DOCTORAATSPROEFSCHRIFT

2009 | School voor Informatietechnologie
Kennistechnologie, Informatica, Wiskunde, ICT

Identifying Dynamic Sparse Interaction Networks

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen: richting informatica, te verdedigen door:

Goele HOLLANDERS

Promotor: prof. dr. M. Gyssens
Copromotoren: prof. dr. R. Westra
prof. dr. K. Tuyls

Universiteit Maastricht

universiteit hasselt

D/2009/2451/45

Dankwoord

Ik maak graag van de gelegenheid gebruik om iedereen te danken die rechtstreeks of onrechtstreeks heeft bijgedragen tot het tot stand komen van dit proefschrift.

Eerst en vooral wil ik mijn begeleiders, Marc Gyssens, Ronald Westra, Karl Tuyls en Geert Jan Bex, danken om mij de mogelijkheid gegeven te hebben een doctoraat voor te bereiden aan de Universiteit Hasselt, en voor hun steun en advies tijdens het tot stand komen ervan. In dit verband gaat een bijzonder woord van dank naar Ronald, die me heeft aangemoedigd om aan dit doctoraat te beginnen. Zonder hem zou ik er zelfs niet aan gedacht hebben. Een bijzonder woord van dank gaat ook uit naar Geert Jan, voor zijn niet aflatende steun, begeleiding en waardevol advies gedurende het hele traject. Hij was altijd beschikbaar voor technische discussies, en hij confronteerde mij met vele interessante en soms zeer uitdagende problemen.

Vervolgens wil ook de leden van onze onderzoeksgroep danken voor de erg stimulerende omgeving waarin ik heb kunnen werken. Ik wil hier Walied Othman noemen met wie ik een bureau deelde, voor zijn aangenaam gezelschap en de talrijke tips om mijn schrijfstijl te verbeteren.

Op meer persoonlijk vlak wil ik mijn ouders en mijn broer danken voor hun onvoorwaardelijke en volgehouden steun. En tot slot dank ik natuurlijk mijn vriend, Stijn, voor zijn voortdurende aanmoedigingen en zijn steun gedurende mijn onderzoekswerk en het schrijven van dit proefschrift.

Diepenbeek, juli 2009,
Goele Hollanders

Voorwoord

Dit doctoraat kwam tot stand in het kader van een "tUL doctoraatsbursaal" gefinancierd vanuit de impulsmiddelen voor de transnationale Universiteit Limburg toegekend aan de Universiteit Hasselt. Initiatiefnemer van dit project is Karl Tuyls, toen postdoctoraal onderzoeker aan de Universiteit Hasselt, en Ronald Westra participeerde als partner van de Universiteit Maastricht. Toen Karl Tuyls docent werd aan de Universiteit Maastricht, werd het Hasseltse team versterkt door Geert Jan Bex. Dit vierkoppige team verzorgde de begeleiding van dit doctoraat, in een hechte transnationale samenwerking. Door beperkende voorwaarden in het doctoraatsreglement van de Universiteit Hasselt zijn officieel enkel Karl Tuyls en Ronald Westra co-promotor van deze thesis. Naar de geest echter, is Geert Jan Bex dit evenzeer.

Marc Gyssens
Promotor

Samenvatting

Interactienetwerken beschrijven een waaier van systemen uit de samenleving en de natuur. Vaak geciteerde voorbeelden zijn het World Wide Web, een netwerk van documenten (webpagina's) gelinkt met elkaar via verschillende hyperlinks (URL's) [5, 52] en de cel [70], een netwerk van chemische (genen) connecties via biochemische reacties.

Een biologisch netwerk kan beschreven worden als volgt. Een organisme, zoals gist, bestaat uit genen, proteïnen en andere elementen. Deze genen en proteïnen zijn de knopen in zulke netwerken en kunnen verbonden worden met andere genen en proteïnen in dat netwerk. Wanneer een gen of proteïne zich onregelmatig gedraagt, kan een ziekte optreden. Om dit te begrijpen, is het dus belangrijk dat deze genen/proteïnen opgespoord kunnen worden.

Celdeling, bijvoorbeeld, zorgt voor het herstel van een weefsel en elke cel bevat genen die de celdeling controleren. Indien een gen zich onregelmatig gedraagt, en dus de celdeling beïnvloedt, kan dit leiden tot een ongecontroleerde groei van weefsel, gekend als kanker. Deze ongecontroleerde groei kan gestimuleerd of afgeremd worden door enerzijds externe factoren, zoals hormonen en chemische substanties, en anderzijds intern door andere genen. Om deze genen te kunnen opsporen, stellen we hun onderlinge wisselwerking voor met behulp van een interactienetwerk. Dit modelleren we als een interactiematrix, die voor elk koppel genen de graad van beïnvloeding tussen beide weergeeft.

De laatste decennia is het onderzoeksgebied van genetica en bio-informatica sterk gegroeid, o.a. dankzij de ontwikkeling van microarray-technologie. Via deze technologie is het mogelijk om gen-en proteïne-expressies—het gedrag van genen en proteïnen over de tijd—te meten.

Vele onderzoeken met betrekking tot het modelleren van gen/proteïne-interacties werden reeds uitgevoerd. Voor een overzicht verwijzen we naar de Jong [22], Bower [16] en anderen [27, 41, 58, 37]. Anders dan in deze studies, zijn wij niet geïnteresseerd in statische modellen, die gefocust zijn op co-occurence, maar in dynamische modellen, waar de aandacht ligt op de causale verbanden.

Probleemstelling

Realistische netwerken bevatten zeer veel knopen. Vanuit de biologie is geweten dat gen-proteïne interactienetwerken uit een duizendtal knopen bestaan. Bijgevolg zijn er miljoenen mogelijke interacties tussen deze knopen. Toch wordt elke knoop maar beïnvloed door een beperkt aantal andere knopen. Realistische netwerken kunnen dus worden voorgesteld als dunne grafen, en daarom zijn we enkel in dit soort netwerken geïnteresseerd.

Om de interacties in zulke netwerken te kunnen identificeren, is een groot aantal experimentele gegevens nodig. Omwille van de dure experimenten die hiervoor nodig zijn, zijn er in de praktijk maar een beperkt aantal metingen beschikbaar in vergelijking met het aantal knopen. We hebben dus te maken met een ondergedetermineerd systeem. Omwille van deze beperking, kunnen identificatieproblemen opduiken, en is het dus belangrijk om zoveel mogelijk informatie over het onderliggende systeem te bekomen.

Ruis, zowel meetruis als systeemruis, is alom tegenwoordig in realistische data, daarom is het belangrijk om een algoritme te ontwikkelen dat hiermee kan omgaan.

Naast het identificeren van dunne interactienetwerken, zijn we ook geïnteresseerd in de opslagcapaciteit van input-output-patronen. Voor elke input bestaat er een output. Door het opslaan van deze patronen kan een systeem leren en op een gepaste manier reageren op een bepaalde input.

We zijn in dit werk dus geïnteresseerd in de relatie tussen het aantal variabelen en het aantal observaties, de verbondenheid, de invloed van ruis en de opslagcapaciteit van netwerken.

Oplossingen

Identificatie van dunne, dynamische interactienetwerken

Om een interactienetwerk te kunnen identificeren, moet een wiskundig model opgesteld worden dat een gedetailleerde beschrijving geeft van het probleem. Gebaseerd op differentiaalvergelijkingen zijn in dit werk twee algoritmes voorgesteld voor het modelleren en identificeren.

In het Single Linear Model wordt de toestandsruimte beschouwd als één geheel. De interacties, voorgesteld door systeemmatrices, worden gemodelleerd door middel van een lineair dynamisch systeem. Omwille van het groot aantal onbekenden, worden oneindig veel oplossingen voor het systeem gevonden, waarvoor de L_1-minimalisatie techniek de meest dunne selecteert.

Voor het tweede model, het Piecewise Linear Model, wordt de toestandsruimte opgesplitst in subsystemen. Hiervoor is een gewichtenmatrix gedefinieerd. Deze geeft het lidmaatschap weer voor elke observatie tot een bepaald subsysteem. Als gevolg van het combineren van de systeemmatrix met de gewichten matrix, kan elk subsysteem bepaald worden door middel van het Single Linear Model.

Uit de experimenten kunnen we concluderen dat zowel het Single Linear Model als het Piecewise Linear Model zeer snel zijn. Verder zijn beide modellen in staat om een dun dynamisch interactienetwerk nauwkeurig te bepalen uitgaande van een beperkt aantal observaties.

Netwerk identificatie geformuleerd als een leer probleem

Om enerzijds de relatie tussen het aantal onbekenden en het aantal observaties en anderzijds de invloed van het dunne karakter meer gedetailleerd te bestuderen, herformuleren we het probleem als een machine learning probleem. In deze context moet een model geleerd worden dat het onderliggende systeem identificeert. Daarnaast moet het model in staat zijn om te generaliseren. Hiermee bedoelen we dat een model, dat geleerd wordt met behulp van een oefenset, ook in staat moet zijn een valideringset, d.w.z. data niet gebruikt door het algoritme, correct met het onderliggende systeem in overeenstemming te brengen.

Uit de experimenten volgt dat, startend van een relatief kleine oefenset, het leerproces een perfecte overeenstemming simuleert. Verder concluderen we

ook dat de overgang naar generalisatie, in functie van het aantal observaties, zeer abrupt is en dat de kwaliteit van de overeenstemming sterk afhangt van het dunne karakter van het netwerk.

Een meer verrassend resultaat is dat voor elke component de inkomende interacties onafhankelijk geïdentificeerd kunnen worden. Hieruit volgt dat de generalisatie voor verschillende systeemgroottes, de verhouding tussen het aantal observaties en het aantal onbekenden, berekend kan worden door middel van het correct schalen van de systeemgrootte.

Helaas zijn beide modellen, het Single Linear Model en het Piecewise Linear Model, niet robuust zijn ten opzichte van ruis. Als een gevolg wordt de L_1-minimalisatie, dat aan de basis ligt voor beide modellen, aangepast en een regularisatieparameter wordt toegevoegd. Deze parameter hangt af van de systeemgrootte en zoekt naar een balans tussen voorwaarden dat het netwerk dun is en de robuustheid van de overeenstemming met het systeem, rekening houdend met ruis.

Opslagcapaciteit van lineaire netwerken

Naast het identificeren van dunne, dynamische netwerken en het bestuderen van de invloed van de systeemgrootte, de mate waarin het netwerk dun is en de ruis hebben we ook nog de input-ouput-opslagcapaciteit van netwerken onderzocht.

Het maximaal aantal outputs dat opgeslagen kan worden hangt af van het aantal inputs, zowel interne inputs van componenten als externe inputs van externe invloeden zoals licht, temperatuur, enzovoort.

Om te bestuderen hoe de patronen opgeslagen worden, kan opnieuw gebruik gemaakt worden van lineair programmeren. Uit de experimenten kan worden afgeleid dat de opslag in drie fasen gebeurd. In de eerste fase slaat het systeem de output op die enkel afhangt van de externe input. In de tweede fase is de opgeslagen output afhankelijk van voornamelijk de interne input, dus van de interacties in het netwerk. In de derde fase, tenslotte, is de opgeslagen output het resultaat van zowel de interne als de externe input.

Voorbeeld van een toepassing

Om dit werk af te ronden, worden het Single Linear Model en het Piecewise Linear Model toegepast op een gist data set van Spellman [70]. Deze gist, *Sac-*

charomyces cerevisiae, is een eencellige schimmel die voorkomt in wijndruiven. Deze data bevatten expressieprofielen voor meer dan 6000 genen gemeten met behulp van microarrays op 24 tijdstippen verspreid over 5 uren.

Alvorens te starten met de identificatie van het netwerk, werden alle genen met missende waarden en waarvoor de genexpressie niet voldoende varieert, verwijderd. Na deze voorbereidingsstap bleven er nog 290 genen over. Verder werden alle genen met gelijkaardige genexpressies, dus met gelijkaardig gedrag, gegroepeerd. Dit levert ons uiteindelijk een dataset op bestaande uit 16 genrepresentatieves en 24 observaties.

Zoals verwacht kan worden uit de voorgaande studies, bevestigen de experimenten dat het Single Linear Model en het Piecewise Linear Model in de originele vorm, dus zonder de regularisatieparameter, geen aanvaardbare overeenstemmingen opleveren. De oorzaak hiervan is de aanwezigheid van ruis in de data. Toch kunnen we besluiten dat het Piecewise Linear Model een dunne interactiematrix oplevert.

Wanneer we het aangepaste Piecewise Linear Model, dus met de toegevoegde regularisatieparameter, toepassen op de gist data, verkrijgen we een nog dunnere interactiematrix met een betere overeenstemming. Toch moeten we bekennen dat de kwaliteit van de overeenstemming nog steeds redelijk laag is.

Toch eindigen we met een positief gevoel. In deze thesis hebben we steeds met grote netwerken— bestaande uit 80 tot 300 componenten — en artificiële data gewerkt. Dit resulteerde steeds in numerieke experimenten die statistisch relevant zijn.

Contents

1

Introduction

1.1 Motivation

Interaction networks describe a wide range of systems in society and nature. Frequently cited examples include the World Wide Web [5, 52], a network of documents (web pages) connected by various hyperlinks (URL's), and the cell, a network of chemicals (genes) linked by biochemical reactions [70].

A biological interaction network can be described as follows. An organism, e.g., yeast, consists out of genes, proteins, and other elements. These genes and proteins are the nodes in such networks and can be connected to other genes and proteins in that network. Each gene/protein has different functions and can thereby activate or disactivate others. A gene/protein expression represents the behavior for each gene/protein at certain time instants. When one or more genes/proteins, regardless of their interactions, behave irregular, a disease can occur. To repair or limit the "damage", it is thus important to detect and understand these genes/proteins and their behavior.

For example, each organism's cell contains genes that control the cell-division when certain tissue needs to be repaired. When these genes act erratically, an uncontrolled growth of tissue by cell-division ensues, known as cancer. This uncontrolled growth can be stimulated or slowed down by external factors such as hormones and chemical substances as well as internally by other genes. Therefore, it is very important to trace these irregular genes and the external-

and internal factors that influence their expression level. By representing the genes/proteins activities of an organism as an interaction network—a matrix which contains the rate of interaction for each pair of components—tracing these interactions can be simplified by evaluating this corresponding interaction matrix.

Over the last two decades, research in the area of genetics and bio-informatics has increased spectacularly, and the gene-protein networks are beginning to be understood. One of the main contributing factors has been the development of microarray technologies, which has enabled the measurement of gene/protein expression levels and profiles on a genome-wide scale. Furthermore, from the experiments it is known that these networks are sparse, each gene only interact with a small number of other genes. This motivates us to try to identify a dynamic, sparse interaction network from noisy measurement data, such as a set of experimental microarray time series. Over the years, a number of different formalisms for modeling the interactions amongst genes and proteins have been presented. For a thorough overview, we refer to de Jong [22], Bower [16], Guthke et al. [42], and others [27, 41, 58, 37].

In contrast to other work, we are interested in dynamical models which focus on cause-effect relations, and do not discuss static models which focus on co-occurrence. A prerequisite for the successful reconstruction of these networks is the way in which the dynamics of their interaction is modeled.

1.2 Problem statement

First, we consider the problem of modeling and identifying sparse, dynamic interaction networks from a given set of observations.

Real-world networks consist of a high number of nodes or components. Hence, there are millions of possible interactions between these components. However, this does not necessarily imply that there are also a high number of actual interactions between the components, however. Most often, components are influenced by only a small number of others. Therefore, we are interested in sparsity criteria.

To identify the interactions for such networks, a very large number of experimental data is required. In practice, however, due to the high costs of experiments, only very few measurements are performed compared to the vast amount of unknown interactions. This means that we have to deal with a

large number of unknowns and a small number of observations (poor data). This substantial lack of data can give rise to an identification problem. It thus becomes crucial to work around this by exploiting as much as possible additional information about the underlying system.

Furthermore, noise is ubiquitous in real-world applications. Hence, it is mandatory to test the robustness of the algorithm to be developed.

Beside the identification of sparse interaction networks, we are also interested in the capacity of these interaction networks to store input-output patterns. For each input, there exists an output. By storing these patterns, a system can learn and react appropriately to a given input. For example, the more input-output patterns an organism can store, the more it learns, and the better it is capable to give the best response.

Summarized, to identify interaction networks, it is important to understand their dynamics. Hence, we are interested in the relation between the number of components and the number of observations, the sparsity of the network, and the influence of noise. Furthermore, we are interested in the storage capacity of sparse interaction networks.

It is useful to have an algorithm that is able to identify networks. Because performing experiments is accompanied with high costs and measurement noise, algorithms have a large advantage if they are able to deal with poor and noisy data sets.

1.3 Preliminaries

As already mentioned, our aim is to identify and reconstruct a sparse interaction network, or *regulatory network*, with a relatively small amount of observations.

Let us denote the number of components, in the biological case the number of gene expression levels, by N and the interaction network by a matrix A that contains the interaction coefficients. Because each component can interact with each other component, the dimension of A is $N \times N$. Furthermore, we have a matrix B that contains the external stimuli coefficients. P external influences can interact with all N components, therefore, the dimension of the external input matrix B is $N \times P$.

The system matrix A is *sparse*, because it consists mostly of zeros. Indeed, most components (genes/proteins) interact with only a relatively small number of other components. We use k to denote the number of non-zero entries per row of A. This k is not constant and can thus be different for each row of A. In our experiments, k will be the same for each row. This assumption is not a restriction, because k will be estimated during the learning. Note that the rate of sparsity has to be much smaller than the number of components, $k \ll N + P$.

To identify such networks, we need a data set. For a biological network, this data set represents microarray readings from samples taken at different time instants. A *microarray* consists of an arrayed series of thousands of microscopic spots, whereby each spot contains DNA of one specific gene/protein, for an example see Figure 1.1. A microarray can be used to measure the gene/protein activity at a certain observation. Thus, to study the changes in expression levels, microarrays are taken at different time points.

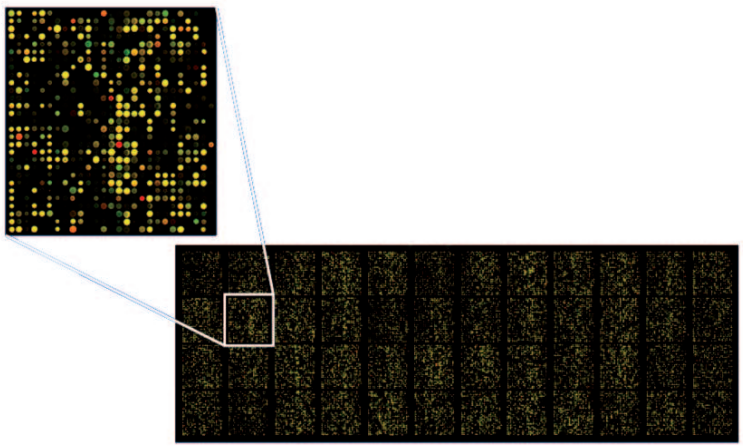

Figure 1.1: An example of a microarray.

Matrix $X \in \mathbb{R}^{N \times M}$ denotes the observations taken at these time instances $t_1 \dots t_M$, such that $X = x[1], \dots, x[M]$. Because of the high costs, only a very few microarray experiments are performed. Therefore, we have to deal with *poor data*: only a small number of observations are available compared to the number of components, i.e., $M \ll N$. Hence, the system is *underdetermined*. One standard approach to circumvent this problem is by dimension reduction

through the clustering of related components. A different perspective is offered by including the sparsity.

Furthermore, it is assumed that there is knowledge of the input signal $u(t)$ for $1 \leq t \leq M$ and accurate estimates of the state derivatives at these time instants can be computed. The way to estimate $\dot{X} = \dot{x}[1], \ldots, \dot{x}[M]$ is by interpolation of the state observations. The derivative $\dot{x}(t_k)$ can be approximated from the observations, e.g., $\dot{x}[k] \approx (x[k] - x[k-1])/(t_k - t_{k-1})$.

In this work, it is thus always assumed that the data matrices $\dot{X}$, X and U—also called *empirical data*—are known. Given these data, the matrices A and B—also known as *system matrices* or *system parameters*—need to be identified.

Noise is ubiquitous in real-world applications. Hence, it is mandatory to test the algorithm's robustness. For the biological network, noise represents system and measurement noise. System noise is divided into intrinsic noise, which varies from cell to cell and over time in a single cell, and extrensic noise, which is global to a single cell but varies from one cell to another. Measurement noise arise from the experiments. So, to complete the system's data set, the contribution of some system noise, ξ, is added to the output data set $\dot{X}$, and the contribution of some measurement noise, ζ, is added to the output measurement set Y. They are normally distributed with zero mean and some standard deviation σ that determines the noise level.

We will also study the identification process as a *machine learning* process. A learning process is defined by Mitchell [53] as searching through a very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by the learner. Machine learning techniques are useful first, when, large databases may contain valuable implicit regularities that can be discovered automatically, second, in poorly understood domains where humans might not have the knowledge needed to develop effective algorithms, and, third, in domains where the program must dynamically adapt to changing conditions.

Learning problems requires an exhaustive search of all possible subsets of features of the chosen cardinality. If large numbers of features are available, this is impractical. For practical learning algorithms, the search is for a satisfactory set of features instead of an optimal set. To select such a subset of relevant features for building a robust learning model, the term *feature selection* is used.

Later in this work, we are interested in the maximum number of linearly independent input-output patterns that can be stored in a network. A *pattern* represents the input at a certain time instance and its optimal network response.

1.4 Theoretical foundation

Real-world networks, studied in Chapter 2, consist of a high number of nodes or components. For the World Wide Web, the unregulated growth of web pages and the interactions between them leads to a huge and complex web, which becomes a large network, close to 20 billion nodes at the end of 2005 (see [2]). From biology, it is known that sparse gene-protein interaction networks consist of thousands of components. Hence, there are millions of possible interactions between these components. Therefore, to identify such sparse networks, we have to deal with poor data.

Before we are able to identify and reconstruct sparse networks, we have to model the interactions between the unknown components. In Chapter 3, these interactions will be considered as reactions so that they can be represented as a set of ordinary differential equations:

$$\dot{x} = f(x, u|\theta) + \xi(t). \tag{1.1}$$

Here, $x(t)$, called the state-vector, denotes the activity of the N unknown components at time t—possibly involving higher order time derivatives, $u(t)$ denotes the P controlled inputs to the system, $\xi(t)$ denotes a stochastic Gaussian white noise term with mean zero. Moreover, this expression involves a parameter vector θ that contains the coupling constants between the components activities.

Starting from this rate equation, we introduce and study two models, the Single Linear Model in Section 3.2.1, and the Piecewise Linear Model in Section 3.2.2.

First, we consider the Single Linear Model. For this model, the entire state-space is considered as one big space, whence the systems behavior is governed by its specific (un)stable equilibrium point, and the interactions are modeled by a linear dynamic state-space system of the form

$$\dot{X} = AX + BU + \xi.$$

This equation is a simplified form of the ordinary differential equation 1.1 with $\dot{X}$, X and U the known data matrices. Because the number of observations is

much smaller than the number of unknowns, infinitely many solutions for the system matrices A and B can be found, from which the linear programming will select the most sparse interaction matrix A. Thus, to identify a sparse interaction matrix, the technique of partial L_1-minimization is used where the sparsity criterion is used as a constraint.

Second, we consider the Piecewise Linear Model. The entire state-space is being partitioned into cells or subsystems, where attractors reign. Thus, the behavior of the state-vector can be described by motion through this collection of subsystems, swiftly moving through subsystems of repellers, until they enter the basin of attraction of an attractor. Under the effects of external agents (via the external input-vector) or by stochastic fluctuations, they can leave this subsystem, and start wandering again, thereby repeating the process. Now, a vital assumption is that in each subsystem the behavior is governed by specific (un)stable equilibrium points. Therefore, it is possible to make a linear approximation of the ordinary differential equation for each subsystem with index ℓ as

$$\dot{X} = A_\ell X + B_\ell U + c_\ell.$$

By introducing a weight matrix—which represents the membership functions of observation m to subsystem ℓ, this piecewise linear differential equation can be reformulated as a single linear differential equation:

$$\dot{X} = H_1 X + H_2 U + \xi.$$

Therefore, the Piecewise Linear Model is computationally strong, because the identification of the Single Linear Model is very fast. Similarly, $\dot{X}$, X and U are the known data matrices. H_1 and H_2 relate to the weight matrix and the—unknown—system matrices such that this is no longer a linear problem. Therefore, we follow a different approach and split the problem into a linear programming (LP) and a "matching" algorithm. Because of this, the technique of L_1-minimization can be used again, controlled by the distance between the data and the model.

In Chapter 4, we want to generalize this process of identification and reconstruction and reformulate it as a learning process. For our model, we have a teacher, $T = (A^*, B^*)$. For the teacher's output on some input X, the following equation holds:

$$T(X) = A^* X + B^* U.$$

The aim is to reproduce the teacher's output for any input perfectly after seeing M examples, so we need a student, $S = (A_{\text{est}}, B_{\text{est}})$, to learn. Learning means that the algorithm has to determine the subset of features used by the teacher. In this context, the algorithm has to search for the interactions, $(A_{\text{est}}, B_{\text{est}})$, using the known data set $(\dot{X}, X, U)$ and the rate equation, introduced above, such that $(A_{\text{est}}, B_{\text{est}})$ is an acceptable fit for the original one, (A^*, B^*). With an "acceptable" fit, we mean that the estimated matrix can differ from the original one with a small but acceptable rate of error.

The unknown components of networks interact with each other, but they also interact with external stimuli. The variety of the stimuli, e.g., for the biological network, is immense. They can be either harmful or beneficial, obligatory or supplementary, lethal or lifesaving. They range from (lack of) illumination, (too low or high) temperature, (shortage of) food and water, to toxic agents and viral or bacterial infections. The efficiency of an organism directly depends on its ability to optimally react to these external inputs.

If a specific influence, from another component or from an external stimulus, is presented with some regularity, it is greatly beneficial for the system to learn and remember its best response. Response here is the output of the system. The pair of the input and the optimal network response defines an input-output pattern, which the interaction network has to "store" in its "memory". For example, the more patterns an organism can store with its current network architecture, the better the organism is equipped to live in its natural environment.

The potential of sparse interaction networks to store sparse input-output patterns is studied in Chapter 5. This problem bears similarity to the engineering task of network identification from experimental data and is based on the folowing equation:

$$\dot{X} = AX + BU + \xi.$$

The central question in Chapter 5 thus concerns the memory capacity of a network of N components which interact according to a simple linear state-space model with P external outputs. It is assumed that the set of best responses to a given input is available.

In Chapter 6, we will apply our identification approach on a real dataset stemming from Spellman et al. [70], which is available at [1]. Yeast *Saccharomyces cerevisiae* is an unicellular fungus found naturally in grapevines which is responsible for wine-making fermenting sugars and producing alcohol. For this

data, the expressions profile of about 6000 genes are measured by microarrays at 24 time points every five hours.

To identify and reconstruct the interactions, the Single Linear Model and the Piecewise Linear Model will be applied on this dataset.

1.5 Numerical experiments and simulations

All experiments were performed on a PC with an Intel Pentium M processor of 1.73 GHz and 1 GB RAM memory under Windows XP Professional. For identification purposes, we have used the following software packages:

1. Matlab 6.5 Release 13, including the Optimization Toolbox. The latter's routine linprog was used to solve LP problems. Its default solution method is a primal-dual interior point method, but an active set method can optionally be used, too. For larger problems, it turned out to be essential for obtaining reasonable computation times that the LP problems were solved by application of the active set method on the dual problem formulation. Therefore, this method was adopted throughout all the experiments.

2. The computer algebra package Maple 9.5. The standard implementation of linear programming in Maple is used, which is very convenient since it allows to specify the objective function and the constraints symbolically.

The advantage of Maple is that an LP problem is formulated in terms of equalities and inequalities. Therefore, constraints can be adapted easily and small experiments can be run fast. In Matlab, the LP is formulated in terms of matrices, which are computed via a number of transformations. These transformations take some time and errors can appear. For large experiments, Matlab is faster.

Since results can depend on the particularities of given data and the original system that generated it, all experiments have been performed on a number of independent runs on randomly selected data and systems. Hence they convey the behavior of our approach "on average".

To measure the quality of the identification process, we will use the following quantities:

- CPU-time, T_c;
- false positives, n_{fpos};

- false negatives, n_{fneg};
- correlation errors, n_{corr};
- zero-threshold, θ_z;
- system error, N_{e};
- student-teacher error, $S \odot T$;
- validation error, ε_{val};
- generalization error, ε_{gen};
- data error, D_{e}.

Using the internal clock, the *CPU-time*, T_c, is used to measure the time required to perform the full computation.

To check whether or not the student and teacher agree, we use the number of *false positives*, the number of *false negatives*, and the *correlation error*. Concretely, n_{fneg} represents the number of interactions from the original system A^* that do not occur in the resulting system, A_{est}, n_{fpos} represents the number of interactions from the resulting system that do not occur in the original system and n_{corr} represents the number of components of the original and resulting system that are significantly non-zero, but have an opposite sign. With k^* the number of non-zeros per row of A^*,

$$\begin{aligned} 0 &\leq n_{\text{fneg}} \leq & Nk^*; \\ 0 &\leq n_{\text{fpos}} \leq & N(N-k^*); \\ 0 &\leq n_{\text{corr}} \leq & Nk^* - n_{\text{fneg}}. \end{aligned}$$

To decide whether a component is zero or not, the *zero-threshold*, θ_z, is introduced. For all experiments $\theta_z = 10^{-5}$. This means that if component $|A_{ij}| > \theta_z$, A_{ij} is assumed to be a non-zero component, and vice versa.

The *system errors*, N_{e}, are generated in the reconstruction by the failure of the algorithm to identify the true non-zero elements of the original sparse interaction matrix A^*. The false positives and false negatives in the reconstructed interaction matrix A_{est} are added up to produce the total number of system errors:

$$N_{\text{e}} = \frac{n_{\text{fpos}} + n_{\text{fneg}} + n_{\text{corr}}}{N}.$$

The *student-teacher error*, $S \odot T$, compares each component of the student with the corresponding component of the teacher:

$$S \odot T = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{z}{N^2} \text{ with } \begin{cases} z & = 1 \text{ if } \quad |A_{\text{est}_{i,j}}| > \theta_z \text{ and } |A^*{}_{i,j}| > \theta_z \text{ or} \\ & \quad\ |A_{\text{est}_{i,j}}| \leq \theta_z \text{ and } |A^*{}_{i,j}| \leq \theta_z; \\ z & = 0 \text{ else.} \end{cases}$$

It is important that the approach is not only capable of reproducing the orignal input, but it should also be able to perform well on input that was not used by the algorithm. To test this generalization ability the validation error, the generalization error, and the data error are defined.

The *validation error*, ε_{val}, is defined as the number of input-output pairs $(x_v, u_v, \dot{x}_v)$ of a validation set of size V (with $1 \leq v \leq V$) that is not reproduced by the student for one independent run:

$$\varepsilon_{\text{val}}(A_i^*, B_i^*, \chi_{tr\,i}) = \sum_{v=1}^{V} H\left(\frac{|A_{\text{est}} x_v + B_{\text{est}} u_v - \dot{x}_v|}{|\dot{x}_v|} - \varepsilon_{\text{err}}\right),$$

with $H(x) = 1$ if $x > 0$ else $H(x) = 0$ and ε_{err} the maximum deviation from zero that is considered insignificant.

