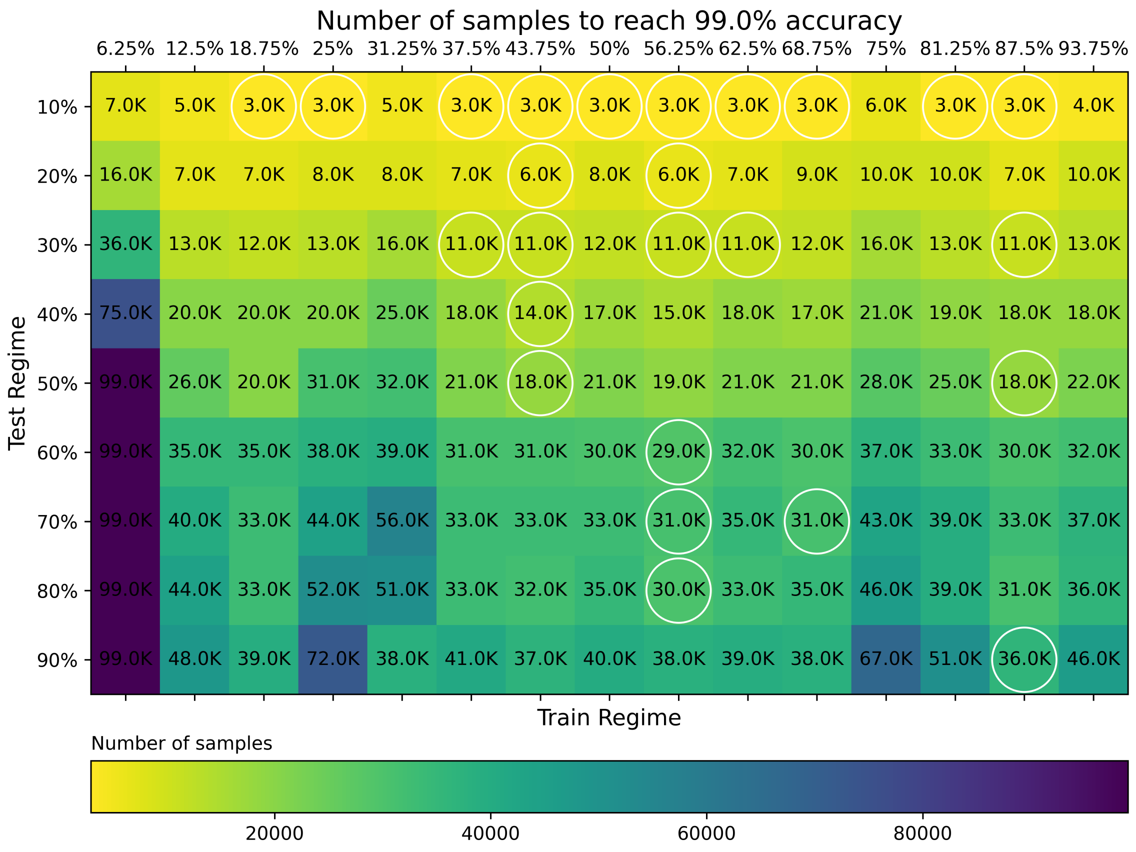


Figure 5.24: Number of samples necessary to reach 99% *accuracy*.

5.2.6 CNN Tree

From the previous results, we had the idea of building a binary tree of CNN models, in which each node of the tree represents a model trained in a specific connectivity regime. This idea is built on the assumption that models trained in a specific connectivity regime are superior in similar regimes than other models trained in different regimes.

The tree works by transversing from the root down to the leaves, in which each non-leaf node predicts the probability of connectivity of the graph from the predicted structure of the network. An example of a tree is shown in Fig. 5.25. If the predicted structure has a higher probability regime than the one from the trained model, we continue down to a model trained in a denser regime, if the predicted connectivity probability is lower, then we continue to a model trained in a sparser regime. If the node is a leaf node, then the model infers the structure as in previous experiments.

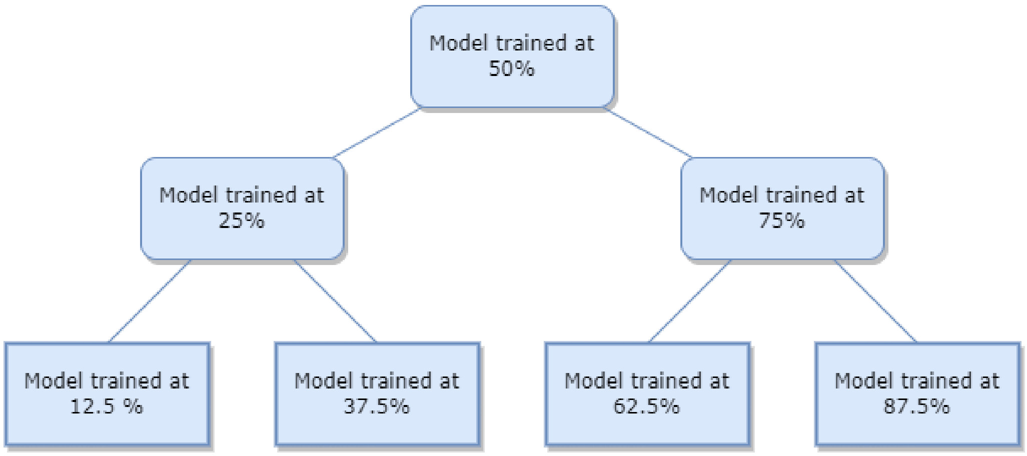


Figure 5.25: Example of a 3-layered tree.

