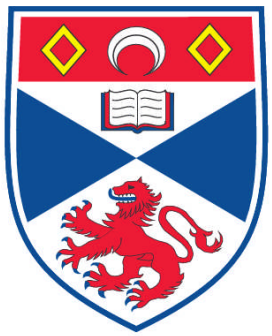
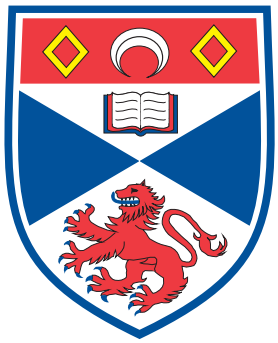
CAUSAL PATTERN INFERENCE FROM NEURAL SPIKE TRAIN DATA

Christoph Echtermeyer

A Thesis Submitted for the Degree of PhD
at the
University of St. Andrews



2009



University
of
St Andrews

School of Biology

PhD Thesis

Causal Pattern Inference from Neural Spike Train Data

by

Christoph Echtermeyer

14th July 2009

Abstract

Electrophysiological recordings are a valuable tool for Neuroscience in order to monitor the activity of multiple or even single neurons. Significant insights into the nervous system have been gained by analyses of resulting data; in particular, many findings were gained from spike trains whose correlations can give valuable indications about neural interplay. But detecting, specifying, and representing neural interactions is mathematically challenging. Further, recent advances of recording techniques led to an increase in volume of collected data, which often poses additional computational problems. These developments call for new, improved methods in order to extract crucial information.

The matter of this thesis is twofold: It presents a novel method for the analysis of neural spike train data, as well as a generic framework in order to assess the new and related techniques. The new computational method, the Snap Shot Score, can be used to inspect spike trains with respect to temporal dependencies, which are visualised as an information flow network. These networks can specify the relationships in the data, indicate changes in dependencies, and point to causal interactions. The Snap Shot Score is demonstrated to reveal plausible networks both in a variety of simulations and for real data, which indicate its value for understanding neural dynamics.

Additional to the Snap Shot Score, a neural simulation framework is suggested, which facilitates the assessment of neural network inference techniques in a highly automated fashion. Due to a new formal concept to rate learned networks, the framework can be used to test techniques under partial observability conditions. In the presence of hidden units quantification of results has been a tedious task that had to be done by hand, but which can now be automated. Thereby high throughput assessments become possible, which facilitate a comprehensive simulation-based characterisation of new methods.

Declarations

I, Christoph Echtermeyer, hereby certify that this thesis, which is approximately 72,500 words in length, has been written by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree.

I was admitted as a research student in September 2006 and as a candidate for the degree of PhD in September 2007; the higher study for which this is a record was carried out in the University of St Andrews between 2006 and 2009.

date _____ signature of candidate _____

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

date _____ signature of supervisor _____

In submitting this thesis to the University of St Andrews we understand that we are giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that my thesis will be electronically accessible for personal or research use unless exempt by award of an embargo as requested below, and that the library has the right to migrate my thesis into new electronic forms as required to ensure continued access to the thesis. We have obtained any third-party copyright permissions that may be required in order to allow such access and migration, or have requested the appropriate embargo below.

The following is an agreed request by candidate and supervisor regarding the electronic publication of this thesis: Access to Printed copy and electronic publication of thesis through the University of St Andrews.

date _____ signature of candidate _____

signature of supervisor _____

Contents

Table of Symbols

$P(A)$	(marginal) probability of proposition A
$P(A \mid B)$	conditional probability of proposition A given B
$\binom{n}{k}$	binomial coefficient $\frac{n!}{k!(n-k)!}$
$\{\dots\}$	set
$\emptyset$	empty set
$\#\{\dots\}$	number of elements in set
$\mathbb{N}, \mathbb{N}_0$	set of natural numbers $\{1, 2, 3, \dots\}$ and $\mathbb{N} \cup \{0\}$, respectively
$\mathbb{Z}$	set of integer numbers $\{ \pm n \mid n \in \mathbb{N}_0 \}$
$\mathbb{R}$	set of real numbers
$[a, b]$	closed interval $I = \{x \in \mathbb{R} \mid a \leq x \leq b\}$ if $a \leq b$, $I := \emptyset$ otherwise
$[a, b]_{\mathbb{Z}}$	set of integers within interval $[a, b] \cap \mathbb{Z}$
$\forall$	universal quantifier, *"for all"*
$\exists$ and $\nexists$	existential quantifier, *"there exists"* and *"there does not exist"*, respectively
$\arg\max_x f(x)$	argument x for which function f is maximal
$\lfloor x \rfloor$	floor function $\lfloor x \rfloor := \max\{z \in \mathbb{Z} \mid z \leq x\}$ for $x \in \mathbb{R}$
$\lceil x \rceil$	ceiling function $\lceil x \rceil := \min\{z \in \mathbb{Z} \mid z \geq x\}$ for $x \in \mathbb{R}$
$a \text{ div } b$	integer division $\left\lfloor \frac{a}{b} \right\rfloor$
$a \text{ mod } b$	modulo operation / remainder $a - b \left\lfloor \frac{a}{b} \right\rfloor$
e	Euler's number $2.71828182846\dots$
$\Gamma(x)$	gamma function $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt$ for $x \in \mathbb{R}$
$n!$	factorial function $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$ for $n \in \mathbb{N}_0$, where $0! := 1$
d	derivative operator
$\log_a(\,\cdot\,)$	logarithm with base a
$\ln(\,\cdot\,)$	natural logarithm $\ln(\,\cdot\,) = \log_e(\,\cdot\,)$
$\hat{=}, \hat{\approx}$	corresponds to, approximately corresponds to
$\overset{!}{=}, \overset{!}{>}$	indicating equation's key condition, i.e. $=$ or $>$, respectively

Preface

Many years of research have been spent in order to understand the principles underlying the central nervous system, and ongoing discoveries indicate that further investigations are required for a comprehensive understanding of higher vertebrate brains. Already, the central nervous system has been shown to exhibit functional organisation in a modular fashion, and interconnecting neural pathways have been identified. With the ability to communicate, different brain regions that are specialised to perform particular computations can cooperate in order to process complex information jointly. Neural action potentials have been identified as a crucial carrier of information being transferred between individual neurons and brain regions; careful observation of these dynamics in vitro and in vivo allows monitoring computations actually being made. Analysing corresponding data can therefore give valuable insights into how the brain works.

Studying the dynamics of a system requires time-series data, a series of observations, whose changes reflect (time-lagged) interaction between different components. In neural systems, the activity of multiple neurons can be recorded with electrophysiological techniques, some of which allow to observe individual action potentials; these can be represented by binary time-series: spike trains. The analysis of such data has led to significant insights, and advancing recording techniques yield more comprehensive and precise observations, such that new, powerful tools are needed in order to analyse the increasingly complex data successfully. During my PhD studies, I thus focused on the construction and evaluation of an efficient analysis technique to detect neural interactions.

Thesis Contents and Conventions

A brief introduction to probabilistic modelling and inference of these models from data prepares the main part. Techniques that can be used to learn neural information flow networks are presented, before a new method, the Snap Shot Score, is introduced. Differences between the new and the existing scores are highlighted, but before that, the new method is illustrated in many exam-

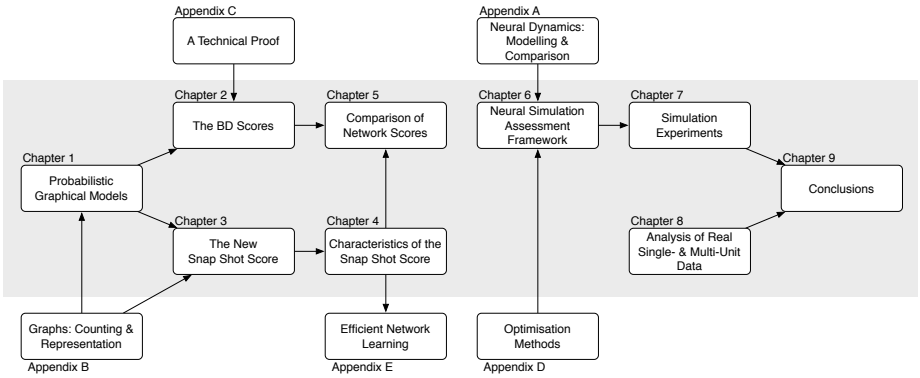


Figure 1: Thesis structure: network of information flow between chapters. The first five chapters introduce the theoretical background, existing techniques are discussed, and they are related to a new method to infer neural information flow networks: the Snap Shot Score. Thereafter, from chapter 6 onwards, the focus is on assessing the novel score practically, and result both for simulated and real data are presented.

ples, which are complemented by a mathematical investigation of its properties. Testing the novel score on both simulated and real data shows its suitability for information flow network inference from spike train data.

Appendices give necessary background information on practical methods for learning networks and the mathematical concept of graphs. Additionally, guides to efficient software implementation are given for both existing and the novel technique, together with details on neuron modelling. References in the main text point to corresponding sections where appropriate.

Parts of this work have been published [Echtermeyer et al., 2009]; these include the Snap Shot Score itself and simple examples (Chapter 3), as well as its assessment in neural simulations (Chapter 7 in parts). Remaining sections (especially Chapters 4 and 8) contain results that have not been reported yet.

The reader is invited to take a sneak preview on the conclusion chapter 9 now, in order to learn more about the main points of this work whose structure is depicted in figure 1. Additionally, two things should be clarified in order to avoid confusion:

1. The term *information flow network* is used in different parts of this thesis and refers to a structure that is recovered from neural data. The name is not meant to be understood in an information theoretic sense; it is simply a descriptive term used in Neuroscience that refers to co-ordinated activity, propagating between different entities.

2. Probabilistic interactions are central to this work and the term *correlation* is used to identify such sporadic effects. Correlation is often associated

with linear effects, but in this thesis it is used in the general context. In case needed, linear-correlation will explicitely be referred to as such.

Finally, the title of this thesis deserves some explanation: As already mentioned, this work is about methods for the inference of information flow networks from spike train data. How such networks relate to causal patterns needs to be explained in two steps. First, the new Snap Shot Score interprets data in terms of a causal model (Chapter 3); learned networks therefore posses a causal semantics (Chapter 5). But secondly, the information conveyed by spike train data is often insufficient in order to resolve equivalence of multiple causal explanations (Section 1.4.2); parts of recovered networks may therefore be ambiguous. Together, inferred networks can therefore not be interpreted on the whole, but as an aggregate of causal patterns.

Acknowledgements

First and foremost I would like to thank V. Anne Smith. With her supervision throughout my PhD studies, I have enjoyed the large freedom to explore; the assurance that I could rely on inspiring input whenever I got stuck; and valuable feedback on several ideas and documents, whose understandability might have been problematic in more than one occasion.

Many thanks go to Tom V. Smulders, who has always been available to discuss theoretical ideas from a biological perspective. He also provided electrophysiological recordings, which were a valuable part for validating the new method on real data.

Further recordings have been kindly provided by Evelyne Sernagor. For these data, as well as for the time she spent in order to select, prepare, and explain them, I am very grateful.

For funding of my PhD studies I would like to thank the Engineering and Physical Sciences Research Council (EPSRC).[1] Additional financial support for the attendance of various meetings in relation to my work was kindly granted by the EU Special Support Action in Neuroinformatics, UK Neuroinformatics Network, Newton Institute Junior Member Grant, William Ramsay Henderson Trust travelling scholarship, EPSRC, Young Physiologists Symposium travel grant programme, and the International Brain Research Organisation (IBRO).

Finally, to all those who inspired me; those who challenged me; those who supported me — Thank you!

[1] My work was supported by the CARMEN e-science project (www.carmen.org.uk) funded by the EPSRC (EP/E002331/1).

Chapter 1

An Introduction to Probabilistic Graphical Models

> A good theoretical model of a complex system should be like a good caricature: it should emphasise those features which are most important and should downplay the inessential details. Now the only snag with this advice is that one does not really know which are the inessential details until one has understood the phenomena under study.
>
> Fisher [1983]

Biological studies of organisms aim at understanding them by examination. Revealing the structure, function, and causal connections between identified elements is often done by a series of inspections concerning both static and dynamic properties of a life form. Initially, distinct components of the organism can often be recognised by static observations. For example, an animal's post-mortem examination reveals facts about its anatomy. Once distinct elements are identified, e.g. separate organs, investigations about their functional properties follow. Static observations, however, may not be sufficient in order to infer the role a particular elements plays. It might, for instance, be hard to imagine that the heart acts as a pump without having ever seen it contracting. Thus, additional information is required, which can be gathered by monitoring the element's in vivo dynamics. The third step, once essential components and their functions are known, is to investigate these parts' interplay, i.e. how their particular functions are combined.

This general scheme in particular applies to studies of the nervous system: Neurons have been identified to constitute the basic building blocks. Their function, the integration of synaptic inputs, has been closely examined on a single cell basis. And after that, studies on the dynamics of neural networks have begun in order to understand how complex computations are performed by combining multiple neural entities. The methods discussed in this thesis are used to detect and describe interactions between multiple neurons; they do thus aim to contribute insights to the phenomena that arise in brain networks. This chapter gives an introduction to the model-paradigm — probabilistic graphical models — that will be used to describe neural interactions.

Generally, successful modelling of a system means to result in a compact description, which allows one to replicate and predict phenomena of its original. The leaky integrate and fire (LIF) neuron or the Hodgkin-Huxley (HH) model (Section A.1) are examples, which both, to some extend, match the dynamics of real neurons. Which one of these two models should be preferred over the other depends on the level of detail and precision it is desired to provide. Generally, modelling is characterised by a trade-off between capability and simplicity: The right kind of model must be flexible enough in order to capture features of the system that are regarded relevant. Both the LIF- and HH-model can be suitable model types at the level of spike train data, but LIF-neurons are generally not capable of generating reasonable sub-threshold dynamics; these are the domain of the more powerful HH-model. However, models that are very good in capturing and replicating effects of the target system can also be too complex to allow insights into its machinery. A simpler, less precise, but expressive model could in fact benefit findings about mechanisms at work. Indeed, the simple LIF-model is very helpful in establishing understanding of the principle of synaptic integration, whereas the HH-model might hinder a clear view through too much detail. Like the many ways to model neurons, there exist numerous alternatives in order to describe their functional interaction. Detecting and representing neural interplay is the subject of this work; therefore, a suitable model type had to be chosen. I decided for a general model type which is combined with concepts that ensure it to be expressive. The balanced trade-off between flexibility and clarity is implemented by probabilistic models, which are discussed in this chapter. Techniques that render them practical tools follow thereafter (Chapters 2 and 3).

1.1 A Primer on Probabilities

Studies of reasoning with uncertain information have led to the field of probability theory. With the corresponding mathematical methodology, doubts can

be formally expressed and analysed with respect to their consequences. Importantly, the theory provides a mechanism for inference from undetermined data by appropriately weighting and integrating different possible initial conditions. This simultaneous consideration of alternatives renders probability theory a powerful extension of two-valued logic [Jaynes and Bretthorst, 2003]: A logical statement can only be either *true* or *false*, but probability theory allows expressing uncertainty about the validity of a statement; any degree of belief in it can be specified. Such might be necessary in scientific reasoning, where incomplete information makes it impossible to decide on the definite validity of a particular statement, for example. Whereas logical reasoning can only be applied if all variables are categorised into one of the extreme cases, probability theory provides the machinery to draw conclusions from indefinite information.

The degree of belief in the validity of a proposition A is called its *probability*. A probability is a real number between zero and one, which is denoted by $P(A)$. The value one corresponds to the fact that the statement is thought to be *true* whereas zero means that it is *false*. Probabilities in between express doubts. At a value of 0.5, for instance, the proposition could either be *true* or *false* without indicating any tendency towards one of the possibilities. As an example consider the statement A= *"It will rain soon"*. Whether this prediction is correct might be completely unknown; however, as additional information becomes available the probability can change. For instance, based on experience, being told that heavy clouds are in the sky, the event of rain becomes more likely than it was before. The probability of the statement A is now conditioned on a given fact B, which is denoted as $P(A \mid B)$ and is called the *conditional probability* of A given B.

Classic definitions of probability often use long run observations of a random experiment whose possible outcomes are called events. The probability of an event is defined as its relative frequency, which is determined by infinitely repeating the experiment (in mind) [Feller, 1950]. One typical example is to toss a coin and count the occurrences of *head* and *tail* for a large number of repetitions. If the coin is unbiased either side of it would show up about equally often, such that $P(head) \approx 0.5 \approx P(tail)$. This concept, known as the *frequentist* approach, is an easy to understand school of thought; however, it fails to assign a probability to events that cannot be repeated [Heckerman, 1997]. In fact, a probability can never be assigned before an experiment is performed (Fig. 1.1).[1] The way probabilities are introduced here — according to the *Bayesian* approach — overcomes these problems as it permits making an educated guess about what the chances of different outcomes are if the random experiment was performed.

[1] A discussion of problems arising by defining probabilities as frequencies and a comparison to the Bayesian probability theory can for instance be found in Cox [1946], Loredo [1990], Jaynes and Bretthorst [2003].

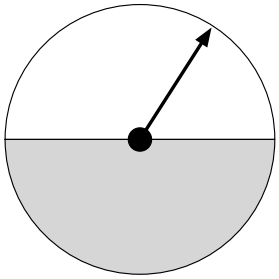


Figure 1.1: Adopted from Heckerman [1997]: Perfectly symmetric wheel of Fortune except for shading of its two halves. In order to assign a probability to the event that the pointer stops in the shaded half, the experiment would have to be performed and repeated a large number of times, if the frequentist approach is used. In contrast, the Bayesian approach allows deduction of a probability from the knowledge about the set-up before ever spinning the wheel.

In order to be consistent, background knowledge that has lead to such a speculative probability must be indicated. This is done correctly by denoting it as a conditional probability $P(A \mid X)$ of the proposition A given the background knowledge X. Any prior information that enters a calculation is made explicit this way.

The assignment of probabilities is often more complicated than in the idealised situation depicted in figure 1.1, where physical properties that are relevant to the assignment are all given. A lack of such precise knowledge is typical for realistic situations where instead only collected data might be available. These data can then be analysed with classical statistical (frequentist) methods whose results represent background knowledge in the Bayesian sense; background knowledge that can be used to assign probabilities. In order to indicate the importance of this subtle step consider the wheel of Fortune again: Observing the wheel to stop 512 out of 1,000 times in the shaded area, i.e. with a relative frequency of 0.512, can be seen as an indication for equal halves of the wheel. Based on the relative frequency, our bias towards both parts of the wheel being equal, and our knowledge about random fluctuations of relative frequencies around their limiting value, we can decide to assign an equal probability to the appearance of each half. The possibility to assign a probability different to the relative frequency introduces subjectivity to the calculus: If the same relative frequency had been determined from a total of 10^6 spins of the wheel, the probability assignment would clearly be different. Because of this, the Bayesian school of thought is often criticised as not being objective; in return, an objective frequentist estimate could be off the correct value (assuming that the wheel is indeed symmetric), because background knowledge is ignored.