The *generalization error*, ε_{gen}, is defined as the ratio of the number of students wherefore $\varepsilon_{\text{val}} > 0$ to the total number of independent runs:

$$\varepsilon_{\text{gen}} = \frac{1}{nrRuns} \sum_{i=1}^{nrRuns} H\left(\varepsilon_{\text{val}}\left(A_i^*, B_i^*, \chi_{tr\,i}\right)\right),$$

with χ_{tr} an independent training set for each run. Note, the validation error is a measure for one run of the algorithm, whereas the generalization error measures the performance of the algorithm.

The difference between the desired system output, $\dot{X}^*$, and the resulting system output, $\dot{X}_{\text{est}}$, is computed by the *data error*:

$$D_{\text{e}} = \frac{|\dot{X}_{\text{est}} - \dot{X}^*|}{|\dot{X}^*|}.$$

1.6 Publications

The chapters in this dissertation are based on the publications [45, 46, 47, 81, 82, 83].

2

Dynamical behavior of interaction networks

2.1 Properties of real-world networks

The simplest and most straightforward form of a network—without apparent design principles— is the random graph, first studied by Rapoport [65, 66, 69] and by Erdös and Rényi [28, 29]. Most of the interesting features of real-world networks that have attracted the attention of researchers in the last few years, however, concern the ways in which networks are not like random graphs. In this Section, we describe the three most prominent features that appear to be common to networks of many different types.

2.1.1 The small-world effect

In a network with the small-world property, most pairs of vertices are connected by a short path through the network. Consider an undirected network, and let us define l to be the mean geodesic (i.e., shortest) distance between vertex pairs in a network:

$$l = \frac{1}{\frac{1}{2}n(n+1)} \sum_{i>j} d_{ij}, \tag{2.1}$$

where n is the number of vertices and d_{ij} is the geodesic distance from vertex i to vertex j. Thus for a small-world network, l is proportional to $\log n$.

The small-world effect has obvious implications for the dynamics of processes taking place on real-world networks. For example, if one considers, e.g., the spread of information across a network, the small-world effect implies that that spread will be fast on most real-world networks. If it takes only a small number of steps for a rumor to spread from any person to any other, for instance, then the rumor will spread much faster than if it takes a hundred steps, or a million.

On the other hand, the small-world effect is also mathematically obvious. Pool and Kochen [64] noted that, if the number of vertices within a distance r of a typical central vertex grows exponentially with r—and this is true for many networks, including the random graph (see Section 2.3.1), the value of l will increase as $\log n$. In recent years, the term small-world effect has taken on a more precise meaning: networks are said to show the small-world effect if the mean geodesic distance l scales logarithmically or slower with network size for fixed mean degree. Logarithmic scaling can be proved for a variety of network models [13, 14, 19, 23, 33] and has also been observed in various real-world networks [4, 55]. Some networks have mean vertex-vertex distances that increase slower than $\log n$. Bollobás and Riordan [15] have shown that networks with power-law degree distributions (see Section 2.1.3) have values of l that increase no faster than $\log n / \log \log n$.

2.1.2 The network transitivity and clustering coefficient

If vertex A is connected with vertex B and vertex B is connected with vertex C, then vertex A is connected with vertex C with high probability. This property is called clustering or network transitivity. In the language of social networks, the friend of your friend is likely also to be your friend. In terms of network topology, the network contains triangles—sets of three vertices that are all connected to each other. To quantify, a clustering coefficient C is defined:

$$C = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of vertices}}, \tag{2.2}$$

where a "connected triple" means a single vertex with two neighbors, which are or are not connected with each other (see Figure 2.1). The definition implies that C lies in the range of $0 \leq C \leq 1$. Notice that $C = 1$ if and only if the network is a clique. More concrete, C is the mean probability that two vertices that are neighbors of the same vertex are neighbors of each other.

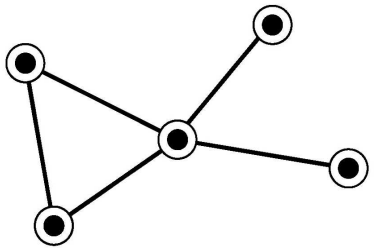

Figure 2.1: Illustration of the definition of the clustering coefficient C (Equation 2.2). This network has one triangle and eight connected triples, and therefore has a clustering coefficient of $3 \times \frac{1}{8} = \frac{3}{8}$.

2.1.3 The degree distribution

Not all nodes in a network have the same number of edges. The degree k of a vertex in a network is the number of connections to that vertex. The spread in the degrees is characterized by a distribution function p_k, which gives the probability that a randomly selected vertex has degree k.

Since in a random graph the edges are placed randomly, the majority of vertices have approximately the same degree, close to the average degree k of the network. The degree distribution of a random graph is a Poisson distribution with a peak at p_k. For most large networks, the degree distribution significantly deviates from a Poisson distribution. In particular, for a large number of networks, including the World Wide Web [5, 52], the Internet [30, 18, 78] or Metabolic Networks [48, 49], the degree distribution has a power-law tail,

$$p_k \sim k^{-\gamma}, \tag{2.3}$$

with, for example, $\gamma = 2.1$ for the World Wide Web [5], $\gamma = 2.2$ for the Internet [30] and $\gamma = 2.2$ for the Metabolic Network of *Escherichia coli* [49]. For more details, see also Section 2.2. Often, a histogram of the degrees of the networks vertices is used to represent the degree distribution for that network. Because of the power-law, histograms are usually rather noisy in the tail. An alternative way of presenting the degree is to make a plot of the cumulative distribution function

$$p_k = \sum_{k'=k}^{\infty} p_{k'}, \tag{2.4}$$

which is the probability that the degree is greater than or equal to k.

2.2 Real-world networks

Complex network structures describe a wide variety of systems of high technological and intellectual importance. For example, a biological cell is best described as a complex network of chemicals connected by chemical reactions; the Internet is a complex network of routers and computers linked by various physical or wireless links; fads and ideas spread on the social network, whose nodes are human beings and whose edges represent various social relationships; and the World Wide Web is an enormous virtual network of web pages connected by hyperlinks. These systems represent just a few of the many examples that have recently prompted the scientific community to investigate the mechanisms that determine the topology of complex networks. In this Section, we describe some examples more into detail.

2.2.1 Information networks: the World Wide Web

An example of an information network is the World Wide Web. This network represents the largest real-world network for which topological information is currently available. Any individual can create a website with any number of documents and links. This unregulated growth leads to a huge and complex web, which becomes a large directed graph whose vertices are the documents (web pages) and the edges are the hyperlinks (URLs) that point from one document to another (see Figure 2.2). Albert, Jeong, and Barabási [5] have found that the size of this network was close to one billion nodes at the end of 1999, and in `http://www.boutell.com/newfaq/misc/sizeofweb.html`, it is shown that this increased to 20 billion at the end of 2005.

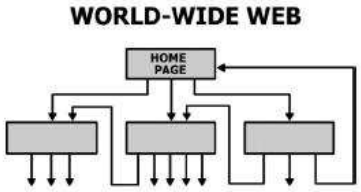

Figure 2.2: Network structure of the World Wide Web. The nodes of the World Wide Web are web documents, connected with directed hyperlinks (URLs).

Since the edges of the World Wide Web are directed, the network is characterized by two degree distributions, the out-degree and in-degree. The out-degree distribution is the probability that a document has k outgoing hyperlinks p_{out}

and the in-degree distribution is the probability that k hyperlinks point to a certain document p_{in}. Several studies [5, 52, 17, 3] have established that both have power-law tails. Albert, Jeong, and Barabási [5] have studied a subset of the World Wide Web containing 325729 nodes and have found $\gamma_{out} = 2.45$ and $\gamma_{in} = 2.1$. Kumar et al. [52] used a 40-million-document crawl by Alexa Inc., obtaining $\gamma_{out} = 2.38$ and $\gamma_{in} = 2.1$.

Despite the large number of nodes, the World Wide Web displays the small-world property. This was first reported by Albert, Jeong, and Barabási, who found that the average path length for a sample of 325729 nodes was 11.2 and predicted, using finite size scaling, that for the full World Wide Web of 800 million nodes that would be a path length of approximately 19. Subsequent measurements by Broder et al. [17] found that the average path length between nodes in a 50-million node sample of the World Wide Web is 16, in agreement with the finite size prediction for a sample of this size.

The directed nature of the World Wide Web does not allow them to measure the clustering coefficient using Equation 2.2. One way to avoid this difficulty is to make the network undirected, making each edge bidirectional. This was the path followed by Adamic [3], who studied the World Wide Web at the domain level using a 1997 Alexa crawl of 50 million web pages distributed among 259794 sites. The nodes that had have only one edge were removed, so a network of 153127 sites was left. While these modifications are expected to increase the clustering coefficient somewhat, Adamic found $C = 0.1078$, orders of magnitude higher than $C_{rand} = 0.00023$, corresponding to a random graph of the same size and average degree.

2.2.2 Biological networks: the Metabolic Network

A number of biological systems can be usefully represented as networks, for which the network of metabolic pathways is a classic one. It is a representation of metabolic substrates and products with directed edges joining them if a known metabolic reaction exists that acts on a given substrate and produces a given product. Studies of the statistical properties of Metabolic Networks have been performed by, for example, Jeong et al. [49, 63], Fell and Wagner [31], and Stelling et al. [71].

Biochemical reactions proceed in one direction, so for each node they distinguish between incoming and outcoming links. As illustrated in Figure 2.3, their results convincingly indicate that the probability that a given substrate participates in k reactions follows a power-law distribution. For instance, in

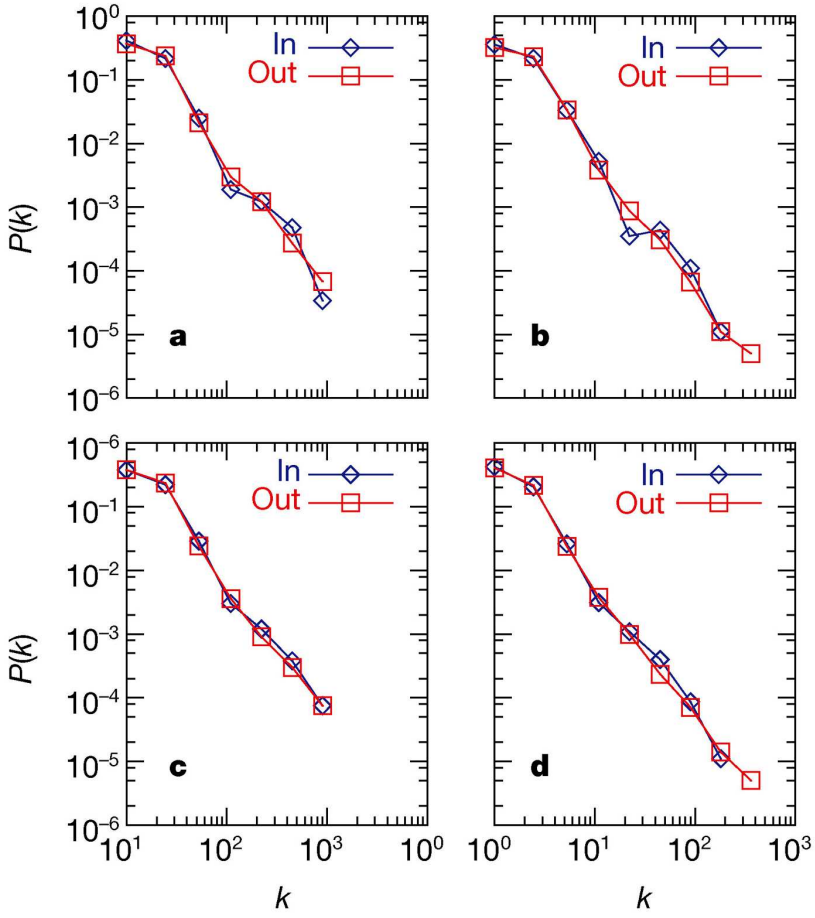

Figure 2.3: Connectivity distributions for substrates: (a) *Archaeoglobus fulgidus* (archae); (b) *Escherichia coli* (bacterium); (c) *Caenorhabditis elegans* (eukaryote), shown on a log-log plot, counting separately the incoming (In) and outgoing links (Out) for each substrate (the parameter k_{in} (k_{out}) corresponds to the number of reactions in which a substrate participates as a product (educt)); (d) the connectivity distribution averaged over all 43 organisms. [49]

Escherichia coli, the probability that a substrate participates as an educt in k metabolic reactions follows incoming degree distribution, with $\gamma_{in} = 2.2$ and the probability that a given substrate is metabolised by k different metabolic reactions follows a similar distribution, with $\gamma_{out} = 2.2$.

Understanding the large-scale structure of cellular networks cannot only provide valuable and perhaps universal structural information, but could also lead to a better understanding of the dynamical processes that generated them.

To study the small-world character, the shortest biochemical pathway averaged over all pairs of substrates is called the network diameter. For non-biological networks examined by Barabási and Albert [6] and Watts and Strogatz [80], the average connectivity of a node is fixed, which implies that the diameter of a network increases logarithmically with the addition of new nodes. For Metabolic Networks, this implies that a more complex bacterium with more enzymes and substrates, such as *Escherichia coli*, would have a larger diameter than a simple bacterium, such as *Mycoplasma genitalium*. They find, however, that the diameter of the Metabolic Network is the same for all 43 organisms, irrespective of the number of substrates found in the given species. This is unexpected and is possible only if, with increasing organism complexity, individual substrates are increasingly connected to maintain a relatively constant metabolic network diameter. Further, they find that the average number of reactions in which a certain substrate participates increases with the number of substrates found within a given organism.

2.2.3 Social networks: the Scientific-Collaboration Network

A social network can be defined as a set of people or groups of people with some pattern of contacts or interactions between them. Examples are the patterns of friendship between individuals, business relationships between companies, and the network of collaborations between scientists, where two nodes are connected if the two scientists have written an article together. To uncover the topology of this complex network, Newman [55] studied four databases spanning physics, biomedical research, high-energy physics, and computer science over a five-year window (1995–1999).

All these networks show a small average path length, but a high clustering coefficient, as summarized in Table 2.1. The degree distribution of the collaboration network of high-energy physicists is an almost perfect power law with an exponent of 1.2, while the other databases display power laws with a larger exponent in the tail.

Barabási et al. [9] investigated the collaboration network of mathematicians and neuroscientists publishing between 1991 and 1998. The average path length of these networks is around $l_{math} = 9.5$ and $l_{nsci} = 6$, their clustering

coefficient being $C_{math} = 0.59$ and $C_{nsci} = 0.76$. The degree distributions of these collaboration networks are consistent with power laws with degree exponents 2.1 and 2.5, respectively.

Network	Size	k	l	l_{rand}	C	C_{rand}	Reference
LANL	52909	9.7	5.9	4.79	0.43	1.8×10^{-4}	[55]
MEDLINE	1520251	18.1	4.6	4.91	0.066	1.1×10^{-5}	[55]
SPIRES	56627	173	4.0	2.12	0.726	3×10^{-3}	[55]
NCSTRL	11994	3.59	9.7	7.34	0.496	3×10^{-4}	[55]
Math.	70975	3.9	9.5	8.2	0.59	5.4×10^{-5}	[9]
Neurosci.	209293	11.5	6	5.01	0.76	5.5×10^{-5}	[9]

Table 2.1: The general characteristics of several co-authorship networks. The number of nodes, the average degree k, the average path length l and the clustering coefficient C are indicated. For a comparison, the average path length l_{rand} and clustering coefficient C_{rand} of a random network of the same size and average degree is introduced. [55, 9]

2.3 Network structures

One of the most interesting developments in the understanding of complex networks was the discovery that, for most large networks, the degree distribution significantly deviates from a Poisson distribution. This has initiated a revival of network modeling in the past few years, resulting in the introduction and study of three main classes of modeling paradigms: the random networks, the small-world networks, and the scale-free networks.

2.3.1 Random networks

A random network is a network of nodes that are connected with each other by a random number of edges.

They are the simplest and most straightforward realization of a complex network and were first studied by Solomonoff and Rapoport [69] and Erdös and Rényi [28] who proposed a simple model $G_{n,p}$ of a random network: starting with n vertices, connect every pair of nodes with probability p to create a network with approximately $pn(n-1)/2$ edges distributed randomly. Actually, $G_{n,p}$ is the ensemble of all such networks in which a network having m edges

appears with probability $p^m(1-p)^{M-m}$—the probability that a node is connected with m other nodes against the probability that this is not true—where $M = n(n-1)/2$ is the maximum possible number of edges.

Many properties of the random graph are exactly solvable in the limit of large graph size, as was shown by Erdös and Rényi in a series of papers in the 1960s [28, 29]. Typically, the limit for large n is taken holding the mean degree $z = p(n-1)$ constant, in which case the model clearly has a Poisson degree distribution, since the presence or absence of edges is random, and hence the probability of a vertex having degree k is

$$p_k = \binom{n}{k} p^k (1-p)^{n-k} \approx \frac{z^k e^{-z}}{k!}, \tag{2.5}$$

with the last approximation tending towards equality in the limit for large n and fixed k.

The random graph reproduces well one of the principal features of real-world networks, namely the small-world effect. The mean number of neighbors a distance d away from a vertex in a random graph is z^d, and hence the value of d needed to encompass the entire network is $z^l \approx n$. Thus, a typical distance through the network is $l = \log n / \log z$, which satisfies the definition of the small-world effect given above. Rigorous results to this effect can be found in, for instance, articles by Bollobás [13, 14].

However, in almost all other respects, the properties of the random graph do not match those of networks in the real world. The random graph has a low clustering coefficient: the probability of connection of two vertices is p regardless of whether they have a common neighbor, and hence $C = p$, which tends to zero as n^{-1} in the limit for large system size [80]. The model also has a Poisson degree distribution [78].

Random networks are extended in a variety of ways to make them more realistic. These are called generalized random graphs. Examples are directed graphs and bipartite graphs. These extensions deal with the problem of the degree distribution, but they still have a shortcoming in that they fail to capture the common phenomenon of transitivity.

2.3.2 Small-world networks

A small-world network is a network in which most nodes are not neighbors of one another, but most nodes can be reached from any other one by a small number of steps.

To create a small-world network, Watts and Strogatz [80] created the small-world model. The idea is to build a network on a low-dimensional regular lattice and then adding or moving edges to create a low density of shortcuts that join remote parts of the lattice to one another.

Starting from a low-dimensional lattice of n vertices with some boundary conditions, e.g., a ring, and connecting each vertex with its k nearest neighbors, we get a system as in Figure 2.4, with nk edges. The small-world network is then created by rewiring a small fraction of edges. Therefore, for each edge, with probability p, one end of that edge will be removed to a new location chosen uniformly at random from the lattice, except that no double edges or self-edges are ever created. This process is illustrated in Figure 2.4.

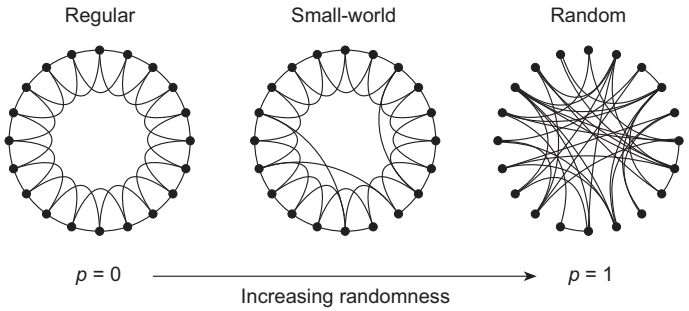

Figure 2.4: The random rewiring procedure of the Watts-Strogatz model, which interpolates between a regular ring lattice and a random network without altering the number of nodes or edges. We start with $n = 20$ nodes, each connected to its four nearest neighbors. For $p = 0$, the original ring is unchanged; as p increases, the network becomes increasingly disordered until, for $p = 1$, all edges are rewired randomly. [80]

Figure 2.4 illustrates that the model interpolates between a regular lattice, $p = 0$, and a random graph, $p = 1$. For $p = 0$, the clustering coefficient is $C = (3k - 3)/(4k - 2)$, which tends to $3/4$ for large k. The regular lattice, however, does not show the small-world effect because the mean geodesic dis-

tance between vertices tends to $n/4k$ for large n. When $p = 1$, each vertex is connected to a new random neighbor. The geodesic distances are of the order of $\log n / \log k$, but the clustering coefficient is very low, $C \approx 2k/n$. However, Watts and Strogatz showed by numerical simulation that there exists a sizable region in between these two extremes for which the model has both low path lengths and high transitivity (see Figure 2.5).

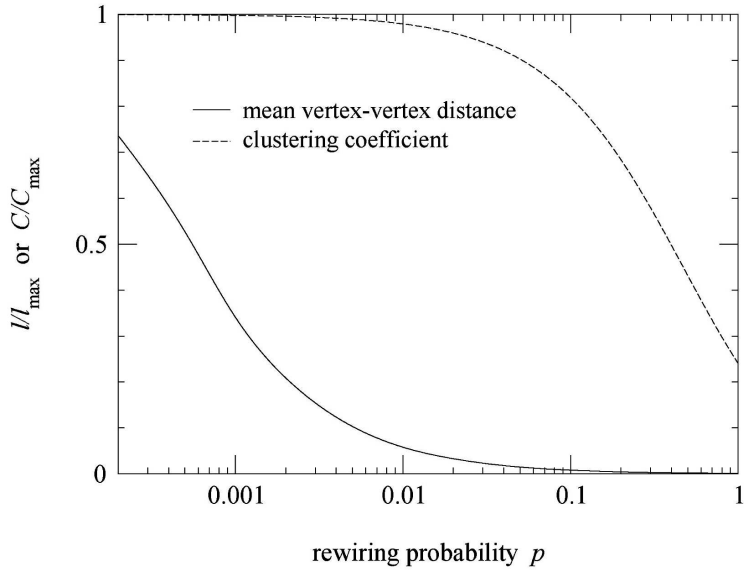

Figure 2.5: The clustering coefficient C and mean vertex-vertex distance l in the small-world model of Watts and Strogatz as a function of the rewiring probability p. For convenience, both C and l are divided by their maximum values, which they assume when $p = 0$. Between the extremes $p = 0$ and $p = 1$, there is a region in which clustering is high and mean vertex-vertex distance is simultaneously low. [80]

Barrat and Weigt [11] computed the clustering coefficient for small-world models as

$$C = \frac{3(k-1)}{2(2k-1)}(1-p)^3. \tag{2.6}$$

The degree distribution of the small-world models does not match most real-world networks very well and is rather complicated. The full expression ex-

plained in [11] is

$$p_j = \sum_{n=0}^{\min(j-k,k)} \binom{k}{n} (1-p)^n p^{k-n} \frac{(pk)^{j-k-n}}{(j-k-n)!} e^{-pk}, \tag{2.7}$$

for $j \geq k$ and $p_j = 0$ for $j < k$.

The disadvantage of this model is that a vertex can become disconnected from the rest of the network. To avoid this, a variant of the model has been proposed by Monasson [54] and by Newman and Watts [57] where no edges are rewired. Instead, shortcuts are added to randomly chosen vertex pairs.

2.3.3 Scale-free networks

A scale-free network is a network with a power-law degree distribution, at least asymptotically.

The network models discussed thusfar assume that one starts with a fixed number n of vertices that are then randomly connected or rewired, without modifying n. In contrast, most real-world networks describe open systems that grow by the continuous addition of new nodes. Starting from a small nucleus of nodes, the number of nodes increases throughout the lifetime of the network by the subsequent addition of new nodes. For example, the World Wide Web grows exponentially in time by the addition of new web pages, and the research literature constantly grows by the publication of new papers.

Further, network models discussed so far assume that the probability that two nodes are connected is independent of the nodes degree, i.e., new edges are placed randomly. Most real networks, however, exhibit preferential attachment, such that the likelihood of connecting to a node depends on the nodes degree. For example, a web page will more likely include hyperlinks to popular documents with already high degrees, because such highly connected documents are easy to find and thus well known, or a new manuscript is more likely to cite well-known and thus much-cited publications than less-cited and consequently less-known papers.

These two ingredients, growth and preferential attachment, inspired the introduction of the Barabási-Albert model, which led for the first time to a network with a power-law degree distribution.

The construction of a scale-free network proceeds as follows. Starting from a network with n_0 nodes, at every time step, add a new node and connect this node with n others already present in the network ($n \leq n_0$). The probability for connecting the new node to node i depends on the degree k_i of that node i. After t time steps, this procedure results in a network with $t + n_0$ nodes and tn edges. Numerical experiments indicate that this network evolves into a scale-invariant state with the probability that a node has k edges following a power-law with an exponent $\gamma_{BA} = 3$ (see Figure 2.6). The scaling exponent is independent of n, the only parameter in the model.

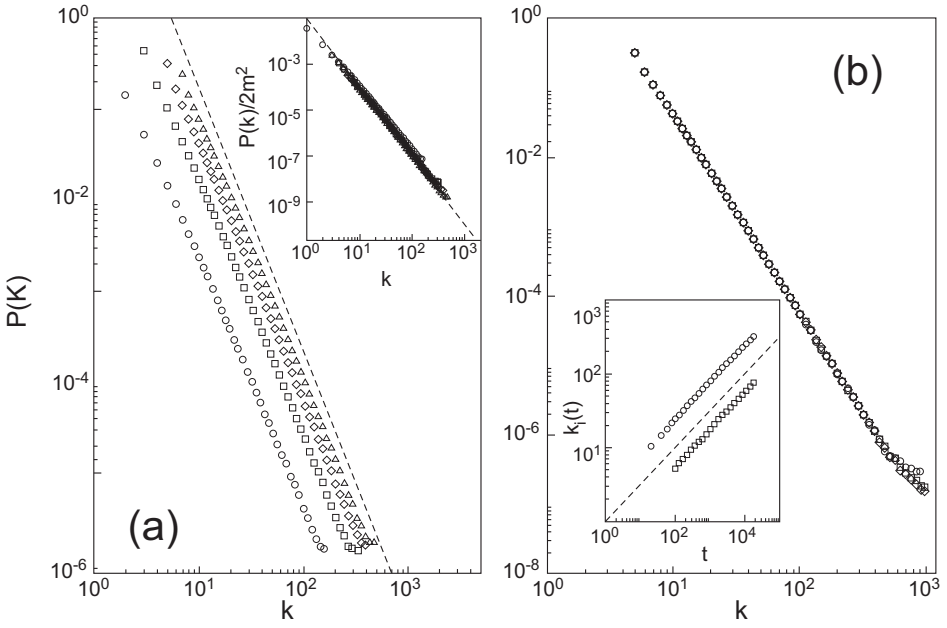

Figure 2.6: Numerical simulations of network evolution: (a) Degree distribution of the Barabási-Albert model, with $N = n_0 + t = 300000$ and $\circ$, $n_0 = n = 1$; $\square$, $n_0 = n = 3$; $\diamond$, $n_0 = n = 5$; and $\triangle$, $n_0 = n = 7$. The slope of the dashed line is $\gamma = 2.9$, providing the best fit to the data. The inset shows the rescaled distribution $p(k)/2n^2$ for the same values of n, the slope of the dashed line being $\gamma = 3$; (b) $p(k)$ for $n_0 = m = 5$ and various system sizes, $\circ$, $N = 5100000$; $\square$, $N = 150000$; $\diamond$, $N = 200000$. The inset shows the time evolution for the degree of two vertices, added to the system at $t_1 = 5$ and $t_2 = 95$. Here $n_0 = n = 5$ and the dashed line has slope 0.5. After Barabási, Albert, and Jeong [7].

Some power-law degree distributions have been observed in a range of networks, for example, the World Wide Web [5, 8], the Internet [18, 78] and Metabolic Networks [48, 49], which are shown in Figure 2.7.

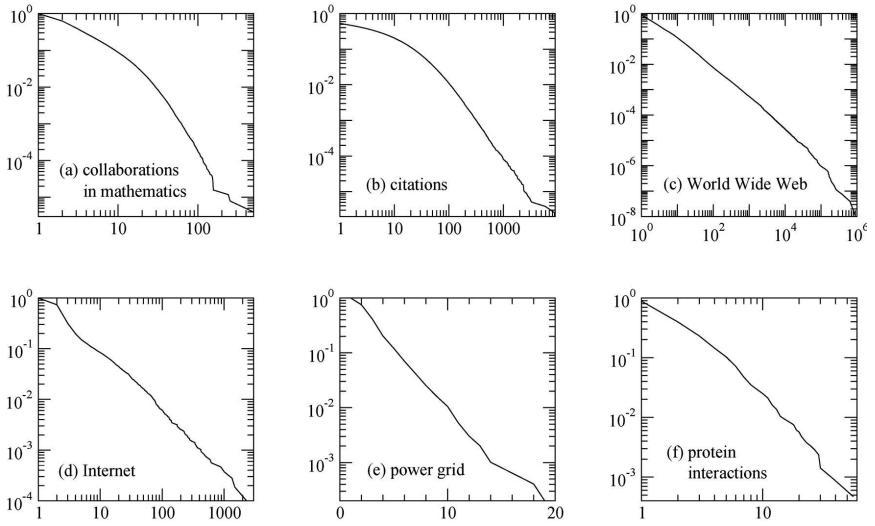

Figure 2.7: Cumulative degree distributions for six different networks. The horizontal axis for each panel is vertex degree k (or in- degree for the citation and Web networks, which are directed) and the vertical axis is the cumulative probability distribution of degrees, i.e., the fraction of vertices that have degree greater than or equal to k. The networks shown are (a) the collaboration network of mathematicians [39]; (b) citations between 1981 and 1997 to all papers cataloged by the Institute for Scientific Information [67]; (c) a 300 million vertex subset of the World Wide Web, circa 1999 [17]; (d) the Internet at the level of autonomous systems, April 1999 [18]; (e) the power grid of the western United States [80]; (f) the interaction network of proteins in the metabolism of the yeast *Saccharomyces cerevisiae* [48].

2.4 Dynamical interaction networks

To identify a regulatory network, a valid mathematical model that gives a detailed description of the given problem must be developed. There exist a lot of different methods to create such a model. Some methods find their foundation in the theory of statistics, others are based on differential or stochastic equations. Mathematics is thus a necessary tool for the modeling of a regulatory

network. For example, activation can be considered as a positive feedback loop and repression as a negative feedback loop. In this Chapter, a literature review is given of some widely used techniques.

2.4.1 Bayesian networks

In the formalism of Bayesian belief networks, studied by [32, 60], the structure of a regulatory system is represented graphically as a directed acyclic graph. Each vertex i in this graph corresponds with some random variable X_i and edges indicate conditional dependencies between these random variables. For genetic regulatory networks, X_i is defined as the expression level of gene i and edges indicate regulatory interactions between genes. An edge from gene 1 to gene 3 means that gene 1 regulates the gene expression of gene 3. A certain value of X_i is written as x_i. The graph defines the joint probability distribution over the variables. The joint probability distribution assigns probabilities to each possible assignment of values $(x_1, \ldots, x_N)$ to $(X_i, \ldots, X_N)$.

The probability of a node i in a Bayesian network is dependent of all nodes in the Markov blanket for that node i (the Markov blanket contains the immediate parents, the children, and the co-parents of these children). We say that gene 1 is a parent of gene 3 if there is a directed path from gene 1 to 3. As an example, consider gene 4 in Figure 2.8. Here, the edges indicate that gene 4 is dependent of gene 1, 2, 3, and 6, but independent of gene 5. This means that, once the expression level of gene 1, 2, 3, and 6 are known, no additional information about the expression level of gene 4 can be obtained from gene 5.