We compared the performance of a tree with 3 layers, a tree with 4 layers and the default CNN trained at 50% connectivity. For the reasons explained in 5.2.5, we introduce the notion of balanced accuracy score, defined by:

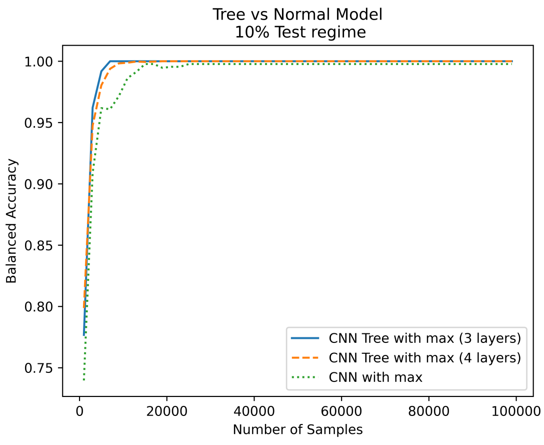
$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right),$$

which avoids inflated performance estimates on imbalanced datasets. This assures that if the conventional accuracy, defined by

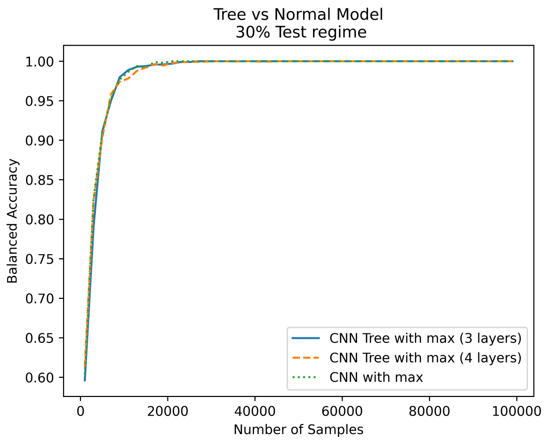
$$\text{Conventional Accuracy} = \frac{TP+TN}{TP+TN+FP+FN},$$

is above chance only because the classifier takes advantage of an imbalanced test set, then the balanced accuracy, as appropriate, will drop to $\frac{1}{2}$.

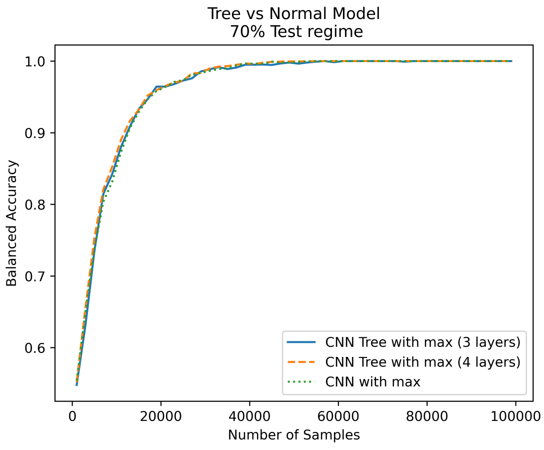
Although the CNN Trees and the default CNN have similar performances across the different regimes, the CNN Tree with 3 layers seems to have a slightly better



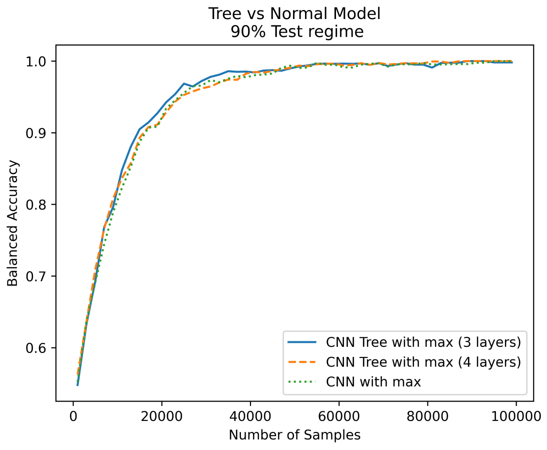
(a) CNN Tree at 10% regime



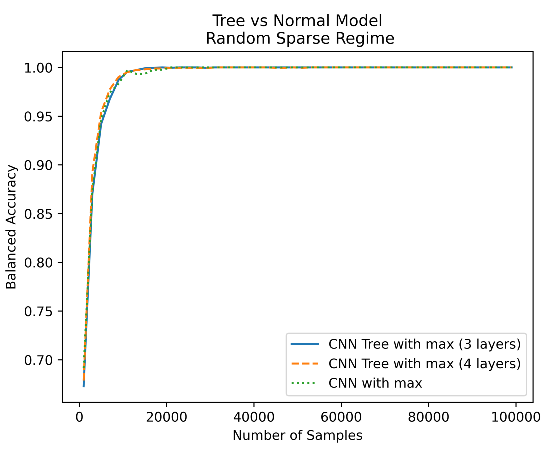
(b) CNN Tree at 30% regime



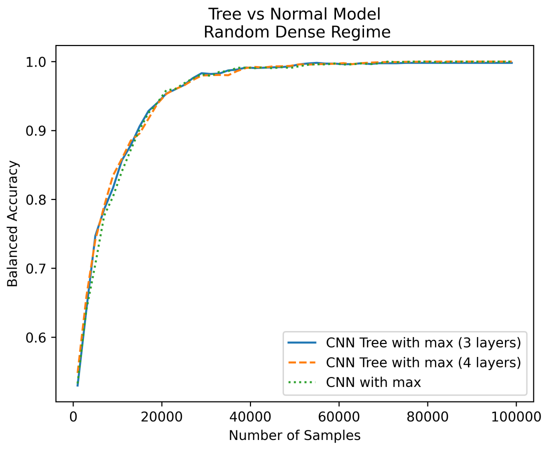
(c) CNN Tree at 70% regime



(d) CNN Tree at 90% regime



(e) CNN Tree at random sparse regime



(f) CNN Tree at random dense regime

Figure 5.26: Comparing the performance of a CNN Tree with 3 layers, a CNN Tree with 4 layers and a CNN model trained at 50% connectivity regime, across different test regimes.

performance in some regimes, namely in the 10% connectivity regime in Fig. 5.26a, in the 30% connectivity regime in Fig. 5.26b and in the 90% connectivity regime in Fig. 5.26d. Despite this, the increase in performance may not be significant and may derive from a somewhat low number of runs (10). Additionally, the exponential increase in the number of nodes in the tree (representing CNN models) with the increase of the height of the tree leads to a high time-consuming model training session, and the increase in computational power from predicting using multiple models make the trade off in performance not that worth. Furthermore, we expected that increasing the number of layers in the tree would result in a better and more robust model but it seems to decrease the tree performance, which is not expected and not trivial.