Ending the example here should already have given an impression about the dispute between the two approaches. This work does not contain further comparisons between the two concepts, and it is left to the reader to decide how results of statistical methods are used to assign probabilities. Henceforth such assignments from data are called *probability estimates*, in order to identify both ways to use statistical methods: either as direct assignment principle or as a technique to derive background knowledge that is used for an assignment.

The next section starts with a brief introduction to probabilistic models, which can often be extended to graphical models, like Bayesian networks. Later on, models that can describe temporal relationships and graphical correspondents of these are discussed. However, dependencies without a temporal component are considered first.

1.2 Probabilistic Models

Before the interactions within a complex system can be modelled, its components first need to be identified. In probabilistic models each of these elements' state is represented by a random variable [Feller, 1950, pp.212]. For the remainder of this work n denotes the number of components that are to be included in the model and $X_1, \ldots, X_n$ denote their state variables. A probabilistic model is simply a joint probability distribution over all of these variables. Such distribution cannot only code for pairwise linear correlation but, more generally, full multivariate relations. The following example illustrates such model.

Example 1 (Probabilistic Model) *Consider the two random variables*

$$\begin{aligned} X_1 &= \textit{the sun is shining} \ , \\ X_2 &= \textit{it is raining} \ , \end{aligned} \tag{1.1}$$

each of which can either be true or false. Assume we are in a sunny region where we find the following marginal probabilities

$$\begin{aligned} P(X_1 \ \textit{true}) &= \ 0.75 \ , \quad P(X_1 \ \textit{false}) = \ 0.25 \ , \ \textit{and} \\ P(X_2 \ \textit{true}) &= \ 0.1 \ , \quad P(X_2 \ \textit{false}) = \ 0.9 \ . \end{aligned} \tag{1.2}$$

The marginal distributions $P(X_1)$ and $P(X_2)$ give a description of how likely either weather phenomenon is regardless of the other; they do not tell anything about chances of observing joint conditions like

$$P(X_1 \ \textit{true}, X_2 \ \textit{true}) \ , \ \textit{or} \quad P(X_1 \ \textit{false}, X_2 \ \textit{true}) \ , \tag{1.3}$$

for example. A probabilistic model links variables such that joint conditions can be evaluated. For instance, assuming that variables X_1 and X_2 were stochastically independent, i.e.

$$P(X_1, X_2) \overset{assumption}{=} P(X_1) \cdot P(X_2) \ , \tag{1.4}$$

would fully specify our probabilistic weather model. With equation (1.4) the joint probability distribution can be evaluated for all combinations of states. However, the unrealistic assumption that sunshine and rainfall are independent of each other might have to be revised: Rain is commonly accompanied by visible clouds, which mask the sun such that it can become invisible. During rainfall, sunshine is thus less likely, which can be expressed in a conditional probability $P(X_1|X_2)$. For instance, the following probability might be assumed

$$P(X_1 \ true \ | X_2 \ true) = 0.05 \implies P(X_1 \ false \ | X_2 \ true) = 0.95 \ . \tag{1.5}$$

With the marginal probabilities $P(X_1)$ and $P(X_2)$ already defined the remaining conditional probabilities $P(X_1|X_2 \ false)$ can be calculated from the identity [Feller, 1950, p.116]

$$P(X_1) \overset{!}{=} P(X_1|X_2 \ true)P(X_2 \ true) + P(X_1|X_2 \ false)P(X_2 \ false) \ , \tag{1.6}$$

with which we find

$$P(X_1 \ true \ | X_2 \ false) = 0.82\bar{7} \implies P(X_1 \ false \ | X_2 \ false) = 0.17\bar{2} \ . \tag{1.7}$$

With these probabilities defined, the weather model can be expressed using the chain rule [Kjaerulff and Madsen, 2008, p.58], which allows to re-write a joint distribution as a product of (conditional) factor distributions. For the two variables X_1 and X_2 this reads as

$$P(X_1, X_2) \overset{chain \ rule}{=} P(X_1|X_2) \cdot P(X_2) \ , \tag{1.8}$$

by which we have defined a more advanced model than that in equation (1.4). The improved model (1.8) not only matches the assumed marginal probabilities about sunshine and rainfall, but it also incorporates the assumed dependency between the two. The effects of the refinement can be seen when evaluating both

models:

model (1.4)

$P(X_1, X_2)$	X_2 *true*	X_2 *false*	$P(X_1)$
X_1 *true*	0.025	0.225	0.25
X_1 *false*	0.075	0.675	0.75
$P(X_2)$	0.1	0.9	

model (1.8)

$P(X_1, X_2)$	X_2 *true*	X_2 *false*	$P(X_1)$
X_1 *true*	0.005	0.745	0.75
X_1 *false*	0.095	0.155	0.25
$P(X_2)$	0.1	0.9	

Both models have the same marginal distributions, but differ in their joint probabilities.

The example gave a demonstration of how existing knowledge can be turned into probabilistic models. As has been shown, these models can give a characterisation of the studied system by capturing interrelations between modelled elements. Models with few variables only (like in the example) can be simple enough to overview and analyse interactions between them, but more complex situations with numerous correlating variables call for clarifying visualisation aids. Illustrating dependencies within a probabilistic model as a network can often provide a good overview; depicted connections help to identify relevant features of the system under study. Models that are associated with particular visualisations are called *graphical models* [Airoldi, 2007, Pearl, 2000, Lauritzen, 1996]. Probabilistic graphical models are fundamental to this thesis and one such model type, Bayesian networks,[2] is discussed in the next section.

1.2.1 Bayesian Networks

Already back in 1921 (and probably even earlier) the advantages of visualising probabilistic dependencies were recognised [Wright, 1921]. Since then, work on graphical models has been continued and resulted in powerful modelling tools of probabilistic and causal relationships (e.g. [Pearl, 1988, 2000]). Out of the developed machinery, here, only a small part is utilised in order to express dependencies as graphs (Section B.1), which can be visualised as networks.

[2]The term *Bayesian network* has been introduced by Pearl [1985]. Three aspects should be emphasised by that name [Pearl, 2000, p.14]: *(1) the subjective nature of the input information; (2) the reliance on Bayes' conditioning as the basis for updating information; and (3) the distinction between causal and evidential modes of reasoning, a distinction that underscores Thomas Bayes' paper of 1763* [Bayes, 1763]. Despite its name, the use of Bayesian methods for these models is neither implied nor a requisite [Murphy, 2001].

The connection between a probabilistic model and its illustration as a network is motivated by an example first. Therefore, techniques will be used that have not been introduced yet; their presentation follows the example, and references to corresponding equations can be skipped when reading it for the first time.

Example 2 (Bayesian Network) *A hypothetical probabilistic model of diseases (cold, flu) and associated symptoms (fever, sore throat, changed blood count) is considered. Each of these 5 variables can either be true or false. The model is expressed by the joint distribution over all states*

$$P(\textit{cold}, \textit{flu}, \textit{fever}, \textit{sore throat}, \textit{blood count}) \ , \tag{1.9}$$

which can be factorised according to equation (1.10) by using the chain rule (Fig. 1.2a). The Bayesian network corresponding to that factorisation (Fig. 1.2b, left) has several links, which indicate the stochastic dependencies in the model. As no information about realistic dependencies between variables has entered the model yet, we find their relations purely dependent on the order of variables when applying the chain rule. In order to improve the model we use the knowledge that diseases cause certain symptoms; we assume that symptoms do not cause other symptoms. For instance, we could assume the causal relations depicted in Fig. 1.2b on the right hand side. This network expresses our assumptions that symptoms are due to certain diseases but not other symptoms; thus, conditioned on their parents, symptoms are stochastically independent of each other. The parents of each network node thus correspond to its minimal dependence set $pa(X_i)$, i.e. all other variables are irrelevant and can be omitted [equation (1.12)], which yields the simplified factorisation of the model (Fig. 1.2c).

The mathematical justification for the example is given in the following steps, which describe how a network structure can be constructed for any particular probabilistic model. As before (Section 1.2), the model is expressed by a joint probability distribution $P(X_1, \ldots, X_n)$. First, the joint distribution is written as a product of conditional distributions by applying the chain rule of probability [Kjaerulff and Madsen, 2008, p.58]:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid X_1, \ldots, X_{i-1}) \ . \tag{1.10}$$

In the next step, the conditional distributions $P(X_i \mid X_1, \ldots, X_{i-1})$ are simplified by omitting irrelevant conditioning variables. Therefore, variables that are

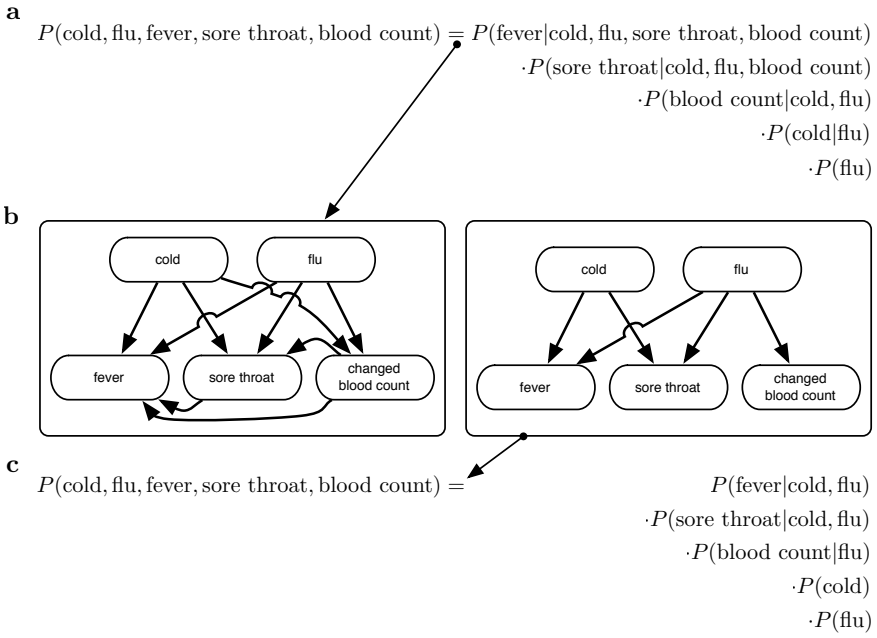


Figure 1.2: Adopted from Jensen [2001] and extended: Hypothetical probabilistic model of diseases (cold, flu) and associated symptoms (fever, sore throat, changed blood count) and related graphical representations. **a** Joint distribution and one possible factorisation according to chain rule [equation (1.10)]. **b** Graphical models corresponding to factorisations shown in (a) (left) and (c) (right). **c** Joint distribution and factorisation according to Bayesian network shown in (b, right). Compared to the factorisation in (a), the number of factor distributions is the same (one for each variable), but distributions in (c) are conditional on fewer variables.

known to be relevant are used to define the minimal dependence set

$$pa(X_i) \subseteq \{X_1, \ldots, X_{i-1}\} \tag{1.11}$$

for each variable X_i such that

$$P(X_i \mid X_1, \ldots, X_{i-1}) = P\left(X_i \mid pa(X_i)\right) \ . \tag{1.12}$$

This means: The minimal dependence sets $pa(X_i)$ are chosen such that no information is gained if more variables are included in the set. Equation (1.10) can therefore be simplified to

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P\left(X_i \mid pa(X_i)\right) \ . \tag{1.13}$$

This equivalent formulation of the model can be easily be converted to an illustrative network. In order to describe the procedure, variables in $pa(X_i)$ are called X_i's *parents* of the *child* X_i. This terminology has been chosen to reflect relations between variables, when representing the factorisation (1.13) as an acyclic directed graph [Heckerman, 1997]: Each variable X_i is assigned a corresponding node, and for each parent of X_i a link pointing from the parent's node to the child's node is added. This converts a factorisation to a graph, which is commonly called a *Bayesian network*. Vice versa, a graph can also imply a certain factorisation, as has already been shown in example 2.

Bayesian networks can give a concise picture of modelled dependencies and, as indicated in the preceding example, their graphical component provides a simple mechanism to translate prior knowledge into the model. The key of the concept is to reduce the minimal dependence sets as far as possible. Thinning out dependency relations not only benefits by producing a clearer representation of the model, but it also reduces the amount of data that is required in order to parameterise the model [Pe'er, 2005]. This is because the dimension of conditional factor distributions is minimised, i.e. they are defined over fewer variables. They thus cover a smaller state space, which requires fewer probabilities to be defined. Finally, this reduces the amount of data that is needed in order to estimate the probabilities. The following example illustrates how this translates into practice.

Example 3 (Reduced Model Dimension) *In order to analyse how model complexity is affected by omitting irrelevant dependencies between variables, recall the model introduced in example 2. The joint distribution (1.9) of* 5 *binary variables is defined over a sample space of* $2^5 = 32$ *possible states. Because probabilities over all states sum to unity, the joint distribution is fully defined by*

31 *probabilities (as the missing probability can be computed from those). Likewise, the factorisation shown in Fig. 1.2a requires 31 probabilities to be estimated. This is because each factor distribution that is conditional on k variables requires* 2^k *probabilities to be defined; hence,*

$$\sum_{k=0}^{n-1} 2^k = 2^n - 1 \tag{1.14}$$

probabilities need to be defined for n binary variables. (Here $n = 5$*, where equation (1.14) evaluates to* $2^5 - 1 = 32 - 1 = 31$*.) By assuming the sparser Bayesian network (Fig. 1.2b, right) the model is made more specific and its associated factorisation (Fig. 1.2c) requires fewer probabilities to be estimated. This is because its factor distributions are of lower dimension and we find that only* 12 *(*$= 2 \cdot 2^2 + 2^1 + 2 \cdot 2^0$*) probabilities are necessary in order to fully specify the model. Both versions of the model describe the same situation, but differ in complexity: By simplifying the dependency relations the number of probabilities to define reduces by* $\approx 61\%$*.*

Explicitly specifying stochastic dependencies can decrease the complexity of a model, which reduces the required amount of data in order to parameterise it. The need for data becomes predominant if the number of modelled variables is increased. In such situations, sparse dependency relations can facilitate the construction of models from limited data.

Biological studies often concern the dynamics of an organism; examples are behavioural studies, attempts to reveal gene regulatory networks, or neural pathways. For such investigations, data is commonly collected as a time-series of observations, which capture the state of multiple entities of interest. Interactions between these entities that manifest by correlation of data channels can be identified by subsequent analysis. How correlation in time-series data can be detected will be discussed later (Chapters 2 and 3). Here the focus is on how the ideas of probabilistic models, and Bayesian networks in particular, can be extended to the temporal domain in order to describe time-lagged relationships.

1.3 Stochastic Processes

Similar to the probabilistic models discussed earlier, stochastic processes aim to describe the dependencies between random variables. Differently to the before mentioned probability distributions, stochastic processes consider time as an inherent factor of relations between variables. In order to be able to express time-lagged effects a time index is added to each variable, which indicates

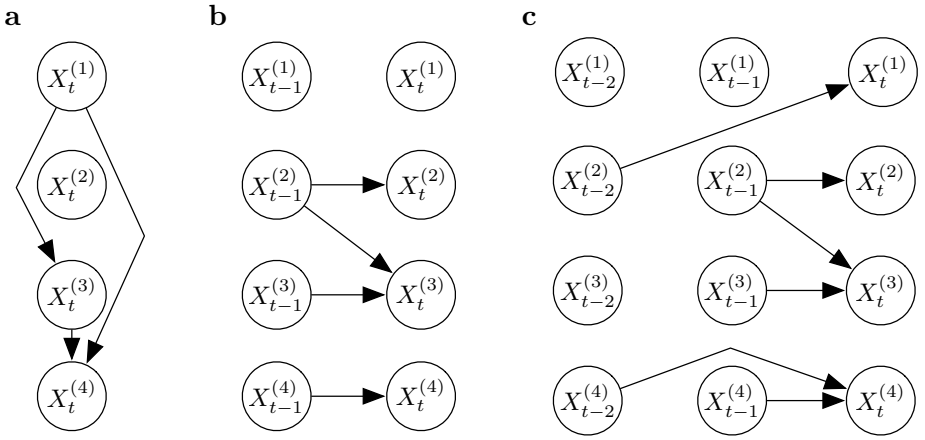


Figure 1.3: Graphical models of different orders. **a** Bayesian network. Interactions between variables occur instantaneously. **b** Dynamic Bayesian Network of 1st order. Variables $X^{(2)}$, $X^{(3)}$, and $X^{(4)}$ are dependent on states of the directly preceding time-slice. **c** Dynamic Bayesian Network of 2nd order. Variables are either dependent on states of one of the preceding time-slices ($X^{(1)}$, $X^{(2)}$, and $X^{(3)}$) or both ($X^{(4)}$).

relative time differences: $X_t^{(i)}$ represents the random variable i at time t. By convention, time t denotes the present; any times greater and smaller than t correspond to the future and past, respectively.

The previously discussed models do not consider time as a factor; interactions between variables are implicitly assumed to be instantaneous. This can be explicitely expressed by variables with time indices (Fig. 1.3a):

$$P\left(X_t^{(i)} \middle| X_t^{(1)}, \ldots, X_t^{(i-1)}, X_t^{(i+1)}, \ldots, X_t^{(n)}\right) . \tag{1.15}$$

In contrast, processes suppose that the present can only be influenced by the past. The probability distribution of a variable is thus conditioned on past states of variables:

$$P\left(X_t^{(i)} \middle| \left(X_{\bar{t}}^{(1)}, \ldots, X_{\bar{t}}^{(n)}\right)_{\bar{t}<t}\right) . \tag{1.16}$$

In order to be practically applicable, processes are commonly constrained with respect to two aspects: The past time-span that actually influences the present is limited, and secondly, time is often assumed to be discrete. In its most simplest form a process is a so called *1st order Markov model*, which states that each variable $X_t^{(i)}$ solely depends on states of variables in the most recent past $t-1$ (Fig. 1.3b), i.e.

$$P\left(X_t^{(i)} \middle| X_{t-1}^{(1)}, \ldots, X_{t-1}^{(n)}\right) . \tag{1.17}$$

The 1st order model is a powerful prototype, which can easily be extended to any order. In an m^{th} order model a variable not only depends on the most recent past, but on a history of m time-steps (Fig. 1.3c):

$$P\left(X_t^{(i)} \middle| \left(X_{\bar{t}}^{(1)}, \ldots, X_{\bar{t}}^{(n)}\right)_{\bar{t}\in\{t-1,\ldots,t-m\}}\right) . \tag{1.18}$$

In the remainder of this work these models, 1st and m^{th} order Markov models, will be considered. These can be visualised as networks similarly to Bayesian networks; however, before turning towards their display as graphs an important aspect of these models should not be overlooked: The stochastic relations between variables are assumed to be *stationary*, i.e. they do not change over time. Models considered here thus do not serve to investigate dependency changes over time, but to encode stable probabilistic relations between random variables, i.e. their lagged influence on each other. Assuming stationarity of the modelled system might seem unacceptable for biological applications at first sight, as organisms change over time; consequently, relations associated with the organism will change, too. However, if the period of data collection is relatively short compared to the scale on which interactions within the organism change, relations in the measured data can indeed appear stable. In cases where no alternations occur while collecting the data it is reasonable to assume stationarity and no inaccuracies are noticeable.