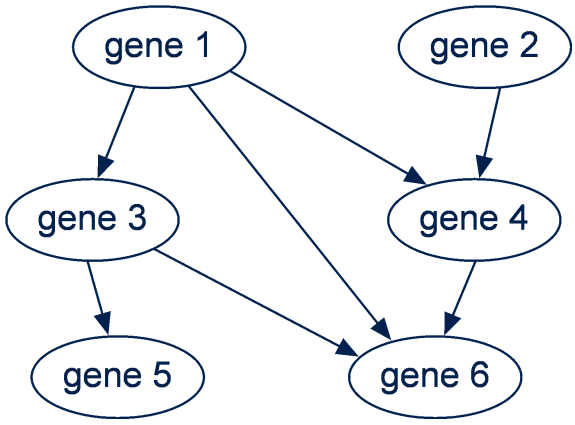

Figure 2.8: A Bayesian belief network.

The Markov condition implies that the conditional probability distributions of a bayesian network represent a factorization of the joint probability distribution as a product of conditional probability distributions of each variable $(x_1, \ldots, x_n)$ given its parents in the graph:

$$P(x_1, \ldots, x_N) = \prod_{i=1}^{N} P(x_i|\text{parents}(X_i)). \tag{2.8}$$

Learning Bayesian networks is attractive because of its solid basis in statistics. Moreover, Bayesian networks can be used when only incomplete knowledge about the system is available and the conditional independence assumptions restrict the search space. However, it is not possible to model feedback relations between nodes because an acyclic directed graph is used. Generalizations such as the dynamical Bayesian network can be used to overcome this problem.

2.4.2 Boolean networks

Boolean networks were proposed by Kauffman [50, 51]. In these networks, the expression for each component is either on or off. This means that for all components i, the expression level x_i is either 0 or 1. The state of the network is defined as a vector $x = (x_1, \ldots, x_N)$. The state of the system at time t is written as $x(t)$. The transition from $x(t)$ to $x(t+1)$ is given by a set of boolean functions $b_i\,(x\,(t))$ $(1 \leq i \leq N)$ where b_i defines the new expression level for component i based on the current state of the network. A sequence of states $(x(t), x(t+1), \ldots, x(t+N))$ connected by transitions is a *trajectory*. As an example, a boolean network is given in Figure 2.9. Suppose that at time $t = 0$ the state of the system is $(0, 0, 0)$. The trajectory of transitions is then given in Table 2.2. When $t = 5$, the network is in the same state as on $t = 2$. We say that the network has reached a cycle.

Because all expression levels are 0 or 1, the number of states in the state space is finite, which yields that the number of states in a trajectory will be finite as well. This means that all initial states of a trajectory will eventually reach a steady state or a cycle.

The disadvantage of this network type is that the expression level of the components can take only two values so the component is considered to be either on or off. Also, transitions between the activation states of the components are assumed to occur synchronously. For real life applications, this assumption is not valid.

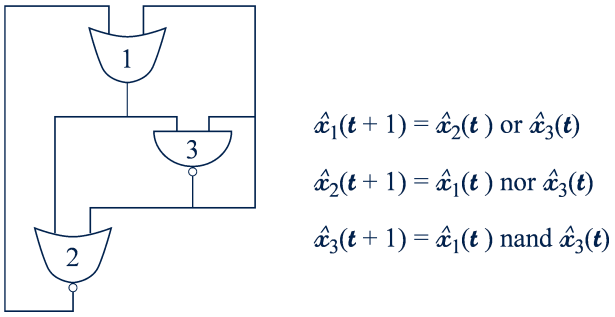

Figure 2.9: Example of a boolean network.

t	$x_1(t)$	$x_2(t)$	$x_3(t)$
0	0	0	0
1	0	1	1
2	1	0	1
3	1	0	0
4	0	0	1
5	1	0	1

Table 2.2: A trajectory of the boolean network

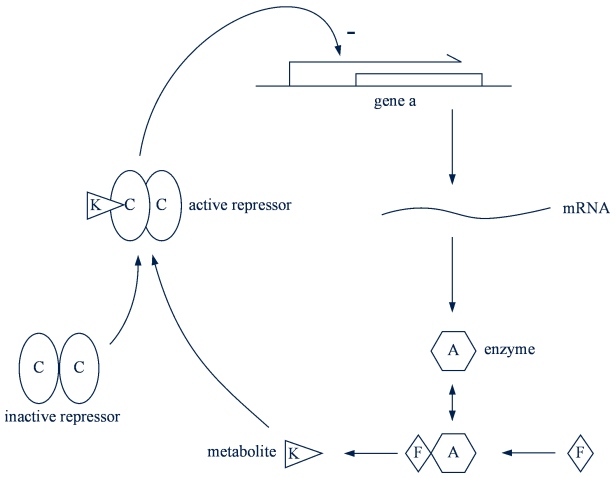

Figure 2.10: Kinetics of a regulatory network.

2.4.3 Ordinary differential equations

Ordinary differential equations (ODEs) are differential equations where the unknown function depends on a single independent variable. A general ODE has the form

$$\dot{x} = \frac{\partial x}{\partial t} = f(x). \tag{2.9}$$

Here, $x(t)$ is called the state-vector in our case, the activity of the unknown components at time t. The function $f(x)$ is called a rate equation and expresses the production rate of a component in the network as a function of the production rate of other components. If we define the kinetics of a component as the change in production rate and the production rate of component i as x_i, the kinetics of each component i in the network is given by the following ODE:

$$\dot{x_i} = \frac{\partial x_i}{\partial t} = f_i(x) \;\; (1 \leq i \leq N), \tag{2.10}$$

with $x = (x_1, \ldots, x_N)^T$ and f_i a nonlinear function. When the f_is are nonlinear, one speaks of nonlinear ordinary differential equations. If necessary, f_i can be extended such that it is not only a function of the production rates, but it also incorporates some external stimulus $u(t)$. ODEs have been widely used to analyze genetic regulatory systems, see [20, 43] for introductions. To illustrate the formalism of the rate equation, consider Figure 2.10. The gene expression level of gene a is repressed by the production rate of metabolite K (observe the negative feedback loop). If x_1 is the concentration level of gene

a, x_2 is the production rate of the enzyme (protein) A and when x_3 is used to denote the production rate of K, the following ODEs represent the kinetics of the network:

$$\begin{cases} \dot{x}_1 &= -\lambda_1 x_1 + r(x_3); \\ \dot{x}_2 &= -\lambda_2 x_2 + w_{21} x_1; \\ \dot{x}_3 &= -\lambda_3 x_3 + w_{32} x_2. \end{cases} \tag{2.11}$$

The factors λ_i are called the degradation constants or decay rates. These terms are used to state that the production rates of genes, proteins, and small molecules decrease by themselves in time. The terms w_{ij} are the production constants or influence rates from j on i. The function $r : \mathcal{R}^N_{\geq 0} \to [0, 1]$ is the so-called regulation function and is used to model positive and negative feedback loops. When there is a negative feedback loop between component j and i, component j represses the production rate of component i. The higher the production rate of j, the stronger the production rate of i is repressed. For the regulation function r, this means that it should be a function that decreases when x_j increases, i.e., formally, $\partial r/\partial x_j < 0$. Similarly, when the production rate of j activates the production rate of i (positive feedback loop), $\partial r/\partial x_j > 0$.

Ordinary differential equations are widely used in mathematics and engineering. One disadvantage, however, is that in most cases they can not be solved analytically.

Piecewise linear differential equations

Ordinary differential equations use differential regulation functions. However, in real-life applications, the components have a more switch-like behavior, i.e., they are activated or they are not activated. More precisely, observations in regulatory networks have led to the belief that the rate of activation of a gene, as a function of the concentration of a regulatory protein, often follows a steep sigmoidal curve. To simulate this behavior, piecewise-linear differential equations (PWLDEs) use step functions as regulation functions. These functions are defined as

$$\begin{cases} s^+(x_j, \theta_{ij}) &= \begin{cases} 0 \text{ if } \ x_j < \theta_{ij}; \\ 1 \text{ if } \ x_j > \theta_{ij}; \end{cases} \\ s^-(x_j, \theta_{ij}) &= 1 - s^+(x_j, \theta_{ij}). \end{cases} \tag{2.12}$$

In these formulas, θ_{ij} is called the threshold and defines when component i will be activated/repressed in function of the production rate x_j. Figure 2.11

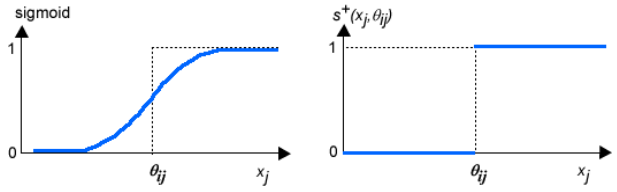

Figure 2.11: A steep sigmoid curve and s^+.

shows a steep sigmoidal curve and the positive step function s^+. The general form of a PWLDE is

$$\frac{\partial x_i}{\partial t} = -\lambda_i x_i + g_i(x) \ \ (1 \leq i \leq N). \tag{2.13}$$

As was the case with ODEs, the λ_is are the degradation constants [74, 73, 36]. The function g_i is defined as

$$g_i(x) = \sum_{l \in L} w_{il} b_{il}(x). \tag{2.14}$$

Here, L is the set with indices of the components that repress or activate component i, w_{il} represents the interaction coefficients (influence rates) and $b_{il}(x)$ is a piecewise-linear function composed of additions and products of step functions. As an example, consider the network consisting of three genes in Figure 2.12. Let x_1, x_2, x_3 be the gene expression levels of gene 1, 2, and 3, then the PWLDEs that model this network are:

$$\left\{ \begin{array}{rcl} \dot{x}_1 & = & -\lambda_1 x_1 + w_{12} s^+(x_2, \theta_{12}); \\ \dot{x}_2 & = & -\lambda_2 x_2 + w_{21} s^-(x_1, \theta_{21}) + w_{23} s^-(x_3, \theta_{23}); \\ \dot{x}_3 & = & -\lambda_3 x_3 + w_{31} s^-(x_1, \theta_{31}) + w_{33} s^-(x_3, \theta_{33}). \end{array} \right. \tag{2.15}$$

The mathematical advantage of PWLDEs over ODEs is that by substituting the nonlinear equations rates of the ODEs by piecewise-linear functions (approximation of the sigmoid functions by step functions), the solution of the system of equations can be found analytically.

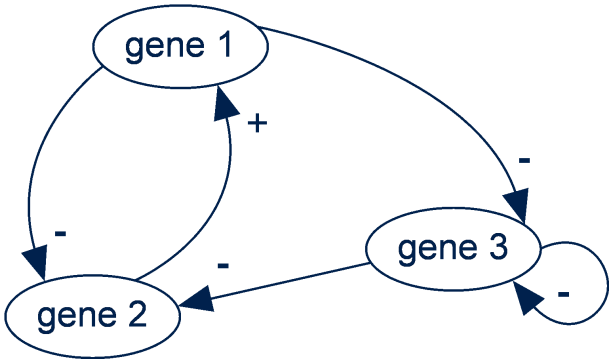

Figure 2.12: A regulatory system consisting of three genes.

3

Identification of sparse dynamic interaction networks

3.1 Requirements of the model

The success of the reconstruction model depends on different factors. Prerequisite is the way in which the dynamics of the interactions are modeled. We will focus on dynamical models, and not discuss static models where the relation between the unknown components are considered fixed in time. The dynamical models we study in this work are the Single Linear Model (Section 3.2.1) and the Piecewise Linear Model (Section 3.2.2). In many engineering applications, the number of observations M available for system identification (also known as reverse engineering) and model validation is usually much larger than the system order N, which represents the number of components. In our work, however, we want to deal with poor data, where $M \ll N$. This substantial lack of data can give rise to an identifiability problem, in which case a large subset of the model class is entirely consistent with the observed data and no unique model results. Since conventional techniques for system identification are not well suited for dealing with such situations, it becomes important to work around this by exploiting as much additional information as possible about the underlying system, in particular the relation between the number of observations and the number of components, the sparsity of the regulatory network and the influence of output noise (see Sections 3.2.1, 3.2.2, and Chapter 4).

3.2 Modeling of the identification problem

To be able to identify and reconstruct a sparse and dynamic regulatory network, we need to model the interactions between the unknown components. Therefore, they will be considered as reactions so that they can be represented as rate equations, i.e., as a set of ordinary differential equations introduced in Section 2.4.3:

$$\dot{x} = f(x, u|\theta) + \xi(t), \tag{3.1}$$

with

- $x(t)$ the state-vector that represents the activity of the N unknown components at time t;
- $u(t)$ the P controlled inputs to the system, with P the number of external stimuli;
- $\xi(t)$ the stochastic Gaussian white noise term with mean zero; and
- θ a representation of the coupling constants between the components activities.

Before we are able to identify the network, we have to model our problem. Therefore, we introduce and study some models to identify the network starting from Equation 3.1.

First, we have the Single Linear Model of Peeters and Westra [61]. In this work, the entire state-space is considered as one big space. Thus, the system's behavior is governed by its specific (un)stable equilibrium point, and the interactions are modeled by a linear dynamic state-space system.

In realistic situations, this model is too simple, however. As was pointed out by Øyehaug et al. [59], real-world systems tend to behave in a switch-like manner, and they determine the switching timepoints using complex modeling.

In contrast, we will determine the switching timepoints by identifying sparse piecewise linear systems. As a consequence, our focus is on modeling the subsystems between the switching points rather than on the dynamics of the switching points themselves, as in, e.g., Plahte et al. [62]. Thus, a Piecewise Linear Model is introduced. The entire state-space is being partitioned into subsystems, each described by a differential equation (see Figure 3.1). Each subsystem has an equilibrium which can be unstable or stable. If the

equilibrium is unstable, the state vector increases exponentially in time and the equilibrium is called a repeller. Otherwise, the equilibrium is stable and is called an attractor. Therefore, the behavior of the state vector can be described by motion through this collection of subsystems, swiftly moving through subsystems of repellers, until they enter the basin of attraction of an attractor. Under the effects of external agents (via the external input-vector) or by stochastic fluctuations, they can leave this subsystem and start wandering again, thereby repeating the process. Now, a vital assumption is that in each subsystem the behavior is governed by specific (un)stable equilibrium points, and, therefore, it is possible to make a linear approximation of Equation 3.1 for each subsystem.

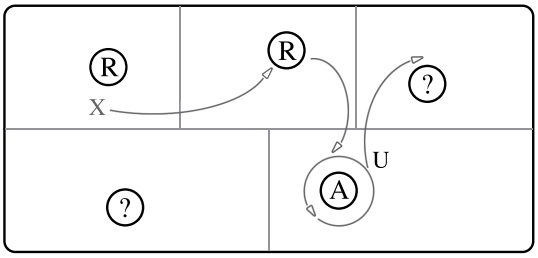

Figure 3.1: The dynamic system can switch over underlying stable states. R: repeller and A: attractor.

3.2.1 The Single Linear Model

We implicitly assume that f is a sufficiently smooth function. In that case, for sufficiently small deviations around the equilibrium value of x, x_{eq}, and for small u, we can approximate f in a Taylor expansion as

$$\dot{x}(t) \approx (x - x_{eq})\frac{\partial f}{\partial x} + u\frac{\partial f}{\partial u} \equiv Ax(t) + Bu(t) + c, \tag{3.2}$$

with A the interaction matrix $A \in \mathbb{R}^{N\times N}$, B the external input matrix $B \in \mathbb{R}^{N\times P}$, and c some small constant. Note that u is the input vector which is selected in such a way that it is small relative to the rate of change of x. In this setting, we use a first-order approximation.

In case of sparse genetic regulatory systems, there are tens of thousands of genes and hundreds of thousands—if not millions—of proteins. Besides the

fact that many of these genes and most of these proteins are currently unknown, and besides the randomizing effect of intrinsic and extrinsic noise, purely from the sheer magnitude of the data it would be completely impossible to implement realistic chemical-physical models and perform computational simulations. For this reason, the work in Peeters and Westra [61], Yeung et al. [85], and Eigen [25] is based on a straightforward and computationally manageable model. The dynamic interaction between gene/protein expression levels in a sparse regulatory network is modeled as a stochastic system of coupled linear differential equations:

$$\dot{x_i}(t) = \sum_{j=1}^{N} a_{ij} x_j(t) + \sum_{p=1}^{P} b_{ip} u_p(t) + \xi_i(t) - \lambda_i x_i(t), \tag{3.3}$$

with

- a_{ij} the influence rate from gene j on gene i;
- $x_j(t)$ the expression of gene j at time t;
- b_{ip} the external stimuli rate from input p on gene i;
- $u_p(t)$ the expression of external stimuli from input p at time t;
- $\xi_i(t)$ any stochastic uncertainty for gene i at time t; and
- λ_i the decay rate.

Such models can be studied in the broader and more flexible context of a state-space system:

- continuous time:

$$\begin{cases} \dot{x}(t) &= Ax(t) + Bu(t) + \xi(t); \\ y(t) &= Cx(t) + Du(t) + \zeta(t), \end{cases} \tag{3.4}$$

- discrete time:

$$\begin{cases} \dot{x}[k+1] &= Ax[k] + Bu[k] + \xi[k]; \\ y[k] &= Cx[k] + Du[k] + \zeta[k], \end{cases} \tag{3.5}$$

with

- $x(t_k) \equiv x[k]$, the state of the system at time t ($N \times 1$ entries);
- $u(t_k) \equiv u[k]$, the external stimuli at time t ($P \times 1$ entries); and
- $y(t_k) \equiv y[k]$, the output of the system at time t ($N \times 1$ entries).

Peeters and Westra's attention goes to the continuous-time, linear state-space model where $x(t) = (x_1(t), \ldots, x_n(t))^T$, $n = N$, $p = N$, and $(A, B, C, D) = (A, B, I_N, 0)$.

Since $C = I_N$ and $D = 0$, the output $y(t)$ is the same as $x(t)$. Furthermore, the state of the system contains the N gene expression levels. The equation of interest in the state-space model is therefore

$$\dot{x}(t) \quad = \quad Ax(t) + Bu(t), \tag{3.6}$$

ignoring the stochastic uncertainty term for now. We will come back on the importance of this term in the experimental Section and in Chapter 4.

The system matrices A and B need to be identified such that the output of the resulting model fits the available data from the microarray experiments well. The available data or observations taken at time instants $t_1, \ldots, t_M$ are given by $X = x[1], \ldots, x[M]$ and define the state of the system at time instant t (by definition of the state of the system). Furthermore, it is assumed that there is knowledge of the input signal $u(t)$ for $1 \leq t \leq M$ and accurate estimates of the state derivatives at these time instants can be computed. The way to estimate $\dot{X} = \dot{x}[1], \ldots, \dot{x}[M]$ is by interpolation of the state observations, which is a disadvantage of the model. The derivative $\dot{x}(t_k)$ can be approximated from the observations, such as $\dot{x}[k] \approx (x[k] - x[k-1])/(t_k - t_{k-1})$. Rewriting Equation 3.6 in matrix form yields

$$\dot{X} = AX + BU. \tag{3.7}$$

It is thus assumed that the matrices $\dot{X}$, X and U are available as empirical data. We will refer to this matrices collectively by $\mathcal{D}$. In this work, matrices in Equation 3.7 will be treated in a row-by-row fashion. We denote the unknown, sparse i-th row of A by the row-vector α_i, the unknown i-th row of B by the row-vector β_i and the known i-th row of $\dot{X}$ by the row-vector δ_i. Using the fact that $(AX)^T = X^T A^T$ and $(BU)^T = U^T B^T$ yields the following decoupled set of N linear systems of equations of size $M \times (N + P)$ (with M the number of observations, P the number of external inputs and N the number of unknowns):

$$\left[\begin{array}{cc} X^T & U^T \end{array} \right] \left[\begin{array}{c} \alpha_i^T \\ \beta_i^T \end{array} \right] = \delta_i^T \quad (1 \leq i \leq N). \tag{3.8}$$

This is a linear system of equations of the form

$$D\chi = R \tag{3.9}$$

with

- $D = \begin{pmatrix} X^T & U^T \end{pmatrix}$, $D \in \mathbb{R}^{M\times(N+P)}$;
- $R = \delta_i^T$, $R \in \mathbb{R}^M$; and
- $\chi = \begin{pmatrix} \alpha_i, & \beta_i \end{pmatrix}^T$, $\chi \in \mathbb{R}^{N+P}$.

Here, D and R are known and we want to find χ. This χ defines a column of the state-space matrices, A and B, and needs to be as sparse as possible as a consequence of the underlying biology. The question of computing a sparse solution to a consistent underdetermined linear system of equation $D\chi = R$ has received some attention [34, 35, 85].

In the literature [34, 35] and some of the references therein, results are given which state conditions under which optimal sparse solutions can be obtained by the technique of L_1-minimization. This technique can be used to find a sparse vector in the solution space S of $D\chi = R$. This method involves two steps. First, S is the set of vectors χ which minimize $\|D\chi - R\|_2$. Hence, S is the set of vectors χ which satisfy $D\chi = R_{proj}$, with R_{proj} the orthogonal projection of R on the columnspace of D. This is true because the shortest distance from vector R to the columnspace of D is of course orthogonal and $D\chi$ is a vector in this columnspace. So, S can be computed by singular value decomposition where $D = Q_1 \sum Q_2^T$. Second, it is well known that this problem can be reformulated as an LP problem. Thus, the difficult combinatorial problem of finding a vector in S having as many values equal to zero as possible is avoided and replaced by finding a vector $\chi \in S$ for which $\|\chi\|_1$ is minimal.

However, in this case, only the first N elements of χ have to be sparse, because the β_i part of χ need not be sparse a priori. Therefore, the matrix $C = [I_N \;\; 0]$ is introduced. Searching for the vector $\chi \in S$ for which $\|C\chi\|_1$ is minimal gives us the best sparse α_i part of χ and can be formulated as

$$\begin{aligned} \min_{\chi\in\mathbb{R}^{(N+P)}} \quad & \|C\chi\|_1 \\ \text{s.t.} \quad & D\chi = R_{proj}. \end{aligned}$$

This technique is called *partial L_1-minimization* and can be rewritten in the form:

$$\begin{aligned} \min_{A\in\mathbb{R}^{(N\times N)}} \quad & \|A\|_1 \\ \text{s.t.} \quad & AX + BU = \dot{X}. \end{aligned}$$

L_1-minimization versus L_0-minimization and L_2-minimization

In the target function, $\|A\|_1$ denotes the 1-norm of the system matrix A, i.e., $\|A\|_1 = \sum_{i,j} |A_{i,j}|$. This choice is motivated by the sparsity constraint on the networks to be identified. If the interaction matrix is fitted correctly, it will be sparse. Hence, we prefer solutions with as few non-zero components as possible.

The minimization of the 0-norm, L_0-minimization, can be used also to get the optimal sparse solution. $\|A\|_0 = \sum_{i,j} |A_{i,j}|^0$, with $0^0 = 0$, thus, the 0-norm denotes the number of non-zero components. But this 0-norm can only be computed by explicit enumeration, it is unsuitable in practice due to the ensuing combinatorial explosion.

J. J. Fuchs [34, 35] has described conditions under which the 1-norm is an acceptable approximation for the 0-norm. These conditions are optimality, separability and sparsity. Though these conditions do not strictly apply here, we find that this 1-norm succeeds in numerical simulations. For more details about this technique, see [61] and [84].

Furthermore, the 1-norm is also an acceptable approximation for the 2-norm, $\|A\|_2 = \sum_{i,j} |A_{i,j}|^2$. The technique of L_1-minimization will automatically set many entries of the solution to zero, whereas L_2-minimization would spread out the error over all components, thus creating many small components.

Numerical experiments

To investigate the performance of the partial L_1-minimization approach proposed in this section, several numerical experiments have been conducted to address the following research questions:

- Is it possible to identify the system matrices, A and B, based a relatively small number of available observations, such that $M \ll N$?
- How does the quality of the identification depend on the number of unknown components, $N(N + P)$?
- Is the algorithm robust against measurement/external noise, ξ?
- Does the quality of the identification depend on the sparsity of the system?

In line with the definitions above, we use the parameters N, M, and P to quantify the size and complexity of the input. In addition, the sparsity of a

solution vector (a row of the interaction matrix) is measured by the number of nonzero entries and denoted by k (which should be much less than N). To complete the system's data set, some stochastic Gaussian white noise is added to the input data set. It is normally distributed with zero mean and some standard deviation σ that determines the noise level. To quantify the quality of the resulting approximation A_{est} of the original A^*, two performance measures are introduced: the system error, N_{e}, and the CPU-time, T_c.

The minimal number of observations For a certain number of unknown components, a sufficient number of measurements has to be available. Therefore, the minimal number of measurements required for a certain number of unknowns, denoted by M_{min}, has been determined. This is the number of measurements so that the total system error N_{e} is acceptably small, $N_{\text{e}} < 10^{-5}$. Figure 3.2 represents the values for M_{min} as a function of the number of unknowns. For comparability reasons, the number of unknown components in

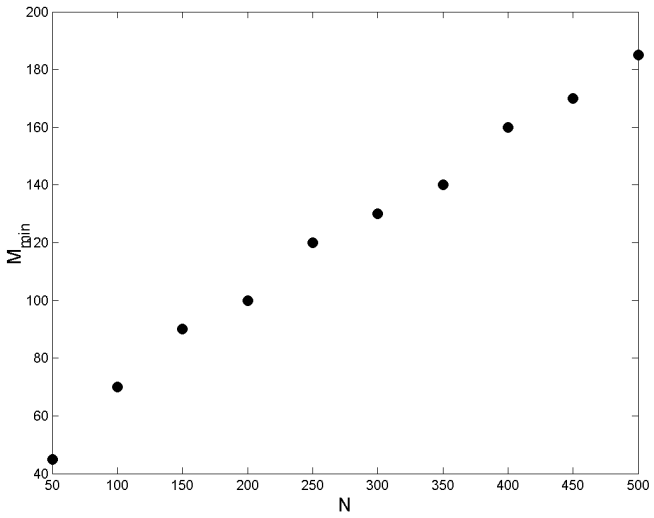

Figure 3.2: Minimal number of required measurements M_{min} as a function of the number of unknowns $N + P$ with $P = 1$ and a sparsity $k = 1$.

the following experiments has been fixed to $N = 150$ and the number of external inputs to $P = 1$. Consequently, the associated minimal number of measurements has been fixed to $M_{\text{min}} = 90$ (see Figure 3.2).

Noise The noise level σ is an other factor where the system error N_e depends on. To test the system's robustness w.r.t. measurement noise, some stochastic Gaussian white noise ξ with mean zero and some standard deviation σ will be added to the output data $\dot{X}$. This standard deviation or noise level lies in the range of $0 \leq \sigma \leq 1$. Figure 3.3 shows how this noise level influences the error rate in our approach. For low noise levels, $\sigma < 0.065$, the system is robust, $N_e < 1$. But, as to be expected, for bigger noise levels, the error increases if the noise level increases, and vice versa.

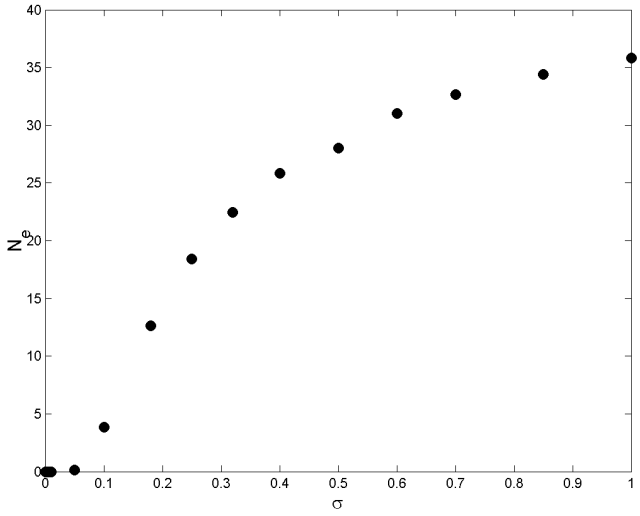

Figure 3.3: The system error N_e as a function of the noise level σ, with $N = 150$, $P = 1$, $M = M_{\min}$, and $k = 1$.

Sparsity A basic assumption in the approach is the sparsity of the underlying coupling matrix, represented by the number of non-zero entries per row, k. If k rises above a certain threshold, the performance of the approach is abruptly and severely affected (see Figure 3.4).

For relatively moderate noise levels and a high degree of sparsity—i.e., a small number k of non-zero elements in the rows of matrix A^*—the approach allows one to reconstruct a sparse matrix with great accuracy from a relative small number of observations $M \ll N$. For example, A^* with rows of 150 components of which all but 3 are equal to zero, can be efficiently reconstructed

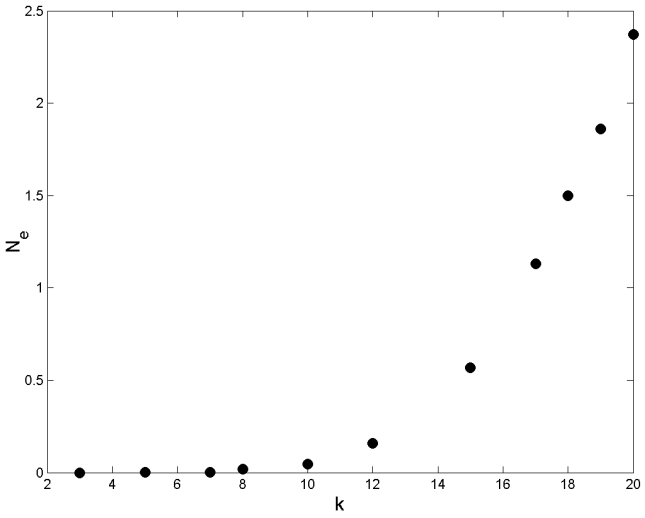

Figure 3.4: Number of errors N_e as a function of the number of non-zero elements per row k with $N = 150$, $P = 1$, and $M = M_\mathrm{min}$.

from just 90 independent measurements (see Figure 3.5). Figure 3.5 shows an initial increase, followed by a decrease. Finally, above a certain threshold value for M, the error N_e is zero. To explain this phenomenon, remember that the system error N_e is the sum of the false positives, the false negatives, and the correlation error in the interaction matrix. The false positives correspond to the non-zero values in the matrix A_est that should be zero, vice-versa for the false negatives, and the correlation error correspond to the non-zeros of A_est and A^* that have an opposite signs.

Turning back to Figure 3.5, the initial increase is caused by false positives. Indeed, as long as $M < M'_\mathrm{min}$, where M'_min is the minimal number of required measurements *in the case of a single row*, $k \approx M'_\mathrm{min}$. As soon as M reaches M'_min, the system becomes completely determined, whence k drops to its proper value. Observe that $M'_\mathrm{min} < M_\mathrm{min}$ due to the absence of effects related to the composition of rows, for more details see Section 4.1. Notice that the false negatives decrease monotonously over the entire range of M.