5.2.7 Precision Matrix Consistency

In our empirical experiments, we discovered that the precision matrix is consistent across different levels of colored noise. Figure 5.27 represents the performance of the precision matrix across different β values, where

$$\beta \in \{0, 1, 16, 64, 128, 256\}.$$

For this experiment, we generated a graph of 40 nodes, with only 20 of them being observed. The time series length ranges from 1000 to 500000, with a step of 10000 samples. We repeated the experiment 40 times and took the average of the runs. This finding is not trivial and may be assumed as a novel discovery.

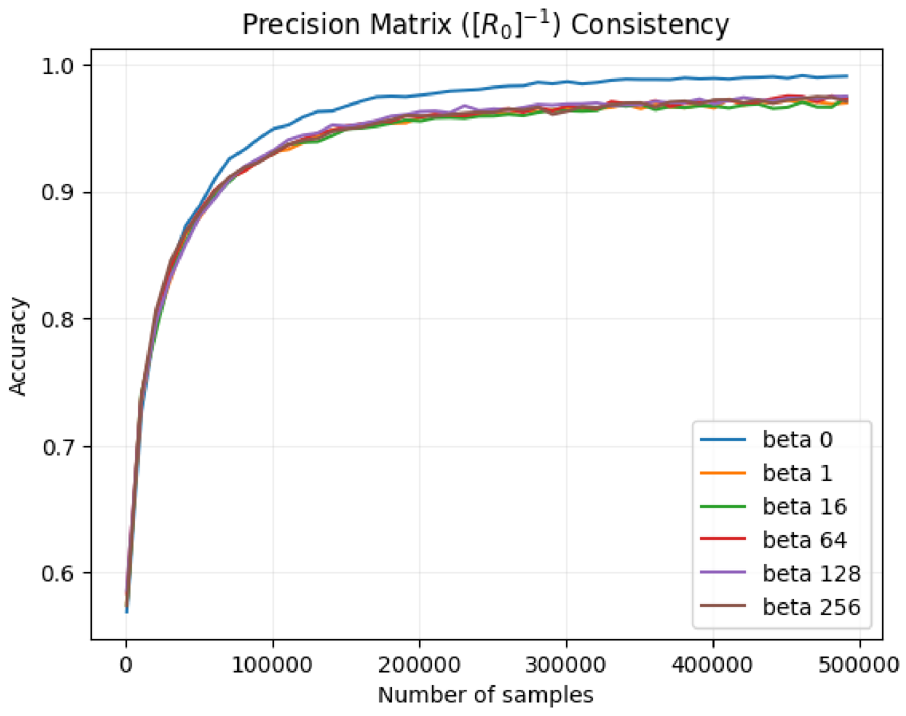


Figure 5.27: Performance of the precision matrix across multiple regimes where the value of β changes.

There is only a slight drop in performance from $\beta = 0$ to $\beta = 1$, and after that, the precision matrix has similar and consistent results for all other values of β. Even though the performance is not the greatest for lower values of β when comparing against the feature-based estimator introduced in the work of Machado et al. [23] or even other methods such as the $R_1 - R_3(NIG)$ estimator, *Granger* or the R_1

estimator, this consistency across different levels of β makes the precision matrix a robust estimator for cases where the strength of the colored noise is higher and may be helpful as a feature for a machine learning approach. Remark that the regime studied is under *partial observability* and that

$$\left[R_0^{-1}\right]_S \neq \left[(R_0)_S\right]^{-1}$$

meaning that the precision matrix calculated from the entire network and the precision matrix calculated only from the observable set are not equal. This should lead to a poor performance in the regime of *partial observability*.

5.2.8 Inversion of the NIG estimator

Along with the precision matrix, we later discovered that the inverse of the $R_1 - R_3$ (NIG) estimator performs relatively well. We compared our method against $(R_1 - R_3)^{-1}$ with GM as post-processing algorithm to cluster the entries of the matrix. In Fig. 5.28, we plot the average accuracy over all runs and we also plot the worst and the best run.

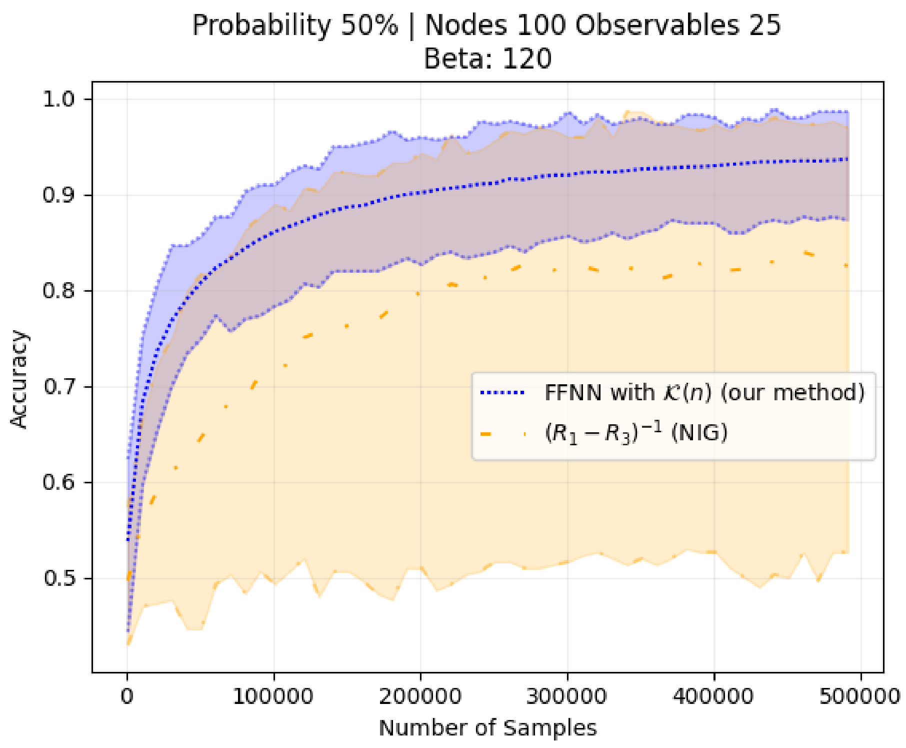


Figure 5.28: Accuracy in a regime of 50% probability of connectivity, a graph consisting of 100 nodes with only 25 of them being observable. β is equal to 120. The middle line of each method represents the average of all the runs, the lower and the higher lines represents the worst and the best run, respectively.