1.3.1 Dynamic Bayesian Networks

Bayesian networks are restricted to modelling instantaneous or *static* relationships between variables. In contrast, Markov models can describe lagged or *dynamic* dependencies. Their graphical representations (Fig. 1.3bc) are therefore referred to as *dynamic Bayesian networks*. In order to describe such model with a graph, the procedure presented for static Bayesian networks (Section 1.2.1) can be adapted to dynamic ones: A minimal dependence set

$$pa(X^{(i)}) \subseteq \left\{X_{t-1}^{(1)}, \ldots, X_{t-1}^{(n)}, X_{t-2}^{(1)}, \ldots, X_{t-2}^{(n)}, \ldots X_{t-m}^{(1)}, \ldots, X_{t-m}^{(n)}\right\} \tag{1.19}$$

is defined for each variable $X^{(i)}$, such that

$$P\left(X_t^{(i)} \middle| \left(X_{\bar{t}}^{(1)}, \ldots, X_{\bar{t}}^{(n)}\right)_{\bar{t}\in\{t-1,\ldots,t-m\}}\right) = P\left(X_t^{(i)} \middle| pa(X^{(i)})\right) . \tag{1.20}$$

Conditioned on its parents $pa(X^{(i)})$ a variable $X^{(i)}$ is thus independent of all others in the past. The full m^{th} order model can therefore be written as

$$P\left(X_t^{(1)},\ldots,X_t^{(n)}\middle|\left(X_{\bar{t}}^{(1)},\ldots,X_{\bar{t}}^{(n)}\right)_{\bar{t}\in\{t-1,\ldots,t-m\}}\right) = \prod_{i=1}^{n} P\left(X_t^{(i)}\middle|\, pa(X^{(i)})\right) \ . \quad (1.21)$$

In order to display the model as a network each variable is represented by $m+1$ nodes; once for each time-layer $t, t-1, \ldots, t-m$. Nodes corresponding to variables in the minimal dependence set $pa(X^{(i)})$ are made parents of the child node belonging to $X_t^{(i)}$ in the current time-layer t.

So far, modelling examples consisted of made up phenomena and an imaginary system which generates them. This allowed derivation of the stochastic dependencies needed to build a corresponding model. In most practical situations, however, the system under study is not completely known and neither are its causal relationships. The system thus cannot be modelled due to a lack of knowledge about its stochastic dependencies, for example. However, to some extent, missing information can be extracted from collected data. How this can be conceptually done is discussed in the next section, which motivates data driven modelling.

1.4 Learning Models From Data

Graphical models of complex systems can benefit their understanding through the compact representation of dependency structures. Such knowledge is particularly interesting when relatively little is known about the studied system. But how can a graphical model be constructed initially, when causal interactions within the system are not known? This section will address this question on a conceptual level and thereby prepare the introduction of two practical approaches, which are discussed later (Chapters 2 and 3). A review of the considerable literature on learning graphical models from data can be found in Buntine [1996], for example.

Research questions can be addressed in various manners and conceptually one can often distinguish a hypothesis driven approach from a data oriented concept. In the first mentioned case a formulated hypothesis is tried to be corroborated or falsified. The required data is collected especially for this purpose and statistical methods are commonly used to decide on the hypothesis' validity. Differently, data driven research is signified by a rather unspecific data collection

phase followed by an analysis step. Typically, the collected data are not used in order to decide whether a particular hypothesis is true or not, but to find hypothesises that can plausibly explain the data. The necessary methodology in order to evaluate and compare hypotheses is provided by Bayesian inference whose key component — Bayes' theorem — is presented next.

1.4.1 Bayes' Theorem

Bayesian inference is concerned with evaluating hypotheses; more precisely, it aims to derive the probability that a particular hypothesis is true. The following ideas assume that the hypothesis to assess is formulated with respect to some data, which is informative (to some degree) about whether the hypothesis is true or not. However, the validity of the hypothesis is seldomly dependent on the data alone, but it also depends on prior information. Both the evidence in the data and background information can be taken into account by Bayesian inference, similar to human reasoning [Kording, 2007]. A simple example shows why this is useful: A hypothesis that is absurd — based on prior knowledge — but cannot be falsified by the data alone can be rejected a priori when incorporating prior information in the analysis. To formalise the combination of background information and data in order to evaluate a hypothesis the following notation is used:

X = prior information,

H = hypothesis to be assessed, and

D = data.

The probability that a hypothesis H is true, based on the information D and X, is inferred with *Bayes' theorem* [Feller, 1950, Jaynes and Bretthorst, 2003]:

$$\underbrace{P(H \mid D, X)}_{\text{posterior}} = \underbrace{P(H \mid X)}_{\text{prior}} \frac{\overbrace{P(D \mid H, X)}^{\text{likelihood}}}{\underbrace{P(D \mid X)}_{\text{evidence}}} \,. \tag{1.22}$$

All terms in this equation play a distinctive role and have thus been named [MacKay, 2003, p.29]; the *posterior* is the actual probability of interest. It is a combination of three factors: *prior* belief in the hypothesis; the *likelihood* that the data could have emerged if the hypothesis was true; and the *evidence*, which is a reflection of the number of explanations for the data. These three factors are discussed separately in the following paragraphs.

The *prior* is the probability that the hypothesis is true without considering the data — it is purely based on the background information. Based on that

knowledge some hypotheses are more likely than others. In fact, any thinkable hypothesis must be (implicitly) assigned a probability of being the explanation for the data. Prior assumptions that are used are made explicit by denoting the prior as a conditional probability, which allows for informed, critical review of the results.[3]

Next consider the *likelihood* of the hypothesis. (Note that the likelihood is a probability of the data, but not of the hypothesis.) It is the probability that, given that the hypothesis is true, the data could have been observed; again, taking into account the background information. The likelihood thus reflects how well the hypothesis explains the data. (Higher values are better.)

The final term, the *evidence*, is the marginal probability of the data. This term's role can be understood in relation to the likelihood, which it normalises. This relativisation renders the dimensionless likelihood expressive, because the goodness of the hypothesis' explanation is compared to all alternatives.[4] The ratio of the two terms can only be high if two conditions are met: the likelihood must be high and only few good alternative explanations exist. The resulting effect is, that the more hypotheses suit the data well, the more important prior probabilities become for their rating.

Combining those three terms according to Bayes' theorem [equation (1.22)] yields the posterior of the hypothesis. This probability might not be very expressive on its own, because a small value does not necessary indicate a bad hypothesis; it can also be an indication for many alternative hypotheses. The total number of considered hypotheses must thus be taken into account for interpretation. It is often easier to use the posterior for comparing, i.e. ranking, hypotheses. In such situations, where hypotheses should only be related to each other, the computations can be simplified by omitting the evidence term, which is the same for all hypotheses. The order of hypotheses with respect to each other is thus the same whether the evidence is considered or not, as their rank solely depends on the prior and the likelihood. This is of practical importance, because it is often intractable to specify the evidence due to insufficient prior knowledge.

In this thesis, hypotheses correspond to functional connectivity networks. Each of these networks (together with its underlying model) is a separate hypothesis, which seeks to explain correlation in the data. Different hypotheses are compared against each other in order to determine the best information flow

[3]The selection of background knowledge used and the assignment of priors itself has been criticised to be subjective in long ongoing debates, which cannot be replicated here. However, *objective* statistical tests have underlying assumptions, too. These are priors in the Bayesian sense and must be equally questioned before applying the test (see e.g. [Loredo, 1990]).

[4]Re-writing the evidence as an integral over all hypotheses illustrates its relationship to the likelihood; their ratio reads as: $\frac{P(D \mid H,X)}{\int P(D,h \mid X)dh}$.

network. Two techniques to implement Bayes' theorem are discussed in this thesis. Their concepts will be explained in the Bayesian context in chapter 5 after they have been introduced (Chapters 2 and 3).

The foregoing section explained how information from different sources needs to be combined for inference. Next it is addressed how precise inferences from data can be. Insights gained in this section are not only important in practice, but also when assessing analysis techniques (Chapter 6).

1.4.2 Limits of Inference

Understanding a system under study is limited by the amount of information about it; any incomplete knowledge causes uncertainty with respect to missing aspects. These doubts can render multiple models explaining the system equally probable, and deciding which of them is more correct would require more information. If enough relevant facts were known, a definite decision on the correct model would be possible; however, generally such equivalence must be accepted due to a lack of data. Depending on how much information is available, ambiguity varies. This section points out three relevant kinds of equivalence, which occur at three different levels of information.

Causal, Observational, and Score Equivalence

In the most informed case it is possible to systematically manipulate the system under study such that all potential combinations of its states can be evoked. Any possible hypothesis on causal interaction can thus be tested; however, for reasonably complex systems it is often infeasible to explore the space of all its states. This can have several reasons, for example: the sheer number of possible states takes too long to explore; some states are impractical to be enforced; or evoking certain states alters causal relations through a learning processes. Certain hypotheses thus cannot be tested, such that causal interactions sometimes cannot be inferred completely. In such case, multiple equally plausible models might explain the system under study. This ambiguity is called causal equivalence [Verma and Pearl, 1990]:

> Two causal models are *causally equivalent* if there is no experiment that could distinguish one from the other.

Often such causal equivalence arises if the system is not fully observable (Fig. 1.4). The number of equivalent models generally increases with lower observability and with tighter restrictions on interaction with the system. In the most extreme case, the system under study cannot be manipulated at all such that

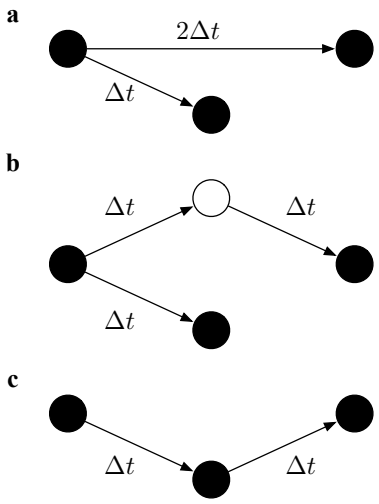


Figure 1.4: Networks visualising information flow between nodes from left to right with different time-lags. Filled nodes are observable while the white node cannot be observed. Three different networks are shown: **a** hub node with connections of different time-lags, **b** homogenous time-lags involving non-observed nodes, **c** chain structure among observed units. An experiment involving intervention can distinguish between alternatives (a) and (c) or between (b) and (c), but not between (a) and (b), which are causally equivalent. The figure further shows that correlation does not imply causation; mere correlation between observed nodes cannot distinguish between any of the three alternatives.

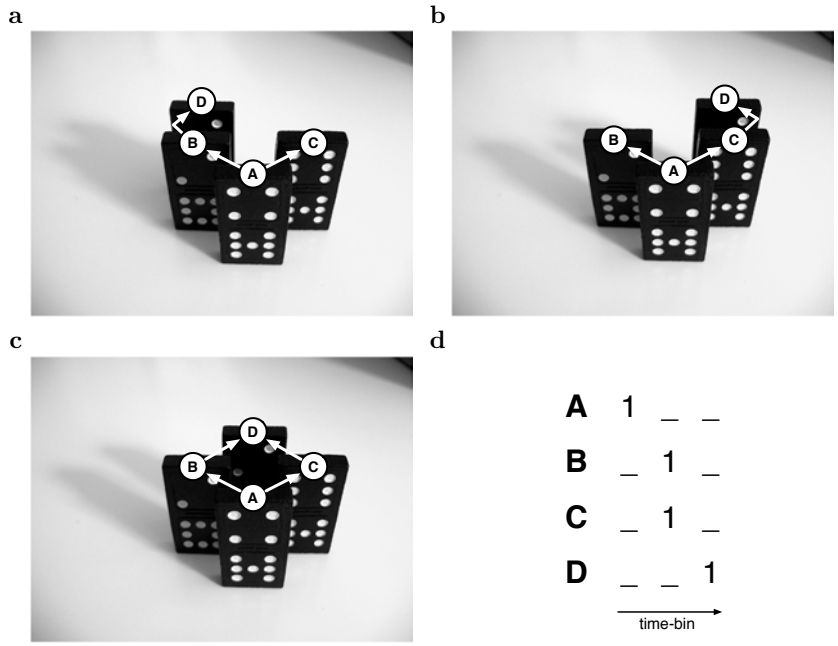


Figure 1.5: Example showing three observationally equivalent models. Graph superimposed on depicted dominoes indicates the chain reaction: Pushing domino A to cause it fall towards B and C will make them fall and trigger tilting of D. Domino D is arranged differently in the photos: **a** D behind B, **b** D behind C, and **c** D behind both B and C. **d** Event sequence for dominoes shown in a, b, and c (falling of a domino corresponds to signal 1). Although the causal chain between set-ups differs, they all produce the same data (d) and are thus observationally equivalent. The three set-ups are not causally equivalent, because an additional experiment — for example, pushing domino B instead of A — can reveal the true arrangement.

data collection limited to pure observation. This non-interfering situation will be considered throughout this work.

Without any intervention with the studied system, but by using purely observational data only, generally less information is available to infer causal relations. This increases uncertainty about the true mechanisms at work and manifests by a larger number of equivalent models. Equivalence resulting from passive data collection is defined as:

> Two causal models are *observationally equivalent* if the given data cannot distinguish one from the other.

Generally, the more models will be observationally equivalent the less information is available. For an example of observational equivalence consider Fig. 1.5.

Finally, the third kind of equivalence to consider can appear on top of either of the two previous ones. It is not directly related to the information that is given about the studied system, but about how well two models can be distinguished from each other. Assume that each model is assigned a score value dependent on the data. (The score of a model could reflect how well it explains the data, for example.) This score value fully characterises the model and no other criterion exists, which can be used to distinguish it from another. Now, two models with the same score value are considered equivalent:

> Two models are *score equivalent* if they have the same score value for the given data.

Depending on how discriminative score values are, information can be lost when reducing models to numbers, since any two models that are assigned the same score value become equivalent. To illustrate this loss of information consider two models A and B, which, on average, explain the data equally well. The difference between the two models is that A explains the first half of the data better than the second, and vice versa for model B. These two models are thus not observationally equivalent, but using their average performance as their score does not capture any difference between the two models. They are thus score equivalent.

Model equivalence can arise from an inability to manipulate the studied system (causal equivalence) or about doubts how to interpret the data (observational equivalence). Additionally, a lack of discrimination between models can render multiple models indistinguishable (score equivalence). When inferring models from data all three types of equivalence can cause ambiguity. The existence of multiple solutions rivalling for explanation poses the question of how to use them. They can, for instance, be processed with model averaging techniques (Section B.3): These methods can reveal a consensus between solutions, which might indicate that a disclosed similarity is indeed a feature of the studied system. However, if instead the number of solutions shall be reduced, a sensible criterion for selection is needed.

Occam's Razor

The principle of Occam's Razor goes back to William of Ockham (1285-1349) and is nowadays also known as law of economy, or law of parsimony [McHenry, 1993].[5] One translation of Ockham's doctrine is [Dowe et al., 2007]:

> *"Plurality should not be posited without necessity."*

[5] The principle is referred to as both Ockham's and Occam's Razor; since the latter variant is more common, it is used in this thesis.

and many re-formulations of the same principle exist. For example by Chatton and Einstein [Groarke, 1992, p.196]:

> *"Use as much explanation as necessary."*

and

> *"Our theories should be as simple as possible, but no simpler."*,

respectively. The concept of choosing the simpler of two equally good models [MacKay, 2003, pp.343] is commonly applied to (causal) probabilistic models [Pearl, 2000, pp.45]. Techniques presented in this thesis also follow the intuitive guideline *simpler is better* although this does obviously not guarantee to select the right, but only the simpler, model.

With the end of this section, preparations are complete to introduce two practical approaches to inference of stochastic relations. These follow in the next two chapters.

Chapter 2

Existing Analysis Methods

Different correlation analysis techniques for neural spike train data exist [Brown et al., 2004], which can be used to infer stochastic relations in order to construct graphical models (Chapter 1). Here, some of these methods are conceptionally related to the new method (the Snap Shot Score), which will be introduced in the next chapter. In this chapter, one of the existing techniques for learning (dynamic) Bayesian networks, called the BD score, will be discussed in detail in order to compare it to the novel method later (Chapter 5).

2.1 Classification of Existing Methods

Neural signal propagation can be modelled with graphical models in order to visualise the spreading of activity as information flow networks. These networks are generally constructed by analysing data with respect to neural correlations, which can indicate flow of signals. Diverse methods can be used for the detection of functional dependencies. Table 2.1 shows a selection of techniques that have been classified according to their characteristics, in order to indicate their differences. For example, one aspect approaches differ on is how they use the data: as spike times or transformations to the frequency domain, like firing rates, for example. Spike time methods utilise the full precision of the data while frequency methods smooth out minor variations, which could be attributed to noise. Another distinction concerns the experimental set-up during data collection: Some methods average over multiple trials and require several repeated recordings under same conditions. Others can be applied to single trial recordings, which is required in situations where conditions cannot be exactly replicated, as in freely moving animals, for example. Additionally, techniques differ in whether or not they require recordings from individual neurons. Methods fitting neuron models to the data require such single-unit data, i.e. all

spikes on one channel originate from exactly one neuron. Other approaches can handle non-spike sorted multi-unit data, which combines spikes of multiple neurons per channel. The final selected feature for categorisation contrasts the before-mentioned ones, as it does not concern the data to analyse, but the type of correlation the technique can identify. Some methods are restricted to the analysis between pairs of channels and might thus fail to pick up correlation that is conditional on more than one channel. Such more complex patterns can be detected by analysis techniques that account for multivariate correlation.

The method I created, the Snap Shot Score (Chapter 3), in particular differs to others listed in Table 2.1 with respect to one aspect: It cannot be classified as being based either on spike times or on frequencies. The Snap Shot Score aims to combine the advantages of both domains by using a mixture of these concepts, as will be seen later. Like other techniques, it can reveal multivariate correlation and it is applicable to both single and multi-unit spike train data without requiring multiple trial recordings. Details on this method are given in the next chapter.

Further discussion cannot cover all mentioned techniques in detail due to their wide conceptional range. The focus is therefore on methods that are explicitely designed to learn graphical models. Within this group of techniques approaches differ in how the quality of a particular network is determined, which could be done by a Bayesian concept or information theoretic ideas, for example. Some methods provide a scoring function, which can be used in order to rate networks. Examples of such scoring functions are the Bayesian Dirichlet (BD) score [Heckerman et al., 1995] or the minimal description length (MDL) [Wallace and Boulton, 1968, Lam and Bacchus, 1994], which is equivalent [Friedman, 1997] to the Bayesian information criterion (BIC) [Schwarz, 1978]. The BD score has several variants [Buntine, 1991, Heckerman et al., 1995], which are known as the K2 score, the Bayesian Dirichlet equivalence (BDe) score, and BDe uniform (BDeu) score. (For references to further methodologies please consult Buntine [1996].) The BD scores have been used to analyse different biological data [Murphy and Mian, 1999, Friedman et al., 2000, Perrin et al., 2003, Kim et al., 2004, Zou and Conzen, 2005, Smith et al., 2006, Junning Li and McKeown, 2006, Li et al., 2007, Rajapakse and Zhou, 2007, Rajapakse et al., 2008, Burge et al., 2009] and have therefore been chosen to be discussed in detail in this thesis. This group of scores is identified as the *family of BD scores*, which will be compared to the new method, the Snap Shot Score, later (Chapter 5).