CPU-time Figure 3.6 shows the CPU time, T_c, used by the algorithm as a function of the number of unknowns N. This function is exponential. It is

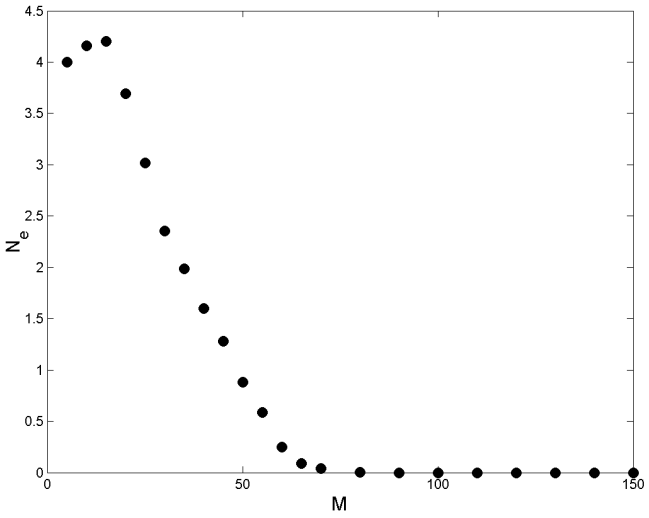

Figure 3.5: The system error N_{e} as a function of the number of measurements M, with $N = 150$, $P = 1$ and $k = 1$.

nearly flat for small values of N, but increases fast as a function of N. From this illustration we can conclude that the Single Linear State Space Model is very fast.

3.2.2 The Piecewise Linear Model

Now, our aim is to obtain the local interaction matrices for each subsystem A_ℓ, that all directly relate to the regulatory network. Again, the matrices B_ℓ provide information on the coupling of unknown components to specific inputs. Now, a vital assumption is that in each subsystem the behavior is governed by its specific (un)stable equilibrium point. In that case, it is possible to approximate the dynamics of Equation 3.1 in cell ℓ—for x near the ℓ-th equilibrium $x_{\mathrm{eq}}^{(\ell)}$ and small u—(except the noise term) as

$$\dot{x}(t) \approx \frac{\partial f(x_{eq}^{(\ell)}, u)}{\partial x}(x - x_{eq}^{(\ell)}) + \frac{\partial f(x_{eq}^{(\ell)}, u)}{\partial u}u \equiv A_\ell x(t) + B_\ell u(t) + c_\ell. \quad (3.10)$$

Thus, the qualitative behavioral dynamics of the interactions between the unknown components is characterized as predominantly linear behavior near the stable equilibria—called the steady states, interrupted by abrupt transitions

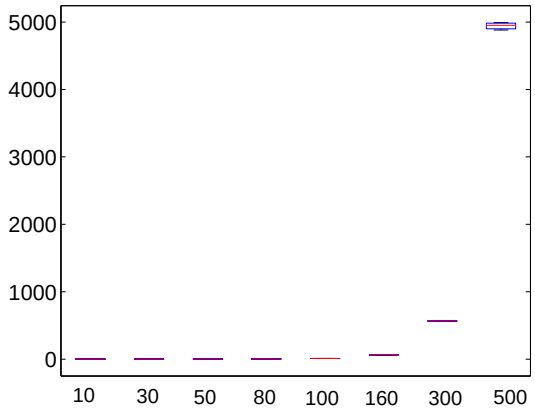

Figure 3.6: The CPU time, T_c, used by the reconstruction approach (in seconds) as a function of the number of unknowns N.

where the system quickly relaxes to a new steady state, either externally induced or by process noise.

General dynamics of switching subsystems

In what follows, let us assume a dynamical input-output system Σ that switches irregularly between K linear time-invariant subsystems $\{\Sigma_1, \Sigma_2, \ldots, \Sigma_K\}$.

Let $S = \{s_1, s_2, \ldots, s_{K-1}\}$ denote the set of—unknown—switching times, i.e., the time instants $t = s_\ell$ when the system switches from subsystem Σ_ℓ, to $\Sigma_{\ell+1}$. Similarly as with the single linear networks, we assume empirical data $X = (x[1], \ldots, x[M])$, $U = (u[1], \ldots, u[M])$, and $\dot{X} = (\dot{x}[1], \ldots, \dot{x}[M])$ at M sampling times $T = \{t_1, t_2, \ldots, t_M\}$, representing full observations of the N states and P inputs, and $x[k] \equiv x(t_k)$. The interval between two sample instants is denoted as $\tau_k = t_{k+1} - t_k$. Here, we assume that the system is sampled on regular time intervals, i.e., that the sample intervals are equal to τ. Within one subsystem Σ_ℓ, the effect of the inputs $u(t)$ is represented as a state-space system of first-order differential (for continuous time systems) or difference equations (for discrete time systems), using an internal vector $x(t)$ spanning the so-called subspace:

- continuous time:

$$\begin{cases} \dot{x}(t) &= A_\ell x(t) + B_\ell u(t) + \xi(t); \\ y(t) &= C_\ell x(t) + D_\ell u(t) + \zeta(t), \end{cases} \tag{3.11}$$

- discrete time:

$$\begin{cases} \dot{x}[k+1] &= A_\ell x[k] + B_\ell u[k] + \xi(t); \\ y[k] &= C_\ell x[k] + D_\ell u[k] + \zeta(t), \end{cases} \tag{3.12}$$

with $x[k] \equiv x(t_k)$.

In the context of piecewise linear systems of gene-protein regulatory systems, the dynamics is slightly different to the case of single linear systems [61]. In our context, we assume that we observe all N genes, and that $C_\ell = I_N$ with I_N the $N \times N$ identity matrix and $D_\ell = 0$ for all ℓ. In the case of continuous time and in the absence of noise, this system can be written as

$$\dot{x}(t) = A_\ell x(t) + \tilde{B}_\ell \tilde{u}(t), \tag{3.13}$$

with $\tilde{B}_\ell = \left(B_\ell | - A_\ell x_{eq}^{(\ell)} \right)$, $\tilde{u}^T = (u^T,\ 1)$, where $x_{eq}^{(\ell)}$ indicates the equilibrium point of the ℓ-th subsystem and A_ℓ and B_ℓ refer to Equation 3.10. We will use this linear expression, and from here on drop the *tilde*. A general disadvantage is that the time evolution of the different genes/proteins, i.e., $x_\nu(t)$, $\nu = 1, \ldots, n$, will strongly correlate, thus obscuring their true relation. This can be avoided by using Equation 3.13 with time series of triplets $\mu[k] \equiv (x[k], u[k], \dot{x}[k])$ with a sufficient amount of statistically independent and varying inputs $u(t_k)$. Practically, this opens the way to combining distinct empirical sets. However, a practical disadvantage of Equation 3.13 is that the derivative $\dot{x}(t_k)$ can only be approximated from the measurements, such as $\dot{x}[k] \approx (x[k] - x[k-1])/(t_k - t_{k-1})$.

We furthermore assume that the system matrices in these equations are constant during intervals $[s_\ell, s_{\ell+1}[$, and abruptly change at the transition between the intervals at $t = s_{\ell+1}$. We assume that on the time scale τ, the system has relaxed to its new state. This means that we do not observe *mixed states*, which would severely complicate the problem of identification, e.g., see [79]. This is accomplished by defining *weights* $W_{k,\ell}$ as the degree to which observation k belongs to subsystem Σ_ℓ. If observation $\mu[k]$ belongs to system Σ_ℓ, then $W_{k\ell} = 1$. Non-integer values in [0,1] can be interpreted as the fuzzy

membership of observation k to system Σ_ℓ. Since we assume that the subsystems $\{\Sigma_1, \Sigma_2, .., \Sigma_K\}$ act disjointly and subsequently, the result can be improved by matching the weights to a block function structure; i.e., $W_{kl} = 1$ for $t_k \in [s_l, s_{l+1}[$ and $W_{kl} = 0$ elsewhere. This may, however, not take into account that the same subsystem can be revisited at different switching intervals. To solve this, some postprocessing step can be done. These considerations lead to the following constraints $\mathcal{C}_{MK}$ on W:

$$\mathcal{C}_{MK}(W) : \begin{cases} W_{1,1} = 1; \\ W_{M,K} = 1; \\ \forall_{k,\ell} \;\; W_{k,\ell} \in [0,1]; \\ \forall_k \sum_\ell W_{k,\ell} = 1; \\ \sum_{k=1}^{M-1} |W_{k+1,1} - W_{k,1}| = 1; \\ \forall_{\ell=2..(K-1)} \sum_{k=1}^{M-1} |W_{k+1,\ell} - W_{k,\ell}| = 2; \\ \sum_{k=1}^{M-1} |W_{k+1,K} - W_{k,K}| = 1. \end{cases} \tag{3.14}$$

Combining the system matrices $\{A, B\}$ with the subsystem weight-matrix W

The assumption that the switching times between the linear subsystems are completely known suits various experimental conditions, as, for instance, when toxic agents are administered. In many biological situations, however, the exact timing between subsystems is *not* known, as during embryonic growth and in many metabolical processes.

When a sufficiently accurate record of estimates of the state derivatives $\dot{X}$ is available, we can simply rewrite this problem as a special case of the method described in the case of a single linear problem [61]. In fact, by exploiting the data $\mathcal{D} = \{X, U, \dot{X}\}$, the problem can be stated as a linear equation in terms of new matrices H_1 and H_2 as

$$\dot{X} = H_1 X + H_2 U. \tag{3.15}$$

In this equation, the matrices H_1 and H_2 relate to the—unknown—system matrices $\{A_1, B_1, \ldots, A_K, B_K\}$ and ditto unknown weights $\{W_{kl}\}$ as

$$\begin{aligned} \mathtt{vec}(H_1) &= W_{kr} \cdot \mathtt{vec}(A); & (3.16) \\ \mathtt{vec}(H_2) &= W_{kr} \cdot \mathtt{vec}(B), & (3.17) \end{aligned}$$

with $\mathtt{vec}(Y)$ the vector of Y, if $Y \in \mathbb{R}^{N\times N}$ then $\mathtt{vec}(Y) \in \mathbb{R}^{N^2\times 1}$. The matrices A, B, and W_{kr} are composed as follows:

$$A = \begin{pmatrix} A_1 \\ \vdots \\ A_K \end{pmatrix}; \quad B = \begin{pmatrix} B_1 \\ \vdots \\ B_K \end{pmatrix}; \tag{3.18}$$

$$W_{kr} = W \otimes I_{N^2} = \begin{pmatrix} W_{1,1}I_{N^2} & \cdots & W_{1,K}I_{N^2} \\ \vdots & \vdots & \vdots \\ W_{M,1}I_{N^2} & \cdots & W_{M,K}I_{N^2} \end{pmatrix},$$

where $\otimes$ is the Kronecker-product and I_{N^2} is the $N^2 \times N^2$ identity matrix. Note that Equation 3.15 is no longer a linear problem, as the unknown matrices A, B, and W_{kr} appear in a non-linear way in the equation. This equation is exactly of the type of single linear networks as in Section 3.2.1 and [61]. Therefore, its solution method is fully applicable, so that an efficient and accurate algorithm is available for solving this problem in terms of H_1 and H_2. However, the problem has now shifted to solving two additional non-linear equations:

$$W_{kr} \diamond A = H_1; \tag{3.19}$$

$$W_{kr} \diamond B = H_2, \tag{3.20}$$

where A, B, and W_{kr} have to be solved from the known—i.e., computed—matrices H_1 and H_2. We use the operation $\diamond$ to represent the relations in Equation 3.16 and 3.17 more concisely.

Equations 3.16 and 3.17 are an underdetermined set of equations that can only be solved by additional information, such as assuming sparsity for A, and a block structure for W_{kr}, as defined in Equation 3.14.

Identification of PWL models with *unknown* switching and *regular* sampling from *poor* empirical data

We will now focus on the general case, that the data X, their derivatives $\dot{X}$, and the external inputs U are available as empirical data $\mathcal{D}$. In this case, the objective of system identification is to compute concurrently the system parameters A, B, *and* weights W_{kr} (and hence the switching times S). Equation 3.15 provides us with the general state-space equations for a PWL system.

In practical experimental conditions, white process and measurement noise adds to the right-hand side. The fit between the empirical data and the system model can be quantified by the weighted difference between observed and expected expression profiles expressed as a linear L_p-criterion:

$$\mathcal{E}_{sys}(A, B, W|\mathcal{D}) = \sum_{k,l} W_{kl} \|A_l x[k] + B_l u[k] - \dot{x}[k]\|_p. \tag{3.21}$$

The criterion furthermore involves the relation between the k-th observation and the ℓ-th subsystem Σ_ℓ; namely the *weight* $W_{k\ell}$ and the *distance* $d_{k\ell}$ between observed and the expected value of observation k relative to subsystem model Σ_ℓ.

In order to handle the underdetermined character of the problem, we furthermore exploit the property of sparsity and allow for hierarchy of the interaction networks. Namely, the interaction matrix is row-sparse, but the columns need not be sparse. In the case of a hierarchical component, the associate column will contain many non-zero components. Under some conditions, this problem is equivalent to minimization of the L_1-norm of the rows of A_ℓ and B_ℓ as argued by J. J. Fuchs [35]. This implies a global minimization such as

$$\mathcal{E}_{sparse}(A|\mathcal{D}) = \sum_{\ell} \|\mathtt{vec}(A_\ell^T)\|_1 \equiv \|A\|_1, \tag{3.22}$$

under the constraints that $\{X, U, \dot{X}\}$ satisfy Equation 3.15.

The problem of estimating the system parameters can thus formally be defined as the search for the vectors A_{est}, B_{est}, and W_{est} that globally minimize $\mathcal{E}$. This can be formulated as a quadratic programming problem, as follows:

QP: given the data $\mathcal{D}$, compute the system matrices A, B and the weight matrix W:

$$\begin{array}{l} (A_{\text{est}}, B_{\text{est}}, W_{\text{est}}) = \arg\min_{(A,B)\in\mathbb{R}^{N(P+N)}, W\in\mathbb{R}^{KM}} \mathcal{E}(A, B, W|\mathcal{D}) \\ \text{subject to:} \\ \quad \left\{ \begin{array}{l} \mathcal{E}(A, B|\mathcal{D}) = \lambda_1 \mathcal{E}_{sys}(A, B, W|\mathcal{D}) + \lambda_2 \mathcal{E}_{sparse}(A|\mathcal{D}) + \lambda_3 \mathcal{E}_{sparse}(B|\mathcal{D}); \\ \mathcal{C}_{MK}(W), \end{array} \right. \end{array} \tag{3.23}$$

for selected λ's with: $\lambda_1 + \lambda_2 + \lambda_3 = 1$, and the constraints $\mathcal{C}_{MK}(W)$ in Equation 3.14.

This is a regularized (or scalarized) convex quadratic optimization problem that is not well posed, because it has a nonsingular Jacobian at the optimum, and becomes ill-conditioned as the approach iterates optimality. Instead of this quadratic programming problem, we will therefore study the following two coupled linear programming problems associated to the original QP:

LP1: given the weight matrix $\tilde{W}$ compute the system matrices $(A_{\text{est}}, B_{\text{est}})$:

$$\begin{aligned}&(A_{\text{est}}, B_{\text{est}}) = \arg\min_{(A,B)\in\mathbb{R}^{N(P+N)}} \mathcal{E}(A, B, \tilde{W}|\mathcal{D})\\ &\text{subject to:}\\ &\quad \mathcal{E}(A, B|\mathcal{D}) = \lambda_1\mathcal{E}_{sys}(A, B, W|\mathcal{D}) + \lambda_2\mathcal{E}_{sparse}(A|\mathcal{D}) + \lambda_3\mathcal{E}_{sparse}(B|\mathcal{D});\end{aligned} \tag{3.24}$$

LP2: given system matrices $(\tilde{A}_\ell, \tilde{B}_\ell)$ apply the L_1-norm $\tilde{d}_{kl} = \|\dot{x}[k] - \tilde{A}_\ell x[k] - \tilde{B}_\ell u[k]\|_1$ to compute the weight matrix W_{est}:

$$\begin{aligned}&W_{\text{est}} = \arg\min_{W\in\mathbb{R}^{KM}} \mathcal{E}_{sys}(\tilde{A}, \tilde{B}, W|\mathcal{D}) = \textstyle\sum_{k=1}^{M-1}\sum_{l=1}^{K} \tilde{d}_{kl}W_{kl}\\ &\text{subject to:}\\ &\quad \mathcal{C}_{MK}(W).\end{aligned} \tag{3.25}$$

The algorithm to estimate the system parameters $\{A, B\}$ and W consists of iteratively solving the two optimizations LP1 and LP2 subsequently, until the criterion has sufficiently converged.

Construction and control of the subsystem weightmatrix

For small values of the regularization terms in $\mathcal{E}$ in LP1 (Equation 3.24), i.e., $\lambda_2, \lambda_3 \ll \lambda_1$, and a simultaneous, extreme underdetermined system, i.e., $\#\Sigma_\ell \ll N$, the tandem {LP1,LP2} proposed above, runs into problems. The problem amounts to the degree of freedom that formulation LP1 offers to match empirical data $\mathcal{D}$ with system $\Sigma = (A, B)$ in order to minimize the distance to the model space $d(\mathcal{D}, \Sigma)$. It is well known that at least $M_{\min} \propto \log(N)$ measurements are required for a good reconstruction of sparse matrices A and B; see for instance [34, 35, 85]. Therefore, when $\#\Sigma_\ell \ll M_{\min}$, the heavily underdetermined system has a high degree of freedom to match the data with the model. This will cause the tandem {LP1,LP2} to halt as the criterion $d(\mathcal{D}, \Sigma) \approx 0$ has been reached.

Avoiding this problem requires (i) the restriction of the maximum number of subsystems to $K < M/\log(N)$, and (ii) the careful control of the weight matrix w during the iteration, such that each subsystem Σ_ℓ has at least $M_{\min}$ elements, i.e., $\#\Sigma_\ell \geq M_{\min}$. For this reason, the following iteration is performed for initializing the weight matrix. (For an example, see Figure 3.7.):

1. Assign the *current measurement* k to 1, and the *current system* ℓ to 1. Initialize W to the $M \times K$ null matrix: $W = 0$.

2. The first $M_{\min}$ measurements are assigned to the current—i.e., first—subsystem: $W_{11} = 1, \ldots, W_{M_{\min},1} = 1$. Now the current measurement k is set to $M_{\min} + 1$.

3. The current measurement, $\mu_k = (x[k], u[k], \dot{x}[k])$, belongs to the current subsystem Σ_ℓ if $d(\mu_k, \Sigma_\jmath)$ is minimized by $\jmath = \ell$. In that case: (i) it is assigned to the current system by setting $W_{k\ell} = 1$, and (ii) the next measurement is considered, i.e., k is increased, and step 3 is repeated.

4. If another system $\Sigma_\jmath$ is closer to μ_k, then this system is assigned to the current system: $\ell = \jmath$, and measurement k is considered as the first of $M_{\min}$ measurements assigned directly to this subsystem, i.e., $W_{k\ell} = 1, \ldots, W_{k+M_{\min}-1,\ell} = 1$, k is set to $k + M_{\min}$, and step 3 is repeated.

This iteration process is continued as long as there are unassigned measurements. When the final subsystem has less then $M_{\min}$ elements, these are discarded. Finally, all measurements will belong to some subsystem, while W obeys all constraints defined in Equation 3.14. One of the advantages of this *matching* algorithm is that it requires no advance knowledge of the number of subsystems.

A tandem algorithm for network reconstruction using the subsystem weight matrix

The procedure for constructing and managing the subsystem weight matrix w, defined earlier in this section, allows for an efficient tandem approach to solving the identification problem.

The non-linear problem $\dot{X} = H_1 X + H_2 U$, defined in Equation 3.15, can be solved in terms of H_1 and H_2, but not in terms of A, B, and W_{kr}, because it is a bilinear problem in terms of A and B for fixed W_{kr}. For these reasons, we again split the problem and follow a tandem approach as discussed in Section 3.2.2.

However, in the present tandem, the construction of the subsystem weight matrix W is performed by the matching approach defined above, rather than by the LP2 defined in Equation 3.25. Both amount to a solution obeying the weight constraints in Equation 3.14, but the matching algorithm will prevent too underdetermined systems that will prematurely halt the iteration as they

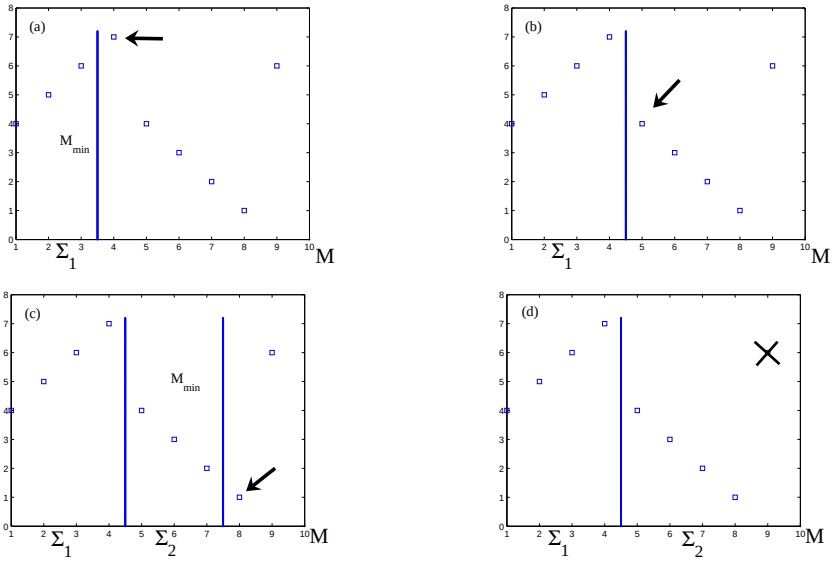

Figure 3.7: Construction of the weight matrix. (a) The first M_{min} measurements are assigned to subsystem Σ_1 and the current measurement $M_{min}+1$ will be dealt with (as in step 3). (b) The next measurement $k = M_{min}+2$ is considered and for this one it is concluded that an other subsystem Σ_2 is closer. (c) The next M_{min} number of measurements is added to this new subsystem and step 3 will be repeated until all measurements belong to a subsystem. In the final figure all measurements—except for one—belong to some subsystem.

generate a fictitious match with the model. The computation of the system matrices (A, B) is again performed by the robust L_1 identification in LP1, with $\lambda_1 = 0$, and $\lambda_2 = \lambda_3$. The tandem is controlled by the distance between the data and the model: $d(\mathcal{D}, \Sigma)) = \mathcal{E}_{sys}(A, B, W_{kr}|\mathcal{D})$, as defined in Equation 3.21. If this quantity has converged below a pre-specified threshold, the iteration is terminated.

Numerical experiments

With our experiments, we want to address the following research questions:

1. Is it possible to identify the system matrices, A and B, based a relatively small number of available observations, such that $M \ll N$?

2. How does the quality of the identification depend on the number of unknowns?

3. Is the algorithm robust against measurement/external noise?

4. Does the quality of the identification depend on the sparsity of the system?

5. How does the number of observations depend on the number of subsystems?

The numerical experiments consist of the comparison of the reconstructed network with the—known—original network structure, and they clearly reveal the range where the approach is effective.

In line with the definitions above, we use the parameters N, M, P and K to quantify the size and complexity of the input. In addition, sparsity of the local interaction matrix A is measured by the number of non-zero entries per row and denoted by k (which should be much smaller than N). To complete the system's data set, some stochastic Gaussian white noise is added to the input data set. It is normally distributed with zero mean and some standard deviation σ that determines the noise level. To quantify the quality of the resulting approximation A_{est} of A^*, two performance measures are introduced: the system error, N_{e}, and the CPU-time, T_c.

Because the computation of the system matrices A and B is performed again by the robust L_1 identification of the LP_1, for questions 1 till 4, we can refer to the experimental results of the Single Linear State Space Model of Section 3.2.1.

For a certain number of genes, a sufficient number of measurements has to be available for each subsystem. In Section 3.2.1, the minimal number of measurements required for a certain number of genes, denoted by $M_{\min}$, has been determined. This is the number of measurements so that the total system error, N_{e}, is acceptably small (see Figure 3.2). This required minimal number of measurements is taken into account for the performance of the next experiments.

Multiple subsystems Some experiments concerning multiple subsystems were performed. Figure 3.8 shows the accuracy of the partitioning of the available measurements into different subsystems. The error measure δ shown in Figure 3.8 is defined as the cumulative distance in terms of time stamps between erroneously classified measurements and the switching point of the class they belong to, relative to the total number of measurements. In the experiment illustrated by Figure 3.8, two subsystems were identified.

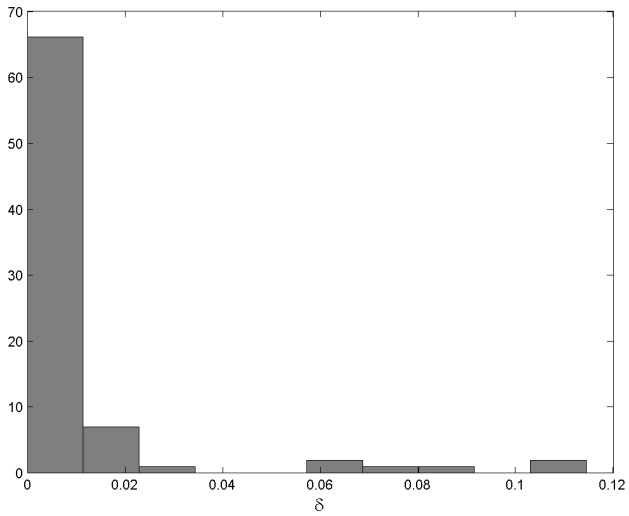

Figure 3.8: The distribution of the error measure δ for partitioning $M = 200$ measurements into subsystems, with $N = 150$. Two subsystems were identified.

CPU-time From Section 3.2.1, we know that the LP_1, the identification of the system matrices A and B, is very fast. Now for the multiple subsystems, we additionally need to compute the weight matrix and this will be repeated until $d(\mathcal{D}, \Sigma))$ is acceptable. The next experiment represents the CPU-time, T_c for two subsystems and shows that, even for the tandem-process, the performance is rather fast.

3.3 Conclusions

Starting from the general ordinary differential equation, we presented two methods to model and identify sparse dynamic regulatory networks from poor data. First, we have the Single Linear Model where the entire state-space is considered as one big space and the interactions are modeled by a linear dynamic state-space system. To get the most sparse solution, linear programming is used. Second, a Piecewise Linear Model is introduced. The entire state-space is partitioned into subsystems, where attractors and repellers reign. Going into detail learned us that each subsystem can be considered as a Single Linear System. Both, the Single Linear Model and the Piecewise Linear Model turned out to be very fast.

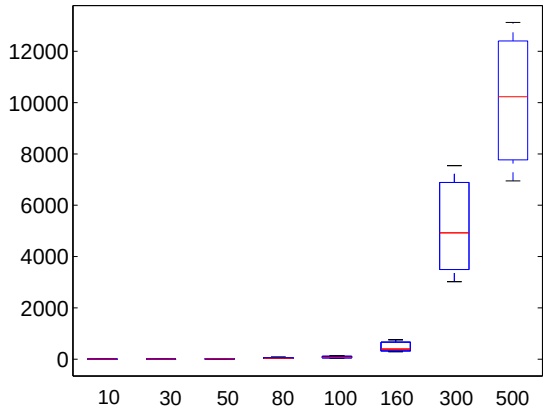

Figure 3.9: The CPU time, T_c, used by the reconstruction algorithm (in seconds) as a function of the number of unknowns N for an approach with two subsystems, $K = 2$.

We can conclude that both the Single Linear and the Piecewise Linear Models resulted in an efficient and fast algorithm that are able to accurately estimate a sparse dynamic interaction matrix from poor data. Unfortunately, both approaches are not robust with respect to noise.

4

Network identification as a learning problem

4.1 The Machine Learning Approach

In this Chapter, we generalize our problem. We want to learn a model that fits the underlying system, capable of generalization. This corresponds to the student-teacher setting in machine learning. Given a training set generated by a teacher, the algorithm should return a network, refered to as the student, that fits the teacher. Furthermore, the student should also perform well on a validation set, input that was not used by the algorithm. In this setting, it is natural to link network identification to feature selection. Only very few components influence the expression level of any given component, so one can restate the problem as selecting exactly those few among the large amount of components under consideration. Hence, the results presented here will not only be applicable to network identification, but also to feature selection.

To translate the problem of network identification formally into machine learning terminology, we assume that a *training set* of M input/output pairs $\chi_{\text{tr}} = \{(x_m, u_m, \dot{x}_m) \mid 1 \leq m \leq M\}$ is given, where $x_m, \dot{x}_m \in \mathbb{R}^N$ and $u_m \in \mathbb{R}^P$. The components of the input vectors x_m are independently and identically distributed so that the elements of the training set are linearly independent. Since the data is assumed to be generated by some interaction

network, this network will be denoted by $T = (A^*, B^*)$ where $A^* \in \mathbb{R}^{N\times N}$ and $B^* \in \mathbb{R}^{N\times P}$. In this context, we refer to T as the unknown *teacher*. For each $(x_m, u_m, \dot{x}_m) \in \chi_{\text{tr}}$, $\dot{x}_m = A^* x_m + B^* u_m$, i.e., $\dot{x}_m$ is the output produced by the teacher T on input x_m and some external stimuli u_m. In general, the teacher's output $\dot{X}^* \in \mathbb{R}^{N\times M}$ on some input $X \in \mathbb{R}^{N\times M}$ and $U \in \mathbb{R}^{P\times M}$ is computed as follows:

$$\dot{X}^* = T(X) \equiv A^* X + B^* U. \tag{4.1}$$

Moreover, we consider sparse networks. Therefore, we assume there exists $k^* \ll N$ such that, for each row of the matrix A^*, k^* components are non-zero. Since the matrix A^* models the interactions in the network, a non-zero value of $A^*_{i,j}$ indicates that component j of input X influences component i of the output $\dot{X}^*$. So, the sparsity constraint implies that each component of the output is determined by k^* components of the input.

4.1.1 The learning algorithm

The learning algorithm should return a network $S = (A_{\text{est}}, B_{\text{est}})$, referred to as the *student*, with $A_{\text{est}} \in \mathbb{R}^{N\times N}$ and $B_{\text{est}} \in \mathbb{R}^{N\times P}$, that reproduces the training set χ_{tr}: $\dot{x}_m = A_{\text{est}} x_m + B_{\text{est}} u_m$ for $m = 1, \ldots, M$. However, the student should also perform well on input that was not used by the algorithm, i.e., the algorithm should be able to generalize beyond the training set χ_{tr}. To test the student's generalization ability, we use a validation set $\chi_{\text{v}} = \{(x_v, u_v, \dot{x}_v) \mid 1 \leq v \leq V\}$ such that $\dot{x}_v = A^* x_v + B^* u_v$ for each $v = 1, \ldots, V$.

The *validation error*, ε_{val}, is defined as the number of input-output pairs $(x_v, u_v, \dot{x}_v)$ of a validation set of size V (with $1 \leq v \leq V$) that is not reproduced by the student for one independent run:

$$\varepsilon_{\text{val}}(A^*_i, B^*_i, \chi_{tr\,i}) = \sum_{v=1}^{V} H\left(\frac{|A_{\text{est}} x_v + B_{\text{est}} u_v - \dot{x}_v|}{|\dot{x}_v|} - \varepsilon_{\text{err}}\right),$$

with $H(x) = 1$ if $x > 0$ else $H(x) = 0$ and ε_{err} the maximum deviation from zero that is considered insignificant.