Although it is clear that our approach performs better overall and its evident that it is more robust across different runs, as the inverse of the (NIG) estimator has some random runs (the worst run is around 50%), the average across all runs of the $(R_1 - R_3)^{-1}$ estimator is relatively good. It would be interesting using this estimator, and maybe similar estimators with different lag moments, as a feature for our approach.

5.2.9 Cross Spectral Density and Magnitude Squared Coherence

The Magnitude Squared Coherence (MSC) between two signals, $x(n)$ and $y(n)$, indicates how much the spectral content of these two signals is correlated at different frequencies. A value of 1 implies perfect coherence, meaning the signals are synchronized in terms of their frequency content, while a value of 0 indicates no correlation between their frequency components.

The MSC of discrete-time signals $x(n)$ and $y(n)$ using Welch's method [50], is defined as:

$$C_{xy} = \frac{\left|S_{xy}(f)\right|^2}{(S_{xx}(f) * S_{yy}(f))},$$

where S_{xx} and S_{yy} are power spectral density of x and y, and S_{xy} is the cross spectral density between x and y. Welch's method computes an estimate of the power spectral density by dividing the data into overlapping segments, computing a modified periodogram for each segment and averaging the periodograms.

The Fast Fourier Transform (FFT) is a computational technique used to efficiently compute the Discrete Fourier Transform (DFT) of a signal. DFT is the discrete equivalent of the continuous Fourier Transform and is employed to analyze a signal's frequency content. In the context of calculating the Power Spectral Density (PSD) and the Cross-Spectral Density (CSD), the FFT is employed to transform the time-domain signals into the frequency domain.

The PSD of a signal describes the power present in the signal distributed across different frequency components. In the context of time series analysis, it helps us understand the frequency content of a signal and how much power it carries at each frequency. Mathematically, PSD is often denoted as $S(f)$, where f represents frequency.

For a given signal $x(t)$, the PSD can be calculated using the Fourier Transform. The formula for PSD, denoted as $S_x(f)$, is:

$$S_x(f) = |X(f)|^2,$$

where $X(f)$ represents the Fourier Transform of the signal $x(n)$.

The CSD is used to understand the relationship between two signals in the frequency domain. It quantifies the frequency-wise correlation between two signals and provides insight into their mutual behavior across different frequencies. For two signals, $x(n)$ and $y(n)$, the CSD, denoted as $S_{xy}(f)$, is formulated as:

$$S_{xy}(f) = X(f).Y^*(F),$$

where $X(f)$ represents the Fourier Transform of the signal $x(n)$ and $Y^*(f)$ is the complex conjugate of the Fourier Transform of the signal $y(t)$.

In Fig. 5.29 we can observe the MSC between all the pairs of nodes in a NDS.

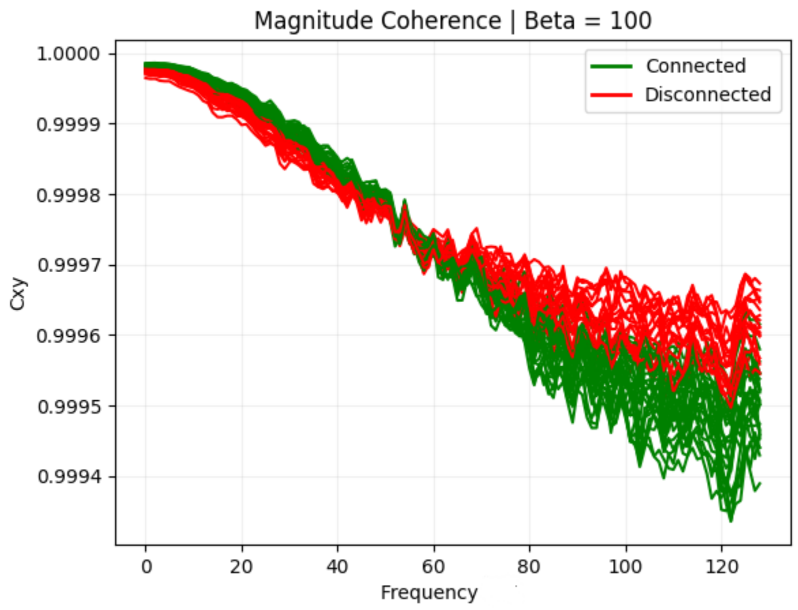


Figure 5.29: MSC (Cxy) of all pairs of a network. It was calculated from 100000 samples, in a regime of 50% connectivity and $\beta = 100$. The green curves represents the magnitude squared coherence from connected pairs, and the red curves are from disconnected pairs.

Each curve in Fig. 5.29 represents the estimate of the MSC of a pair of nodes in a network. It is evident that there is a separation between the MSC between connected nodes and disconnected nodes (the curves in red and the curves in green are not clustered), even in the context of colored noise. For this reason, the MSC could be used to further enhance the feature-vector used in our approach. However, it should be noted that the range of values in the y-axis is particularly small, and for this reason, the inverse of the correlation lag-moments may be easier to separate.

Furthermore, the inverse of the CSD matrix also reveals some interesting results, shown in Fig. 5.30. From the performed experiences, the inverse of the CSD seems to hold for all levels and amplitudes of the *colored* noise and different connectivity regimes (varying levels of p). Similar to the MSC, adding the CSD to our feature-vector as a feature may be considered and could improve the performance of our approach. Nonetheless, an increase in the number of nodes leads to a more complex system and the inverse of the CSD becomes, seemingly, less separable.