Table 2.1: Classification overview of techniques that can be used for inference of neural information flow networks from electrophysiological data. Legend: ■ required/intended use, □ possible use, `NA` not applicable.

method (based on)	spike time /frequency	multiple trials	single-unit	multi-unit	correlation type
Joint Peristimuls Time Histogram (JPSTH)[a]	time	■	■	□	pairwise
Cross-Correlation[b]	frequency[k]	■	■	■	pairwise
Information Theory[c]	frequency[k]	■	■	□	multivariate
Single Neuron Model[d]	time	□	■	`NA`	multivariate
Gravity[e]	time	□	■	□	multivariate
Dynamic Bayesian Network (DBN)[f]	frequency (Hz)	□	`NA`	■	multivariate variate
Partial Directed Coherence (PDC)[g]	frequency (Hz)	□	■	■	multivariate
Generalised Linear Model (GLM)[h]	time	□	■	■	multivariate
Granger Causality[i]	frequency (Hz)	□	□	■	multivariate
Direct Transfer Function (DFT)[j]	frequency (Hz)	□	□	■	multivariate
Snap Shot Score (SSS)	both	□	■	■	multivariate

[a] [Gerstein and Perkel, 1969, Aertsen et al., 1989]

[b] [Perkel et al., 1967]

[c] [Rieke et al., 1999, Borst and Theunissen, 1999, Dayan and Abbott, 2005]

[d] [Nykamp, 2005, Makarov et al., 2005]

[e] [Gerstein et al., 1985, Gerstein and Aertsen, 1985, Lindsey and Gerstein, 2006]

[f] [Smith et al., 2006]

[g] [Sameshima and Baccalá, 1999, Baccalá and Sameshima, 2001, Astolfi et al., 2006, Takahashi et al., 2007]

[h] [Chornoboy et al., 1988, Okatan et al., 2005, Truccolo et al., 2005, Pillow et al., 2008]

[i] [Granger, 1969, Cadotte et al., 2008]

[j] [Kaminski and Blinowska, 1991, Eichler, 2006]

[k] Frequency meant in terms of a frequentist's probability estimate (p.6). Large amounts of data are generally required for these estimates, although not necessarily multiple identical trials.

2.2 The BD-Score Family

Scoring functions of the BD-score family have been successfully applied to learn dynamic Bayesian networks from different kinds of biological data: For example, gene expression profiles [Murphy and Mian, 1999, Friedman et al., 2000, Perrin et al., 2003, Kim et al., 2004, Zou and Conzen, 2005], fMR images [Junning Li and McKeown, 2006, Li et al., 2007, Rajapakse and Zhou, 2007, Rajapakse et al., 2008, Burge et al., 2009], and also neural electrophysiological multi-unit data [Smith et al., 2006]. The BD score is derived by a fully Bayesian approach, and it exists in different variants [Heckerman et al., 1995]: Depending on properties the score is desired to have and the prior probabilities of networks, the K2, BDe, or BDeu score can originate from the same equation. These variants will be discussed later to show how they especially differ with respect to the explicit incorporation of prior information into the assessment of networks.

The following section introduces the notation and variables that are necessary in order to specify the BD score, which is given thereafter.

2.2.1 Notation and Preparation

The BD score is proportional to the posterior probability of a Bayesian network with respect to some data.[1] Succeeding preparations therefore assume that a directed acyclic graph (DAG, Section B.1) G is given as well as a data set D. The number of vertices in G and the dimension of the data must be equal, such that each data-channel can be identified with one vertex. The data are assumed to be a set of m vectors d_t:

$$D = \{d_1, \ldots, d_m\} \ . \tag{2.1}$$

Let n denote the number of variables that were observed. Each data vector d_t then consists of n discrete[2] coordinate entries, each corresponding to one variable, i.e.

$$d_t = \left(d_t^{(1)}, \ldots, d_t^{(n)}\right), \text{ where } \ d_t^{(i)} \in \mathbb{Z} \ \text{ observation of variable } i \ . \tag{2.2}$$

[1] The BD score is actually a joint probability $P(G, D)$ of a graph G and some data D [Cooper and Herskovits, 1992, Chickering et al., 1995]. The score is generally expressed as a product of the graph's marginal probability $P(G)$ and the conditional probability $P(D|G)$ of the data given the graph, i.e. $P(G, D) = P(G) \cdot P(D|G)$. In this thesis the concern is to assess different graphs for some fixed data, such that the probability of interest is that of the graph given the data; but for fixed data D $P(G|D)$ is proportional to the joint probability, since $P(G|D) = P(G)\frac{P(D|G)}{P(D)} \propto P(G)P(D|G) = P(G, D)$.

[2] For practical application continuous measurements may have to be discretised. Associated problems are not discussed here, but the data are assumed to be in an appropriate form with few discrete values for each variable.

In the following, different variables are defined to determine statistics of the data, which will be used by the scoring function. The first quantity is r_i, which denotes the number of different states variable i takes over the whole data-set, i.e.

$$r_i = \# \left(\bigcup_{t=1,\ldots,m} \left\{ d_t^{(i)} \right\} \right) . \tag{2.3}$$

For simplicity, each of the r_i states is identified with an arbitrary but fixed number between 1 and r_i.

Next, the count of different potential states is determined for multiple variables together. The given graph G determines which variables are to be combined; namely, the parents of any node i. Using the described relation between a graph and the factorisation of a probability distribution (Section 1.2.1), the *parent set* pa_i is determined from G. Let p_i denote the number of node i's parents, which itself are denoted by $p_1^{(i)}, \ldots, p_{p_i}^{(i)}$; the corresponding parent set is then

$$pa_i = \left\{ p_1^{(i)}, \ldots, p_{p_i}^{(i)} \right\} \subseteq \{1, \ldots, n\} . \tag{2.4}$$

The next entity to determine is the number of different joint states the variables in the parent set pa_i can take. In other words, if all variables that are not in the parent set are omitted from each data vector, i.e. the reduced vector

$$d_t^{(pa_i)} = \left(d_t^{\left(p_1^{(i)}\right)}, \ldots, d_t^{\left(p_{p_i}^{(i)}\right)} \right) \tag{2.5}$$

is considered, how many different sub-vectors could be observed? Counting such potential *joint parent states*, or *joint states* for short, is straightforward with equation (2.3); the number of different joint states q_i is

$$q_i = \prod_{p \in pa_i} r_p \tag{2.6}$$

and $q_i = 1$ if $pa_i = \emptyset$. Finally, in order to define the statistics for the BD score, each joint parent state is identified with an arbitrary but fixed number between 1 and q_i. The preparations are now complete.

2.2.2 Sufficient Statistics of the BD Score

Using the agreed notation we turn towards the actual score calculation. Assume that suitable data have been prepared and that we are given a network to score. The data is analysed according to the procedures outlined before and each node's states are assigned numbers $(1, \ldots, r_i)$; likewise for all joint parent states

$(1, \ldots, q_i)$. The sufficient statistics[3] for the BD score are counts of combined child-parent states, i.e. how often node i was observed in a particular state k while its parents were in joint state j. These numbers are represented by N_{ijk}, which can be formalised as

$$N_{ijk} = \# \left\{ t \in \{1, \ldots, T\} \mid d_t \in D : d_t^{(pa_i)} \hat{=} j,\ d_t^{(i)} \hat{=} k \right\} . \tag{2.7}$$

Using this count one can easily calculate the number of times node i's parents were in state j — regardless of the state of the child:

$$N_{ij} = \# \left\{ t \in \{1, \ldots, T\} \mid d_t \in D : d_t^{(pa_i)} \hat{=} j \right\} = \sum_{k=1}^{r_i} N_{ijk} . \tag{2.8}$$

These numbers are then used in order to determine the score value according to the formula given in

Definition 1 (BD score) *For the data D, the BD-score of a graph G is*

$$\begin{aligned} BD(G|D) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})} \\ \textit{where} \quad N'_{ij} = \sum_{k=1}^{r_i} N'_{ijk} . \end{aligned} \tag{2.9}$$

Formula (2.9) contains yet undiscussed entities: the prior $P(G)$ and the pseudo counts N'_{ijk}. These terms incorporate prior knowledge into the score calculation. How prior information translates to the missing terms N'_{ijk} is discussed next.

2.2.3 Integration of Prior Information

Often, previous investigations of the studied system have revealed insights, which should not be ignored when analysing data with the BD score. Instead, the score should have a tendency to assign higher scores to networks that match the prior information. This bias towards consistent network structures should be overcome if the data give strong indications towards other networks that did not seem to be likely beforehand. As a Bayesian approach, the BD score facilitates the combination of evidence in the data and prior information. Therefore the latter has to be expressed in two parts: the prior on network structures and the prior on the data any graph is associated with.

[3] The sufficient statistics of a model are the minimal set of parameters that fully specify it. From the model's point of view, any additional statistics are irrelevant. The sufficient statistics thus determine which information is extracted from the data to be used; any data-features that are not conveyed by these statistics remain unknown to the model. See Jaynes and Bretthorst [2003, Chapter 8] for a detailed discussion and examples.

The term $P(G)$ is the prior on network structures G; it expresses the belief that the network G is plausible in the context of the data. Previous work, for instance, might indicate interaction between two variables, such that networks which contain corresponding links are assumed to be more likely than those that lack dependencies between them. Further interpretation and the role of the prior has been subject in section 1.4.1; the following discussion is thus turned towards the likelihood.

The likelihood $P(D \mid G)$ of a network G is the probability that the data D are observed, given that the network is correct (Section 1.4.1). The user of the BD score does not employ that probability to the scoring function directly; instead, a likelihood function is parameterised by the terms N'_{ijk} (so called hyper-parameters). (For details on assumptions shaping the likelihood function the reader is referred to the literature [Heckerman et al., 1995, Heckerman, 1997, Pe'er, 2005].) The pseudo counts N'_{ijk} correspond to the data a particular network is expected to generate and how a network translates to its expected counts is considered next. For any network G, the user needs to specify the probability of any parent-child state combination that can occur in this network. For each node i this yields a probability distribution

$$P_i(\text{parents } pa_i \text{ in state } j, \text{child } i \text{ in state } k) = P_i\left(d_t^{(pa_i)} \mathrel{\hat{=}} j,\ d_t^{(i)} \mathrel{\hat{=}} k\right) , \quad (2.10)$$

which is specific to the network G. Additionally, the user must specify a weight, which expresses the confidence in the prior information P_i $(i = 1, \ldots, n)$ relative to the data to analyse: the *equivalent sample size* N'. The number N' expresses the reliability of prior information in terms of data-points and thereby allows to adjust between relying more on prior information or on the data. Setting the equivalent sample size to the number of data points m corresponds to balance between the two sources of information. A relatively large equivalent sample size indicates very reliable prior information, for which the influence of the data on the network's score is marginal. In contrast, with no prior information available at all, i.e. the equivalent sample size is zero, scoring solely depends on the data. Together with each node's probability distribution P_i [equation (2.10)], the equivalent sample size determines the missing pseudo counts N'_{ijk}:

$$N'_{ijk} = N' \cdot P_i\left(d_t^{(pa_i)} \mathrel{\hat{=}} j,\ d_t^{(i)} \mathrel{\hat{=}} k\right) . \quad (2.11)$$

These terms can be understood as the expected number of joined state observations for a particular graph G weighted by the confidence about the expected data. Given the pseudo counts N'_{ijk}, the likelihood function is fully specified

and can be used to evaluate the network.

All entities in the BD score formula (2.9) are now known, such that networks can be scored. In order to apply the score in practical dimensions, however, the assignment of priors must often be generalised due to the large number of networks (Section B.2). In situations where too many networks exist in order to specify priors by hand, fundamental assumptions can be made in order to automatically derive the pseudo counts for each network. Different approaches to these prior assignments resulted in variations of the BD score, which are presented next.

2.2.4 BD Score Variants

The first special case of the BD score to consider is the so called K2 score.[4] It arises from the BD score when an uninformative prior on state combinations is used [Heckerman et al., 1995]. This means that the prior probabilities in equation (2.10) and the equivalent sample size N' are defined such that the expected counts for state combinations fulfil

$$N'_{ijk} = 1, \quad \text{and hence,} \quad N'_{ij} = r_i \ . \tag{2.12}$$

For a given network any child-parent state combination is thus equally likely; but networks whose nodes have fewer parents are implicitly assigned a higher likelihood. The counts N'_{ijk} and N'_{ij} are integer values, which can be used in order to simplify the formula of the BD score (2.9) by substituting the Γ-functions with factorials via the relation [Bronstein et al., 1999, p.456]

$$\Gamma(x+1) = x!, \quad \text{for} \quad x \in \mathbb{N} \ . \tag{2.13}$$

The resulting variation of the BD score is called the K2 score, which has been proposed by Cooper and Herskovits [1992] before the more general BD score was known.

Definition 2 (K2 score) *For the data D, the K2-score of a graph G is*

$$K2(G|D) = P(G) \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}! \tag{2.14}$$

[4] The BD score and its variants are measures of how well a network structure fits the data. Therefore, these scores are sometimes referred to as *scoring metrics* or *Bayesian scoring metrics* (e.g. [Heckerman et al., 1995]). Metrics are commonly defined on a set X for which the distance between *any* two points in X can be measured (e.g. [Heuser, 2000, pp.81]). The BD score cannot measure the distance between two networks, for example, and it is thus not a metric. Throughout this thesis, functions to assess network structures are therefore called *scores*, which is in accordance with other publications (e.g. [Pe'er, 2003]).

With the K2 score prior information can only be incorporated via the prior on network structures $P(G)$. Since no further parameters exist, the K2 score is easy to use and, as will be seen later, its simplicity facilitates a straightforward interpretation of the BD score family. However, the simplifications of the score come at a cost: it lacks a property called *likelihood equivalence*.[5] Another variant of the score possesses this property: the BDe score (e for equivalence) [Heckerman et al., 1995]. However, this score is not considered further, but a sub-case of it, the BDeu score (u for uniform distribution) is presented. This form of the score arises by using another uninformative prior, by which the expected counts are set to

$$N'_{ijk} = \frac{N'}{r_i q_i}, \quad \text{and hence,} \quad N'_{ij} = \frac{N'}{q_i} \ . \tag{2.15}$$

For any network every child-parent state combination is assumed to be equally likely, but different to the K2 score, the pseudo counts N'_{ijk} are normalised such that every network has the same likelihood. Also, this assignment of pseudo counts does not imply a preference of sparser networks by the likelihood function; however, by specifying the prior on network structures $P(G)$, the user can still direct the score.

The previous sections were concerned with the use of the BD score. Next, a short analysis of the score shows its working principles from a practical perspective.

2.2.5 A Brief Characterisation of the BD Score

The BD score is a fully Bayesian approach, which calculates the likelihood of a graph for given data. Its mathematical derivation fills several pages, which can be found in the literature [Heckerman et al., 1995] and will therefore not be replicated here. Instead of replicating all steps that lead to the scoring function in definition 1, the formula will be explored in a reverse engineering manner in order to get insights into its machinery.

The assignment of priors has already been discussed (Section 2.2.3), such that the following analysis concentrates on how the statistics N_{ijk} affect the score value. For this purpose it is most convenient to survey the K2 score (Definition 2), which has the simplest formula. Insights into the K2 score's working principle generalise to the other members of the BD score family because their difference is in prior assignments only; otherwise they are the same. From equation (2.14) the factors that determine the score contribution of a single node i

[5] Heckerman et al. [1995] introduced the postulation that any two network structures with identical assertions of conditional independence should have equal likelihoods. This property has been termed *likelihood equivalence*.

can be identified as

$$\prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} N_{ijk}! \;\leq\; 1 \;. \tag{2.16}$$

Knowing that the K2 score is a product over probabilities of each node's parent configuration it is evident that the inequality (2.16) is true; however, it is also possible to show this directly (Appendix C). Understanding why the inequality holds without using knowledge about its probabilistic origin helps to derive the conditions which must be met in order to yield a particularly high or low score value. In the next two paragraphs these conditions will be identified, making use of the fact that the faculty is the most quickly increasing function in equation (2.16).

First, equation (2.16) is analysed to see what facilitates a high score value for any node i. The role of three terms involved can be quickly revealed: The first term $(r_i-1)!$ is solely determined by the number of different states the child node i takes in the observed data. This number is thus the same constant for any network. In contrast, the terms $N_{ijk}!$ and $(N_{ij}+r_i-1)!$ depend on the structure of the network, since the counts N_{ijk} and N_{ij} can be different for two differing networks. This is because the network structure determines whose variables' states are considered jointly [equations (2.5) and (2.7)]. The counts can therefore vary if different channels of the data or even a different number of channels are used to determine them. For a high score the product $\prod_{k=1}^{r_i} N_{ijk}!$ needs to be maximised. The highest value possible would indeed be taken if all factors were equal to 1 except one; i.e. if all except one of the counts N_{ijk} were equal to 0. With respect to the data this means that whenever node i's parents were observed in state j the child node was always in state k, i.e. the state of the parents has always evoked a particular state of the child. The product of factors $N_{ijk}!$ is thus maximal if the child state is a deterministic function of the joint parents state.[6]

On the other hand, a low score value would arise if the ratio between the product $\prod_{k=1}^{r_i} N_{ijk}!$ and the denominator $(N_{ij}+r_i-1)!$ is small. This requires the value of the product to be minimal, which is achieved when all counts N_{ijk} are about equal, since the factorial is increasing faster than the product of the resulting factors. Thus, if the child states are not affected by the parent state, i.e. their conditional distribution is a uniform distribution, the product is minimised. This is because the denominator is only affected by the structure of the network, but not the child's conditional distribution, as this does not affect

[6] If the state of a child node is a deterministic function of its joint parents' states, the probability distribution of the child conditioned on its parents corresponds to a Dirac-δ-function [Bronstein et al., 1999, p.712].

the counts N_{ij}. Thus, if there is no relation between the state of the parents and the child, i.e. the child is independent, the score value is lowest.

From the formula of the score it can also be seen that the number of links in the network influence the score value; namely, networks with many links are less likely to have a high score. This is because the conditions for a high score value (as discussed above) must be met for every joint state j of the parents. The more parents a node has, the more joint states exist for which the conditions must be fulfilled. If any of the joint states yields a low score value this affects the overall score via the product over joint states j. In general, if only few joint states exist, it will be more likely that each of them contributes a high factor to the score than if there are many. Since the number of parents determines the number of joint states, sparser networks, which only have few parents per node, are more likely to be high scoring.

To summarise, the K2 score generally favours sparse graph structures with parent child relations for which the joint state of the parents is a reliable predictor of the child state. Correspondingly, complex graph structures with many links and the ones for which the child states are found to be independent of joint parent states have low score values. As already mentioned, these characteristics are shared by the other BD scores as well.

The BD score family has been introduced and characterised, but nothing has been said about how these scores can be used to learn other than static Bayesian networks. In the next section it will be discussed how the BD scores can be applied in order to analyse the dynamics of the data.