The *generalization error*, ε_{gen}, is defined as the ratio of the number of students wherefore $\varepsilon_{\text{val}} > 0$ to the total number of independent runs:

$$\varepsilon_{\text{gen}} = \frac{1}{nrRuns} \sum_{i=1}^{nrRuns} H\left(\varepsilon_{\text{val}}\left(A^*_i, B^*_i, \chi_{tr\,i}\right)\right),$$

with χ_{tr} an independent training set for each run. The learning task can now be formulated as follows: *the algorithm should produce a student S given χ_{tr} such that ε_{gen} is minimal.*

The algorithm we use is a reformulation of the problem in terms of linear programming: the objective is to minimize $\|A_{\text{est}}\|_1$ subject to the M constraints $\dot{x}_m = A_{\text{est}} x_m + B_{\text{est}} u_m$. This choice is motivated by the sparsity constraint on the networks to be identified. If the student S reproduces the teacher T, the result must be sparse, hence we prefer solutions with as few non-zero components as possible.

The constraints can be written more explicitly as

$$\sum_{i=1}^{N} A_{\text{est}_{i,j}} x_{j,m} + B_{\text{est}_{i,p}} u_{p,m} = \dot{x}_{i,m}, \quad j = 1, \ldots, N;\ p = 1, \ldots, P;\ m = 1, \ldots, M. \tag{4.2}$$

Hence, each row of A and B is a solution to a set of M equations and can be determined independently, an observation to which we will return later on. For $M \leq N$, infinitely many solutions can be found, from which linear programming will select the most sparse. Trivially, for $M = N + 1$, the set of equations will have a unique solution: the teacher T. This implies that one can expect a generalization error $\varepsilon_{\text{gen}} \approx 1$ for very small training sets, i.e., $M \ll N$, while $\varepsilon_{\text{gen}} \approx 0$ for $M \approx N$. We may conclude that ε_{gen} will be a function of the training set size M. We denote the number of patterns such that $\varepsilon_{\text{gen}} = 1/2$ by M_{gen}, the *generalization threshold*.

Although the generalization error is a good measure to evaluate the student's quality, it will nevertheless be useful to consider a measure to compare the student's structure to that of the teacher. Since our setting is that of identifying interaction networks, the presence or absence of such an interaction in the inferred model S is important. This can be characterized by the following three quantities: the number of *false negatives*, n_{fneg}, the number of *false positives*, n_{fpos}, and the number of *correlation errors*, n_{corr}. These three quantities measure the quality of the identification process. By definition, $0 \leq n_{\text{fneg}} \leq Nk^*$, $0 \leq n_{\text{fpos}} \leq N(N-k^*)$ and $0 \leq n_{\text{corr}} \leq Nk^* - n_{\text{fneg}}$. Note that these error measures can all be zero, even if the student does not generalize well, i.e., $\varepsilon_{\text{gen}} > 0$. Also notice that $0 \leq n_{\text{fneg}} + n_{\text{fpos}} + n_{\text{corr}} \leq N^2$. Therefore, we aggregate these three measures into the operator $S \odot T = (N^2 - n_{\text{fneg}} - n_{\text{fpos}} - n_{\text{corr}})/N^2$ that measures the quality of the identification.

4.1.2 Relation to feature selection

From Equation 4.2, it is clear that the problem of identifying the interactions within a network modeled by the matrix A^* can be decomposed into identifying the N rows of that matrix. Since, apart from the sparsity constraint, interactions in the teacher are completely random, these rows can be determined independently. We can now shift the perspective slightly and reformulate the original problem in terms of N simpler ones: given an input vector $x \in \mathbb{R}^N$, which of the N components of x will effectively contribute to the output $\dot{x} \in \mathbb{R}^N$? This can be viewed as a feature selection problem, since the sparsity of the teacher implies that only very few components will contribute. As for network identification, we can define the generalization error for feature selection, $\varepsilon_{\text{gen}}^{\text{fs}}$. This $\varepsilon_{\text{gen}}^{\text{fs}}$ represents the ratio of the number of students wherefore $\varepsilon_{\text{val}}(a^*, b^*, \chi_{tr\,i}) > 0$ to the total number of independent runs. With $a^* \in \mathbb{R}^N$ and $b^* \in \mathbb{R}^P$. At this point, it is useful to note that the generalization error can be interpreted as the probability that the student will not compute the correct output on a random input. The probability that N independent feature selection problems will *all* compute the correct answer is thus given by $(1-\varepsilon_{\text{gen}}^{\text{fs}})^N$, which allows us to compute the generalization error for network identification ε_{gen} from that for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}$ as follows:

$$\varepsilon_{\text{gen}} = 1 - (1 - \varepsilon_{\text{gen}}^{\text{fs}})^N. \tag{4.3}$$

This relation will be verified experimentally in Section 4.1.3.

4.1.3 Numerical experiments

By means of experiments, we will consecutively address the following research questions that arise naturally in this context:

1. Is it possible to identify T with a training set that contains less than $N+1$ input-output pairs? If so, what is the value of the generalization error ε_{gen} as a function of the training set size?

2. How does the generalization error ε_{gen} depend on the teacher's sparsity?

3. What is the evolution of the similarity of the student to the teacher when compared with the teacher as a function of the training set size?

4. Is the algorithm robust against noise?

5. How does the generalization error ε_{gen} scale with the system size N?

To facilitate the discussion, we first introduce some convenient notation. The ratio of the training set size to the system size is denoted by $\alpha = M/N$. In particular, $\alpha_{\text{gen}} = M_{\text{gen}}/N$. The fraction of non-zero components per row of a system is denoted by $\kappa = k/N$. In particular, $\kappa^* = k^*/N$. The amplitude of the noise should be considered relative to the amplitude of the signal, i.e., we define $\sigma = \sigma_{\text{noise}}/\sigma_{\dot{x}}$, where $\sigma_{\dot{x}}$ is the standard deviation of the output vectors' components $\dot{x}_{i,m}$. The kT non-zero components of the teacher A^*, the components of B^*, and of the input x_m are drawn from an uniform distribution over $]-1, 1[$.

Generalization error

To determine the generalization error, we randomly generate a set of M input vectors and a random teacher system so that we can compute the output to obtain a training set χ_{tr}. The algorithm produces a student, for which we calculate the generalization error. Since its value depends on the particular selection of input and teacher, we independently repeat this procedure many times to compute the average. Figure 4.1 shows the observed generalization error, ε_{gen}, as a function of the training set size α.

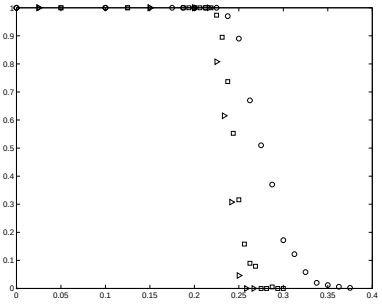

Figure 4.1: The generalization error ε_{gen} as a function of the training set size α for $N = 80$ ($\circ$), $N = 160$ ($\square$) and $N = 300$ ($\triangleright$) for $\kappa^* \approx 0.03$.

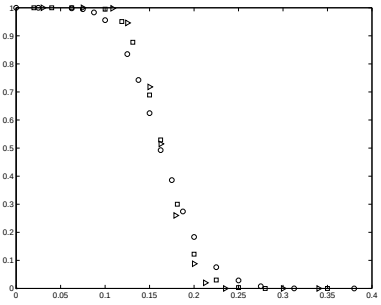

Figure 4.2: The generalization error for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}$ as a function of the training set size α for $N = 80$ ($\circ$), $N = 160$ ($\square$) and $N = 300$ ($\triangleright$) for $\kappa^* \approx 0.03$.

This result is surprising in two respects: first, the generalization error decreases to zero for a training set size $\alpha < 1$, and second, the transition towards generalization is quite abrupt. It is also clear from Figure 4.1 that the transition is increasingly abrupt for increasing system size N. It is instructive to relate the generalization error ε_{gen} for network identification to that for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}$ in Figure 4.2. The latter figure illustrates that $\alpha_{\text{gen}}^{\text{fs}}$, the training set size

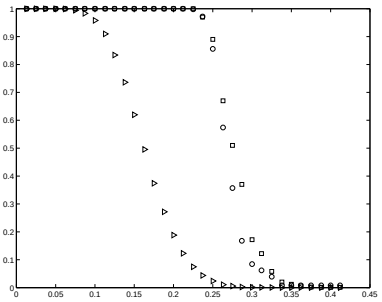

Figure 4.3: The observed (□) versus the computed (○) generalization error ε_{gen} as a function of α for $N = 80$, $\kappa^* \approx 0.03$. The curve representing $\varepsilon_{\text{gen}}^{\text{fs}}$ (▷) is given as reference.

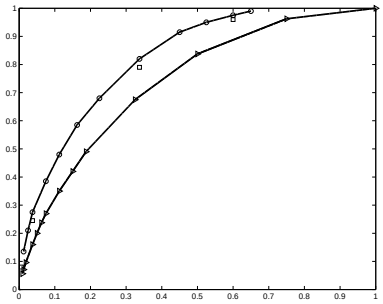

Figure 4.4: The generalization threshold α_{gen} as a function of the sparsity κ^* for $N = 80$ (○) and a few values for $N = 160$ (□). The generalization threshold for feature selection $\alpha_{\text{gen}}^{\text{fs}}$ (▷) is given as reference.

α for $\varepsilon_{\text{gen}}^{\text{fs}} = 1/2$, is independent of the system size N and that $\varepsilon_{\text{gen}}^{\text{fs}}$ converges to a step-function for $N \to \infty$. So, we observe a first-order phase transition between a regime for $\alpha < \alpha_{\text{gen}}^{\text{fs}}$ where the student simply reproduces the training set, and another regime for $\alpha > \alpha_{\text{gen}}^{\text{fs}}$ where he is able to reproduce the teacher's output perfectly.

The relation between the generalization error for the network identification problem ε_{gen} and that for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}$, given by Equation 4.3, is illustrated in Figure 4.3. It is clear that ε_{gen} for the network identification problem can reliably be estimated from $\varepsilon_{\text{gen}}^{\text{fs}}$ for the feature selection problem. Values beyond α_{gen} are less reliable since inaccuracies for $\varepsilon_{\text{gen}}^{\text{fs}}$ are amplified considerably due to the mathematical form of Equation 4.3.

Note that in our experiments, k is equal for each row. But, this k does not need to be constant and can thus vary for each row of A. Since the probability to generalization depends on the sparsity (see the next Section of sparsity), we will represent the sparsity of the matrix as the maximum sparsity over all rows. As a consequence, α_{gen} for the matrix will be over estimated, which is a "safe" error.

Sparsity

The next question concerns the relation between the sparsity of the teacher and the generalization threshold. For a non-sparse teacher, i.e., $\kappa^* \approx 1$, one would need a training set of size $\alpha \approx 1$ since each of the $N+1$ components has to be determined. However, as Figure 4.1 illustrates, the fact that the teacher is sparse simplifies the identification process considerably. Figure 4.4 shows the generalization threshold α_{gen} for network identification as a function of κ^*. It is clear that training sets of increasing size α are required to facilitate the transition to the generalization regime as κ^* increases, i.e., as the sparsity decreases. As expected, for $\kappa^* \approx 1$, we have that $\alpha_{\text{gen}} \approx 1$. It is clear that the advantage sparsity offers to the efficiency of the learning algorithm virtually vanishes for $\kappa^* \approx 0.5$. However, it is very pronounced for $\kappa^* < 0.2$. As before, these results have been obtained for many independent instances of the training set and teacher.

Learning process

To gain a better understanding of the learning process, i.e., the evolution of the student with respect to the teacher as a function of the training set size, we first consider a fixed training set and teacher. Thereto, define a sequence of training sets χ_m for $m = 1, \ldots, M$ such that $\chi_m \subset \chi_{m+1}$ and $|\chi_{m+1}| = |\chi_m| + 1$. These sets are used to determine a sequence of students S_m for $m = 1, \ldots, M$. Figure 4.5 shows $S_{\alpha N} \odot T$ as a function of the training set size α. For $\alpha N = 1$, the number of false negatives is $N\kappa_T$ and the number of false positives is 0. For increasing α, the number of false positives increases approximately linearly with α, while the number of false negatives decreases very slowly. The plot illustrates clearly that the transition to generalization is very sudden: at α_{gen}, i.e., $S_{\alpha_{\text{gen}} N} \odot T = 1$. Figures 4.6 and 4.7 confirm that the scenario sketched above is indeed the typical behavior when it is averaged over many independent training sets and teachers. The latter plot illustrates the explanation given above for the behavior of $S_{\alpha N} \odot T$.

Scaling with system size

How the system scales with the system size has already been illustrated in Figures 4.1, 4.4, 4.6, and 4.7. However, it is Figure 4.2 that provides the most insight. It turns out that the generalization error curves for various system sizes can be computed by applying the correct scaling on the system size α. Suppose we have a curve for $\varepsilon_{\text{gen}}^{\text{fs}}$ versus α for system size N_0, then the curve

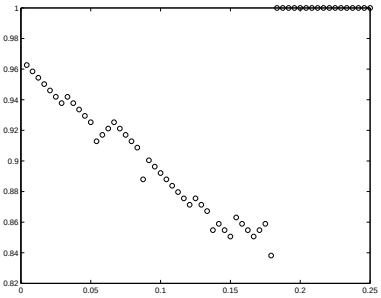

Figure 4.5: The learning process characterized by $S_m \odot T$ as a function of the size of the training set m/N for an individual run.

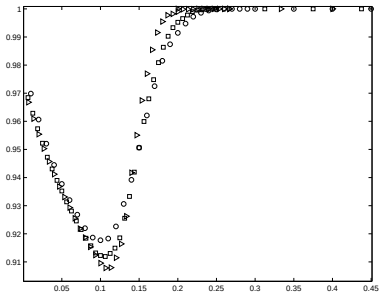

Figure 4.6: The learning process characterized by $S_m \odot T$ as a function of the size of the training set m/N, system sizes $N = 80$ ($\circ$), $N = 160$ ($\square$) and $N = 300$ ($\triangleright$) for $\kappa^* \approx 0.03$.

for system size N can be obtained by scaling:

$$\alpha(N) = \alpha_{\text{gen}}^{\text{fs}} + \sqrt{N_0/N}\,\left(\alpha(N_0) - \alpha_{\text{gen}}^{\text{fs}}\right). \tag{4.4}$$

The result is shown in Figure 4.8 for system sizes $N = 80$ and $N = 160$ with sparsity $\kappa^* = 0.03$. The curve computed for $N = 160$ by scaling that for $N = 80$ is in very good agreement with the one observed for that system size.

4.2 The impact of measurement noise

Gene/protein expression is a stochastic or noisy process. On the one hand, this noise is due to randomness in transcription and translation and comes about in two ways. The inherent stochasticity of biochemical processes such as transcription and translation generates intrinsic noise. In addition, fluctuations in the amounts or states of other cellular components lead indirectly to variation in the expression of a particular gene and thus represent extrinsic noise. For more details, see [26, 72]. In general, the amount of protein produced by a particular gene varies from cell to cell. Therefore, the system noise is divided in two components: extrinsic noise, which is global to a single cell but varies from one cell to another, and intrinsic noise, which varies from cell to cell and over time in a single cell. Fluctuations in even one single component may potentially affect the performance of the entire system. Furthermore, there is experimental noise on microarray data which are used for this process; see [77]. Remark that similar considerations apply to other application domains.

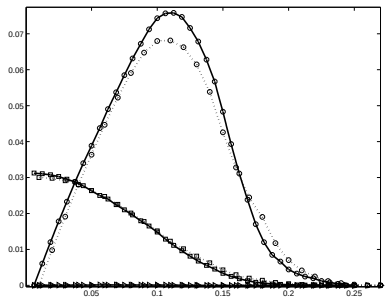

Figure 4.7: Measures n_{fneg} (□, $N = 80$ dotted line, $N = 160$ solid line), n_{fpos} (○, $N = 80$ dotted line, $N = 160$ solid line) and n_{corr} (▷, $N = 80$ dotted line, $N = 160$ solid line) as a function of the training set size α for $\kappa^* \approx 0.03$.

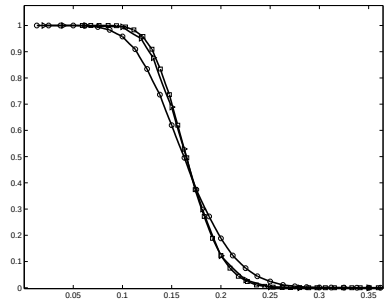

Figure 4.8: Figure 4.2 using Equation (4.4) for $N = 80$ (○), $N = 160$ (□) computed, $N = 160$ observed (▷), $\kappa^* \approx 0.03$.

In this work, regulatory networks are identified from poor data. As mentioned above, noise is ubiquitous, thus it is of large importance that the approach is robust towards this—intrinsic, extrinsic, and measurement—noise.

To test the robustness of the Single Linear and Piecewise Linear algorithm, some system (intrinsic and extrinsic) noise ξ will be added to the output data set $\dot{X}$ and some measurement noise ζ will be added to the output measurement set Y (see Sections 3.4 and 3.11). They are normally distributed with zero mean and some standard deviation σ that determines the noise level.

For the identification of sparse regulatory networks from poor and noisy data, the following equation is thus of interest:

$$\dot{X} = AX + BU + \xi.$$

In line with the notations used before, matrix X represents the input data, matrix A represents the interactions between the unknown components and matrix B is the input matrix with the external inputs defined by the matrix U. To select the most sparse and robust solution, the technique of L_1-minimization was used and thus this equation is reformulated as the following linear pro-

gramming (LP) problem:

$$\min_{A \in \mathbb{R}^{(N \times N)}} \quad \|A\|_1$$
$$\text{s.t.} \quad AX + BU = \dot{X}_\xi,$$

where $\dot{X}_\xi = (1 + \xi)\dot{X}$, with $\xi \in N(0, \sigma)$.

Figure 4.9 shows the relative deviation of the respective output of student and teacher $\delta\dot{x}$ as a function of the noise level σ. As can be expected for a linear system, the quality is acceptable for low noise levels only. In particular, $\sigma < 0.01$ still yields a reasonably accurate output. The breakdown for higher noise levels is illustrated by Figure 4.10 which shows a very large increase in the number of false positives n_{fpos} for an increasing noise level σ. Although these components are very small, they nevertheless allow perfect identification of the network.

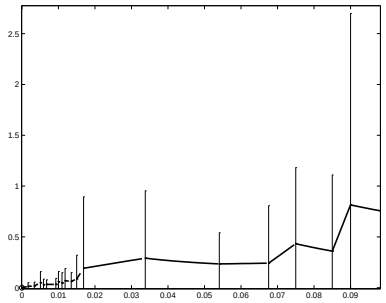

Figure 4.9: Deviation of the student and teacher output $\delta\dot{x}$ as a function of the noise level σ on the training set.

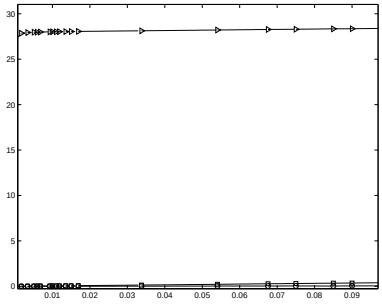

Figure 4.10: n_{fneg} ($\square$), n_{fpos} ($\triangleright$) and n_{corr} ($\circ$) as a function of the noise level σ for $N = 80$, $\kappa^* \approx 0.03$.

The goal is thus to find an approximate and stable solution for the introduced LP above. This Section proposes a method to obtain an approximate solution to make our identification approach robust with respect to noise. Therefore, a regularization parameter is introduced to the LP. This parameter makes it possible to introduce additional information in order to solve this noise problem by imposing certain distributions on the optimalization terms.

4.2.1 The Tikhonov regularization

The standard approach to solve an underdetermined system of linear equations is of the form

$$\min_{\chi \in \mathbb{R}^{(N+P)\times N}} \quad \chi \qquad \text{s.t.} \quad D\chi = R, \tag{4.5}$$

with

- $D = \begin{pmatrix} X^T & U^T \end{pmatrix}$, $D \in \mathbb{R}^{M\times(N+P)}$;
- $R = \dot{X}_\xi^T$, $R \in \mathbb{R}^{M\times N}$; and
- $\chi = \begin{pmatrix} A^T \\ B^T \end{pmatrix}$, $\chi \in \mathbb{R}^{(N+P)\times N}$.

This approach seeks to minimize the residual $\|D\chi - R\|_2$ where $\|.\|_2$ is the Euclidian norm. In this case, however, R is noisy. So, although this approach has a solution, solving it directly will not give the right solution.

In order to reliably find a solution, as to control the effect of noise, a regularization technique can be used. A simple form of regularization was proposed by Tikhonov [75], which introduced a trade-off between fitting the data and reducing a norm of the solution. This Tikhonov regularization proposes to minimize χ by solving

$$\min \|D\chi - R\|_1 + \lambda\|\chi\|_1, \tag{4.6}$$

with $\lambda > 0$ called the additional regularization parameter and can be interpreted as a penalty for model complexity.

4.2.2 Including the regularization parameter into network reconstruction

The goal is to find an approximated stable solution. Therefore, the presence of noise and sparsity have to be taken into account.

The solution is sparse if A is the null-matrix and the solution is robust with respect to noise if $\|\dot{X}_\xi - \dot{X}^*\|_p = \|\dot{X}_\xi - AX - BU\|_p \approx 0$, with $\dot{X}_\xi$ the system output with some stochastic Gaussian white noise, $\dot{X}^*$ the desired system output, and $\|.\|_p$ a suitable L_p-norm. Note that, if $\|\dot{X}_\xi - \dot{X}^*\|_p = 0$, then

$\dot{X}_\xi = \dot{X}^*$, and thus the estimated system $\dot{X}_{\text{est}}$ which should fit $\dot{X}^*$, will contain noise as well. Hence, to get a robust system, the difference between both systems has to be very small but non-zero. Both conditions are important to reconstruct sparse interaction networks from poor and noisy data.

Given the empirical data $\mathcal{D} = (X, \dot{X}, U)$, the L_1-minimization presented in Chapter 3 (see Equation 4.5) satisfies both conditions. Given the nature of this LP, the solution satisfies the robustness of the fit, because $\|\dot{X}_\xi - AX - BU\|_1 \approx 0$. But, as shown in the noisy data experiment of Section 4.1.3, the solution is non-sparse, because $\kappa^* \ll \kappa^* - (n_{\text{fneg}} - n_{\text{fpos}} - n_{\text{corr}})/N^2$. Again, κ^* is the fraction of non-zero components per row of the desired interaction matrix A^*, N is the number of unknowns and n_{fneg}, n_{fpos}, and n_{corr} are the number of false negatives, false positives, and correlation errors, respectively.

Therefore, the L_1-minimization of the original approach (see Equation 4.5) is adapted as follows:

$$\min \quad \|A\|_1 + \|\dot{X}_\xi - AX - BU\|_1.$$

Let us denote $\|A\|_1$ as the *sparsity term* and $\|\dot{X}_\xi - AX - BU\|_1$ as the *fitting term*. Now, the fitting is more flexible, because $\|\dot{X}_\xi - AX - BU\|_1$ does not need to be zero. As a consequence, the LP seeks a balance between the sparsity and the robustness. In this situation, the minimization of both terms have the same priority. Note that this equation is less strict then the original one.

In order to give preference to a particular solution with desirable properties, the regularization parameter λ is included in this minimization:

$$\min \quad \lambda\|A\|_1 + \|\dot{X}_\xi - AX - BU\|_1. \tag{4.7}$$

This regularization parameter λ has a remarkable influence on the proposed LP. The larger the parameter value, the more the sparsity term and the less the fitting term will be emphasized. Vice versa, the smaller the parameter ($\lambda \to 0$), the less sparse and the lower the fitting term will be. Thus, in this situation, it is important to get a good understanding about this regularization parameter to get an acceptable estimation and balance. This is worked out in the experimental Section 4.2.3.

As noted in Section 4.1.2, each row of the interaction matrix can be determined independently. This means that the problem of identifying the interactions within a network can be decomposed into identifying N rows of that

matrix. Therefore, we define the regularization parameter as a vector $\lambda \in \mathbb{R}^N$. Now for each λ_i—the regularization parameter for row i—an acceptable value has to be stipulated such that there is a good balance between both optimization terms.

Network reconstruction from noisy data

The learning algorithm should return a network, also called student, $S = (A_{\text{est}}, B_{\text{est}})$, with a sparse interaction matrix $A_{\text{est}} \in \mathbb{R}^{N\times N}$ and input matrix $B_{\text{est}} \in \mathbb{R}^{N\times P}$, that reproduces the noisy training set $\chi_{\xi_{\text{tr}}}$: $\dot{x}_{\xi_m} \approx A_{\text{est}}x_m + B_{\text{est}}u_m$ for $m = 1, \ldots, M$. For this problem as well, the student should be able to generalize beyond the training set $\chi_{\xi_{\text{tr}}}$. To test the student's generalization ability, we use a validation set $\chi_{\xi_{\text{v}}} = \{(x_v, u_v, \dot{x}_{\xi_v}) \mid 1 \leq v \leq V\}$ such that $\dot{x}_{\xi_v} \approx A^*x_v + B^*u_v$ for each $v = 1, \ldots, V$.

The algorithm we use is based on the "Machine Learning" algorithm of Section 4.1 and is a reformulation of the problem in terms of linear programming: the objective is to minimize $\lambda\|A\|_1 + \|\dot{X}_\xi - AX - BU\|_1$. To compare the student's structure with the teacher's, three quantities are used: the number of *false negatives*, n_{fneg}, the number of *false positives*, n_{fpos}, and the number of *correlation errors*, n_{corr}. By definition, $0 \leq n_{\text{fneg}} \leq Nk^*$, $0 \leq n_{\text{fpos}} \leq N(N - k^*)$ and $0 \leq n_{\text{corr}} \leq Nk^* - n_{\text{fneg}}$.

Influence and estimation of the regularization parameter

If $\lambda = 0$, it can be expected that the LP seeks a non-sparse but good fitting system, whereas, for larger values of λ, the LP emphasizes the sparsity of the interaction matrix and less the quality of the fit. Therefore, the system error N_{e} between the desired and the resulting system is considered to keep the interval $[\lambda_{min}, \lambda_{max}]$ as small as possible. More precisely, the false negatives n_{fneg} and the false positives n_{fpos} will be considered. An increasing λ ensures a transition from zero to non-zero for n_{fneg} and vice versa for n_{fpos}. To obtain the interval $[\lambda_{min}, \lambda_{max}]$ two quantities are defined, λ_i^- and λ_i^+ with λ_i^- represents λ_i for which n_{fneg} becomes non-zero and λ_i^+ represents λ_i for which n_{fpos} becomes zero. Then $\lambda_{min} = \min\{\lambda_i^-, \lambda_i^+ | 1 \leq i \leq N\}$ and $\lambda_{max} = \max\{\lambda_i^-, \lambda_i^+ | 1 \leq i \leq N\}$.

To evaluate the student's quality, the *data error* D_{e} is defined as the error between the student's system output and the desired system output:

$$D_{\text{e}} = \frac{|\dot{X}_{\text{est}} - \dot{X}^*|}{|\dot{X}^*|}.$$

This data error will be a function of the regularization parameter λ. Starting from a small λ_i, the system is robust with respect to noise, because the fitting term $\|\dot{X}_\xi - AX - BU\|_1 \approx 0$, and, thus, D_{e_i} is small. The bigger λ_i the less robust the system, and the bigger D_{e_i} becomes. Therefore, this data error D_{e} can be used to estimate λ such that there is a balance between both optimization goals, sparsity and robustness.

4.2.3 Numerical experiments

In line with the definitions above, we use the parameters N and M to quantify the size of the input and $\alpha = M/N$ to represent the ratio of the training set size to the system size. The fraction of non-zero components per row of the system matrices is denoted by $\kappa = k/N$. As before, the components of the teacher and the input data X and U are drawn from a uniform distribution over $]-1, 1[$.

The resulting system has to be sparse and robust with respect to noise. Both optimization goals will be influenced by the size of the regularization parameter. If $\lambda = 0$, the LP seeks a non-sparse system with a small fitting term, whereas for larger values of λ, the sparsity will be emphasised more and the fitting term will be emphasized less. Therefore, in the first experiment, the regularization parameter λ'_i, for each row separately, varies from 0 to 50 in steps of 0.1. The intention is to estimate a limited and representative interval for further experiments.

Using the L_1-minimization of Equation 4.7 and $\lambda_i \in [0, 50]$, the false negatives n_{fneg} and the false positives n_{fpos} are considered to fix the interval $[\lambda_{min}, \lambda_{max}]$. The frequencies of λ_i^- and λ_i^+, for different training set sizes $\alpha_1, \ldots, \alpha_6$, are shown in Figure 4.11. λ_i^- is the value of λ_i for which n_{fneg} becomes non-zero and λ_i^+ is the value of λ_i for which n_{fpos} becomes zero. To get a representative result, the experiment is run 100 times, thus 100 systems are generated for these training set sizes $\alpha_1, \ldots, \alpha_6$, and a fixed sparsity $\kappa^* \approx 0.03$. Figure 4.11 shows that the size of the interval depends on the number of unknowns N, whereas λ_{min}, and hence also λ_{max}, depends on the number of observations M.

For example, from Figure 4.11, we may conclude that the interesting interval, for a system with $N = 80$, $M = 40$ and a sparsity $\kappa^* = 0.03$, lies between $\lambda_{min} = 6.8$ and $\lambda_{max} = 13.8$.

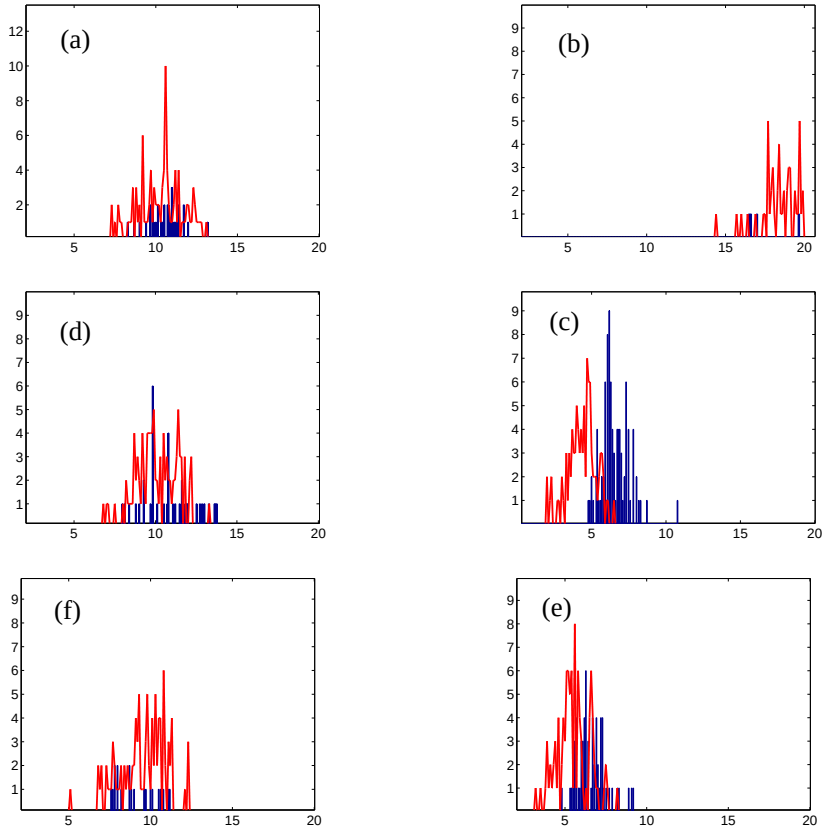

Figure 4.11: The frequentie of λ_i^- (lines) and λ_i^+ (histogram) for training set sizes $\alpha_1, \ldots, \alpha_6$, and a fixed sparsity $\kappa^* \approx 0.03$. (a) $\alpha_1 = 50/100 = 0.5$, (b) $\alpha_2 = 75/100 = 0.75$, (c) $\alpha_3 = 40/80 = 0.5$, (d) $\alpha_4 = 20/80 = 0.25$, (e) $\alpha_5 = 30/60 = 0.5$, (f) $\alpha_6 = 18/60 = 0.3$.