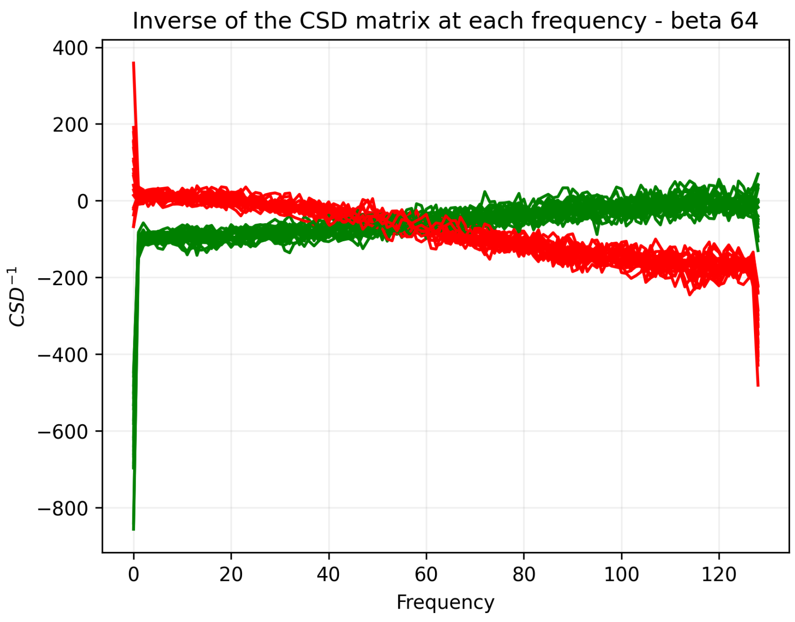


Figure 5.30: Inverse of the CSD of all pairs of a network. It was calculated from 100000 samples, in a regime of 50% connectivity and $\beta = 64$. The green curves represents the magnitude squared coherence from connected pairs, and the red curves are from disconnected pairs.

Chapter 6

Conclusion

Our research focused on recovering the network structure from time series data produced by a linear NDS. This particular problem is generally considered to be ill-posed. It is widely accepted that the feasibility of inference tasks is essentially limited by the different parameter regimes and levels of noise excitation. Specifically, the hyperparameters - namely, the observability, noise level and type, and other parameters - determine whether information about the network structure in present in the time series or not, making the problem unsolvable in the latter case. We faced the complex challenge of *partial observability* under *colored noise*. From our work, two major contributions arise. First, we developed a unique identifiability condition for linear NDS under this setting, where we grant feasibility in identifying the underlying structure of the NDS. Secondly, we propose a new approach to identify and recover the hidden structure under *partial observability* and *colored noise*. For this, we adopted a feature-based approach that assigns a feature vector, generated from the time series data, to each pair of nodes in the network, rather than relying on scalar-based methods that are commonly used in structure identification.

Furthermore, we analyzed the impact of the noise structure on the identification properties of the set of features, and we proposed a novel set of features. To recover the network structure, we trained FFNN with our features and clustered them. We conducted several distinct numerical experiments to test our approach's performance across different connectivity, observability, and noise correlation regimes, and we found it to be highly competitive against state-of-the-art estimators.

6.1 Contributions

As a result of this Dissertation, we have two major contributions. Firstly, we establish a regime of parameters where we guarantee *structural consistency*. Secondly, we propose a novel set of features and develop a machine learning approach that outperforms other state-of-the-art estimators across multiple regimes of *partial observability* and *colored* noise.

The work was, in part, submitted for publication [1]. This paper summarizes the work developed in this Dissertation, namely, the development of a novel set of features and the introduction of a novel structure identification feasibility condition in the *colored* noise regime under *partial observability*.

In addition, we created a GitHub repository [1] with all the code necessary to replicate our results and to use in future work.

6.2 Future Work

The study of dynamical systems offers a powerful framework for interpreting neuroimaging data from a range of different contexts. In the work of John et al. [51], it is shown that dynamical systems theory has the potential to revolutionize the analysis of neuroimaging data, recorded via Functional Magnetic Resonance imaging (fMRI) signals, made possible due to the advances in computational power necessary for the simulations of non-linear dynamical systems. Although the existence of significant distinct nonlinear interactions between brain regions is proven by Poskanzer and Anzellotti [52], it is important to note that the linear components of these interactions were still an order of magnitude larger than the nonlinear. This is an important remark to consider since, in our approach, we assume a linear dynamical system for the generation of the feature vector. Furthermore, in the work of Liégeois et al. [53], it is shown that estimating Functional Connectivity (FC) from precision matrix based estimators allows more meaningful Structural Connectivity (SC) - FC comparisons. In particular, it was shown that certain powers of the precision matrix (computed from fMRI time series) exhibit nontrivial similarity with the actual SC matrix (estimated from the white matter tracts). Pursuing this work and applying the proposed methods to real data streams, such as Electroencephalography (EEG) or fMRI signals from various brain regions is a future promising direction, as we have analyzed this approach for real world networks with synthetically generated time series but never with real data streams.

Additionally, we only delved into the field of linear NDS. There is great promise in exploring the intricacies of nonlinear NDS, since many real-world systems exhibit nonlinear and complex behaviors that cannot be adequately described by linear models. It is, therefore, highly promising to pursue this avenue of inquiry while implementing, analyzing, and refining the proposed methodology to effectively adapt to the complexities inherent in these systems.

Another interesting application would be in predicting the interaction and network structure of food-webs, more concretely, in the case of predator-prey species, in a directed manner. Moreover, in the context of a pandemic, where the network of interactions between regions is relevant for the design of mitigation policies and predicting if the strain of the virus will survive or will die out, the topology of the network may be inferred from the number of infected across distinct regions.

[1] `https://github.com/seabrapt/brain_underlying_structure_identification`

References

[1] Augusto Santos, Diogo Rente, Rui Seabra, and José M. F. Moura. Learning the causal structure of networked dynamical systems under latent nodes and structured noise. In *submitted*, 2023.

[2] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97, Jan 2002. doi: 10.1103/RevModPhys.74.47. URL `https://link.aps.org/doi/10.1103/RevModPhys.74.47`.

[3] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, Jun 1998. ISSN 1476-4687. doi: 10.1038/30918. URL `https://doi.org/10.1038/30918`.

[4] M. E. J. Newman. Spread of epidemic disease on networks. *Physical Review E*, 66(1), jul 2002. doi: 10.1103/physreve.66.016128. URL `https://doi.org/10.1103%2Fphysreve.66.016128`.

[5] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, 2008. doi: 10.1017/CBO9780511791383.

[6] Mason A. Porter and James P. Gleeson. Dynamical systems on networks: A tutorial, 2015.

[7] Stephen Eubank, Hasan Guclu, Sritesh Kumar, Madhav Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modeling disease outbreaks in realistic urban social networks. *Nature*, 429:180–4, 06 2004. doi: 10.1038/nature02541.