2.2.6 Using BD Scores to Learn DBNs

Static Bayesian networks can represent stochastic relationships between variables (Section 1.2.1). The intention behind the BD scores is to learn such models, but they can also infer dynamic Bayesian networks. However, for learning DBNs, the BD score cannot be applied directly to the data, but it must be transferred in order to present time-lagged interactions as instantaneous ones. Details are discussed in this section.

As has been outlined earlier, the BD score favours network structures for which the parents of a node are good predictors of the child variable's state. In order to assess to which degree this is the case for a particular network, the counts of state combinations N_{ijk} are used to calculate a score value (see previous sections). All information the score uses about the data is encoded in these counts; they are the sufficient statistics of the score, which means that any information that is not conveyed by these counts is hidden to the score.

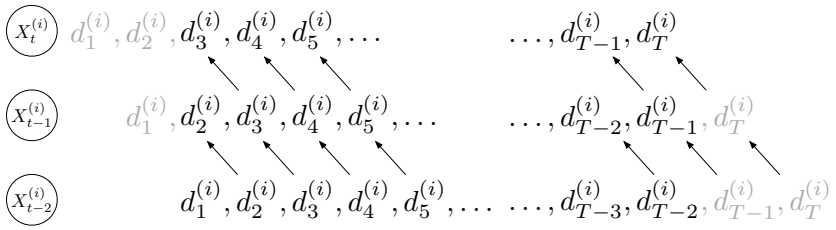


Figure 2.1: Shifted data duplication to facilitate learning of a m^{th} order DBN with the BD score. The original data of unit i are assumed to be ordered in time to represent a time-series $\left(d_t^{(i)}\right)_{t=1,\ldots,T}$. Each variable i is represented $m+1$ times: $X_{t-k}^{(i)}$, once for each time-layer $k = 0, \ldots, m$. The data for each variable $X_{t-k}^{(i)}$ are given by the shifted time-series $\left(d_{t-k}^{(i)}\right)_{t=m+1,\ldots,T}$ shown in black for a 2^{nd} order model.

Looking at how these counts are defined [equation (2.7)] shows why they do not convey any information about time-lagged effects between variables: These are the number of observations of a particular child state and a joined state of its parents; a combination of states, which occurred simultaneously. No information about temporal relationships is extracted from the data, such that the BD score cannot be directly be applied in order to learn DBNs. However, a little trick makes it possible to incorporate temporal information into the sufficient statistics of the score without changing them. DBNs can thereby be learned with the BD score and the procedure in order to facilitate this, as well as its drawbacks, is discussed next.

In a DBN each variable is represented multiple times; once for each time-layer of the model (Fig. 1.3bc). The state of all variables at a particular point in time can be derived from time-series data by time-shifted data-duplication (Fig. 2.1): Implicitly, the data are duplicated and shifted in time such that successive data points of one variable become aligned for its different temporal representations. This imaginary transformation facilitates the application of the BD score on these new data, which are informative about temporal relations: Time-lagged effects are represented within a single time-layer. These data are used to score a DBN by determining the corresponding statistics N_{ijk}. The statistics are (still) the number of times a joint parent state was observed together with a particular child state; but, as states at different times are now represented as simultaneous events, the counts can reflect variable state-combinations across different time-layers. By duplicating and shifting the data, the BD score can thus be used to learn DBNs.[7]

[7]For practical implementation the data are not actually duplicated and shifted, but it is more efficient to reference to the original time-series with an appropriate time-offset.

The time-shifted data-duplication procedure maps variables from different time-layers into a common layer. Any acyclic network connecting the corresponding nodes can be scored with the BD score; however, not every of these may be plausible. Graphs connecting nodes across time-layers with links directing backwards in time contradict common understanding of causality, which implies that effects cannot precede their causes. Hence, networks suggesting a variable in the present to be influencing variables in the past are absurd in this sense. Such implausible networks should thus not be considered to be scored. The computational costs saved by not evaluating incongruous networks is heavily needed for the large number of remaining candidates (Section B.2).

The total number of potential networks indeed becomes a serious issue for high dimensional data and/or higher order DBNs. This is because the number of networks grows super-exponentially in the number of variables to consider [equation (B.6)]. For DBNs the total number of variables is the number of channels in the data times the number of time layers to model. Thus, modelling data with a 3^{rd}-order model instead of a 2^{nd} order one, for example, can already lead to a significant increase in the number of networks to score [equation (B.7)]. For practical dimensions, generally far too many networks exist in order to score all of them; heuristics or sampling methods are thus needed to select a fraction of the networks to score (Appendix D). Computational constraints can therefore hinder the inference of stochastic relations from high dimensional data in the form of higher order DBNs. Indeed, it has been shown that learning the structure of a BN is NP-complete when the BDe score is used [Chickering, 1996]. This means that no algorithm is known to solve this problem in polynomial-time [Cormen et al., 2001, Chapter 34]. Learning DBNs can be expected to be similarly complex, since, by the procedure outlined above, these networks are presented as BNs to the score.

Computational demands limit the applicability of the BD scores to high dimensional data in practice. Therefore a new score has been developed, which is introduced in the next chapter. It has been designed with problems of high dimensionality in mind and it reduces computational costs by considering a less precise model than a DBN. The new score is not as flexible as the BD scores, because it is specifically adapted to spike train data; but its sufficient statistics efficiently capture multiple time-lags, by which it becomes applicable to practically sized data-sets.

Chapter 3

The Snap Shot Score

This chapter will present the Snap Shot Score (SSS), which is a novel technique that has been designed to learn information flow networks from spike train data. The score is introduced and briefly interpreted in the following section. Thereafter, examples follow, which demonstrate some of its characteristics.

3.1 Introduction of the Snap Shot Score

The Snap Shot Score, which is introduced later in this section, can be used for correlation analyses of spike train data. Detecting and quantifying stochastic relations in neural data has been performed with a variety of different techniques (Table 2.1, p. 27). Even graphical models, namely DBNs, were inferred from multi-unit data using the BDe score [Smith et al., 2006]. However, up to date no publication seems to report the application of the BD scores for the analysis of real spike train data;[1] presumably, because these data are not suitable for a direct application of the scoring functions (Chapter 5). Although the BD scores have proven their applicability for different data types, it seems that the special characteristics of spike trains hinder their broad application in this domain. In order to infer information flow networks from this special kind of binary data, the Snap Shot Score (SSS) has thus been developed.

Similar to other scoring functions the SSS can account for dependencies over multiple time-lags. This is achieved by converting each spike train with a low-pass filter to an activity level series: All spike times are preserved in the activity level series and additionally — in inter spike intervals — it is enriched by information about past neural activity. The actual score values are then calculated

[1] So far the only publication of which I am aware applying DBNs to (simulated) spike train data is Eldawlatly et al. [2008], who used a coarse representation of 3 msec time-bins for the data. The large time-bins are likely to have prevented the problems associated with the imbalance in numbers of spiking to non-spiking bins (Section 5.2.1).

using both the spike trains and activity level series, by taking snapshots of the activity level at all spike times. Multivariate correlation, i.e. situations in which an effect has multiple causes, is accounted for by joining several activity level series before calculating the score. The mathematical details of the SSS are given next.

Consider spike trains of n channels being given by the n-dimensional time series $s = (s_{k,t})_{t=1,\dots,T}^{k=1,\dots,n}$ with $s_{k,t} = 1$ if a spike was detected at time t on channel k and $s_{k,t} = 0$ otherwise. Corresponding *activity level series* $a = (a_{k,t})_{t=1,\dots,T}^{k=1,\dots,n}$ are defined by

$$a_{k,t} = \max_{j=0,\dots,t-1} s_{k,t-j} - j \cdot d \tag{3.1}$$

for some *decay constant* $d \in [0, 1]$. In most of the examples shown the decay constant is considered to be $d = 3^{-1}$, such that the activity level of channel k at time t is determined by $s_{k,t}, s_{k,t-1}$, and $s_{k,t-2}$, only.[2] The further a spike occurred in the past the less influence it has on the activity level, because the weight of a spike $1 - j \cdot d$ decreases as j increases. Spikes in the more recent past (i.e. a smaller j) or the present ($j = 0$) have highest weights. Due to the maximum taken in equation (3.1), the spike with the highest weight supersedes any others (that occurred earlier). Spikes whose activity is fully decayed do not contribute to the activity level at all. In detail, after $\lceil 1/d \rceil$ time-bins the weight becomes negative, such that any subsequent spike or even current silence (value 0) is selected by the maximum instead.

The *joined activity level series* $a_{(k_1,\dots,k_m)}$ of channels $k_1, \dots, k_m$ is defined as the maximum over channels for each time t:

$$a_{(k_1,\dots,k_m),t} = \max_{j=1,\dots,m} a_{k_j,t} \ . \tag{3.2}$$

For $m = 1$, the joined activity level series (*join*) is identical to the activity level series of the single channel. Joins and activity level series of single channels are both identified with the term activity level series. After these preparations the scoring function can be introduced in the following

Definition 3 (Snap Shot Score) *For a given pair (a, s) of activity level series $a = (a_t)_{t=1,\dots,T}$ and a spike train $s = (s_t)_{t=1,\dots,T}$ the Snap Shot Score is*

[2] Other choices for the decay constant are possible: the smaller d is chosen, the larger the range of time-lags considered for correlation detection. Extreme values where $d = 0$ (activity level constantly 1 once a spike occurred on the channel) or $d \approx 0$ (activity decaying extremely slow) are unlikely to deliver sensible results. The decay constant has been chosen $d = 1/3$ to keep examples expressive and clear. For real data the decay constant can be derived from the anticipated maximal correlation lag (in time-bins) or by a using a parameter series, as shown in chapter 7, later.

defined as

$$\mathrm{SSS}\,(a, s; \Delta t) = \frac{\sum_{t=1}^{T-\Delta t} a_t \cdot s_{t+\Delta t}}{\sum_{t=1}^{T-\Delta t} a_t} \tag{3.3}$$

if $\sum_{t=1}^{T-\Delta t} a_t \neq 0$*, and 0 otherwise. The parameter* $\Delta t \in \mathbb{N}$ *is called the shift constant; it defines the minimal time-lag with which causal effects occur.*

In this thesis simple examples use shift constant $\Delta t = 1$, i.e. at least one time step lies in between a cause and its effect. Larger time-lags are considered in more complex situations, which come along with practical guidance for choosing the shift constant (Chapter 7).

The SSS quantifies the excitatory effect of an activity level series on a spike train (Fig. 3.1): The score value is determined by spikes occurring within the *lag-window* $W \subset \mathbb{N}$ defined by the

$$\textit{minimal response lag} \;\; \Delta t \;, \tag{3.4}$$

which is equal to the shift constant, and the

$$\textit{maximal response lag} \;\; \lceil d^{-1} \rceil + \Delta t - 1 \;. \tag{3.5}$$

Spikes that occur during nil activity (i.e. $a_t = 0$) are not assigned any weight (since $a_t \cdot s_{t+\Delta t} = 0$) and thus do not contribute to the score value. (How this can be interpreted biologically is discussed in section 5.2.2, later.)

Formal definitions associated with the SSS are thereby complete. It remains to be explained how the score can be practically used to learn information flow networks; this will be described in section 3.1.2. But before that, the following section gives some background information on the SSS in order to understand its relation to stochastic processes (Section 1.3) and neuron models (Section A.1).

3.1.1 Interpretation of the Snap Shot Score

The SSS can be easily interpreted by dividing both the numerator and denominator in equation (3.3) by $T - \Delta t$ to render both terms time-averages or frequencies. Understanding these frequencies as probability estimates, the left hand side is a conditional probability [Feller, 1950], such that the scoring function informally reads as:

$$P\left(\begin{array}{c}\textit{spikes will follow,}\\ \textit{given that activity is high}\end{array}\right) = \frac{P(\textit{spikes following high activity})}{P(\textit{high activity})} \;. \tag{3.6}$$

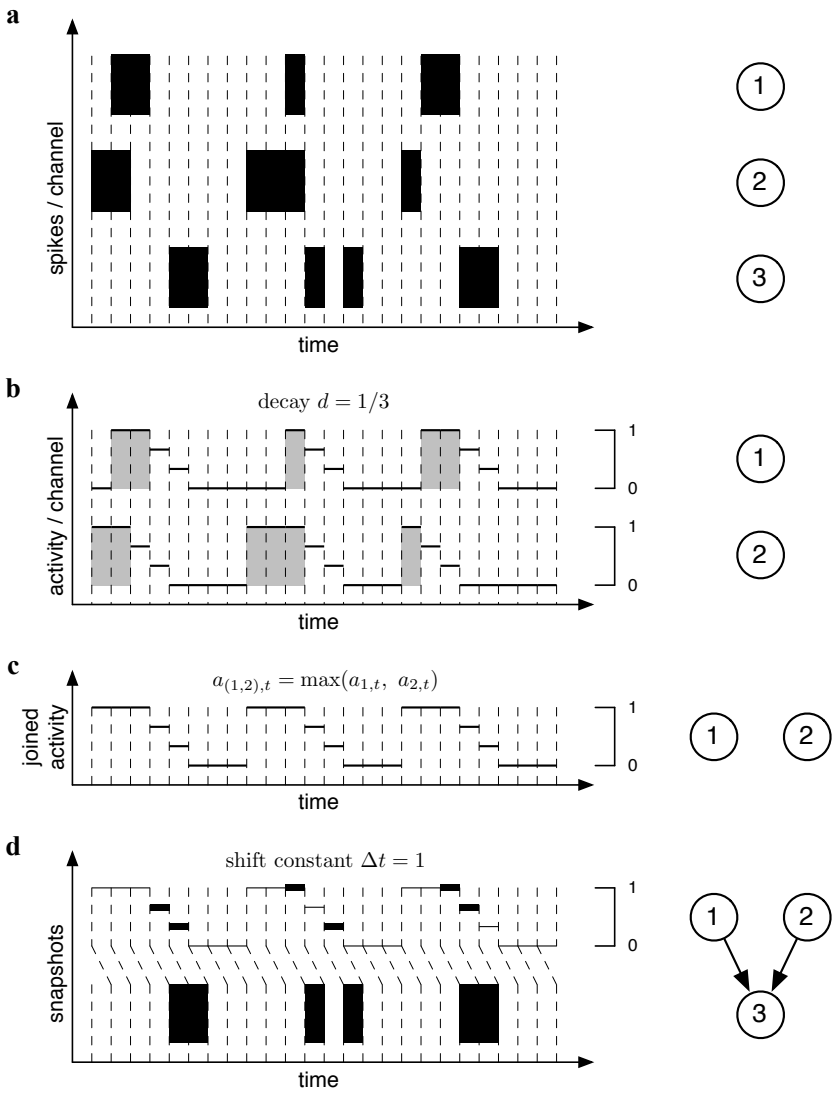


Figure 3.1: Graphical interpretation of the Snap Shot Score. Spike trains and activity level series on the left; corresponding network nodes on the right. **a** Spike trains for three channels. The channel on the bottom is excited by the upper two channels. **b** Activity level series of upper two channels for given activity level decay (decay constant $d = 1/3$). **c** Joined activity level series (join) of upper two channels. **d** Snapshots of joined activity level series taken at spike times of bottom channel corrected by shift constant ($\Delta t = 1$). Normalising the snapshot-values ($\frac{2}{3} + \frac{1}{3} + 1 + \frac{1}{3} + 1 + \frac{2}{3} = 4$) by the accumulated joined activity ($1 + 1 + 1 + \frac{2}{3} + \frac{1}{3} + 1 + 1 + 1 + \frac{2}{3} + \frac{1}{3} + 1 + 1 + 1 + \frac{2}{3} + \frac{1}{3} = 12$) yields the SSS value $\left(\frac{4}{12} = \frac{1}{3}\right)$ of the parent-configuration of node 3 shown right next to it.

The model underlying the SSS assumes that the spike trains are generated by a stochastic process $\mathbf{X}_t = \left(X_t^{(1)}, \ldots, X_t^{(n)}\right)$ where each neuron's future activity depends on past neural activity, i.e.

$$P\left(X_{t+1}^{(i)} | \mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_0\right) \ . \tag{3.7}$$

As discussed below, the SSS can be used to learn a network, which gives more specific information about the process $\mathbf{X}_t$: First, dependencies over time are limited by the score's lag-window W and secondly, parent child relations in the learned network describe which subset of neurons $pa(X^{(i)})$ was found to be relevant for each neuron $X^{(i)}$; namely its parents. Equation (3.7) thus simplifies to

$$P\left(X_{t+1}^{(i)} | pa(X^{(i)})_{\bar{t} \in \{(t+1)-l \ | \ l \in W\}}\right) \tag{3.8}$$

and describes the process $\mathbf{X}_t$ more concretely. Equation (3.8) does not imply one particular neuron model (Section A.1) in order to interpret learned networks. Instead, any model for which this process is a reasonable characterisation can be chosen. This could for instance be a leaky integrate and fire neuron where the leakage current is chosen such that temporal summation of synaptic inputs only occurs over a time-window corresponding to the score's lag-window W. Stochastic dependencies in the resulting spiking process $\mathbf{X}_t$ would then match equation (3.8). Changing assumptions about the neuron model leads to a distinct understanding of the network. The large number of possible models prevents discussing all of them; an easily interpretable template model is thus presented in which the learned links between observed units stand for chains of hidden units. The aim of this model is to give an illustrative example of how to interpret recovered links in neural terms; it is not suggested to reflect realistic connectivity. The model is formed by the following assumptions:

- All neurons act as unreliable relay units, i.e. spikes received through synaptic transmission are forwarded to connected neurons with a certain probability.
- All postsynaptic potentials are excitatory and synaptic transmission takes one time-bin per synapse.
- Each recovered link between observable neurons represents a connection between these neurons via a number of synapses (*connection length*). This number of synapses ranges from Δt (minimal response lag) to $\lceil d^{-1} \rceil + \Delta t - 1$ (maximal response lag). The connections are formed by chains of hidden units; one less than synapses in the chain. For shift $\Delta t = 1$ and decay constant $d = 3^{-1}$ (as chosen before) there are 1 to 3 synapses between

connected units, i.e. observable neurons either connect directly to each other or by up to 2 hidden units in between.

This descriptive model gives a simple interpretation for links, which allows explanation of the SSS's tradeoff between model complexity and explanatory power. In detail, the following examples 5 and 6 will show that making a node's parent configuration more complex lowers the score value, unless compensated by explanatory benefit. In terms of the model, complexity means two things: (1) the number of neurons upon which a neuron's firing is dependent and (2) the number of hidden units in a chain that correspond to each particular link. As will be demonstrated, the score generally favours configurations with fewer parents, but also, units are favoured, which respond with minimal rather than maximal response-lag. Thus, the SSS aims at explaining the dependence among (observed) units by linking them using few and short connections.

3.1.2 Learning Networks Using the Snap Shot Score

Potential information flow networks are assessed by identifying each data channel with one network node. Every node is then assigned a score value depending on the nodes linked to it. A child-node is scored by applying the SSS to the join of all parent-channels and the child's spike train.[3] If a node does not have any parents, its score value requires the join $a_{(1,\ldots,n)}$ of all channels. With the child's spike train s, the score of the parent-less node is $SSS\left(a_{(1,\ldots,n)}, s; \Delta t\right)$ if this value is non-zero, and 1 otherwise. Finally, the score of the full network is the product of all its nodes' scores.