Now that an appropriate interval has been determined, a particular value for λ_i, $i = 1, \ldots, N$ needs to be selected. Therefore, the optimization terms, i.e., the robustness of the fit and the sparsity, are considered in the next Section.

Phase transitions as a consequence of the regularization parameter

The regularization parameter λ influences the quality of the system output—the robusteness of the fit —, and thus influences the data error D_e. Furthermore, the regularization parameter λ also influences the sparsity of the system output. Thus, to determine the parameter, the data error and the sparsity for each row $i = 1, \ldots, N$ and for each possible $\tilde{\lambda}$, $\lambda_{min} \leq \tilde{\lambda} \leq \lambda_{max}$, needs to

be computed. Therefore, we need a student. For this purpose, we randomly generate a set of M noisy input vectors and a random teacher so that we can compute the output to obtain a noisy training set $\chi_{\xi_{\text{tr}}}$.

Figure 4.12 represents the data error per row for $\tilde{\lambda}_i$, $\lambda_{min} \leq \tilde{\lambda}_i \leq \lambda_{max}$ and Figure 4.13 represents the sparsity per row for this $\tilde{\lambda}_i$.

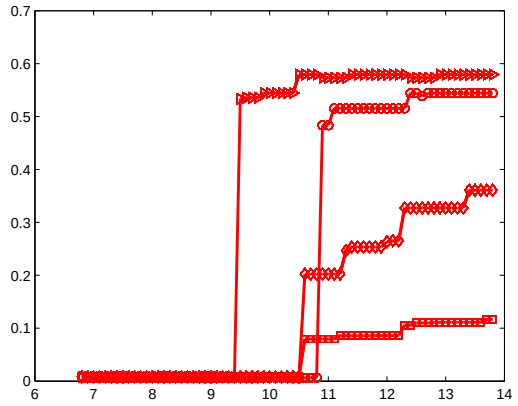

Figure 4.12: The data error D_{e_i} for row $i = 1, 2, 3, 4$ as a function of $\tilde{\lambda}_i$, $6.8 \leq \tilde{\lambda}_i \leq 13.8$, with $\alpha = 40/80$ and $\kappa^* \approx 0.03$.

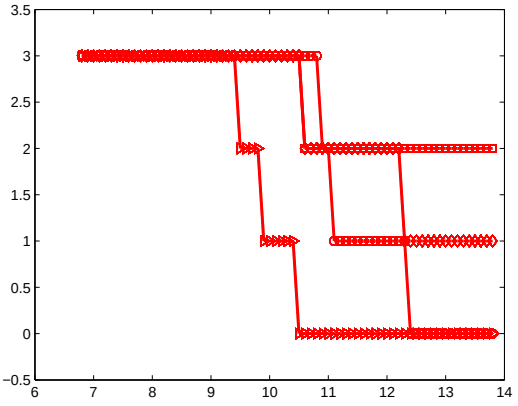

Figure 4.13: The sparsity, k_i, for row $i = 1, 2, 3, 4$ as a function of $\tilde{\lambda}_i$, $6.8 \leq \tilde{\lambda}_i \leq 13.8$, with $\alpha = 40/80$ and $\kappa^* \approx 0.03$.

From Figure 4.12, it is clear that for each row i, there is a transition for the data error, $D_{e_i} \approx 0$ to $D_{e_i} >> 0$, indicating that the resulting system becomes less robust. For the sparsity in Figure 4.13 we can conclude the same. At the same values for $\tilde{\lambda}_i$, there is a decrease for the number of non-zeros for each specific row i. For example, for the first row, $i = 1$, the transition happens for $D_{e_1} = k_1 \approx 9.4$ and, thus, $\tilde{\lambda}_1 \approx 9.4$. Note that $\tilde{\lambda}_1 \in [6.8, 13.8]$.

This means that, at these values for $\tilde{\lambda}_i$, the equilibrium between both optimization terms disappears. Therefore, the value of the regularization parameter is determined as $\lambda_i = \tilde{\lambda}_i - 0.1$ for each row i.

Generalization error

Now that an appropriate interval for the regularization parameter λ has been found, we obtain a training set, χ_{tr}, as the output of M randomly generated input vectors and a random teacher. In this Section, we want to work with a noisy training set, $\chi_{\xi_{\text{tr}}}$. Therefore, system noise is added to the training set. This noise is normally distributed with zero mean and a standard deviation of $\sigma = 0.01$. From this noisy training set, the algorithm produces a student, for which we calculate the generalization error for feature selection, $\varepsilon_{\text{gen}}^{\text{fs}}$. The value of $\varepsilon_{\text{gen}}^{\text{fs}}$ depends on the particular selection of input and teacher. Therefore, this procedure will be repeated many times to compute the average.

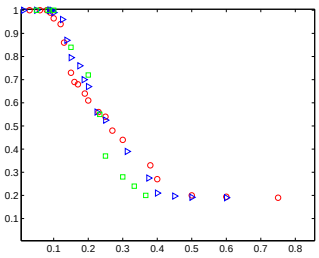

Figure 4.14: The generalization error (with $\varepsilon_{\text{err}} = 10^{-5}$) for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}$ as a function of the training set size α for $N = 80$ ($\circ$), $N = 100$ ($\triangleright$), and $N = 120$ ($\square$).

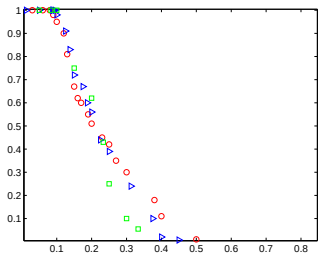

Figure 4.15: The generalization error (with $\varepsilon_{\text{err}} = 10^{-3}$) for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}{}'$ as a function of the training set size α' for $N = 80$ ($\circ$), $N = 100$ ($\triangleright$), and $N = 120$ ($\square$).

Figure 4.14 shows the observed generalization error, $\varepsilon_{\text{gen}}^{\text{fs}}$, as a function of the training set size, α. As for the simple model in Section 4.1, Figure 4.14 il-

lustrates that the transition to generalization is still quite abrupt. However, Figure 4.14 shows two important differences with the simple model of Section 4.1. First, it shows that the generalization error decreases to *almost* zero for a training set size $\alpha < 1$. This is due to the definition of the generalization error:

$$\varepsilon_{\text{gen}} = P\left(\frac{|A_{\text{est}}X + B_{\text{est}}U - \dot{X}^*|}{|\dot{X}^*|} > \varepsilon_{\text{err}}\right),$$

with ε_{err} the maximum deviation from zero that is considered insignificant. Since, for the fitting term, we have that $\|\dot{X}_\xi - \dot{X}^*\|_1 = \|\dot{X}_\xi - AX - BU\|_1 \approx 0$, the threshold, $\varepsilon_{\text{err}} = 10^{-5}$, must be relaxed to be able to compare the results with those of Section 4.1. For the adapted threshold, $\varepsilon_{\text{err}} = 10^{-3}$, the results are shown in Figure 4.15. From this Figure, it is clear that $\alpha_{\text{gen}}^{\text{fs}}{}'$, the training set size α' for $\varepsilon_{\text{gen}}^{\text{fs}} = 1/2$, is independent of the system size N and that $\varepsilon_{\text{gen}}^{\text{fs}}$ converges to a step-function for $N \to \infty$.

Second, we need to emphasize that, for generalization, more observations are needed, because $\alpha_{\text{gen}}^{\text{fs}}{}' > \alpha_{\text{gen}}^{\text{fs}}$, with $\alpha_{\text{gen}}^{\text{fs}}$ the training set size for the model of Section 4.1.

Furthermore, we consider a standard deviation or noise level of $\sigma = 0.05$ and $\sigma = 0.1$. From Figure 4.16, it is clear that for an increasing noise level, $\alpha_{\text{gen}}^{\text{fs}}{}'$ increases and the minimum value for the generalization error, $min(\varepsilon_{\text{gen}}^{\text{fs}})$, increases as well. Still, the student is able to fit the teacher.

Learning process

Now, we consider a fixed noisy training set and teacher to compute a sequence of students, S_m, for $m = 1 \ldots M$, with respect to the teacher as a function of the training set size, χ_m, such that $\chi_m \subset \chi_{m+1}$ and $|\chi_{m+1}| = |\chi_m| + 1$.

The number of false negatives, n_{fneg}, false positives, n_{fpos}, and the correlation error, n_{corr}, are represented in Figure 4.17 as a function of the training set size α. Figure 4.17 shows almost the same phenomenons as for Chapter 4.1: for $\alpha N = 1$, $n_{\text{fneg}} = Nk^*$ and $n_{\text{fpos}} = n_{\text{corr}} = 0$. For increasing α, n_{fneg} decreases slowly to zero and n_{fpos} initially increases and than abrupt decreases to approximately 0. Again, due to the property of the fitting term: if $n_{\text{fneg}} = n_{\text{fpos}} = n_{\text{corr}} = 0$, then $\dot{X}_\xi - AX - BU = 0$, which means that the noisy system is correctly identified. Here, n_{fneg} decreases to 0.002. Thus, over 500 runs, there is one run in which one non-zero component is not identified.

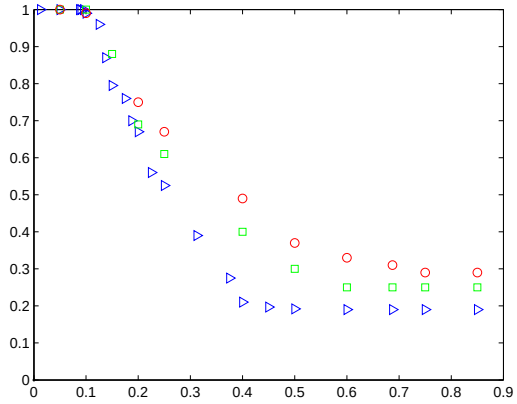

Figure 4.16: The generalization error (with $\varepsilon_{\text{err}} = 10^{-5}$) for feature selection $\varepsilon_{\text{gen}}^{\text{fs}}$ as a function of the training set size α for $N = 80$ and different noise levels $\sigma = 0.01$ ($\triangleright$), $\sigma = 0.05$ ($\square$), and $\sigma = 0.1$ ($\circ$).

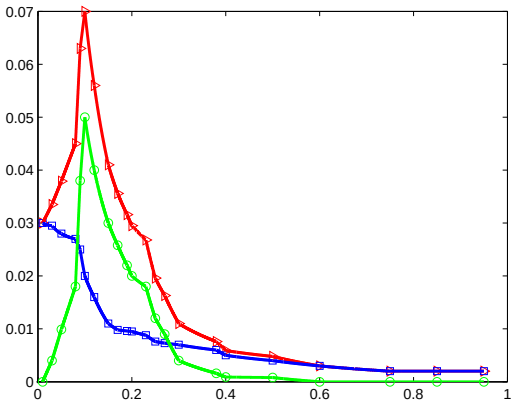

Figure 4.17: The n_{fneg} ($\square$), n_{fpos} ($\circ$), and their sum $n_{\text{fneg}} + n_{\text{fpos}}$ ($\triangleright$) as a function of the training set size $\alpha = m/N$ and system size $N = 80$, for $\kappa^* \approx 0.03$.

Because $0 \leq n_{\text{fneg}} + n_{\text{fpos}} + n_{\text{corr}} \leq N$, these three measures are aggregated into the operator $S \odot T = (N^2 - n_{\text{fneg}} - n_{\text{fpos}} - n_{\text{corr}})/N^2$ to represent the quality of the identification and is shown in Fig 4.18. This Figure confirms the scenario sketched above.

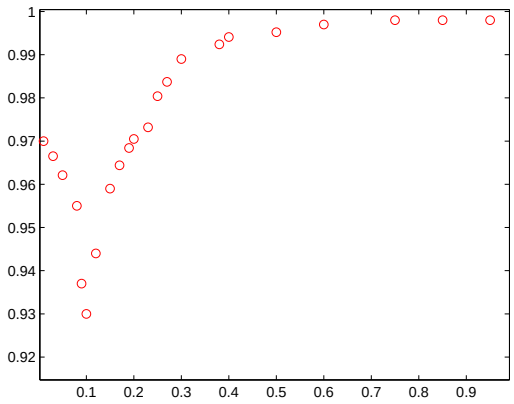

Figure 4.18: The learning process characterized by $S_m \odot T$ as a function of the size of the training set α for $\kappa^* \approx 0.03$.

4.3 Conclusions

In this Chapter, we went into detail about the impact of the number of observations to the number of unknown components, the role of the sparsity constraint, and the influence of noise on the identification.

First, we reformulated our programming problem in the context of machine learning. It is quite remarkable that a simple model such as the one considered here exhibits so many interesting features. With respect to the research questions addressed, we may conclude that the algorithm identifies a network with $N(N+P)$ components using a training set size of considerably smaller size. This can be observed from the generalization error, ε_{gen}. This error decreases to zero for a training set size, $\alpha < 1$, and is increasingly abrupt for increasing system sizes N. To get a better understanding, the number of false negatives, n_{fneg}, the number of false positives, n_{fpos}, and the correlation error, n_{corr}, were considered. For $\alpha N = 1$, the number of false negatives is $n_{\text{fneg}} = Nk^*$, and the number of false positives and the correlation error are $n_{\text{fpos}} = n_{\text{corr}} = 0$. For increasing α, n_{fneg} decreases slowly to zero and n_{fpos} decreases later, but fater, to zero. In other words, the identification of the non-zeros starts immediately, at $M = 1$, whereas the identification of the zeros starts later, but happens very fast.

Furthermore, we found that the algorithm strongly depends on the sparsity. The less sparse the teacher is, i.e., with increasing k^*, the more observations

are required to facilitate the transition to generalization. From the experiment, it turned out that, for 80 components, the algorithm performs well for a sparsity $k^* < 0.2$. Thus, for $N = 80$, the algorithm is able to identify an interaction matrix with up to 16 non-zero components per row.

An interesting observation is the relation between the identification of the interaction matrix and the independent identification of each row of this interaction matrix, which is called feature selection. Experiments, performed on the N independent feature selection problems, show that the training set size for $\varepsilon_{\text{gen}}^{\text{fs}} = 1/2$ is independent of the system size N and that this generalization error converges to a step-function for $N \to \infty$. From this observation, it turned out that the generalization error for various system sizes can be computed by applying the correct scaling on the system size α.

Second, in order to deal with noise, we added the regularization parameter to the L_1-minimization. This regularization parameter makes it possible to introduce additional information by imposing certain constraints on the optimization terms. From the experiments, it was shown that the optimal value for this parameter varied over an interval, $\lambda_{min} \leq \lambda \leq \lambda_{max}$. This interval depends on the number of unknown components, N, and on the number of observations, M. For increasing N, λ_{max} increased, and for increasing M, both λ_{min} and λ_{max} increased approximately equal, such that the interval approximately keeps the same size.

Furthermore, for this new LP, we may conclude again that the algorithm is able to identify a network with a high number of unknown components using a training set of a smaller size. From the experiments, it was clear that the generalization error still decreases abruptly to zero for a training set size $\alpha' < 1$ and that the training set size for $\varepsilon_{\text{gen}}^{\text{fs}} = 1/2$ is independent of the system size N. After we compared the training set size ${\alpha_{\text{gen}}^{\text{fs}}}'$ with those of the L_1-minimization without the regularization parameter, $\alpha_{\text{gen}}^{\text{fs}}$, we must determine that the generalization threshold for noisy data is larger than that for data without noise. Furthermore, the generalization error also converges to a step-function for $N \to \infty$.

5

Memory capacity of linear networks

All biological processes result from the interactions of genes and proteins with external stimuli. Because these stimuli can cause a threat to the survival of the organism or give a potential boost to its existence, it is important for the organism to learn and remember its best response for that specific external stimulus. The pair of the external stimulus and the optimal network response will be defined as an input-output pattern. These patterns have to be stored in the memory of the gene-protein interaction network. In this Chapter, we are interested in the maximum number of linearly independent patterns that can be stored in a network with N components, which interact with P inputs.

Here, it is assumed that to a certain combination of inputs U^*, there exists an optimal state of the system X^*, i.e., values of the gene expressions and protein levels that have been attained externally, e.g., through evolutionary learning. Given such a set of M learnt optimal input-output patterns, the design question here is to find a sparse network structure for the interaction matrix A and the coupling matrix B. This problem is formulated as an optimization problem in a linear programming setting.

In the neighborhood of a suitable smooth equilibrium, the activity of the components, represented by the matrix X, will develop according to the interaction matrix A and the input matrix B with the external inputs, defined

by the matrix U, as

$$\dot{X} = AX + BU + \xi(t). \tag{5.1}$$

Here, $\xi(t)$ represents white Gaussian noise with mean zero and standard deviation 1. Such Single Linear Models allow for a limited set of behaviors, but, here, they fully suit our purpose as a simple and transparent metaphor for studying the number of patterns M that the system can store as a function of the number of unknown compoments N, the number of external inputs P, and the sparsity in the interaction matrices A and B.

5.1 Storage of dynamic patterns of time series and its relation to network reconstruction

In this context, we define a *pattern* as a time series that represents the desired dynamical behavior of the state vector $x(t)$ in response to a given input vector $u^*(t)$, such as a finite impulse or a harmonic signal. In this way, the desired system outputs $x^*(t)$ and $\dot{x}^*(t)$ for a given input $u^*(t)$ can be compared with the observed outputs $x(t)$ and $\dot{x}(t)$. Suppose we have sets of M desired input-output triples $\{(u^*_m, x^*_m, \dot{x}^*_m) \mid 1 \leq m \leq M\}$, which can be used to define the three data matrices $X = (x^*_1 \dots x^*_M)$, $\dot{X} = (\dot{x}^*_1 \dots \dot{x}^*_M)$, and $U = (u^*_1 \dots u^*_M)$. In this setting, one could attempt to minimize the error $\|x(t) - x^*(t)\|$ as a function of the free parameters in A and B. Or, alternatively, one could attempt to minimize the error in the time derivatives $\|\dot{x}(t) - \dot{x}^*(t)\|_p = \|\dot{x}(t) - Ax(t) - Bu(t)\|_p$, where $\|\cdot\|_p$ represents a suitable L_p-norm. This error expresses the difference between the observed rate of change $\dot{x}(t)$ and predicted model output $Ax(t) + Bu(t)$.

Given X, $\dot{X}$, and U, one could compute the matrices A and B that minimize this error function. In case that the problem is underdetermined, it is useful to involve the constraint that the interaction network is sparse. In this sparse scenario, the minimization is performed over the number of zeros in A and B, while Equation 5.1 is considered as a constraint:

$$\begin{aligned} \min_{A,B} \quad & \|A\|_1 + \|B\|_1 \\ \text{s.t.} \quad & \dot{X} = AX + BU. \end{aligned}$$

As indicated above, this optimization problem coincides precisely with the problem of reconstructing the underlying interaction network from a time series of (e.g., microarray) measurements (see Chapter 3).

5.2 Memory storage of static equilibrium state patterns in linear networks

5.2.1 Linear state space formulation

In the static context, we assume that the system has converged to an equilibrium state X^* and that $\dot{X} = 0$. For the gene-protein interaction matrix, this means that, to a given input U^*, there is an optimal value X^* for the gene expressions and protein densities in the network such that $AX^* + BU^* = 0$. In our context, a (static) input-output pattern is defined as such an input-output pair (U^*, X^*). This pattern itself can be considered as the outcome of a long evolutionary learning process, where the best combination of gene/protein activations X are selected relative to the observed combination of external inputs (e.g., toxic agents, virus infections) U.

Here, we are interested in the maximum number M_{max} of linearly independent patterns that can be stored in a network of N genes or proteins, and P external inputs. Suppose that there exists a set of M linearly independent patterns, i.e., external inputs as a matrix $U \in \mathbb{R}^{P\times M}$ with $U = (u[1], \ldots, u[M])$, and the associated learnt states as a matrix $X \in \mathbb{R}^{N\times M}$, with $X = (x[1], \ldots, x[M])$. For each input-pattern pair $(x[j], u[j])$, Equation 5.1 implies that $Ax[j] + Bu[j] = 0$. For the entire set of patterns X and U, this means that the unknown matrices A and B feature in a linear way in the resulting matrix equation: $AX + BU = 0_{MN}$. This equation can be rewritten as

$$\begin{pmatrix} X^T & U^T \end{pmatrix} \begin{pmatrix} A^T \\ B^T \end{pmatrix} = O_{MN}.$$

This matrix equation can be treated in a row-by-row fashion. Denoting the—unknown—i-th row of A by the row vector α_i^T, and the—unknown—i-th row of B by the row vector β_i^T, this yields the following decoupled set of N linear systems of equations of size $M \times (N + P)$:

$$\begin{pmatrix} X^T & U^T \end{pmatrix} \begin{pmatrix} \alpha_i^T \\ \beta_i^T \end{pmatrix} = 0, \quad i = 1, 2, \ldots, M. \tag{5.2}$$

This is a linear system of equations of the form

$$D\chi = 0, \tag{5.3}$$

with

- $D = \begin{pmatrix} X^T & U^T \end{pmatrix}$, $D \in \mathbb{R}^{M \times (N+P)}$; and
- $\chi = \begin{pmatrix} \alpha_i, & \beta_i \end{pmatrix}^T$, $\chi \in \mathbb{R}^{N+P}$.

The objective now is to find sparse matrices A and B that satisfy this condition. However, there are two practical problems with the formulation above. First, as Equation 5.3 holds for each combined row χ of A and B, its solution also holds for all rows, and, therefore, A and B consist of dependent columns in the form $A_{i,j} = v_i \chi_j$, for some non-zero vector v. Second, and more seriously, the sparse matrices $A = B = 0$ are a trivial but valid solution to Equation 5.3. This can be overcome by introducing more information, such as the natural degradation of proteins, the autocatalyzation of certain proteins, or other empirically known biochemical characteristics. For instance, natural degradation or autocatalyzation stipulates that each protein decays or grows with a specific rate λ, and, therefore, the associated component of the gene/protein interaction matrix A has the value $\pm\lambda$. The known non-zero influence of an input j on a certain component i indicates that the $(i, j)-$component of matrix B is non-zero. Thus, in general, certain specified components of A and B have non-zero values.

Moreover, there is a scaling invariance in the solution—if χ is a solution of $D\chi = 0$, then $\lambda\chi$ is also a valid solution for all $\lambda \in \mathbb{R}$. The exclusion of $A = B = 0$ can thus be obtained by fixing the B-part of χ to a specific value, which expresses the value of A relative to B. This can be obtained by using constraints such as $\|B\|_1 = constant$, or $\sum_{ij} b_{ij} = constant$. In this study, this constraint is realized as

$$Q^T \chi = 1, \tag{5.4}$$

where the first N components, the "A-part" of Q, are zero, and the remaining P components q_i, $(i = N + 1..N + P)$, the "B-part", satisfy $|q_i| \in [\frac{1}{2}, 2]$. This vector was generated with an uniform distribution on $[-2, -\frac{1}{2}] \cup [\frac{1}{2}, 2]$ and $Q^T \in \mathbb{R}^{N+P}$.

5.2.2 Robust estimation as an approximation to the L_1-norm of A

Because, as mentioned in Chapter 3, optimal sparse solutions can be obtained by the technique of L_1-minimization, this problem can be reformulated as a linear programming problem. In our case, we can thus reformulate the original quest for a sparse vector χ (associated to corresponding rows in A

and B) as the problem of finding the minimum value of the L_1-norm of χ, i.e., $\|\chi\|_1$, under the given constraints. As we may weigh the importance of sparsity in A and B differently, we introduce a regularization parameter γ. If y_A denotes the first N components of χ, and y_B the other P components, the regularized minimum value of the L_1-norm of χ for the given data of M patterns for N components and P external inputs follows from the following linear programming formulation:

$$\chi_{\text{est}} = \arg\min_{\chi \in \mathbb{R}^{N+P}} \|y_A\|_1 + \gamma \|y_B\|_1 \tag{5.5}$$

subject to:

$$\begin{cases} \chi & = & y_A + y_B; \\ \chi & = & 0; \\ Q^T \chi & = & 1; \\ \chi_i & = & r_i, i = 1, \ldots, C, \end{cases}$$

where r_i represents a random non-zero constant. This defines LP1 as a regularized L_1-minimization problem over the affine subspace of vectors χ satisfying the given constraints.

5.3 Numerical experiments

The formulation in Equation 5.5 allows for numerical experiments with different settings of the model parameters X, U, M, N, and P.

The value of the number of patterns M varies between 0 and $N+P-1-C$, as will be shown later in this Section. Here, C is the number of extra constraints in LP1. The experiments were performed with a specific interest in partial sparse solutions for A and B. We introduce k_A as the number of non-zero elements in matrix A, and, similarly, k_B for B. Figure 5.1 shows two results from such numerical experiments. The patterns X and U were drawn uniform randomly from the interval [-1,1].

As Figures 5.1 and 5.2 clearly show, it was found that—for fixed regularization parameter $\gamma = 1$—the sparsity in A and B exhibits different behavior as the number of patterns crossed the value $M_{C1} \approx 0.5P$. Thus, a scale-free representation for studying the sparsity is obtained by defining:

- the relative sparsity in A: $p_A = k_A/N$;
- the relative sparsity in B: $p_B = k_B/P$;
- the scaled number of patterns, μ:

– if $M < M_{C1}$, then $\mu = M/M_{C1}$;

– if $M \geq M_{C1}$, then $\mu = (M + N - 1)/(2N + P - 2)$.

This scaling ensures that μ, p_A, and p_B are between 0 and 1. The scaled variables are useful in comparing situations with different components N and inputs P. Other values of the regularization parameter γ gave quantitatively different results that, however, could again be scaled to become qualitatively similar.

In these plots, the sparsity in the matrix is plotted versus the scaled number of patterns μ. The lower curve gives the relative non-zeros p_A in matrix A. It exhibits a clear transition at $\mu \approx 0.5$. The upper curve gives the relative non-zeros in matrix B. In the scaled representation, these plots are almost identical for different values of P and N, though the morphology somewhat depends on $N + P$.

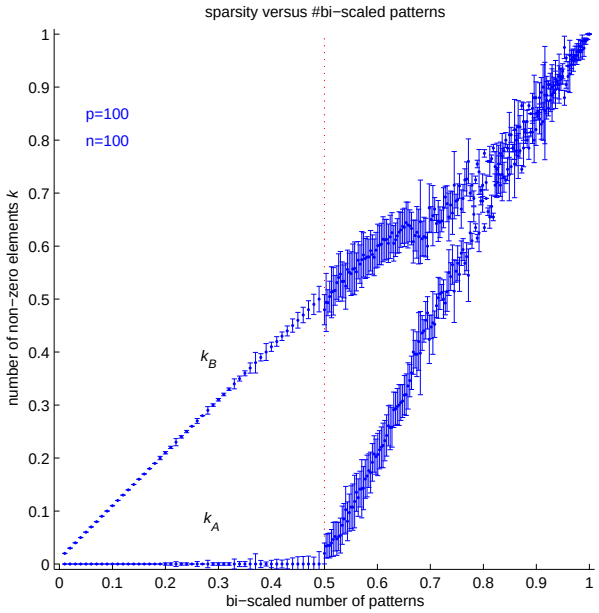

Figure 5.1: The scaled matrix sparsity versus the scaled number of patterns for $N = 100$ and $P = 100$. The solutions of the LP-approach to memory storage in linear networks, exhibit a clear phase transition at approximately $0.5P$ patterns.

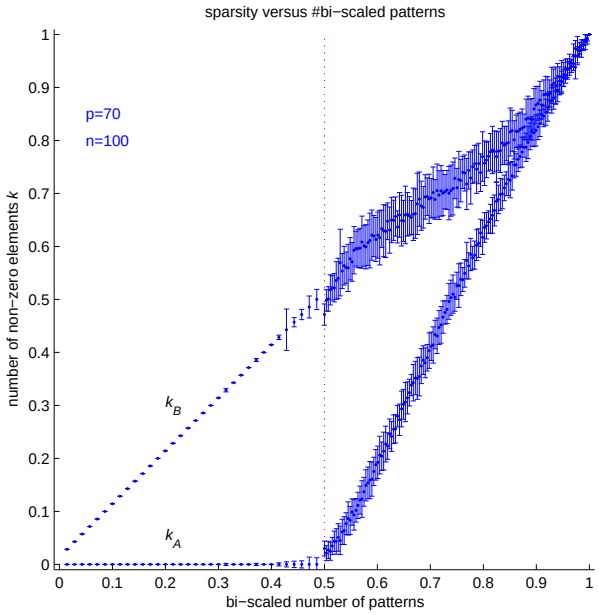

Figure 5.2: The scaled matrix sparsity versus the scaled number of patterns for $N = 70$ and $P = 100$. The solutions of the LP-approach to memory storage in linear networks, exhibit a clear phase transition at approximately $0.5P$ patterns.

5.3.1 Considerations from the underlying geometry

The observed relations between the parameters k_A, k_B, M, N, and P can be understood from an analysis of the underlying geometric constraints of the optimization process in $\mathbb{R}^{N+P}$. For an overview of the employed matrix algebra, consult [38]. The solution $\chi_{\text{est}} \in \mathbb{R}^{N+P}$ to the problem lies on the intersection between the two major constraints of the problem.

The first constraint is $D\chi = 0$, where D is the data matrix $\begin{pmatrix} X^T & U^T \end{pmatrix}$, and the first N components of χ correspond to a row of matrix A, and the last P components to the corresponding row of matrix B. Thus, the solution χ lies in the $(N + P - M)$-dimensional linear null-subspace, or version space, V_1 of D, because there are $N + P$ unknowns and M patterns.

Second, the solution χ is bound by the $(N + P - 1)$-dimensional affine space V_2 defined by $Q^T\chi = 1$, where the first N components of Q are zero, and the P remaining components $q_i, i = N + 1, .., N + P$, satisfy $|q_i| \in [\frac{1}{2}, 2]$.

The intersection $V_3 = V_1 \cap V_2$ contains the solution χ and V_3 is a $(N + P - M - 1)$-dimensional affine space. Therefore, the short answer to the maximum number M_{max} of patterns that can be stored is $M_{max} = N + P - 1$. Adding $C - 1$ extra constraints of the form $\chi_i = constant$ effectively lowers the dimension, but does not alter the basic topology, and leads to

$$M_{max} = N + P - C. \tag{5.6}$$

The situation becomes more complex if we analyze the number of exact zeros, $k_A + k_B$ in χ, as the optimization is aimed at increasing this amount. Here, k_A denotes the number of non-zeros in the first N components of χ, and k_B in its last P components. In this formalism, the maximum sparsity in A is obtained when $k_A = 0$. We avoid the complication that our model represents a situation without interactions, as the system output is entirely defined by the interactions to the external inputs. For the original equation,

$$\begin{pmatrix} X^T & U^T \end{pmatrix} \begin{pmatrix} \alpha_i^T \\ \beta_i^T \end{pmatrix} = 0, \tag{5.7}$$

the condition $k_A = 0$ means that the associated row of A is zero, i.e., $\alpha_i = 0_{1,N}$. Hence, $U^T \beta_i^T = 0$, meaning that β_i^T lies in the null-space of U^T. The condition $k_A = 0$ implies that $x_1 = x_2 = \ldots = x_N = 0$, which defines a P-dimensional linear subspace V_4 in $\mathbb{R}^{N+P}$.