[8] Alfredo Braunstein, Luca Dall'Asta, Guilhem Semerjian, and Lenka Zdeborová. Network dismantling. *Proceedings of the National Academy of Sciences*, 113(44):12368–12373, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1605083113.

[9] Luqing Wei, Chris Baeken, Daihong Liu, Jiuquan Zhang, and Guo-Rong Wu. Functional connectivity–based prediction of global cognition and motor function in riluzole-naive amyotrophic lateral sclerosis patients. *Network Neuroscience*, 6(1):161–174, 02 2022. ISSN 2472-1751. doi: 10.1162/netn_a_00217. URL `https://doi.org/10.1162/netn_a_00217`.

[10] L Q Uddin, D R Dajani, Voorhies W, Bednarz H, and R K Kana. Progress and roadblocks in the search for brain-based biomarkers of autism and attention-deficit/hyperactivity disorder. *Translational Psychiatry*, 7(8):e1218–e1218, 08

2017. ISSN 2158-3188. doi: 10.1038/tp.2017.164. URL https://cir.nii.ac.jp/crid/1364233270762710528.

[11] Gonzalo Mateos, Santiago Segarra, Antonio G. Marques, and Alejandro Ribeiro. Connecting the dots: Identifying network structure via graph signal processing. *IEEE Signal Processing Magazine*, 36(3):16–43, 2019. doi: 10.1109/MSP.2018.2890143.

[12] Augusto Santos, Vincenzo Matta, and A. H. Sayed. Local tomography of large networks under the low-observability regime. *IEEE Transactions on Information Theory*, 66:587 – 613, 01 2020. doi: 10.1109/TIT.2019.2945033.

[13] Vincenzo Matta, Augusto Santos, and Ali H. Sayed. Graph learning under partial observability. *Proceedings of the IEEE*, 108:2049 – 2066, 11 2020. doi: 10.1109/JPROC.2020.3013432.

[14] J. Mei and J. M. F. Moura. Signal processing on graphs: Causal modeling of unstructured data. *IEEE Transactions on Signal Processing*, 65(8):2077–2092, April 2017. ISSN 1053-587X. doi: 10.1109/TSP.2016.2634543.

[15] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, 1968. doi: 10.1109/TIT.1968.1054142.

[16] E.M.M. Kivits and Paul M.J. Van den Hof. Identification of diffusively coupled linear networks through structured polynomial models. *IEEE Transactions on Automatic Control*, pages 1–16, 2022. doi: 10.1109/TAC.2022.3191406.

[17] Songting Li, Yanyang Xiao, Douglas Zhou, and David Cai. Causal inference in nonlinear systems: Granger causality versus time-delayed mutual information. *Physical Review E*, 97, 05 2018. doi: 10.1103/PhysRevE.97.052216.

[18] Nir Friedman, Iftach Nachman, and Dana Pe'er. Learning bayesian network structure from massive datasets: The "sparse candidate" algorithm, 2013.

[19] George Sugihara, Robert May, Hao Ye, Chih-hao Hsieh, Ethan Deyle, Michael Fogarty, and Stephan Munch. Detecting causality in complex ecosystems. *Science (New York, N.Y.)*, 338, 09 2012. doi: 10.1126/science.1227079.

[20] Jonathan Mei and José M. F. Moura. Silvar: Single index latent variable models. *IEEE Transactions on Signal Processing*, 66(11):2790–2803, 2018. doi: 10.1109/TSP.2018.2818075.

[21] Zheng Chen, Zengyou He, Hao Liang, Can Zhao, and Yan Liu. Correlation-based community detection, 2019.

[22] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 12 2007. ISSN 1465-4644. doi: 10.1093/biostatistics/kxm045. URL https://doi.org/10.1093/biostatistics/kxm045.

[23] Sergio Machado, Anirudh Sridhar, Paulo Gil, Jorge Henriques, Jose M. F. Moura, and Augusto Santos. Recoverying the graph underlying networked dynamical systems under partial-observability: a deep learning approach. In *Proceedings of the 37th AAAI Coference on Artificial Intelligence*, AAAI'37. AAAI, 2023.

[24] P. Geiger, K. Zhang, B. Schölkopf, M. Gong, and D. Janzing. Causal inference by identification of vector autoregressive processes with hidden components. In *Proc. International Conference on Machine Learning*, volume 37, pages 1917–1925, July 2015.

[25] Vincenzo Matta, Augusto Santos, and Ali H. Sayed. Graph learning over partially observed diffusion networks: Role of degree concentration. *IEEE Open Journal of Signal Processing*, 3:335–371, 2022. doi: 10.1109/OJSP.2022.3189315.

[26] Divyanshu Vats and José M. F. Moura. Necessary conditions for consistent set-based graphical model selection. In *2011 IEEE International Symposium on Information Theory Proceedings*, pages 303–307, 2011. doi: 10.1109/ISIT.2011.6034133.

[27] Divyanshu Vats and José M. F. Moura. Finding non-overlapping clusters for generalized inference over graphical models. *IEEE Transactions on Signal Processing*, 60(12):6368–6381, 2012. doi: 10.1109/TSP.2012.2214216.

[28] Guy Bresler, Frederic Koehler, Ankur Moitra, and Elchanan Mossel. Learning restricted boltzmann machines via influence maximization, 2018.

[29] Animashree Anandkumar and Ragupathyraj Valluvan. Learning loopy graphical models with latent variables: Efficient methods and guarantees. *Ann. Statist.*, 41(2):401–435, 04 2013. doi: 10.1214/12-AOS1070. URL `https://doi.org/10.1214/12-AOS1070`.

[30] M.A. Porter and J.P. Gleeson. *Dynamical Systems on Networks: A Tutorial*. Springer International Publishing, 2016. ISBN 9783319266411.

[31] Faryad Darabi Sahneh, Caterina Scoglio, and Piet Van Mieghem. Generalized epidemic mean-field model for spreading processes over multilayer complex networks. *IEEE/ACM Transactions on Networking*, 21(5):1609–1620, 2013. doi: 10.1109/TNET.2013.2239658.

[32] Vishwaraj Doshi, Shailaja Mallick, and Do Young Eun. Competing epidemics on graphs - global convergence and coexistence. In *2012 INFOCOMM*, May 2021.