Learning an information flow network from data generally involves scoring many potential structures. Ideally, the highest scoring one would be found. Because of the score's *decomposability* (Section E.1.1), the best scoring network can be assembled from each node's best scoring parent configuration. Thus, full network scores need not to be calculated for learning, but it is sufficient to determine each node's optimal parent configuration. In order to identify these with certainty, all 2^n possible joins for each node would have to be evaluated (Fig 3.2).[4] However, for practical dimensions (like a 60 electrode array, for example) there are far too many joins for an exhaustive evaluation (Section B.2.1). To circumvent this problem, the set of information flow networks to score can be limited to those with sparse connectivity or by limiting the number of parents per node. The number of potential child-parent relations might also be reduced by excluding connections ruled out by factual knowledge (like large physical

[3] Recall that a link's source node is called a *parent* of the destination node (*child*) (Section B.1). A loop-link renders a node parent and child at the same time. Such configurations will be referred to as *self-exciting*.

[4] For notes on efficient software implementation please consult appendix E.

distance between electrodes, for example). Additionally or alternatively, search heuristics and Monte Carlo methods can be used to select promising configurations to assess. For a discussion of suitable techniques see appendix D.

Incorporating Prior Knowledge

Network inference can be assisted by prior knowledge about the studied system. Here available information will be used in order to derive a separate *link-acceptance-threshold* (LAT) for each network node. During network learning any parent configuration with a score value lower than the child node's LAT will be rejected. This selection removes irrelevant links and can lead to sparser, more relevant networks.

The LAT is chosen to reflect the best explanation for the data at a particular level of complexity. The actual level of complexity is determined by the prior information at hand: Knowledge about the studied system constrains the space of potential parent configurations for each node. For example, self-excitation might be excluded or observed units are known to only have few interaction partners. The space of potential configurations can thus be restricted to a particular level of complexity, i.e. number of parents. Configurations at the highest permitted level of complexity determine the LAT, which is the highest score value of these configurations. This highest scoring configuration (*LAT-configuration*) reflects the best explanation for the data at a level of complexity, which could not be limited further by using prior knowledge. In other words, the LAT-configuration represents all background information formulated in terms of the SSS. Better explanations than the LAT-configuration might exist; these are simpler configurations with scores equal or above the LAT. Calculating the SSS for several parent configurations might reveal such superior explanations. Ultimately, we seek to find the simplest among the best scoring configurations consistent with prior knowledge. Thus, any configuration with a score value below LAT should be omitted from result lists, as it gives a worse explanation for the data than prior knowledge, i.e. the LAT-configuration. For a demonstration of how to determine the LAT consider the following

Example 4 (LAT) *The spike trains shown in Fig. 3.1a are analysed using the SSS ($d = 3^{-3}$, $\Delta t = 1$). Assume that no prior information is available, which facilitates any restriction of potential parent configurations. We thus consider all possible configurations including self-excitation of units (Fig. 3.2). In order to determine the LAT for node 1 its most complex parent configurations must be found and scored. Here, only one configuration of highest complexity exists; namely that where nodes 2, 3, and node 1 itself are parents of node 1 (Fig. 3.2 top right). This configuration's SSS value is* 0.26*, which is thus node 1's LAT-*

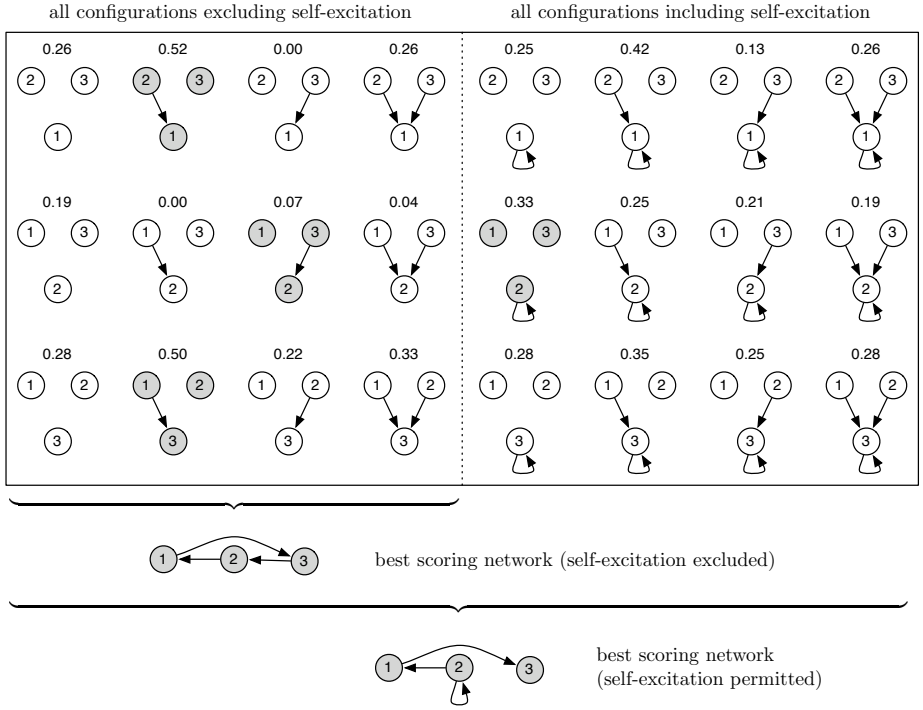


Figure 3.2: Exhaustive evaluation of all possible parent configuration of 3 nodes (top) for spike trains shown in Fig. 3.1a. Each row shows the configurations for one child node with the rounded SSS value above each ($d = 3^{-3}$, $\Delta t = 1$). All configurations excluding self-excitation are located on the left half; considering self-excitation doubles the number of configurations (left and right part together). Best scoring parent configurations (high-lightened in gray) combined to full networks (bottom). Depending on whether or not self-excitation is considered, top-scoring networks differ.

threshold. Similarly, for nodes 2 and 3 we find thresholds 0.19 *and* 0.28 *respectively. The best scoring parent configurations of each node exceed each node's LAT; they are thus all included in the combined network (Fig. 3.2 bottom). (Otherwise, i.e. if the LAT was not out-valued by the best scoring configuration, corresponding links would be omitted.)*

The preceding sections showed how to interpret and how to use the SSS. It has already been noted that the score favours sparse networks, but no practical demonstration of this and other features has been given yet. The next section uses simple examples to illustrate the score's characteristics, which are investigated formally later (Chapter 4).

3.2 Simple Examples

To illustrate the functionality of the SSS two examples are discussed next. In the first one, basic features of the score are worked out to illustrate its working principles. Thereafter, all possible parent configurations of one node are evaluated for different sample data sets and the score's quantification of these is discussed.

Example 5 (Basic Snap Shot Score features) *In order to illustrate first basic features of the score, consider an unnatural data-set with 6 channels (Fig. 3.3a) for which activity level series were calculated (Fig. 3.3f) to score selected information flow networks (Fig. 3.3b-e). The links of these networks were chosen to illustrate central features of the SSS:*

- *The score value is zero if cause and effect do not appear within the lag-window of minimal and maximal response lag:* $[\Delta t,\ \lceil d^{-1} \rceil + \Delta t - 1]_{\mathbb{Z}} = [1, 3]_{\mathbb{Z}}$ *(Fig. 3.3b). This is especially true when effects precede causes (Fig. 3.3c). Note that in figure 3.3bcd, node B's contradictory parent configurations render the full networks inconsistent, which is reflected by their zero score values.*

- *The score value is maximal if putative cause and effect occur exactly at the minimal time-lag, and it is lower if effects occur later (Fig. 3.3b). The SSS thus favours units as causal ones to which the response-lag is minimal (Nodes D, E, F in Fig. 3.3b vs. 3.3c).*

- *Increasing complexity of a parent configuration by adding more parents may get penalised by the SSS: If higher complexity is not balanced out by a significant explanation benefit, the score value decreases (Nodes D, E, F in Fig. 3.3c vs. 3.3d).*

- *Different parent configurations can have the same score value if spike trains are identical for different units (Fig. 3.3e). In this situation, different configurations are equivalent explanations for the data and their score value is thus the same. The ability to present alternatives as such is generally desirable; however, since the score cannot distinguish between identical single units and their join, more complex structures can have the same score value as simpler ones without giving a better explanation. According to Occam's razor (Section 1.4.2), the simpler structure should then be favoured over the configuration with more parents; but this can be easily realised by adding such preference to the network learning procedure. Also, as discussed in the following example in more detail, for realistic data, it is extremely unlikely that two units have precisely identical spike trains; it*

Fig. 3.3 legend: Simplistic spike trains and SSS values of selected networks (left), activity level series used for scoring and detailed calculation example (right). **a** Spike trains of 6 units (A-F). **b-e** Snap Shot Score values shown for selected parent-child configurations (near child nodes) and for full networks ($\prod = \ldots$ next to network). **b** Unit A is single parent of all other nodes. Scores of nodes B and F are zero, as these units do not *respond* within the defined time-lag-window: Unit B undershooting minimal response-lag ($\Delta t = 1$); unit F overshooting maximal response-lag ($\lceil d^{-1} \rceil + \Delta t - 1 = 3$ time-steps). Decreasing non-zero scores of nodes C through E reflect the Snap Shot Score's preference of short time-lags. **c** Unit C is single parent of all other nodes. Scores of nodes A and B are zero, as their response undershoots the minimal response-lag: Links $C \rightarrow A$ and $C \rightarrow B$ are directing *backwards in time*. Scores of nodes D, E, and F are larger for parent node C than for parent node A (b), as their response-lag to unit C is smaller than to A. **d** Units A and C are joined parents of nodes B, D, E, and F. All non-zero scores are smaller than those where unit C is the only parent (c); the explanatory benefit of two parents (A and C) does not balance out raised complexity of the network. **e** Nodes A or B are parents of node C either exclusively or jointly; a chain is formed of C to D to E to F. For the data in (a), these three resulting networks are the best scoring ones. Their structure differs with respect to the parent configuration of node C, but links between nodes C to F are unambiguous. **f** Activity level series of all units (top) and joins $a_{(A,B)}$ of A and B, and $a_{(A,C)}$ of A and C, respectively (bottom). Join $a_{(A,B)}$ is used in (e) and equals individual activity level series of A and B. Join $a_{(A,C)}$ was used to calculate scores in (d). **g** Detailed score calculation for node F with parents A and C as shown in (d).

is thus extremely unlikely that several networks are assigned the same high score value anyway.

While the preceding example showed how the SSS works for selected child-parent configurations, the following one illustrates how it operates on the full space of parent configurations of one particular node. Only few channels are included in the example in order to be able to display results appropriately; systems of higher dimensions will be discussed in chapter 7, later.

Example 6 (Exhaustive evaluation of parent configurations) *To demonstrate the Snap Shot Score's selectivity, different activity patterns (4 channels) were chosen to evaluate the score of node 1 for all its possible parent configurations (Fig. 3.4). Scores of individual configurations are compared against the mean score of all configurations for illustrative purposes, only.*[5] *Despite the low dimensionality several characteristics of the SSS can be seen in the depicted sit-*

[5]Generally the mean score is unknown because an exhaustive evaluation is computationally impossible in practical dimensions.

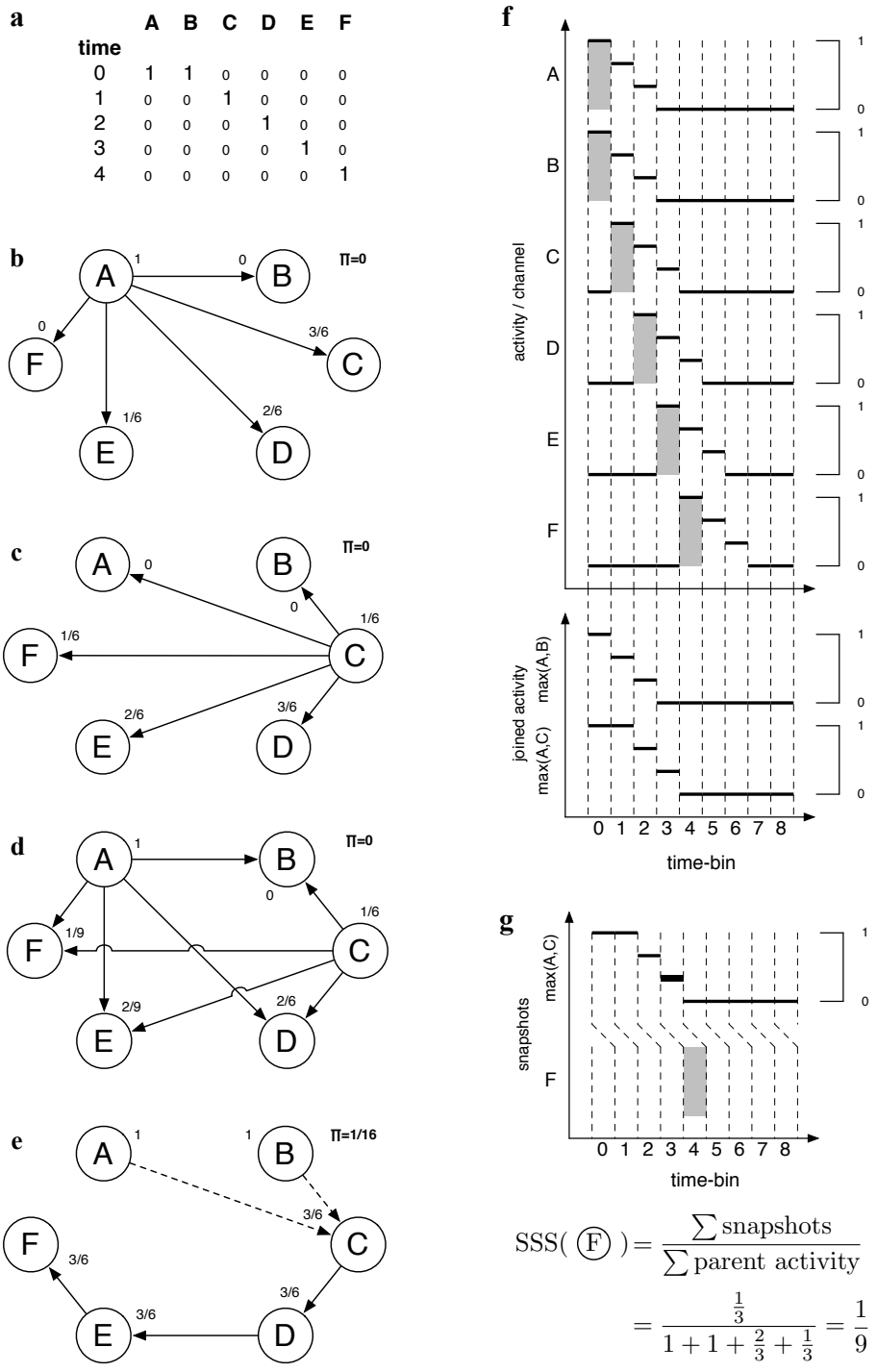


Figure 3.3: Legend in separate box on page 47.

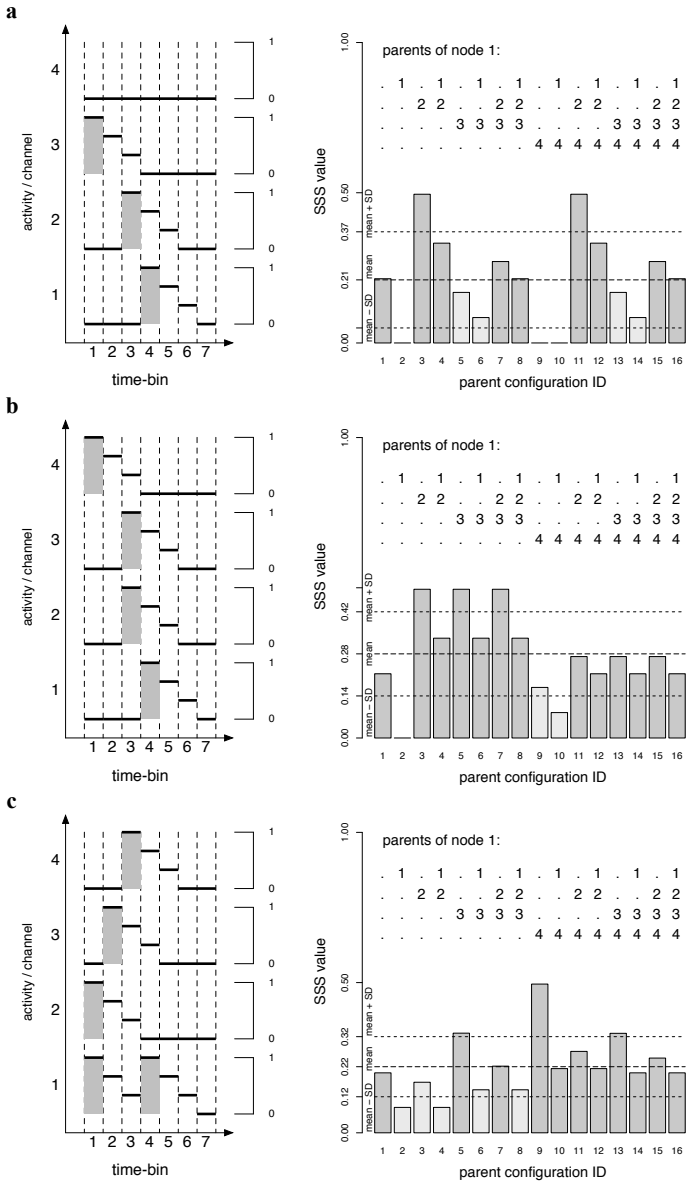


Figure 3.4: Spike train patterns and activity level series (left) and corresponding Snap Shot Score values of channel 1 for all its possible parent configurations (right). Spikes are indicated by grey bars with superimposed lines indicating the resulting activity levels. The adjacent bar plot shows the score values of all possible parent configurations for node 1 (parent nodes given above each bar). Mean score value and mean plus/minus one standard deviation (root mean square) shown by dashed lines for illustrative purposes. Bars indicating the score values are coloured light grey if below the link-acceptance threshold (score of most complex configuration 16) and dark grey otherwise. **a** Channel 4 silent; joining it has no effect on SSS value. **b** Channels 2 and 3 identical; joining more than one of them has no effect on SSS values. **c** Close spikes on channel 1; self-exciting configuration (number 2) with positive SSS value.

uations. For example, joining silent channels has no effect on the score value: configurations in Fig. 3.4a have the same score value whether they contain the silent channel 4 *as a parent or not. Also, joining completely identical channels is effectless compared to using only one of them (Fig. 3.4b, channels 2 and 3). Both of these effects are due to the* max*-operation in equation (3.2). More precisely, the score stays unaltered if the activity level series of the channel to join does not raise the activity level any further (Fig. 3.4c, join of channels 1 and 2 equals channel 1). This occurs in only three special cases: (1) joins of identical spike trains, (2) joins where all spikes on one channel occur simultaneously with a spike on any other channel in the join, and (3) joins including silent channels. This is not optimal, as according to Occam's razor (Section 1.4.2), one would rather penalise the effectless complication of a structure. However, this behaviour is a consequence of the simplicity of the score. For real data, it seems unlikely that identical spike trains, or spike trains that echo precisely a subset of another, are observed on different channels; thus, the special cases (1) and (2) are expected to have little effect in practical application. The third special case can be completely excluded by preprocessing: Inactive channels can be removed from the data (as would be likely practice in any case). Thus, in practice the SSS values are likely to be different for every parent configuration (Fig. 3.4a, configurations 1-8). The different score values can be used to order parent configurations hierarchically for subsequent inspection and result selection.*