The intersection $V_3 \cap V_4$ contains all fully sparse A-solutions that satisfy the two constraints. So, the condition $k_A = 0$ can be reached whilst $V_3 \cap V_4 \neq \emptyset$. The intersection $V_3 \cap V_4$ is defined by $N + M + 1$ equations with $N + P$ variables that imply an upper bound $M_A^{up} = P - 1$ for $k_A = 0$. However, as is apparent from Figure 5.3, k_A already abruptly starts deviating from zero at $M = M_C \approx 0.5P < M_A^{up}$. Although, in that case, $V_3 \cap V_4$ is not empty, the L_1-optimization process favors another solution with a smaller value of the L_1-norm of matrix A—but an unsolicited larger value for the number of non-zeros k_A. Finding a solution with $k_A = 0$ results in solving the linear equation $Z\chi = z$, with

$$Z = \begin{pmatrix} D \\ Q^T \\ I_N \;\; 0_{N,P} \end{pmatrix}; \quad z = \begin{pmatrix} 0_{M,1} \\ 1 \\ 0_{N,1} \end{pmatrix}. \tag{5.8}$$

This system has a consistent solution if z is a vector in the column space of Z. If this is not the case, the affine solution space S can be described implicitly as the solution of the consistent linear system of equations $Z\chi = z_{proj}$, where

the vector z_{proj} is the orthogonal projection of z onto the column space of Z. An explicit description of S follows from computing the solution space of the system $Z\chi = z_{proj}$, which can be achieved in a numerically reliable way by employing singular value decomposition. Thus, S can be formally written as

$$\hat{\chi} = W(Z)\xi + Z^{+}z, \tag{5.9}$$

where $W(Z)$ is the $N \times \dim(\text{null}(Z))$ matrix whose columns consist of the basis of the $\dim(\text{null}(Z))$-dimensional null-space of Z, and Z^{+} is the pseudo-inverse of Z. Here, ξ is a $\dim(\text{null}(Z))$-dimensional vector that parameterizes S. The support vector $Z^{+}z$ may indeed have a lower k_A value than the solution χ_{est} of the LP, as can be seen in Figure 5.3. This is an artifact of the L_1-minimization process that can favor a smaller value of the absolute value of a non-zero component rather than a smaller value of k_A. Hence, in general $criterion(\chi_{\text{est}}) = \|\alpha_{\text{est}}\|_1 + \gamma\|\beta_{\text{est}}\|_1 < \|\hat{\alpha}\|_1 + \gamma\|\hat{\beta}\|_1 = criterion(\hat{\chi})$.

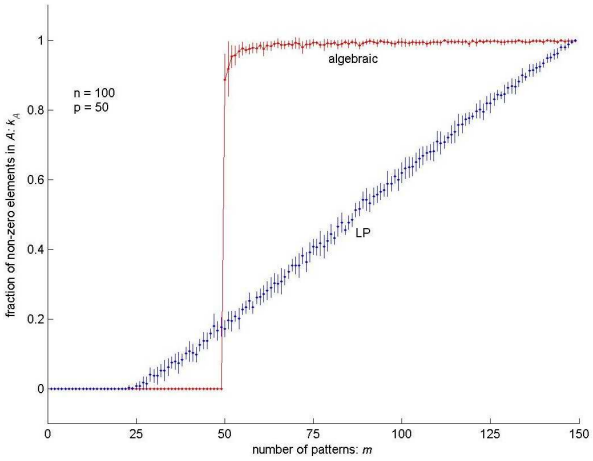

Figure 5.3: Comparison of LP-solution (diagonal line) and the algebraic solution (step function). M varies between 1 and $N + P - 1 = 149$. At $M = P - 1 = 49$ the algebraic solution suddenly jumps from 100% to 0% sparsity of A, while the LP-solution exhibits a 1st order phase transition at $M \approx 0.5P = 25$.

In the general case, when $k_A > 0$, the subset in V_3 that contains exactly k_A non-zeros in the first N and k_B non-zeros in the last P coordinates of χ is defined by V_3 , the $(N + P - k_A)$-dimensional linear subspace with k_A non-zeros in the first N coordinates of χ, and the $(N + P - k_B)$-dimensional linear subspace with k_B non-zeros in the last P coordinates of χ. There are

respectively

$$\binom{N}{N-k_A} \quad \text{and} \quad \binom{P}{P-k_B} \tag{5.10}$$

possible combinations of the form $\chi_i = 0$ to realize such equations. This defines $M + 1 + N - k_A + P - k_B$ equations with $N + P$ variables. In general this has a solution if

$$k_A + k_B \geq M + 1. \tag{5.11}$$

This relation is indeed found empirically as shown in Figure 5.4, where k_B/M is plot against k_A/M. The location of the critical value of the phase transition $M_c/P \approx 0.5$ is a function of the regularization parameter γ.

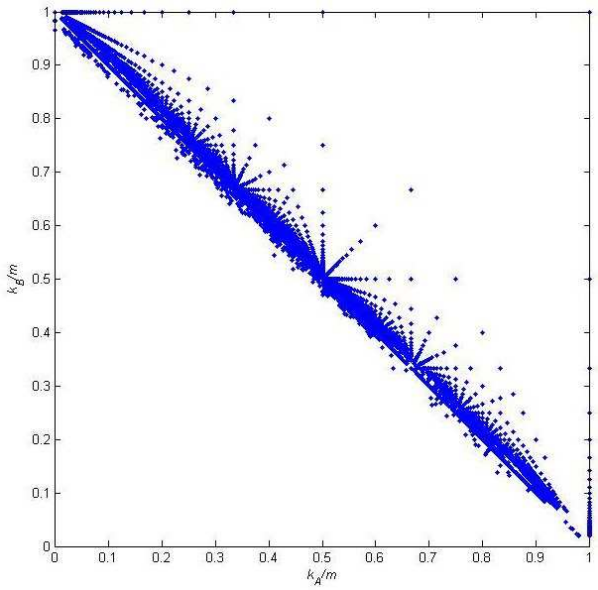

Figure 5.4: This plot exhibits the relation $k_B/M + k_A/M = 1$.

5.4 Phase transitions in the matrix sparsity define distinct learning strategies

Using the rescaled quantities p_A, p_B, and μ, different experimental settings can be straightforwardly compared. In Figure 5.5, the smoothed plot for the relative sparsities $p_A = k_A/N$ and $p_B = k_B/P$ versus the scaled number of

patterns μ is depicted for $N = 100$ components and $P = 100$ inputs, averaged over 10000 measurements. Again, all results are shown for fixed regularization parameter $\gamma = 1$.

This plot exhibits the typical characteristics representative for all the networks in our context. Firstly, the behavior of p_A for $\mu_{C1} \approx 0.5$ changes abruptly from constant zero to a near linear increase. Secondly, the characteristic plateau visible in p_B for $\mu_{C2} \approx 0.67$ is not an artefact due to noisy measurements or processing deficiencies, but is present in all such graphs. Next, inspection of Figure 5.5 and numerical analysis show that both graphs converge to a line $L \approx 0.67\mu - 0.5$. Finally, for $\mu > 1$, the system of (independent) inequalities becomes inconsistent and, therefore, has no solution. This graph therefore

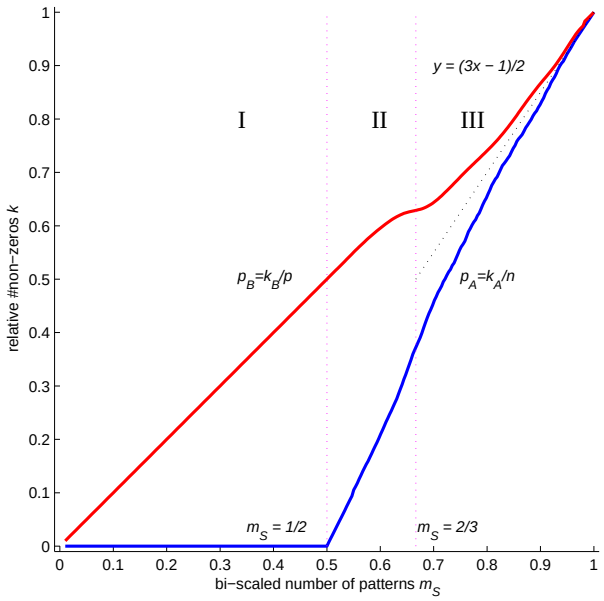

Figure 5.5: The connectivity probabilities $p_A = k_A/N$ and $p_B = k_B/P$ for $N = 100$, $P = 100$, and $C = 0$, averaged over 10000 measurements, versus the scaled number of patterns μ. This indicates three major regions I, II, and III, where the network memory behaves different. The transitions occur for the scaled $\mu \approx 0.5$ for the transition I to II, and $\mu \approx 0.67$ for II to III. In the final phase of region III both probabilities converge to $L \approx 0.67\mu - 0.5$.

shows three different types of storing behavior separated by phase transitions. The presence of phase transitions in the storage of information in linear ran-

dom sparse networks is reminiscent of the situation for the propagation of information through these networks.

It is well known that sparse networks under certain conditions exhibit phase transitions. Starting from a sparse regular network, the gradual addition of random links reduces the direct path between any pair of vertices in the network from being very long to being very short. This change is achieved abruptly as a phase transition to a "small-world" network, characterized by short overall path lengths, small overall connectivity, high information processing time, and high local clustering (see Watts and Strogatz [24], Barabasi et al. [10], Newman [56], and Schäfer [68]). This phase transition occurs at a critical threshold p_C for the probability of rewiring a given connection in the network. The system then changes abruptly from slow to fast information processing.

The numbers k_A and k_B of non-zero elements in our random networks can be related to this rewiring probability, as the probability that a connection between two components exists equals $p_A = k_A/N$, and the probability that a connection between a component and a certain input exists equals $p_B = k_B/P$.

5.4.1 Influence of the extra constraint

In the LP, i.e., Equation 5.5, explicit constraints of the form $\chi_i = constant(i)$, $i = 1, 2, \ldots, C$, were added to avoid the matrices A and B from becoming equal to zero. This addition, however, has little impact on the behavior of the system.

With an increasing number C of such constraints, the phase transition at M_c disappears, until at $C = N^2 + NP$ all values of A and B are specified by the constraints. Also, with an increasing number of constraints C, the morphology to the right of M_c gradually flattens, causing the disappearance of the transition II/III, as is evident in Figure 5.6, containing $C = 10$ constraints. For $C = 0$ and μ below μ_{C1}, the solution to the LP yields the matrix $A = 0$ as solution. Thus, besides the desired patterns $(\chi_{\text{est}}, u_{\text{est}})$, there are many other unsolicited equilibria, as all (y, u_{est}) are also valid solutions, because $A\chi_{\text{est}} = Ay = 0$. This condition allows all possible values for the unknown components, because the input u_{est} is directly coupled the desired equilibrium χ_{est}. Biologically, this means that, here, the genes are simply not involved, and the system is a chemical buffer that has evolutionarily learned to counter the important stimulus "impulsively".

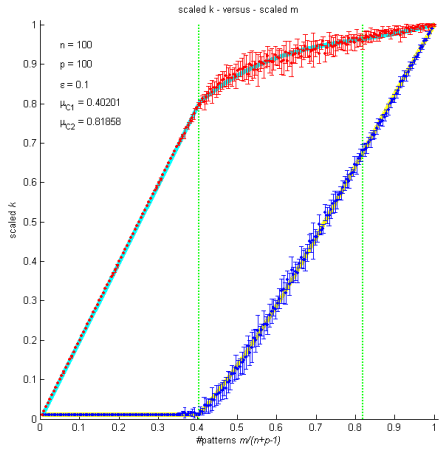

Figure 5.6: The connectivity probabilities k_A/N and k_B/P for $N = 100$ and $P = 100$, with $C = 10$ extra constraints of the form $\chi_i = constant$. This also exhibits the strong phase transition at M_c. Right to this value, the morphology however is flatter than for less extra constraints.

However, to the right of the critical point μ_{C1}, an increasing number of components of A and B become non-zero. Therefore it is *not* necessary to impose the extra constraints on the LP, because it does not change the behavior of p_A and p_B below μ_{C1} (except for a vertical translation), and, above μ_{C1}, there are sufficient non-zeros in both schemes anyway.

Biologically, the genes and proteins are involved in selecting the equilibrium. The network processes the input information, and then "decides"—computes—the best system response χ_{est}.

5.5 Stability of the stored patterns

To study the stability of the patterns, a set of $M = 64$ linearly independent patterns was created, and offered to a system with $N = 100$ components and $P = 6$ inputs, so the vector χ contains 106 elements. In addition, $C = 1$ extra constraint was used, i.e., $\chi_1 \leq -1/3$. The associated input u_i consists of the 6-bit coding of the index-number $i \in \{1, \ldots, 64 = 2^6\}$ of the pattern M_i.

Numerical experiments were performed such that a given input u_i was offered to the system in a random equilibrium state χ_0. The subsequent convergence

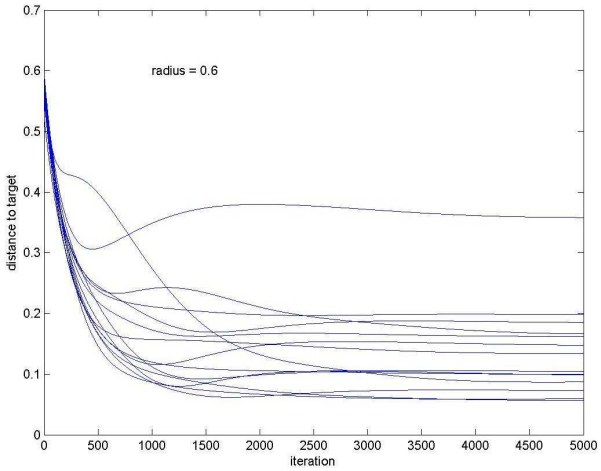

Figure 5.7: Dynamical behavior of a network which has "learned" to respond with a system state χ_1 to an input u_1. This plot shows the evolution of the distance of the actual system state $\chi(t)$ to the target system state χ_1, when an input u is created a distance r away from the input u_1. In one case the system apparently converges to a mixed pattern between the target patterns.

relative to the pattern χ_i is studied as a function of the distance between χ_0 and χ_i. This is performed by computing the evolution of $\chi(t)$ using $\dot{\chi} = A\chi + Bu_i$ and $\chi(0) = \chi_0$. Numerical analysis shows that A only has negative eigenvalues. However, the largest eigenvalue is just below zero, i.e., $\max(\text{eigenvalues}(A)) \approx 10^{-11}$. Consequently, the state vector $\chi(t)$ will converge, but not necessarily to the pattern χ_i associated to the input u_i. If the distance between χ_0 and χ_i becomes too large, namely in the order of the distance between the patterns themselves, it is likely that the state vector converges to a mixed state somewhere between the patterns. Figure 5.7 indicates that the end result, when offered input u_2 to a system in equilibrium $\chi = \chi_1$, may converge to a state other than the target pattern χ_2. Obviously, it has converged to a mixed state somewhere in between pattern χ_1 and pattern χ_2. It also shows the evolution of the distance to pattern χ_1 if a random pattern is created within an Euclidean distance $r = 0.6$ from pattern χ_1. Furthermore, Figure 5.7 shows that most random patterns converge to a state near pattern χ_1, so close that it is practically identical.

5.6 Conclusions

In this Chapter, we discussed the characteristics of the (genetic/proteomic) memory of simple linear and sparse random networks. It is found that a linear time-invariant state space formulation provides a comprehensible and transparent model that is accessible to numerical and mathematical analysis. Even this simple model exhibits a range of complex behaviors for storing input-output patterns in a sparse representation. Most important is the fact that the storage of both dynamical and static patterns in sparse networks exhibit distinct phase transitions. In the case of static input-output memory, there are two second-order continuous phase transitions. These transitions occur for a number of patterns that equals $M_{C1} \approx 0.5P$ and $M_{C2} \approx 0.3N + 0.67P - 0.3$, respectively, or in scaled coordinates, for $\mu_{C1} \approx 0.5$ and $\mu_{C2} \approx 0.67$, respectively. These transitions divide the learning in three distinct regions.

In the first region, where $M < M_{C1}$, all information is exclusively stored in the input-output matrix B, so that this state represents a high degree of order. In the second region, $M_{C1} < M < M_{C2}$, the information is stored increasingly in the interaction matrix A. Finally, in the third region, for $M > M_{C2}$, the information is stored evenly in both matrices.

The parameters N, P, and γ influence the behavior of the system to a different extent. The model is symmetrical in χ and u, as $A\chi + Bu = 0$, and the criterion function is $\|A\|_1 + \|B\|_1$, as the regularization parameter γ was set to 1. However, N and P have basically different influences on the behavior. Using the rescaled quantities p_A, p_B, and μ, it was possible to obtain an almost scale-free representation of the graphs of resulting sparsity versus number of stored patterns. Below the first critical point M_{C1}, the relation is essentially scale-free. However, above this value the morphology of the graph depends on N and P, as can be seen in Figure 5.1. Especially the plateau near the second critical point μ_{C2} becomes better visible for higher P. Moreover, the first critical point itself is found to occur at approximately $0.5P$, and is therefore independent of N.

In this study, we considered only linearly independent patterns. However, real gene-protein networks are often considered to be modular. In this case, different sub-patterns may combine to give the global pattern. Therefore, the global patterns are not necessarily linearly independent, but can be constructed from a set of linearly independent sub-patterns.

As a final point, the stability of the stored patterns was examined. It was found that the patterns are stable, each pattern having a well-defined basin of attraction. However, the sizes of these basins vary with the number of stored patterns, which is intuitively clear as the available space remains the same, so the space-per-patterns decreases. However, even close to the patterns, there are paths to mixed states in which some patterns are "confused".

These simple experiments hint to some characteristics to be expected for memory storage in real-world networks, i.e., the gene-protein interaction network. Firstly, memory storage can be understood as fixing the free model parameters such that a presented input matches with the desired output. In the natural world, this learning process is performed by the actions of biological evolution. Secondly, also in real biological information networks, it is possible to directly couple the input to the output without altering the gene-protein interactions. This creates an "instinctive" reaction to the presented input, that only indirectly alters the states of the genes and proteins. This also includes epigenetic memory storage, as it can be understood as a specific state vector of the proteome without regard to the states of the genome. Thirdly, realistic networks potentially may exhibit phase transitions as the number of stored patterns grows. These transitions can divide the storage process in distinct regions with their own learning strategies. Finally, even in real biological networks, this has led to predominantly stable patterns, just as in the context of the Single Linear Networks studied here.

6

An example application

In this Chapter, we will apply our identification approach on a real dataset stemming from Spellman et al. [70]. Yeast *Saccharomyces cerevisiae* is an unicellular fungus found naturally in grapevines and is responsible of wine-making fermenting sugars and producing alcohol. This data is available at `http://cellcycle-www.stanford.edu`. From being budded off from its parent cell, to reproducing its own offspring, each yeast cell goes through a number of typical steps that also involve changes in gene expression, turning complete pathways on and off. Today, the study of such phenomena is possible thanks to the technology of microarrays that can measure the expression level of many genes simultaniously. For this data, the expressions profile of about 6000 genes are measured by microarrays at 24 time points every five hours.

In this approach a number of preprocessing steps are usual in litirature [40]. In the reconstruction of the yeast regulatory network, the first step is the preprocessing of the microarray data in Section 6.1. The second step is to group genes that behave similarly within the cell cycle. Therefore, the K-means clustering method is proposed in Section 6.2. The third and last step is the identification. In Section 6.3, the Single Linear Model and the Piecewise Linear Model will be applied in order to find the gene interactions involved in the yeast cell cycle.

6.1 Preprocessing

As it happens frequently, the original matrices, resulting from the microarrays, have various entries with missing values. This is because of the measurement error or the construction error of the array. Furthermore, due to the high number of unknowns, we are not interested in gene expressions with low variability and low amplitude.

6.1.1 Missing values

The dataset contains numerous genes with one or more missing values. There are several different methods that deal with these missing values, which have been studied by Troyanskaya et al. [76]. For example, an unreliable way to deal with this problem is to replace missing values with a zero or with the average expression value for that gene. The latter method, called the row average method, assumes that the expression of a gene in one of the experiments is similar to its expression for other experiments, but this is often not true for microarray data. Therefore, this naive method is disregarded. In our approach, genes with at least one missing value in their expression profiles are removed permanently from the dataset. In this way, almost 2000 genes are deleted.

6.1.2 Filtering to increase variance

Next, several filtering steps, based on the study of [21], were performed. Analysis of the data has revealed that some expression profiles are flat and not significantly different from the others. This flat data indicates that the genes associated with these profiles are not significantly affected by others. We are interested in the genes with large changes in expression. Therefore, genes with small variance over time are removed. More specificly, genes that have an absolute expression value below 1.5, i.e., genes for which there is no expression level higher than 1.5 or lower then -1.5, are thus removed. Furthermore, expression profiles with an amplitude in the lowest 20 percent are filtered. After these filtering steps, 290 gene expressions remain.

6.2 Clustering

The goal of clustering is to find a partitioning $C = \{C_1, \ldots, C_K\}$ of objects $x_1, \ldots, x_N$ into a number of disjoint groups, K, such that objects in one group are similar to each other and are as different as possible from the objects in the other groups.

In this context, the objects are the vectors $x_i = x_i[1], \ldots, x_i[M]$, i.e., object i is the measured gene expression profile of gene i. Clustering can then be used to find the so-called co-regulated genes. Co-regulated genes are genes with similar expression profiles, for which the up- and down regulation is very similar. When the most suitable clusters are found, for each cluster one object (gene) is chosen out of each cluster which will represent that whole cluster. These genes are called the cluster representatives. Once found, the next step in the construction of the genetic regulatory network will only depend on these genes.

In the literature [12], some cluster methods are available. The wide variety in clustering algorithms is due to the fact that there is no best clustering algorithm for a given dataset or for a specific task.

The distances between the clusters can be defined in a variety of ways. Each distance measurement leads to different shapes of clusters.

6.2.1 K-means clustering

The K-means algorithm is an iterative clustering algorithm. The idea of this algorithm is to start with some initial clustering of the objects and then to reassign objects to clusters such that the score function has a better value. This process is repeated until no improvement in the score function or no change in the cluster memberships occurs. This process leads at least to a local optimum. If one wants to use the K-means algorithm, the number of clusters K should be known in advance. The basic algorithm begins by randomly selecting K centroids and then it assigns each object to the cluster with the centroid that is the most similar to the object. Next, the centroids are recalculated as the mean vector of the objects in the clusters. This process is repeated until no changes in the cluster memberships occur.

6.2.2 Selecting the number of clusters

In [44], a clear image has been obtained concerning the number of clusters that is suitable for the K-means clustering method. They have found that $K \in [8, 12]$ for the data set of yeast.

For our approach, the number of clusters is increased to 16, which is still a representative number. The larger K, the better the generalization error, ε_{gen} is (see Figures 4.1, 4.2, 4.14, and 4.15). However, K is also related to the limited number of observations $M = 24$ which needs to be split into

a training set and a validation set. The smaller K, the smaller the size of the training set and, thus, the larger the validation set is. The result of the clustering is shown in Figures 6.1 and 6.2.

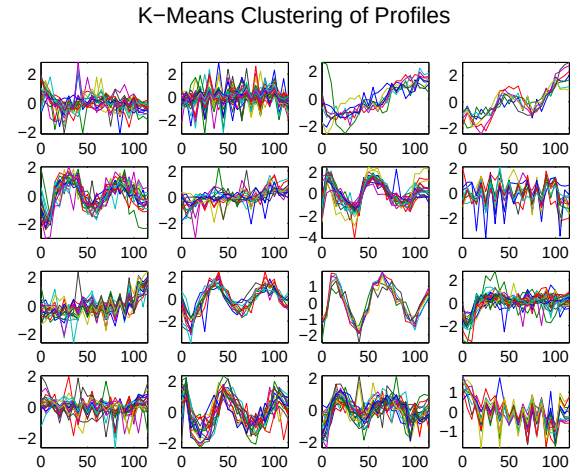

Figure 6.1: The gene expressions per cluster, for K-means with 16 clusters.

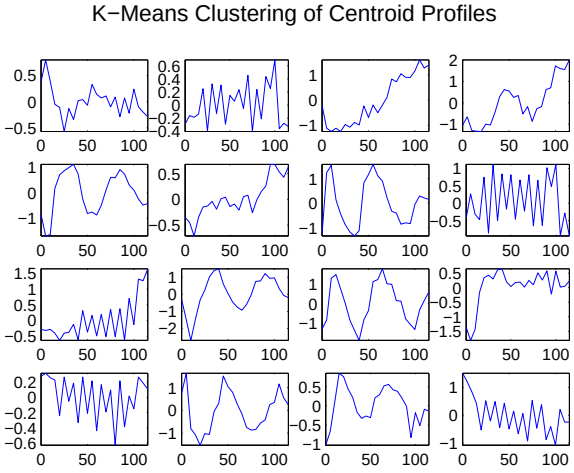

Figure 6.2: The gene representatives per cluster, for K-means with 16 clusters.

6.3 Network reconstruction

After the pre-processing and the clustering steps, the yeast data can be represented by 16 clusters. Each cluster has a centroid that represents the mean of the expression levels of the genes for that cluster. These centroids are called the genes' representatives and, thus, we reduced the problem to a data set of $16 \times 16 + 16$ unknown components and 16×24 measurements or observations. In this Section, we will apply the Single Linear Model and the Piecewise Linear Model in the context of learning as in Chapter 4. To learn, we need a training set of T input/output pairs $\chi_{\text{tr}} = \{(x_t, u_t, \dot{x}_t) \mid 1 \leq t \leq T\}$ and a validation set $\chi_{\text{v}} = \{(x_v, u_v, \dot{x}_v) \mid 1 \leq v \leq V\}$.

For this application, the data matrix, $X \in \mathbb{R}^{N \times M}$, represents the expression levels of the $N = 16$ gene representatives and the output matrix, $\dot{X} \in \mathbb{R}^{N \times M}$, is estimated by interpolating these gene expressions over the 24 time points. Note that $U = 1^{P \times M}$, with $U \in \mathbb{R}^{P \times M}$, because there is no information known about the external stimuli.

6.3.1 The Single Linear Model

The Single Linear Model was introduced and studied in Section 3.2.1. For this model, the entire state-space is considered as one large parameter space and the interactions are modeled by a linear state-space system of the form

$$\dot{X} = AX + BU + \xi,$$

with A and B the unknown system matrices and ξ stochastic Gaussian white noise.

From biology, it is known that gene-protein interaction networks are sparse [85] and so is the yeast network. As discussed before, Linear Programming (LP) is potentially a good approach to tackle sparsity. Thus, to identify the network, the Single Linear Model uses the technique of L_1-minimization, where the sparsity criteria is used as a constraint:

$$\begin{aligned} \min_{A \in \mathbb{R}^{(N \times N)}} \quad & \|A\|_1 \\ \text{s.t.} \quad & AX + BU = \dot{X}. \end{aligned}$$

Training set versus validation set

From this yeast data set, only $M = 23$ time step observations are available. Note that one observation is lost due to the estimation of the output matrix, $\dot{X}$. To train the data, a minimal number of observations is necessary, but to validate the result, some observations are needed too. Therefore, we first focus on the partition of these 23 observations.

How the system scales with the system size was studied and illustrated in Section 4.1. Therefore, the generalization error is introduced as the probability that the student will not compute the correct output on a random input. In Section 4.1, it turns out that the generalization error for various system sizes can be computed by applying the correct scaling on the system size $\alpha = M/N$:

$$\alpha(N) = \alpha^{\text{fs}}_{\text{gen}} + \sqrt{N_0/N}\,(\alpha(N_0) - \alpha^{\text{fs}}_{\text{gen}}). \tag{6.1}$$

The generalization error for feature selection—for one row of the input data X and $\dot{X}$— versus α for system size $N_0 = 80$ is represented in Figure 4.2. Using the information of this Figure and of Equation 6.1, the generalization error for feature selection, $\varepsilon^{\text{fs}}_{\text{gen}}$, for system size $N = 16$ can be computed.

If, e.g., $\varepsilon^{\text{fs}}_{\text{gen}} = 0.1$, then $\alpha(80) = 0.22$. Furthermore, the scaling factor equals $\sqrt{80/16}$, thus, $\alpha(16) = 0.49$. This means that, for $M = \alpha N \approx 7$ observations, the interactions can be identified with 90 percent reliability. For $M = 5$, $M = 6$, and $M = 8$, there is a reliability of 35, 50, and 95 percent, respectively.

Numerical experiments

The training set contains M_{tr} randomly selected observations, such that $\chi_{\text{tr}} = \{(x_t, u_t, \dot{x}_t) \mid 1 \leq t \leq M_{\text{tr}}\}$. For this number of observations, the Single Linear Method is able to return a pair of system matrices, $(A_{\text{est}}, B_{\text{est}})$, that produces the training set χ_{tr}: $\dot{x}_t = A_{\text{est}}x_t + B_{\text{est}}u_t$, for $t = 1, \ldots, M_{\text{tr}}$. The data error, D_{e}, is defined to measure the quality of the system matrices. For a training set size $M_{\text{tr}} = 15$, $D_{\text{e}} = |A_{\text{est}}x_t + B_{\text{est}} - \dot{x}_t|/|\dot{x}_t| = 1.43 \times 10^{-9}$. Here, $M_{\text{tr}} = 15$, which is the maximum number of observations such that $M < N$. For, $M_{\text{tr}} = 15$ the generalization error is zero, hence, $M_{\text{tr}} = 15$ has a reliablility of 100 percent. Thus, if the approach does not perform well, it can not be improved by changing the value of M_{tr}.

Note that, in this setting, we do not have an original system, known as the teacher, to compare the resulting system with.

Of course, the resulting system, $(A_{\text{est}}, B_{\text{est}})$, should also perform well on a data set that has not been used by the algorithm. So, to test the systems generalization ability, we use the validation set that contains the remaining $M_{\text{v}} = M - M_{\text{tr}} = 8$ observations, $\chi_{\text{v}} = \{(x_v, u_v, \dot{x}_v) \mid 1 \leq v \leq V\}$. Again, the data error is used to show the generalization quality of $(A_{\text{est}}, B_{\text{est}})$.

v	$D_{\text{e}} = \lvert A_{\text{est}} x_v + B_{\text{est}} - \dot{x}_v \rvert / \lvert \dot{x}_v \rvert$
1	1.79
2	1.97
3	1.25
4	2.11
5	1.54
6	1.34
7	1.29
8	0.86

Table 6.1: The generalization quality of $(A_{\text{est}}, B_{\text{est}})$.

Table 6.1 illustrates that the resulting system $(A_{\text{est}}, B_{\text{est}})$ does not perform well on the validation set. This is to be expected for two reasons. First, the approach only performs well on linear systems, but, here, the gene expression levels are inherently non-linear (see Figures 6.1 and 6.2). Therefore, the system will be split into subsystems, so the Piecewise Linear Model will be used (see Section 6.3.2). And, second, because of the microarray technique, we can expect the yeast data to be noisy. However, as concluded in Section 3.2.1, the Single Linear Model is not capable to deal with this noise. In Section 6.3.3, we try to find a sparse result with respect to noise.