[33] Ali H. Sayed. Adaptation, Learning, and Optimization over Networks. *Found. Trends Mach. Learn.*, 7(4-5):311–801, 2014. ISSN 1935-8237. doi: 10.1561/2200000051.

[34] L Hufnagel, D. Brockmann, and T Geisel. Forecast and control of epidemics in a globalized world. *Proceedings of the National Academy of Sciences of the United States of America*, 101:15124–9, 11 2004. doi: 10.1073/pnas.0308344101.

[35] Vittoria Colizza, Alain Barrat, Marc Barthelemy, Alain-Jacques Valleron, and Alessandro Vespignani. Modeling the Worldwide Spread of Pandemic Influenza: Baseline Case and Containment Interventions. *PLoS Medicine*, 4: e13, 2007. URL `https://hal.science/hal-00126956`. 16 pages.

[36] Michael Breakspear. Dynamic models of large-scale brain activity. *Nature Neuroscience*, 20:340–352, 02 2017. doi: 10.1038/nn.4497.

[37] Joaquín Goñi, Martijn P. van den Heuvel, Andrea Avena-Koenigsberger, Nieves Velez de Mendizabal, Richard F. Betzel, Alessandra Griffa, Patric Hagmann, Bernat Corominas-Murtra, Jean-Philippe Thiran, and Olaf Sporns. Resting-brain functional connectivity predicted by analytic measures of network communication. *Proceedings of the National Academy of Sciences*, 111(2):833–838, 2014. doi: 10.1073/pnas.1315529111. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1315529111`.

[38] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009. ISBN 0262013193.

[39] Andreas Eberle. Markov processes, June 2017. URL `https://wt.iam.uni-bonn.de/fileadmin/WT/Inhalt/people/Andreas_Eberle/Markov_Processes_1617/MPSkript1617.pdf`.

[40] Yupeng Chen, Zhiguo Wang, and Xiaojing Shen. An unbiased symmetric matrix estimator for topology inference under partial observability. *IEEE Signal Processing Letters*, 29(02):1257–1261, 2022. doi: 10.1109/LSP.2022.3177076.

[41] D. Materassi and M. V. Salapaka. Network reconstruction of dynamical polytrees with unobserved nodes. In *Proc. IEEE Conference on Decision and Control (CDC)*, pages 4629–4634, Maui, Hawaii, Dec 2012. doi: 10.1109/CDC.2012.6426335.

[42] D. Materassi and M. V. Salapaka. On the problem of reconstructing an unknown topology via locality properties of the Wiener filter. *IEEE Transactions on Automatic Control*, 57(7):1765–1777, July 2012.

[43] Mishfad Shaikh Veedu and Murti V. Salapaka. Topology identification under spatially correlated noise, 2023.

[44] Mishfad Shaikh Veedu, Harish Doddi, and Murti V. Salapaka. Topology learning of linear dynamical systems with latent nodes using matrix decomposition. *IEEE Transactions on Automatic Control*, 67(11):5746–5761, 2022. doi: 10.1109/TAC.2021.3124979.

[45] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[46] F. Chung. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series. Conference Board of the Mathematical Sciences, 1994. ISBN 9780821889367.

[47] Sergio Machado. Learning the graph of networked dynamical systems under partial-observability via artificial neural networks, September 2022.

[48] Antonín Škoch, Barbora Rehák Bučková, Jan Mareš, Jaroslav Tintěra, Pavel Sanda, Lucia Jajcay, Jiří Horáček, Filip Španiel, and Jaroslav Hlinka. Human brain structural connectivity matrices–ready for modelling. *Scientific Data*, 9 (1):486, 2022.

[49] Lloyd Goldwasser and Jonathan Roughgarden. Construction and analysis of a large caribbean food web: Ecological archives e074-001. *Ecology*, 74(4): 1216–1233, 1993.

[50] P. Welch. The use of fast fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, 15(2):70–73, 1967. doi: 10.1109/TAU.1967.1161901.

[51] Yohan J. John, Kayle S. Sawyer, Karthik Srinivasan, Eli J. Müller, Brandon R. Munn, and James M. Shine. It's about time: Linking dynamical systems with human neuroimaging to understand the brain. *Network Neuroscience*, 6(4):960–979, 10 2022. ISSN 2472-1751. doi: 10.1162/netn_a_00230. URL `https://doi.org/10.1162/netn_a_00230`.

[52] Craig Poskanzer and Stefano Anzellotti. Functional coordinates: Modeling interactions between brain regions as points in a function space. *Network Neuroscience*, 6(4):1296–1315, 10 2022. ISSN 2472-1751. doi: 10.1162/netn_a_00264. URL `https://doi.org/10.1162/netn_a_00264`.

[53] Raphael Liégeois, Augusto Santos, Vincenzo Matta, Dimitri Van De Ville, and Ali H. Sayed. Revisiting correlation-based functional connectivity and its relationship with structural connectivity. *Network Neuroscience*, 4(4):1235–1251, 2020. doi: 10.1162/netn_a_00166.

[54] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Fundamentals of Convex Analysis*. Grundlehren Text Editions. Springer-Verlag Berlin Heidelberg, 2001. ISBN 978-3-540-42205-1. doi: 10.1007/978-3-642-56468-0.

Appendices

Appendix A

K-means algorithm

K-means is a popular unsupervised machine learning clustering algorithm that partitions a set of data points into a predefined number of clusters, or groups, based on the distance between the points and the centroid (center) of each cluster. The algorithm works by iteratively assigning each data point to the cluster with the nearest centroid and then updating the centroid to be the mean of all the points in the cluster.

The general process for running k-means on a set of data points is as follows:

1. Initialization
   (a) Choose the number of clusters (k)
   (b) Initialize the centroids for each of the k clusters.
2. Assignment step
   (a) For each data point x, calculate the distance between the data point and the centroid of each cluster
   $$D_{xc} = \sqrt{(x_1 - c_1)^2 + (x_2 - c_2)^2 + ... + (x_N - c_N)^2}, \tag{A.1}$$
   where D_{xc} is the distance from point x to centroid of cluster c and N is the dimensionality of the problem.
   (b) Assign x to the cluster with the smallest distance
   $$\text{class } x = argmin[D_{xc(1)}, D_{xc(2)}, ..., D_{xc(k)}], \tag{A.2}$$
   where $c(k)$ is the centroid of the k-th cluster and $D_{xc(k)}$ is the distance between x and the centroid of the k-th cluster.
3. Update step
   (a) For each cluster, calculate the new centroid by taking the mean of all the points in the cluster.
   $$C(k) = mean(X), \forall x \in M, \tag{A.3}$$

where $C(k) = [c_1(k), c_2(k), ...c_N(k)]$ represents the centroid of the k-th cluster, $X = [x_1, x_2, ..., x_N]$ represents a data point, and M represents the subset of points that belong to cluster k.