In Fig. 3.4a-c the SSS assigns distinct top-scores to its most favoured configurations. To grasp the score's characteristics for more variable data, the spike patterns in Fig. 3.4 were concatenated in three different ways to yield the spike trains shown in Fig. 3.5. Interpreting the resulting spike trains with respect to the question "Which channels are exciting channel 1*?" can be harder or easier than before; the SSS values reflect this: In Fig. 3.5a channel* 1 *seems to be clearly excited by channel* 2*, as all parent configurations except one including this channel have score values above the mean. However, except from the clear peak for configuration number* 3 *the score is high and undecided about some others, especially configurations* 7 *and* 11*, and configurations* 4 *and* 5*. The spike train does not contain enough information to clearly prefer one of these parent configurations over the other; we find high scores close to the mean plus one standard deviation (SD) for all four configurations. A similar situation occurs in Fig. 3.5b: One configuration is clearly favoured, too, but high scores (about one SD above the mean) are reached by only two other configurations. The spike train in Fig. 3.5b is thus more meaningful to the SSS than the preceding one (Fig. 3.5a).*

Score value peaks become flat when the data contains unclear information: In Fig. 3.5c spikes on channels $2, 3$ *and* 4 *all occur as favoured (with time-*

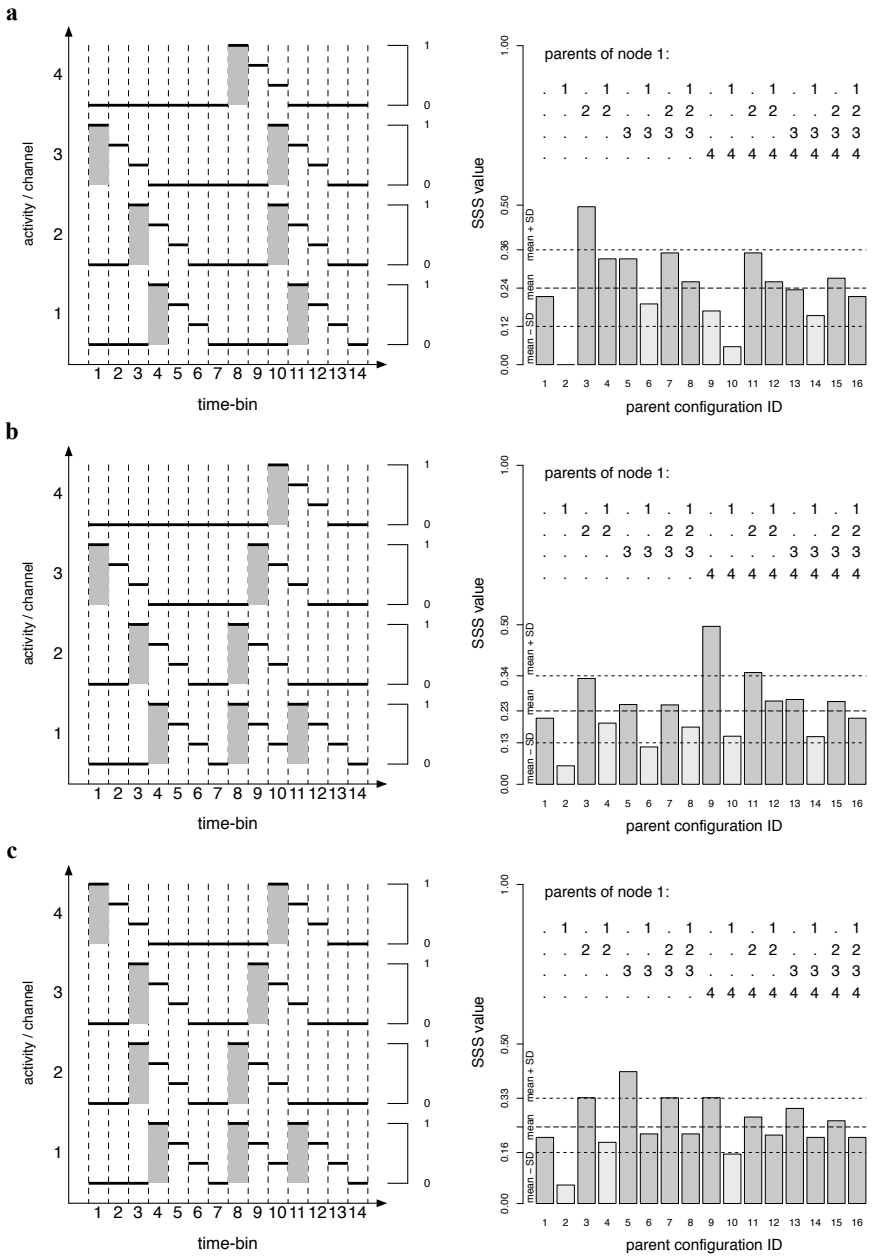


Figure 3.5: Exhaustive score evaluations of all parent configurations for channel 1 with same semantics as in Fig. 3.4. Spike train patterns are concatenations of those in Fig. 3.4 (**a** Fig. 3.4ab, **b** Fig. 3.4ac, **c** Fig. 3.4bc). **a** Distinct peak and four parent configurations with score value close to mean plus one SD. **b** Distinct peak and two parent configurations with score value close to mean plus one SD. **c** Score value of most configurations close to mean score and vague peak only.

lag 1*) once, but also with larger lags another time. The prominence of the top configuration number* 5 *is less distinct than in other cases and the overall score distribution close to the mean score. More informative data would be needed for the formation of a distinct peak; indeed, if one of the two spike patterns (Fig. 3.4bc) is concatenated twice, the score's favour is more distinct and on fewer configurations (not shown).*[6]

In both Figures 3.4 and 3.5 even numbered configurations include node 1 *as its own parent; for configuration number* 2 *node* 1 *is its only parent. This exclusive self-exciting configuration can have non-zero score values when two spikes on channel* 1 *occur close enough to each other, i.e. within the lag-window (Figs. 3.4c, 3.5c); any two spikes that are too close or too far apart do not contribute to the score value, as do single spikes (Fig. 3.4ab). If (exclusive) self-exciting configurations seem implausible for the single-unit spike train data they should not be considered.*

The new SSS has been introduced and illustrated by examples in this chapter. Later, in chapters 7 and 8, its practical applicability to spike train data will be investigated. But before that, the score is investigated mathematically in order to learn more about its characteristics (Chapter 4) and a discussion relates it to the BD scores (Chapter 5).

[6]For independent random spike trains SSS values of all parent configurations lie within mean $\pm$ SD and approach mean score ($SD \searrow 0$) for increasing length of spike trains (not shown).

Chapter 4

Characterising the SSS

The SSS has been introduced and motivated by examples, which showed the score's preferences in simplified situations (Section 3.2). For instance, simpler parent configurations were seen to be preferred over more complex ones if those could not explain the data significantly better. The given examples demonstrate characteristics of the SSS, but these illustrations do not justify conclusions about the generality of properties shown. In this chapter, key features of the score are proven to hold as general principles. Thorough investigation will give useful insights for efficient software implementation (Appendix E) and interpreting learned networks.

4.1 A Conditional Score Limit on Complex Configurations

The main result of this section is a limit on the score value for configurations with multiple parents. The mathematical formulation of this (Theorem 1) requires new terms to be introduced first. Also, two propositions follow, which are used to prove the theorem. We begin with the following definition, which formalises (nearly) simultaneous activity on different channels.

Definition 4 (active-time) *Let* $a = (a_{k,t})_{t=1,\dots,T}$ *be unit* k*'s activity level series. The set of time-points*

$$T_k = \{t \in T \mid a_{k,t} > 0\} \tag{4.1}$$

is called node k*'s active-time. For two nodes* i *and* j *we further define:*

$$\textit{nodes } i \textit{ and } j \textit{ have non-overlapping activity} \;:\Leftrightarrow\; T_i \cap T_j = \emptyset \tag{4.2}$$

Accordingly, a set of nodes $\{k_1, \ldots, k_m\}$ is said to have non-overlapping activity, if all possible pairs of different nodes have non-overlapping activity, i.e.

$$\forall\ i, j \in \{k_1, \ldots, k_m\} :\ i \neq j\ \Rightarrow\ T_i \cap T_j = \emptyset\ . \tag{4.3}$$

(Note the special case: This definition implies that a set containing only one node is considered to have non-overlapping activity.) The extend to which statements about the SSS can be made fundamentally depends on whether or not activity of different channels overlaps, as we shall see throughout the section. Next, two propositions are introduced, which will be helpful for proofs and interpretation of the score thereafter.

Proposition 1 *For two nodes i and j with activity level series $a_k = (a_{k,t})_{t=1,\ldots,T}$ $(k = i, j)$ that have non-overlapping activity we find*

$$\forall\ t :\quad \max(a_{i,t}, a_{j,t}) = a_{i,t} + a_{j,t}\ . \tag{4.4}$$

Accordingly, for a set $M = \{k_1, \ldots, k_m\}$ of m nodes with non-overlapping activity and activity level series $a_k = (a_{k,t})_{t=1,\ldots,T}$ $(k = k_1, \ldots, k_m)$ we find

$$\forall\ t :\quad \max_{k=k_1,\ldots,k_m} a_{k,t} = \sum_{k=k_1,\ldots,k_m} a_{k,t}\ . \tag{4.5}$$

Proposition 2 *Let $a_i, b_i \in \mathbb{R}_{\geq 0}$ where $b_i \neq 0$ $(i = 1, 2)$. Then:*

$$\begin{aligned} \frac{a_1}{b_1} \geq \frac{a_2}{b_2}\ &\Leftrightarrow\ a_1 b_2 \geq a_2 b_1 \\ &\Leftrightarrow\ a_1 b_1 + a_1 b_2 \geq a_1 b_1 + a_2 b_1\ \Leftrightarrow\ \frac{a_1}{b_1} \geq \frac{a_1 + a_2}{b_1 + b_2} \end{aligned} \tag{4.6}$$

and thus

$$\max\left\{\frac{a_1}{b_1}, \frac{a_2}{b_2}\right\} \geq \frac{a_1 + a_2}{b_1 + b_2}\ . \tag{4.7}$$

This completes the preparations in order to formulate the following theorem 1, which states an upper bound on the score value of configuration with multiple parents: If all parent channels involved have non-overlapping activity, a configuration with only one of the parents is guaranteed to exist, which exhibits the same or a better score value than the joined parents. Counter-examples show that the constraint of non-overlapping activity is indeed a necessary condition for this result (Fig. 4.1).

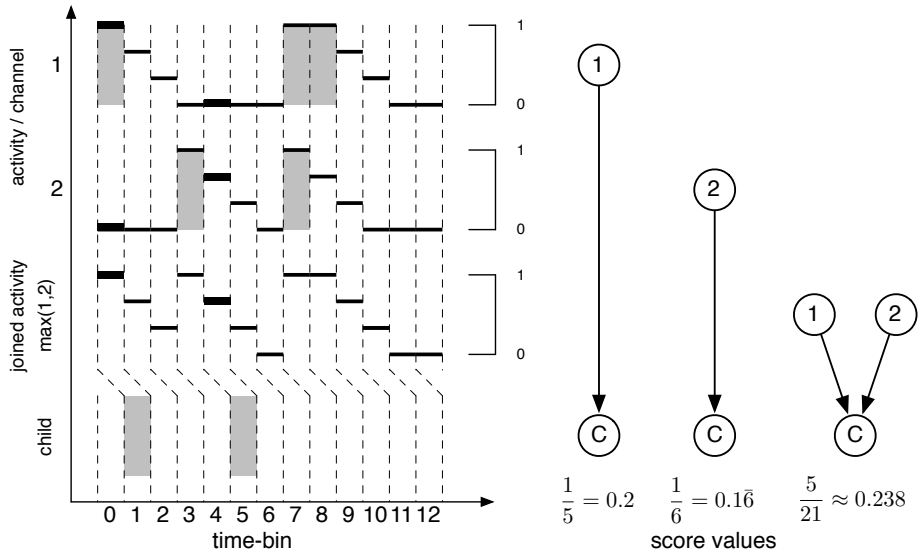


Figure 4.1: Example spike trains show that joined parents can yield a better score value than single ones. Spike trains and activity level series on the left; parent configurations with SSS values below on the right hand side ($d = 3^{-1}$, $\Delta t = 1$). Note that parent nodes 1 and 2 have overlapping activity (in time-bins 8, 9, and 10) such that the conditions of theorem 1 are not met.

Theorem 1 (Conditional upper limit) *Let $a_{(k_1,k_2)} = (a_{(k_1,k_2),t})_{t=1,\ldots,T}$ be the join of two channels k_1 and k_2 and $s = (s_t)_{t=1,\ldots,T}$ denote a spike train. If k_1 and k_2 have non-overlapping activity, then*

$$\mathrm{SSS}\left(a_{(k_1,k_2)}, s; \Delta t\right) \;\leq\; \max\left\{\mathrm{SSS}\left(a_{k_1}, s; \Delta t\right),\; \mathrm{SSS}\left(a_{k_2}, s; \Delta t\right)\right\} \;. \tag{4.8}$$

Accordingly, for a join $a_{(k_1,\ldots,k_m)}$ of m channels $k_1, \ldots, k_m$ with non-overlapping activity we find:

$$\mathrm{SSS}\left(a_{(k_1,\ldots,k_m)}, s; \Delta t\right) \;\leq\; \max_{j=1,\ldots,m} \mathrm{SSS}\left(a_{k_j}, s; \Delta t\right) \;. \tag{4.9}$$

Proof: Equation (4.8) is a combination of the score's definition [equation (3.3)] and previous results:

$$\mathrm{SSS}\left(a_{(k_1,k_2)}, s; \Delta t\right) \overset{(3.3)}{=} \frac{\sum_t a_{(k_1,k_2),t} \cdot s_{t+\Delta t}}{\sum_t a_{(k_1,k_2),t}} \tag{4.10}$$

$$\overset{Prop.1}{=} \frac{\sum_t a_{k_1,t} \cdot s_{t+\Delta t} + \sum_t a_{k_2,t} \cdot s_{t+\Delta t}}{\sum_t a_{k_1,t} + \sum_t a_{k_2,t}} \tag{4.11}$$

$$\overset{Prop.2}{\leq} \max\left\{ \frac{\sum_t a_{k_1,t} \cdot s_{t+\Delta t}}{\sum_t a_{k_1,t}}, \frac{\sum_t a_{k_2,t} \cdot s_{t+\Delta t}}{\sum_t a_{k_2,t}} \right\} \tag{4.12}$$

$$\overset{(3.3)}{=} \max\left\{\mathrm{SSS}\left(a_{k_1}, s; \Delta t\right),\ \mathrm{SSS}\left(a_{k_2}, s; \Delta t\right)\right\} \ . \tag{4.13}$$

Now the more general equation (4.9) is proven by induction:

Induction hypothesis ($\star$): Equation (4.9) holds for m channels $k_1, \ldots, k_m$ with non-overlapping activity.

Basic step: For $m = 2$ the induction hypothesis ($\star$) corresponds to equation (4.8), which has already been proven above.

Inductive step: Let k_{m+1} be another channel such that $k_1, \ldots, k_m, k_{m+1}$ have non-overlapping activity. Because $k_1, \ldots, k_{m+1}$ have non-overlapping activity channel k_{m+1} and the join $a_{(k_1,\ldots,k_m)}$ have non-overlapping activity; equation (4.8) can thus be applied to $a_{(k_1,\ldots,k_m,k_{m+1})}$, which yields

$$\mathrm{SSS}\left(a_{(k_1,\ldots,k_{m+1})}, s; \Delta t\right) \overset{(4.8)}{\leq} \max\left\{ \mathrm{SSS}\left(a_{(k_1,\ldots,k_m)}, s; \Delta t\right), \mathrm{SSS}\left(a_{k_{m+1}}, s; \Delta t\right) \right\} . \tag{4.14}$$

In the next step, the induction hypothesis ($\star$) is applied to replace the term $\mathrm{SSS}\left(a_{(k_1,\ldots,k_m)}, s; \Delta t\right)$. We find

$$\mathrm{SSS}\left(a_{(k_1,\ldots,k_{m+1})}, s; \Delta t\right) \overset{(\star)}{\leq} \max\left\{ \max_{j=1,\ldots,m} \mathrm{SSS}\left(a_{k_j}, s; \Delta t\right), \mathrm{SSS}\left(a_{k_{m+1}}, s; \Delta t\right) \right\} . \tag{4.15}$$

Rewriting the the right hand side of equation (4.15) as follows,

$$\max\left\{ \max_{j=1,\ldots,m} \mathrm{SSS}\left(a_{k_j}, s; \Delta t\right), \mathrm{SSS}\left(a_{k_{m+1}}, s; \Delta t\right) \right\} = \max_{j=1,\ldots,m,m+1} \mathrm{SSS}\left(a_{k_j}, s; \Delta t\right) \ , \tag{4.16}$$

shows that (4.9) is proven. ∎

As a direct consequence of theorem 1 we find the following

Corollary 1 *Let $k_1, \ldots, k_m$ be m channels with non-overlapping activity. Further let $\emptyset \neq \{j_1, \ldots, j_r\} \subseteq \{k_1, \ldots, k_m\}$ be an arbitrary sub-set of these channels. We then find the score of the join $a_{(j_1,\ldots,j_r)}$ limited:*

$$\mathrm{SSS}\left(a_{(j_1,\ldots,j_r)}, s; \Delta t\right) \;\leq\; \max_{j=j_1,\ldots,j_r} \mathrm{SSS}\left(a_j, s; \Delta t\right) \;\leq\; \max_{k=k_1,\ldots,k_m} \mathrm{SSS}\left(a_k, s; \Delta t\right) \;. \tag{4.17}$$

The corollary extends the claim of the theorem; it not only gives a limit for the join of all non-overlapping channels, but for the join of any sub-set of these, too. The score value of any such join is limited by the score of at least one single parent configuration. It is thus known that the simplest and best scoring parent configuration for non-overlapping channels has exactly one parent. In situations of non-overlapping activity the SSS thus shows a very strong tendency towards sparse networks. There is also another important implication of the theorem, which affects the expansion of an existing join.