6.3.2 The Piecewise Linear Model

For the Piecewise Linear Model, the entire state-space is partitioned into subsystems. The general form for this model is

$$\dot{X} = H_1 X + H_2 U, \tag{6.2}$$

where H_1 and H_2 are related to the weight matrix and the system matrices $(A_{\text{est},\ell}, B_{\text{est},\ell})$. The weight matrix, $W \in \mathbb{R}^{M \times K}$, with K the number of sub-

systems, represents the membership functions of observation m to subsystem ℓ. For more details, see Section 3.2.2.

To partition the entire system into subsystems, the weight matrix has to be initialized by using the matching algorithm. For this algorithm, the number of subsystems does not have to be known in advance. When the switching points for each subsystem are known, the Single Linear Model can be used to identify each subsystem.

Numerical experiments

The training set contains $M_{\text{tr}} = 18$ randomly chosen observations, such that $\chi_{\text{tr}} = \{(x_t, u_t, \dot{x}_t) \mid 1 \leq t \leq M_{\text{tr}}\}$ and still enough observations are left over to validate the resulting subsystems. To split the system into subsystems, the weight matrix, W, has to be initialized from this training set. Therefore, the minimal number of required observations, M_{min}, has to be known. From Section 4.1 and Section 6.3.1, we know how the system scales with the system size and thus M_{min} can be estimated.

The first M_{min} observations are assigned to the first subsystem. Once an assignment of M_{min} observations to a subsystem is made, the next observation $M_{min} + 1$ will be considered. To decide whether an observation, $\mu_k = (x[k], u[k], \dot{x}[k])$, belongs to a subsystem ℓ, the distance between the observation and the current subsystem, $d(\mu_k, \ell)$, must be computed. If the distance to the current subsystem is sufficiently small, this observation will be added to the current subsystem, otherwise, the next M_{min} observations, starting from μ_k, will be added to the next subsystem $\ell + 1$. This process is repeated until all observations belong to a subsystem. This procedure will be refered to as the matching algorithm. For more details, see Section 3.2.2.

Let us assume that $M_{min} \in \{5, 6, 7, 8\}$. After a run, the corresponding *possible* weight matrices are respectively as in Figure 6.3.

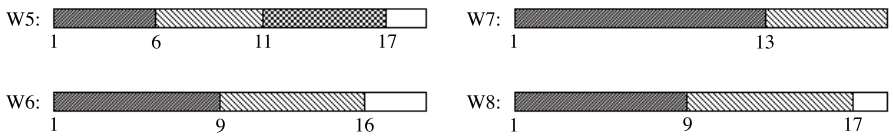

Figure 6.3: Split of the subsystems for $M_{min} \in \{5, 6, 7, 8\}$.

So, the matching algorithm for $M_{min} = 5$ splits the system into 3 subsystems, $M_{min} = 6$, $M_{min} = 7$, and $M_{min} = 8$ are split all into 2 subsystems, see Figure 6.3.

After the initialization of the weight matrix, the Single Linear Model will be used to identify all subsystem matrices $(A_{\text{est},\ell}, B_{\text{est},\ell})$. We measure the quality of the identification from the validation set χ_{v} by

$$D_{\text{e},\ell} = \frac{|A_{\text{est},\ell} x_{v,\ell} + B_{\text{est},\ell} - \dot{x}_{v,\ell}|}{|\dot{x}_{v,\ell}|}.$$

Note that to compute the data error for each subsystem, $D_{\text{e},\ell}$, the validation vectors for that specific subsystem $x_{v,\ell}$ and $\dot{x}_{v,\ell}$ are needed . Table 6.2 shows the data error, for each specific subsystem, and this for $M_{min} \in \{5, 6, 7, 8\}$. Table 6.2 illustrates that, even after splitting into subsystems, the resulting system matrices $(A_{\text{est},\ell}, B_{\text{est},\ell})$, for $\ell = 1 \ldots K$, do not perform well on the validation set. This is to be expected, because the data is still strongly noisy, and, as studied in Section 3.2.2, this Piecewise Linear Model is not capable to deal with noisy data sets.

M_{min}	subsystem 1	subsystem 2	subsystem 3
5	1.05	0.79	2.14
6	1.22	0.77	/
7	0.98	1.04	/
8	1.25	0.92	/

Table 6.2: Data error per subsystem for $M_{min} = 5$, $M_{min} = 6$, $M_{min} = 7$, and $M_{min} = 8$.

In Table 6.3, the sparsity, i.e., the number of non-zeros per row of $A_{\text{est},\ell}$, for each subsystem and $M_{min} \in \{5, 6, 7, 8\}$, is shown. From Table 6.3, it is clear that the resulting system matrices $A_{\text{est},\ell}$, for $\ell = 1 \ldots K$, are relatively sparse. Here we employ our earlier definition of sparsity, namely the number of non-zeros per row of the connection matrix. A sparsity of 4, which yields the best result, means that, on average, 1/4th of the unknown gene representatives are non-zero. This is an acceptable result, because a large number of genes, which are not significantly affected by others, were removed in the filtering step. And the remaining non-affected gene expressions were clustered into the same cluster.

M_{min}	subsystem 1	subsystem 2	subsystem 3
5	4	4	5
6	7	6	/
7	11	5	/
8	7	7	/

Table 6.3: Sparsity per subsystem for $M_{min} = 5$, $M_{min} = 6$, $M_{min} = 7$, and $M_{min} = 8$.

6.3.3 Dealing with noise

The data available for the identification and reconstruction of the yeast cell cycle is a result of microarrays taken at 24 time points. To obtain each microarray, a complex biological process is executed. Therefore, it can be expected that the data contains measurement noise. Besides this measurement noise, the system can contain also system noise, noise which occurs in and between the cells.

As studied in Sections 3.2.1 and 3.2.2, and the experiments above, the Single Linear Model and the Piecewise Linear Model in the current form cannot deal with noisy data. Therefore, in Section 4.2, the original L_1-minimization is adapted and a regularization parameter is introduced:

$$\min \quad \lambda\|A\|_1 + \|\dot{X}_\xi - AX - BU\|_1 \tag{6.3}$$

The regularization parameter, λ, imposes certain distributions on the optimization terms $\|A\|_1$ and $\|\dot{X}_\xi - AX - BU\|_1$. The first term, $\|A\|_1$, represents the sparsity constraint, and the second term, $\|\dot{X}_\xi - AX - BU\|_1$, measures the quality of the fit. As discussed in Section 4.2, $\|\dot{X}_\xi - AX - BU\|_1 \approx 0$ to have an acceptable fit of the original system.

Numerical experiments

From Section 4.2, it is known that a larger training set size is needed to obtain an acceptable fit of the original system. Therefore, the experiments are performed on a training set of size $M_{\text{tr}} = 21$. Note that 21 observations are the absolute maximum, because only 2 observation are left over for the validation set ($M_{\text{v}} = 2$). To split into subsystems, we perform runs for $M_{min} = 5, \ldots, M_{min} = 10$.

For this LP, the regularization parameter, λ, needs to be identified to balance between the sparsity and the robustness of the fit. Notice that λ depends on the number of unknowns and the number of observations. Applying the study introduced in Section 4.2, $\lambda_{min} \leq \lambda \leq \lambda_{max}$ for $N = 16$, the unknown components can be estimated. To fit each subsystem, λ for $M = M_{min}, \ldots, M_{\mathrm{tr}} - M_{min}$ must be computed. The results are shown in Table 6.4.

M	λ_{min}	λ_{max}
5	1.4	3.2
6	1.3	4.1
7	1.8	4.5
8	1.2	5.0
9	1.5	5.1
10	3.4	6.1
11	3.3	7.1
12	3.9	6.9
13	5.0	7.7

Table 6.4: Representation of λ_{min} and λ_{max} for $N = 16$ and $M = 5, \ldots, 13$.

After performing the matching algorithm, $M_{min} = 5$, $M_{min} = 6$ and, $M_{min} = 7$ are split into three subsystems. Hence, at least one subsystem does not have a corresponding validation vector. $M_{min} = 8$, $M_{min} = 9$ and, $M_{min} = 10$ are split into two subsystems.

To represent the performance quality of the Piecewise Linear Model for this LP, the data error for each subsystem, $D_{\mathrm{e},\ell}$, is defined as before and the results are shown in Table 6.5. For each subsystem, it is shown that the data error is somewhat lower than for the Piecewise Linear Model without the regularization parameter. Figures 6.4 and 6.5 represent the robustness of the fit for the model with $M_{min} = 7$, which results into 3 subsystems. Of course, we are also interested in the sparsities of the systems which are represented in Table 6.6. It is clear that by introducing the regularization parameter, the systems have a better sparsity.

M_{min}	subsystem 1	subsystem 2	subsystem 3
5	0.70	/	0.51
6	0.84	0.69	/
7	0.73	/	1.32
8	0.61	0.98	/
9	0.59	0.97	/
10	0.83	1.02	/

Table 6.5: The data error for each subsystem for $M_{min} = 5, \ldots, M_{min} = 10$ and a training set size of $M_{\mathrm{tr}} = 21$.

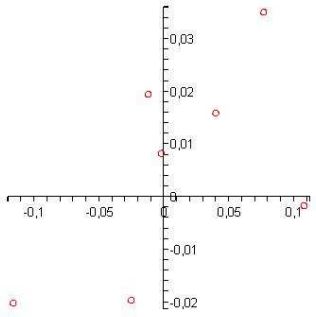

Figure 6.4: $\dot{X}$ vs. $\dot{X}_{est}$ for $M_{min} = 7$, subsystem 1.

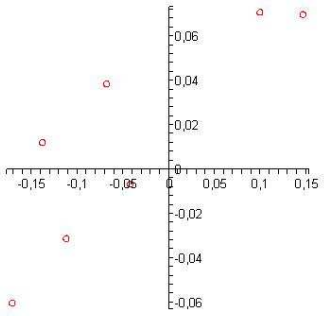

Figure 6.5: $\dot{X}$ vs. $\dot{X}_{est}$ for $M_{min} = 7$, subsystem 3.

M_{min}	subsystem 1	subsystem 2	subsystem 3
5	2.88	/	1.13
6	1.44	2.68	/
7	1.63	/	0.94
8	1.75	1.62	/
9	1.18	2.18	/
10	0.62	1.12	/

Table 6.6: Sparsity per subsystem for $M_{min} = 5, \ldots, M_{min} = 10$ and a training set size of $M_{\mathrm{tr}} = 21$.

6.4 Conclusions

Data sets for real-world networks, e.g., the metabolic network, and the World Wide Web, consist of a large number of nodes or components. In practice, e.g., due to the high costs of microarray hardware and the unregulated growth of the number of web pages, there are only few reliable real-world data sets available for our framework.

In this Chapter, we considered the data set of yeast *Saccharomyces cerevisiae* consisting of about 6400 genes and 24 time steps of genome-wide measurements. We found that there are various gene expressions with missing values and that some others are not interesting, because of their low variability and low amplitude. These genes are removed from the data set. From the 6400 original genes, some 4000 were left over. Obviously, 24 measurements for 4000 genes are insufficient. Therefore, the gene expressions were clustered in such a way that genes with the same behavior are represented by one gene representative. In this study, K-means clustering was found to be optimal for 16 clusters. As a result, we obtained a data set of 16 gene representatives and 24 observations which have been used for the identification process.

For identification purposes, a training set of M_{tr} random selected observations was defined, and to measure the quality of the resulting fit, a validation set of $M_{\mathrm{v}} = M - M_{\mathrm{tr}}$ remaining observations was defined.

First, the Single Linear Method was applied with a training set size $M_{\mathrm{tr}} = 15$, which is the maximal number of observations such that the condition $M < N$ still holds. From Section 6.3.1, we concluded that the resulting system $(A_{\mathrm{est}}, B_{\mathrm{est}})$ does not perform well on a data set that has not been used by the algorithm. On the one hand, this result is due to the non-linearity of the

original data. To deal with this non-linearity, the data was split into linear parts. On the other hand, the result was negatively influenced by the noise level of the original data. In Section 3.2.1, it was shown that the Single Linear Model cannot deal with noisy data.

Second, the Piecewise Linear Model was applied. With this model, the data was split into parts and each subsystem was identified by the Single Linear Model. For all the experiments, the training set size $M_{\text{tr}} = 18$ and, thus, a validation set of $M_{\text{v}} = M - M_{\text{tr}} = 5$ observations was left over. Splitting the system depended on the minimal number of required observations, M_{min} and the generalization ability of each subsystem was measured by the data error. From Section 6.3.2, it is clear that the resulting system $(A_{\text{est},\ell}, B_{\text{est},\ell})$, with $\ell = 1 \dots K$, is sparse, but it does not generalize beyond the training set. For this Piecewise Linear Model, it was shown, in Section 3.2.2, that it is not capable to deal with noise either.

Third, to fit the system with respect to noise, a regularization parameter was added to the L_1-minimization to find a balance between the sparsity and the robustness of the system fit. Again, the Piecewise Linear Model was applied with a training set of size $M_{\text{tr}} = 21$. By introducing the regularization parameter, it became clear that the sparsity for each row of $A_{\text{est},\ell}$ is lower than for the simpler LP models and the data error is, in general, somewhat lower too. But, still, we have to conclude that the resulting system is not capable to perform well on data that has not been used by the algorithm. This is explained by the fact that only 16 unknowns and 23 observations were available. Until now, we only worked with large networks, of about 80 till 300 components, on artificial data. This resulted in numerical experiments which were statistically relevant.

The application on this Yeast data set is not novel, but it illustrates how our algorithms can be run on practical data after doing some preprocessing and clustering. As mentioned before, it is difficult to find data sets with a realistic number of observations.

7

Conclusion and discussion

In this work, we were interested in the dynamical multi-agents systems, as a means to identify their underlying sparse dynamic interaction networks. Real-world networks, e.g., information networks, biological networks, and social networks, consist of a high number of nodes, representing the agents, and are mostly extremely sparse. Furthermore, in practice, it is difficult to obtain reliable real-world data. As a consequence, experimental data consist of a high number of unknown components and only a few experimental observations compared to the amount of components. This underdetermined system can lead to an identifiability problem, because more solutions are possible. Therefore, we were interested in the sparsity of the network, the relation between the number of components and the number of observations, and the influence of noise, because they define extra constraints that limit the solution space considerably.

To identify an interaction network, a valid mathematical model that gives a detailed description of the given problem had to be developed. There exist numerous different methods to create such a model. Some methods find their foundation in the theory of statistics, others are based on differential or stochastic equations. The interactions between the unknown components in our work, are modeled by rate equations, i.e., by a set of ordinary differential equations (ODE). Starting from this ODE, two algorithms to model and identify have been presented.

First, we have considered the Single Linear Model where the entire state-space is seen as one large space. The interactions, and thus, the system matrices are modeled by a linear dynamic state-space system. To get the most sparse solution, the technique of L_1-minimization, which is based on linear programming, is used.

Second, we have introduced a novel approach, the Piecewise Linear Model. Here, the state-space is partitioned into different subsystems, where different attractors reign. By combining the system matrices with a weight matrix, which represents the membership of each observation to a particular subsystem, it turned out that each subsystem can be considered as one Single Linear System.

From our theoretical analysis in Chapter 4, and the analysis of the example-application in Chapter application, we can conclude that the Single Linear Model and, thus, also the Piecewise Linear Model result in efficient and fast algorithms that are able to accurately estimate a sparse dynamic interaction network from poor data.

To study the relation between the available number of observations and the number of system components, to the impact of sparsity more into detail, the problem has been reformulated in the context of machine learning. In this setting, a model must be learned that fits the underlying dynamic interaction system and is capable of generalization. Therefore, efficient training sets and validation sets were defined.

As an interesting and published result, we found that for a relatively small training set size the learning process is capable to find a perfect fit, and that the transition towards generalization is quite abrupt. To explain this phenomenon, the unknown components were considered. For an increasing training set size, the non-zeros were fitted weakly whereas the fitting of the zeros occurred for higher numbers of observations, but approximately linear with the training set size. Furthermore, the ability to identify networks from poor data turned out to be a consequence of the sparsity of the original regulatory network. The less sparse the network, the more observations were needed to fit.

Besides the results concerning the system sizes and the sparsity, the experiments in the context of machine learning also led to another, important conclusion. It turned out that each row of the linearized interaction matrix can be determined independently. Therefore, the learning problem can be viewed as a feature selection problem. This has led to the observation that the gen-

eralization for various system sizes can be computed by applying the correct scaling on the system size.

Unfortunately, our studies in the context of Machine Learning firmly establish that both the Single Linear Model and the Piecewise Linear Model are not robust with respect to noise. As a consequence, the L_1-mimimization, which is the basis for both models is extended and a regularization parameter is included. The regularization parameter depends on the number of unknown components and the number of observations, and searches for a balance between the sparsity constraint and the robustness of the fit with respect to noise.

In the context of regularization, more observations were required to find a perfect fit than in the previous, more simple linear programming model. The regularization algorithm is able to identify a network with a high number of unknown components together with a training set size of considerably smaller size. More observations were necessary to identify the zero- and non-zero components of the system. In general, however, the behavior of our identification methodology is comparable to the more simple L_1 regression model. For an increasing training set size, the identification of the non-zeros started directly, but weakly, whereas the identification of the zeros occurred later, but fast and efficient and, thus, the transition towards generalization remains still quite abrupt.

After we had found answers to the impact of sparsity of the connectivity matrix of the network structure, the relation between the number of components and the number of observations, and the impact of noise, we were interested in the network's storage capacity.

For each possible combination of system inputs there exists an output vector which is the optimal network response. In the context of the environment of the system. For example, a bacterium exposed to a toxic agent. By storing these input-output patterns, system can learn and react on an appropriate manner to a certain input. The more input-output patterns an organism can store, the more it has learned, the better it is capable to give the best response in its world. It was found that the linear programming formalism can be applied again, and that the maximal number of observations that can be stored depends on the number of unknown components and the number of external inputs. In addition, the storage of the patterns in sparse networks exhibits two distinct phase transitions. These phase transitions divided the system into three regions with different memory characteristics. In the first region, a relatively small number of patterns are stored in the external input matrix

and none in the interaction matrix. In the second region, information is preferentially stored in the interaction matrix. And finally, in the third region, there is no clear preference for storing information in either the external input matrix or the interaction matrix and both matrices are now almost entirely full.

In other words, first, the system stores output that depends exclusively on the external inputs, i.e., light and temperature. Second, the stored output is a result of the learning where mainly interactions between the components are taken into account. Third, the output is a result of the learning of both, the external input and the interactions between the components.

Finally, the Single Linear Model and the Piecewise Linear Model were applied on a real-world data set of the yeast *Saccharomyces cerevisiae*. To identify the interaction network, some preprocessing and clustering was necessary. This resulted in a collection of 16 gene representatives with 23 independent time step observations.

We found that the Single Linear Model and the Piecewise Linear Model were not able to fit the interaction network. This was to be expected because of the noisy data set and the underlying non-linearity. Nevertheless, the Piecewise Linear Model reconstructed a sparse network. To deal with the noise, a regularization parameter was added to the L_1-mimimization, following our approach in Chapter 4. Now, the Piecewise Linear Model was still not able to fit, but the fitting error was lower and the network was more sparse. These results can be attributed to the small number of components and observations.

In this work, we worked predominantly with large networks, of about 80 till 300 components, on artificial data. This resulted in numerical experiments which yielded statistically relevant results. The application on the yeast data set is not novel, but it illustrates how our models can run on practical data after performing the required preprocessing and clustering.

Summarized, in this work, we studied the dynamics of sparse, dynamic interaction networks from poor data. Primarily, this means that we examined the impact of sparsity to the network, the role of the system sizes to the network, and the influence of noise. To show the applicability of this approach, a real-world network was applied. Furthermore, we were interested in the storage capacity of these sparse networks.

Our study results in a number of remaining research questions. What comes to our mind first involves the constraints of hierarchical construction for networks. In this study, we worked with sparse networks, each component is influenced by a small number of other components. But does this imply that a small number of components influence a high number of others? And if so, how does this constraint influence the linear programming?

Second, to perform numerical experiments, we worked with a random interaction matrix wherefore the sparsity constraint must hold. It turned out that most real-world networks are not just random, but have a small-world or scale-free structure. The question that arises is whether our sparse networks have a small-world structure or a scale-free structure? To test the small-world structure, the algorithm of Watts and Strogatz [80] can be used, which is able to construct such a small-world network. To construct a scale-free network, the algorithm of Barabási and Albert [4] can be used. Finally, to confirm, a statistical relevant test, i.e., the χ^2-test, needs to be performed for these simulated networks and our sparse network.

Bibliography

[1] Cell cycle regulated yeast genes. `http://cellcycle-www.stanford.edu`.

[2] Www faqs: How many websites are there? `http://www.boutell.com/newfaq/misc/sizeofweb.html`.

[3] L.A. Adamic and B.A. Huberman. Internet: Growth dynamics of the world-wide web. *Nature*, 401:131, 1999.

[4] R. Albert and A.L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[5] R. Albert, H. Jeong, and A.-L. Barabási. The diameter of the world wide web. *Nature*, 401:130–131, 1999.

[6] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[7] A.-L. Barabási, R. Albert, and H. Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–187, 1999.

[8] A.-L. Barabási, R. Albert, and H. Jeong. The topology of the world wide web. *Physica A*, 281:69–77, 2000.

[9] A.-L. Barabási, H. Jeong, E. Ravasz, Z. Néda, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaborations. *Physica A*, 311(3):590–614, 2002.

[10] A.-L. Barabási and A. Reka. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[11] A. Barrat and M. Weigt. On the properties of small-world network models. *European Physica B*, 13(3):547–560, 2000.

[12] J.C. Bezdek and S.K. Pal. *Fuzzy models for Pattern Recognition.* IEEE-Press, New York, 1992.

[13] B. Bollobás. The diameter of random graphs. *Trans. Amer. Math. Soc.*, 267:41–52, 1981.

[14] B. Bollobás. *Random Graphs.* Academic Press, New York, 2001.

[15] B. Bollobás and O. Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2002.

[16] J.M. Bower and H. Bolouri. *Computational Modeling of Genetic and Biochemical Networks.* The MIT Press, Massachusetts, 2001.

[17] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajalopagan, R. Stata, A. Tomkins, and J. Wiener. Graph structures in the web. *Computer Networks*, 33(16):309–320, 2000.

[18] Q. Chen, H. Chang, R. Govindan, S. Jamin, A.J. Shenker, and W. Willinger. The origin of power laws in internet topologies revisited. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, 2000.

[19] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *PNAS*, 99(25):15879–15882, 2002.

[20] A. Cornish-Bowden. Kinetics of multi-enzyme systems. *Biotechnology*, 9:121–136, 1995.

[21] N. Cristianini and M. W. Hahn. *Introduction to Computational Genomics.* Cambridge University Press, 2006.

[22] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Computational Biology*, 9:67–103, 2002.

[23] S.N. Dorogovtsev, J.F.F. Mendes, and A.N. Samukhin. Metric structure of random networks. *Nuclear Physics B*, 653:307–338, 2003.

[24] J.W. Duncan and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[25] M. Eigen. Kinetics of reaction control and information transfer in enzymes and nucleic acids. *Nobel Symp.*, 5:333–366, 1967.

[26] M.B. Elowitz, A.J. Levine, E.D. Siggia, and P.S. Swain. Stochastic gene expression in a single cell. *Science*, 297:1183–1186, 2002.

[27] D. Endy and R. Brent. Modeling cellular behavior. *Nature*, 409:391–395, 2001.

[28] P. Erdös and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[29] P. Erdös and A. Rényi. On the strength of connectedness of a random graph. *Acta Mathematica Scientia Hungary*, 12:261–267, 1961.

[30] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *Computer Communication*, volume 29, pages 251–262, 1999.

[31] D.A. Fell and A. Wagner. The small world of metabolism. *Nature Biotechnology*, 18:1121–1122, 2000.

[32] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Computational Biology*, 7:601–620, 2000.

[33] A. Fronczak, P. Fronczak, and J.A. Holyst. Average path length in random networks. *Physical Review E*, 70, 2002.

[34] J.J. Fuchs. More on sparse representations in arbitrary bases. In *13th IFAC Symposium on System Identification*, pages 1357–1362, 2003.

[35] J.J. Fuchs. On sparse representations in arbitrary redundant bases. *IEEE Transactions on Information Theory*, 50(6):1341–1344, 2004.

[36] L. Glass and S.A. Kauffman. The logical analysis of continuous non-linear biochemical control networks. *Theoretical Biology*, 39:103–115, 1973.

[37] A. Goldbeter. Computational approaches to cellular rhythms. *Nature*, 420:238–245, 2002.

[38] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 1989.

[39] J.W. Grossman and P.D.F. Ion. On a portion of the well-known collaboration graph. *Congressus Numerantium*, 108:129–131, 1995.

[40] R. Guthke, U. Möller, M. Hoffmann, F. Thies, and S. Toepfer. Dynamic network reconstruction from gene expression data applied to immune response during bacterial infection. *Bioinformatics*, 21:1626–1634, 2005.

[41] J. Hasty, D. McMillen, F. Isaacs, and J.J. Collins. Computational studies of gene regulatory networks: in numero molecular biology. *Nature Reviews Genetics*, 2(4):268–279, 2001.

[42] M. Hecker, S. Lambeck, S. Toepfer, E. van Someren, and R. Guthke. Gene regulatory network inference: Data integration in dynamic models. *Biosystems*, 96:86–103, 2009.

[43] R. Heinrich and S. Schuster. *The Regulation of Cellular Systems*. Chapman and Hall, New York, 1996.

[44] G. Hollanders. Reconstruction of gene-interaction networks from microarray timeseries: a comparison of two methods on real data sets. Master's thesis, Transnational University Limburg, 2005.

[45] G. Hollanders, G. Bex, M. Gyssens, K. Tuyls, and R. Westra. Learning sparse networks from poor data. In *Benelearn 2007*, pages 30–36, Amsterdam, the Netherlands, 2007.

[46] G. Hollanders, G. Bex, M. Gyssens, K. Tuyls, and R. Westra. On phase transitions in learning sparse networks. In *BNAIC 2007*, pages 359–360, Utrecht, the Netherlands, 2007.

[47] G. Hollanders, G. Bex, M. Gyssens, K. Tuyls, and R. Westra. On phase transitions in learning sparse networks. *Lecture Notes in Artificial Intelligence*, 4701:591–599, 2007.

[48] H. Jeong, S. Mason, A.-L. Barabási, and Z.N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411:41–42, 2001.

[49] H. Jeong, B. Tombor, R. Albert, S.N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

[50] S.A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Theoretical Biology*, 22:437–467, 1969.

[51] S.A. Kauffman. *Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[52] R. Kumar, P. Raghavan, S. Rajalopagan, and A. Tomkins. In *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, 1999.

[53] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[54] R. Monasson. Diffusion, localization and dispersion relations on 'small-world' lattices. *European Physic B*, 12:555–567, 1999.

[55] M.E.J. Newman. The structure of scientific collaboration networks. *PNAS*, 98:404–409, 2001.

[56] M.E.J Newman. Power laws, pareto distibutions and zipf's law. *Contemporary Physics*, pages 46–323, 2005.

[57] M.E.J. Newman and D.J. Watts. Renormalization group analysis of the small-world network model. *Physica A*, 263:341–346, 1999.

[58] B. Novak and J.J. Tyson. Modeling the control of dna replication in fission yeast. *PNAS*, 94:9147–9152, 1997.

[59] L. Øyehaug, E. Plahte, and S.W. Omholt. Targeted reduction of complex models with time scale hierarchy–a case study. *Mathematical Biosciences*, 185:123–152, 2003.

[60] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Francisco, 1988.

[61] R.L.M. Peeters and R.L. Westra. On the identification of sparse gene regulatory networks. In *The 16th Intern. Symposium on Mathematical Theory of Networks and Systems (MTNS)*, 2004.

[62] E. Plahte, T. Mestl, and S.W. Omholt. A methodological basis for description and analysis of systems with complex switch-like interactions. *Mathematical Biology*, 36:321–348, 1998.

[63] J. Podani, Z.N. Oltvai, H. Jeong, B. Tombor, A.-L. Barabási, and E. Szathmary. Comparable system-level organization of archaea and eukaryotes. *Nature Genetics*, 29:54–56, 2001.

[64] I. Pool and M. Kochen. Contacts and influence. *Social Networks*, 1:1–48, 1978.

[65] A. Rapoport. Contribution to the theory of random and biased nets. *Bulletin of Mathematical Biophysics*, 19:257–277, 1957.

[66] A. Rapoport. Cycle distribution in random nets. *Bulletin of Mathematical Biophysics*, 10:145–157, 1968.

[67] S. Redner. How popular is your paper? an empirical study of the citation distribution. *European Physica B*, 4:131–134, 1998.

[68] J. Schäfer and K. Strimmer. An empirical bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 36:754–764, 2005.

[69] R. Solomonoff and A. Rapoport. Connectivity of random nets. *Bulletin of Mathematical Biophysics*, 13:107–117, 1951.

[70] P.T. Spellman, G. Sherlock, M.Q. Zhang, V.R. Iyer, K. Anders, M.B. Eisen, P.O. Brown, D. Botstein, and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyes cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9:3273–3297, 1998.

[71] J. Stelling, S. Klamt, K. Bettenbrock, S. Schuster, and E.D. Gilles. Metabolic network structure determines key aspects of functionality and regulation. *Nature*, 420:190–193, 2002.

[72] P.S. Swain, M.B Elowitz, and E.D. Siggia. Intrinsic and extrinsic contributions to stochasticity in gene expression. *PNAS*, 99(20):12795–12800, 2002.

[73] R. Thomas and R. D'Ari. *Biological Feedback*. CRC Press, Boca Raton Florida, 1990.

[74] R. Thomas, D. Thieffry, and M. Kaufman. Dynamical behaviour of biological regulatory networks. *Bulletin of Mathematical Biology*, 57:247–276, 1995.

[75] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-posed Problems*. V H Winston and sons, Washington D.C., 1977.

[76] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R.B. Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

[77] Y. Tu, G. Stolovitzky, and U. Klein. Quantitative noise analysis for gene expression microarray experiments. *PNAS*, 99(22):14031–14036, 2002.

[78] A. Vázquez, R. Pastor-Satorras, and A. Vespignani. Large-scale topological and dynamical properties of the internet. *Physical Review E*, 65, 2002.

[79] V. Verdult and M. Verhaegen. Subspace identification of piecewise linear systems. In *The 34rd IEEE Conference on Decision and Control (CDC)*, volume ISSN 3838–3843, 2004.

[80] D.J. Watts and S.H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[81] R. Westra, G. Hollanders, G. Bex, M. Gyssens, and K. Tuyls. Reconstruction of flexible gene-protein interaction networks using piecewise linear modeling and robust regression. In *Workshop on Adaptation in Artificial and Biological Systems*, volume 3, pages 180–188, Bristol, England, 2006.

[82] R. Westra, G. Hollanders, G. Bex, M. Gyssens, and K. Tuyls. The pattern memory of gene-protein networks. *AI Communications*, 20(4):297–311, 2007.

[83] R. Westra, G. Hollanders, G. Bex, M. Gyssens, and K. Tuyls. Piecewise linear modeling of gene-protein interaction networks. *Lecture Notes in Bioinformatics*, 4366:157–170, 2007.

[84] R.L. Westra. Piecewise linear dynamic modeling and identification of gene-protein interaction networks. In *Nisis/JCBWorkshop reverse engineering*, Jena, Germany, 2005.

[85] M.K.S. Yeung, J. Tegnr, and J.J. Collins. Reverse engineering gene networks using singular value decomposition and robust regression. *PNAS*, 99:6163–6168, 2002.