4. Repeat steps 2 and 3 until the new centroids are equal to the old centroids or until the maximum number of iterations is reached.

One way to evaluate the quality of the clusters produced by k-means is to use the within-cluster sum of squares (WCSS) measure, which is the sum of the squared distances between each point in a cluster and the centroid of that cluster. A smaller WCSS value indicates that the points in the cluster are more tightly grouped around the centroid.

Appendix B

Gaussian Mixture algorithm

Gaussian mixture is a probabilistic model that represents a set of data points as a mixture of multiple Gaussian distributions. It is a type of clustering algorithm that is used to group data points into clusters based on their similarity.

The Gaussian mixture model assumes that the data is generated from a mixture of multiple underlying Gaussian distributions, each with its own mean and covariance. The model estimates the probability of each data point belonging to each of the Gaussian distributions and assigns the point to the cluster corresponding to the distribution with the highest probability.

The general process for fitting a Gaussian mixture model to a set of data points is as follows:

1. Initialization
    (a) Choose the number of clusters, k, and the initial values for the parameters of the model (means, covariances, and mixing coefficients).
2. Expectation-Maximization (EM) algorithm.
    (a) In the E-step, calculate the probability of each data point x belonging to each cluster c
    (b) In the M-step, update the parameters of the model based on the probabilities calculated in the E-step.
3. Repeat step 2 until the model converges or a maximum number of iterations is reached.

One way to evaluate the quality of the clusters produced by a Gaussian mixture model is to use the log-likelihood measure, which is the log of the probability of the data given the model. A higher log-likelihood indicates a better fit of the model to the data.

Appendix C

Principal Component Analysis

PCA is a statistical technique used to analyze the relationships among a set of variables to identify patterns and trends. It is a helpful tool for reducing the dimensionality of large datasets and for visualizing complex datasets in a more manageable and interpretable form. PCA is a linear method that assumes that the relationships between the variables are linear. It works by finding the directions (called "principal components") in the data that capture the most variation or variance. These directions are represented by the eigenvectors of the covariance matrix of the data, and the amount of variation captured by each direction is represented by the corresponding eigenvalue.

The general process to apply PCA is as follows:

1. Centralization or Normalization of the features

   We first center the data to apply PCA by subtracting the mean from each feature. This centralization is done because PCA is sensitive to the scale of the variables, and centering the data helps to give each variable an equal weight in the analysis.

   To perform centralization:

   $$x'_{ij} = x_{ij} - \mu_j, \tag{C.1}$$

   where x_{ij} is the value of the i-th sample for the j-th feature, and μ_j is the mean of the j-th feature.

   We can normalize the features utilizing a standard scaler, by removing the mean and scaling to unit variance, instead of only centralizing them:

   $$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j}, \tag{C.2}$$

   where x_{ij} is the value of the i-th sample for the j-th feature, μ_j is the mean of the j-th feature and σ_j is the variance of the j-th feature.

2. Computing the covariance matrix

   Next, we compute the covariance matrix of the centered data. The covariance matrix is a square matrix that contains the pairwise covariances between the variables. The diagonal elements of the covariance matrix represent the variances of the individual variables, and the off-diagonal elements represent the covariances between pairs of variables.

3. Computing the eigenvectors and eigenvalues

   Finally, we compute the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors are the principal components, and the eigenvalues are the variance captured by each principal component.

4. Selecting the principal components

   We can then select the top n principal components, where n is the number of dimensions to which we want to reduce the data. We can then transform the original data onto the new, lower-dimensional space defined by the principal components.

PCA is a very useful technique for data exploration and visualization, and it is also used as a preprocessing step for many machine learning algorithms. In addition, it can help reduce the data's complexity, eliminate noise and redundant features, and improve the model's performance.

Appendix D

Properties of the function Osc

In this appendix, we present some properties of the function $\mathsf{Osc}(\cdot)$:

Property 1. Given a symmetric stochastic matrix $\overline{A}$, we have

$$\mathsf{Osc}\left(\overline{A}v\right) \leq \mathsf{Osc}\left(v\right),$$

as each entry of the vector $\overline{v} \triangleq \overline{A}v$ lies in the convex hull [54] of the set $\{v_1, v_2, \ldots, v_N\}$ comprised by the entries of the vector $v \in \mathbb{R}^N$ and, in particular, $\overline{v}_i \in [v_{\min}, v_{\max}]$ for all $i = 1, 2, \ldots, N$.

Property 2. Observe that

$$\mathsf{Osc}\left(\alpha v\right) = |\alpha|\, \mathsf{Osc}\left(v\right),$$

for all $v \in \mathbb{R}^N$ and $\alpha \in \mathbb{R}$.

Property 3. The function $\mathsf{Osc}(\cdot)$ is sub-additive in that

$$\mathsf{Osc}\left(B + C\right) \leq \mathsf{Osc}\left(B\right) + \mathsf{Osc}\left(C\right),$$

for any matrices $B, C \in \mathbb{R}^{N \times N}$.

Property 4. The function $\mathsf{Osc}(\cdot)$ is sub-multiplicative in the following sense: If

$$\left\{ \begin{array}{lcl} \mathsf{Osc}\left(Bv\right) & \leq & k_b \mathsf{Osc}\left(v\right) \\ \mathsf{Osc}\left(Cv\right) & \leq & k_c \mathsf{Osc}\left(v\right) \end{array} \right. \forall v \in \mathbb{R}^N$$

then,

$$\mathsf{Osc}\left(CBv\right) \leq k_c \mathsf{Osc}\left(Bv\right) \leq k_b k_c \mathsf{Osc}\left(v\right)$$

with $k_b, k_c > 0$.