Corollary 2 *Let $a_{(j_1,\ldots,j_r)}$ be a join of r channels. Further let k be a channel whose activity does not overlap with $a_{(j_1,\ldots,j_r)}$. We then find the score of the join $a_{(j_1,\ldots,j_r,k)}$ limited:*

$$\mathrm{SSS}\left(a_{(j_1,\ldots,j_r,k)}, s; \Delta t\right) \;\leq\; \max\left\{\mathrm{SSS}\left(a_{(j_1,\ldots,j_r)}, s; \Delta t\right), \mathrm{SSS}\left(a_k, s; \Delta t\right)\right\} \;. \tag{4.18}$$

This seemingly technical result has significant practical importance. It can be read as: *Expanding a join by a channel whose activity does not overlap with it cannot increase the score value beyond that of the better scoring of the two.* Or vice versa: *If the members of a join can be split into two groups which have non-overlapping activity, at least one of the groups' join scores as least as good as that of all channels together.* This formulation especially sheds light on each nodes' optimal configuration: It cannot be split into groups with non-overlapping activity, because a simpler configuration with equal or better score would otherwise exist. In practice, corollary 2 can be utilised in order to avoid the evaluation of too complex joins. Details on the implementation can be found in section E.2.2.

The preceding considerations yield strong results in case activity between channels does not overlap. In practical situations, however, it is likely that at least few channels show overlap in their activity. The following section therefore investigates the behaviour of the SSS under circumstances in which the conditions of theorem 1 and its corollaries are not met.

4.2 A Close View on the Join Operation

The previous section investigated situations where joining multiple channels does not lead to a higher score than that of individual channels. The sufficient condition for the corresponding results is that channels have non-overlapping activity. Next, in order to understand which conditions must be met such that a join can have a higher score than its individual channels, the join operation is investigated closely. Therefore, different re-formulations of the SSS are discussed to highlight aspects that facilitate high score values of joins. For this consider two nodes i and j, which are potential parents of a child node. The score of configurations where nodes i and j are both exclusive parents of the child will be compared to that of their joint parenthood. Therefore let $a_k = (a_{k,t})_{t=1,\dots,T}$ $(k = i, j)$ denote the corresponding activity level series and let $s = (s_t)_{t=1,\dots,T}$ be the child's spike train.

First, the definition of the SSS is repeated by plugging in the join operation [equation (3.2)] directly into the score's formula [equation (3.3)]:

$$\mathrm{SSS}\left(a_{(i,j)}, s; \Delta t\right) = \frac{\sum_t \max\{a_{i,t}, a_{j,t}\} \cdot s_{t+\Delta t}}{\sum_t \max\{a_{i,t}, a_{j,t}\}} \ . \tag{4.19}$$

If the activity of channels i and j is non-overlapping proposition 1 can be applied to replace the join operation max by the sum of its arguments, which yields:

$$\mathrm{SSS}\left(a_{(i,j)}, s; \Delta t\right) \overset{i,j\text{non-overlapping}}{=} \frac{\sum_t a_{i,t} \cdot s_{t+\Delta t} \quad + \quad \sum_t a_{j,t} \cdot s_{t+\Delta t}}{\sum_t a_{i,t} \qquad + \qquad \sum_t a_{j,t}} . \tag{4.20}$$

Here, the score of the join $a_{(i,j)}$ is the sum of the snapshots of the individual channels and accumulated activity, respectively. However, equation (4.20) does not hold when channels have overlapping activity (Fig. 4.1) because the condition of proposition 1 is violated. In order to yield generally valid re-formulations of equation (4.19), the join operation is replaced by different identities, which hold whether channels have non-overlapping activity or not. The first identity to consider is given in

Proposition 3 *Let $a, b \in \mathbb{R}$. Then*

$$\max\{a, b\} = a + b - \min\{a, b\} \tag{4.21}$$

holds.

Proposition 3 is used to reformulate equation (4.19) by applying it to the joined

activity $\max\{a_{i,t}, a_{j,t}\}$ at any time t. This yields:

$$\mathrm{SSS}\left(a_{(i,j)}, s; \Delta t\right) = \frac{\sum_t a_{i,t} \cdot s_{t+\Delta t} \quad + \quad \sum_t a_{j,t} \cdot s_{t+\Delta t} \quad - \quad \sum_t \min\{a_{i,t}, a_{j,t}\} \cdot s_{t+\Delta t}}{\sum_t a_{i,t} \qquad + \qquad \sum_t a_{j,t} \qquad - \qquad \sum_t \min\{a_{i,t}, a_{j,t}\}}. \tag{4.22}$$

In this formula we find the left and the middle sums corresponding to the terms that make up the score of configurations where node i and node j are exclusive parents. The two sums on the right gather snapshots and activity of both channels i and j while their activity overlaps. The score is calculated by adding the snapshots and accumulated activity of both channels and correcting it by the overlap of channels. Indeed, if no such overlap exists, i.e. in the case where channels i and j have non-overlapping activity, the right sums both evaluate as 0, such that equation (4.22) reduces to (4.20). Finding equation (4.20) to be a special case of (4.22) verifies its more general validity; however, it does not give any explicit insights into what maximises the score of a join. This is because it is not sufficient to maximise the ratio of the left and middle sums while minimising the right sums ratio; in order to optimise the score of the join the overall ratio must of course be maximised. Due to the sheer number of possible combinations of values for the six sums, it is not possible to derive a simple, general condition, which is sufficient to guarantee a high score value of a join. It is, however, possible to identify special cases of overlapping activity in which joining channels can *not* have higher score values than their separate channels. In order to spot these cases we use the following basic

Proposition 4 *Let $a, b \in \mathbb{R}_{\geq 0}$. Then*

$$\max\{a, b\} = a + \max\{b - a, 0\} \tag{4.23}$$

holds.

Proposition 4 is applicable to activity level series as these, by definition, are non-negative at all times. We thus find equation (4.19) equivalent to

$$\mathrm{SSS}\left(a_{(i,j)}, s; \Delta t\right) = \frac{\sum_t a_{i,t} \cdot s_{t+\Delta t} \quad + \quad \sum_t \max\{a_{j,t} - a_{i,t}, 0\} \cdot s_{t+\Delta t}}{\sum_t a_{i,t} \qquad + \qquad \sum_t \max\{a_{j,t} - a_{i,t}, 0\}}. \tag{4.24}$$

This formulation of the SSS can be interpreted when recognising that the sums on the left resemble the score of channel i alone (without joining with j); the right sums constitute the change to the score of channel i when joining i and j. If the activity $a_{j,t}$ of channel j is lower or equal than that of channel i at all

times t, the right sums both evaluate as 0, such that the score of the join is equal to that of solely using channel i. (This has already been observed in examples 5 and 6 in section 3.2.) However, if channel i does not out-value channel j at all times the score of the join will be different; it can either be higher or lower than that of i's score. Whether the score actually increases or decreases by joining channels i and j depends on the ratio of the right sums compared to the score of node i as single parent: Proposition 2 shows that the score of the join $a_{(i,j)}$ will only be higher, if the ratio of the right sums is higher than that of channel i alone.

The preceding investigations in this chapter aimed at specifying the SSS's general characteristics. Under specific assumptions about the (non-)simultaneous activity on different channels strong results could be obtained. Considerations of more general cases gave at least an impression about mechanisms at work. In order to be more specific, the next section addresses particular questions, which might have arisen from previous results.

4.3 Questions, Presumptions, and (Counter) Examples

The previous sections confirmed that the SSS has a strong favour of simple configurations under a variety of conditions. The reader might already wonder about whether this characteristic of the score can be generalised even further. For example, given the score's tendency towards configurations with fewer parents, a legitimate question to ask is:

Q1 Do any examples exist where the best scoring parent configuration has more than one parent?

Indeed, simple ad-hoc examples often give the impression that the SSS would always favour one parent configurations, but persistently constructing more examples shows:

A1 Yes, indeed such situations exist. Figure 4.1 shows one example where two joint parents have a higher score value than either of them alone.

Given an example of a case where a joint parent configuration is highest scoring, it is reasonable to ask questions about sufficient conditions for the existence of such cases. The previous sections already revealed that overlapping activity of joined channels is a necessary condition; however, it is not sufficient.[1] One

[1] As an example consider Fig. 3.1 on page 41 where all channels have overlapping activity. The exhaustive evaluation of all configurations (Fig. 3.2 on page 45) shows that the best

might thus wonder instead, if it is possible to find a sufficient condition in order to exclude the existence of higher scoring joins. For example, a fair speculation would be:

Q2 Consider a node's best scoring parent configuration p_k with k parents and let $\tilde{p}_k$ denote its score value ($k = 1, \ldots, n$). If the best scoring single parent configuration p_1 has a higher score value than the best scoring two-parent configuration p_2 (i.e. $\tilde{p}_1 > \tilde{p}_2$), will $\tilde{p}_1$ be greater than the score $\tilde{p}_k$ of any configuration p_k that is more complex, i.e.

$$\tilde{p}_1 > \tilde{p}_2 \;\Rightarrow\; \forall\, k \in \{3, \ldots, n\} :\; \tilde{p}_1 > \tilde{p}_k \quad ? \tag{4.25}$$

Such property would be very valuable since it could be used to shorten the network learning process: Evaluating all one- and two-parent configurations and finding $\tilde{p}_1 > \tilde{p}_2$ would render the evaluation of all other configurations (with more parents) superfluous, since their score could not out-value $\tilde{p}_1$. Unfortunately, it turns out that equation (4.25) does not hold generally:

A2 No, there exist counter examples, which disprove equation (4.25). One such example is shown in Fig 4.2 where we find $\tilde{p}_1 > \tilde{p}_2$ but $\tilde{p}_1 < \tilde{p}_3$. However, scores of activity level series with non-overlapping activity will always out-value the score of their join (Section 4.1).

This answer is clear about the generality of equation (4.25), but it does not give any insights into how common its violation actually is. A direct follow-up question might thus be:

Q3 Is the counter example used to disprove equation (4.25) a common, practical situation or rather an especially constructed set-up, which is very unlikely to be relevant in applications?

This is a good question, but for which no definite answer can be given. The reason for this is that the characteristics of the data need to be clearly specified, as the answer clearly depends on them. However, a brute-force approach, considering all possible spike trains, yields the following quantitative answer:

A3 Considering all possible combinations of spikes that can occur on 4 separate spike trains of length 7, we find that in about 0.056% of them equation (4.25) is violated.[2] The low relative frequency of these cases might

scoring configuration for each node has one parent only. Overlapping activity is thus not a sufficient condition in order to render a join higher scoring than individual channels.

[2]The total number of spike trains considered here is $(2^7)^4 = 268,435,456$ out of which $150,635$ violate equation (4.25) for SSS parameters $d = 3^{-1}$ and $\Delta t = 1$. The enormous growth of spike trains to consider prevents evaluations for longer data or more channels.

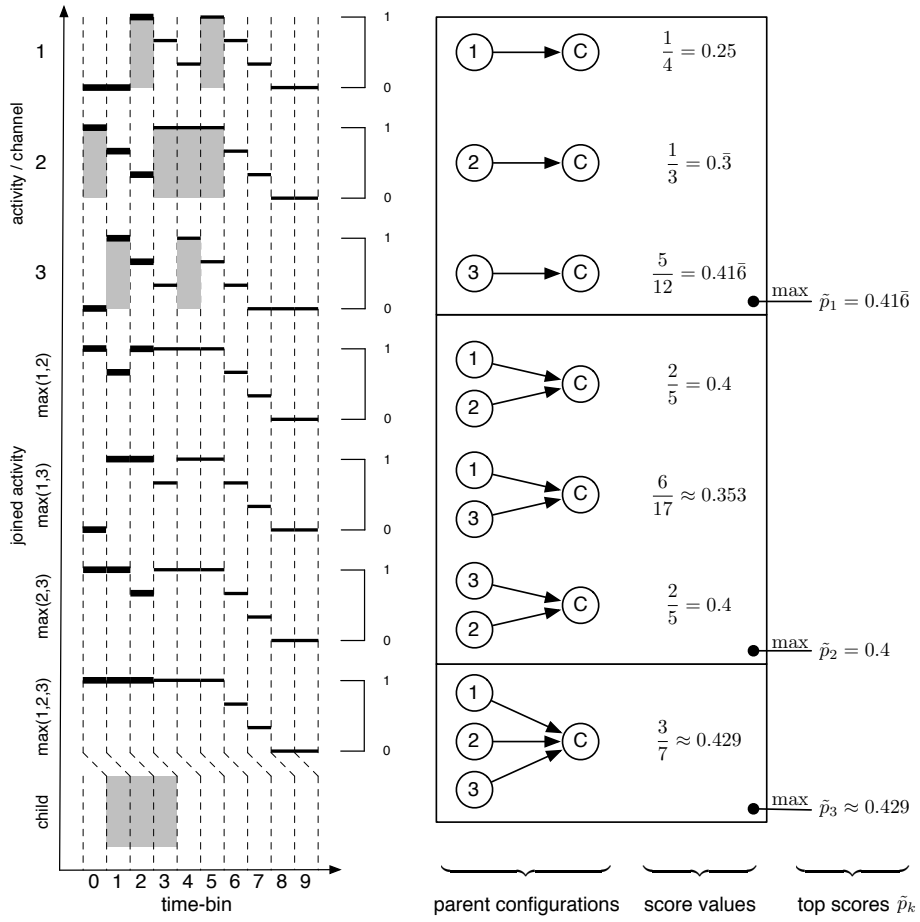


Figure 4.2: Example spike trains to show that single parents can yield a better score value than two joined parents, but a worse score than three parents. Spike trains and activity level series on the left; parent configurations and SSS values the right hand side ($d = 3^{-1}$, $\Delta t = 1$). Best score of configurations with k parents denoted as $\tilde{p}_k$. All two parent configurations are out-valued by best single parent configuration, which is outperformed by best three parent configuration (i.e. $\tilde{p}_2 < \tilde{p}_1 < \tilde{p}_3$).

indicate a rather rare phenomenom; however, if more channels are considered, violations become more likely.

Many more questions could be considered, but most of them can only be answered with respect to a specific data-set. Thus, in order to gain further general insights to the SSS, the next section is concerned with another aspect of the score: its sufficient statistics. As these determine which information in the data gets used by the score, they are central to its understanding.

4.4 Sufficient Statistics of the SSS

Likewise to the BD score, the sufficient statistics of the SSS are counts of state combinations; but the statistics of both scores differ: For the BD score state combinations are counted within a time-layer (Sections 2.2.2 and 2.2.6), whereas those of the SSS are effectively determined across multiple time-layers. We shall look at the SSS's statistics in detail next and use them in order to re-formulate the scoring function in terms of the counts later.

For a formal description of the SSS's sufficient statistics consider n-channel spike trains $s = (s_{k,t})_{t=1,\dots,T}^{k=1,\dots,n}$ of length T and the corresponding activity level series $a = (a_{k,t})_{t=1,\dots,T}^{k=1,\dots,n}$. From the calculation of the activity level series [equation (3.1)] follows that for a decay constant d there exist $r := \lceil d^{-1} \rceil + 1$ different values of activity level. Let $v_1, \dots, v_r$ denote these values. Note that joined activity level series take the same values.[3] As before, Δt denotes the shift constant. We then find:

> The SSS's statistics are the counts of how often parent nodes took each value of activity level, once while a spike of the child followed, and once while the child did not spike. More formal, let M_{ijk} denote the number of times where the we find the joined activity level $a_{(pa_i),t}$ of node i's parents pa_i to be v_j, while later, at time $t+\Delta t$, the child i is in state k, i.e.:
>
> $$M_{ijk} = \#\left\{ t = 1, \dots, T - \Delta t \;\middle|\; a_{(pa_i),t} = v_j,\ s_{i,t+\Delta t} = k \right\} \ . \quad (4.26)$$
>
> The state of the child can either be spiking ($k = 1$) or non-spiking ($k = 0$), such that for each node i we get $r \cdot 2$ counts: M_{ij1} and M_{ij0} ($j = 1, \dots, r$). In total, the sufficient statistics of the SSS consist of $n \cdot r \cdot 2$ different counts.

[3] As before, it is assumed that the join operation max is used. Modifications of the score might use another join operation (e.g. *mean* or *product*), which can render more than $d + 1$ different values of activity level possible. In such cases, r denotes the number of all possible values $v_1, \dots, v_r$ of activity level that could occur.

The SSS is defined as a ratio of sums over time (Def. 3.3), where the activity level series and spike trains are used. It is also possible to express the score in terms of its sufficient statistics: Given the statistics for node i, M_{i**}, an equivalent formulation of the score is:

$$\mathrm{SSS}\left(M_{i**}\right) = \frac{\sum_{j=1}^{r} M_{ij1}}{\sum_{j=1}^{r} M_{ij0} + M_{ij1}} \tag{4.27}$$

if $\sum_{j=1}^{r} M_{ij0} + M_{ij1} \neq 0$, and 0 otherwise. In contrast to the original formulation of the score [equation (3.3)], we find the sums being calculated over values of activity level rather than time. Sums over time correspond to the practical implementation of the score, whereas the representation using counts benefits interpretation. The sufficient statistics can also be used for the following closed expression for the score of a full network; given the sufficient statistics for all nodes, M_{***}, we find:

$$\mathrm{SSS}\left(M_{***}\right) = \prod_{i=1}^{n} \frac{\sum_{j=1}^{r} M_{ij1}}{\max\left\{\sum_{j=1}^{r} M_{ij0} + M_{ij1},\ \min\{v_k > 0 \mid k = 1, \ldots, r\}\right\}} \,. \tag{4.28}$$

Note that the maximum and minimum in the denominator are of pure technical nature in order to prevent a division by zero; this eliminates the need for discussing special cases where a ratio is not well defined.

This ends the theoretical investigation of the SSS as it has been introduced; the next section considers generalisations of the score. The more abstract presentation of the SSS is intended to give an additional perspective aiding the score's interpretation. Building upon this, possible modifications of the SSS are presented and briefly compared to each other.

4.5 General Form of the Snap Shot Score

Spike trains are time-series in which each data point corresponds to a time-interval (e.g. 1 msec). The sums in the formula of the SSS are taken over these time-bins by which time is treated as a discrete entity. Mathematically, time can also be considered to be continuous, such that spike-times can be specified more precisely than in a spike train. For the remainder of this section time will be treated continously in order to formulate ideas conveniently. Technological limitations do not allow real continuous measurements, such that spike trains will always be defined over discrete time. However, all ideas expressed below can be translated to this case.

In order to formulate the SSS in continuous time let $s(t)$ denote the *spike function* of a neuron. I.e. $s(\,\cdot\,)$ is a function that takes the value 1 if a spike occurred at time t; and 0 otherwise. Accordingly, the *activity level function* $a(t)$ maps any point in time to the associated activity level. (Details on $a(t)$ are discussed later.) With these two functions (instead of time-series) the SSS can be written as

$$\mathrm{SSS}\,(a, s; \Delta t) = \frac{\int_0^{T-\Delta t} a(t) \cdot s(t+\Delta t)\ dt}{\int_0^{T-\Delta t} a(t)\ dt} \tag{4.29}$$

if $\int_0^{T-\Delta t} a(t)\ dt \neq 0$, and 0 otherwise. (It is assumed that s and a are well defined over the time-period $[0, T]$.) This reformulation of the SSS in continuous time does not provide any insights itself since it is heavily based on the activity level function $a(\,\cdot\,)$, which is the actual key to the SSS. The following sections therefore discuss how activity levels can be calculated and joined in continuous time.

4.5.1 Alternative Definitions of the Activity Level

The activity level can basically be understood as a low-pass filtered version of the spike function; i.e. abrupt and short-lived spike-events are converted to a less transient form. Such is generally done by any method for firing rate estimation from spike trains (e.g. [Gabbianti and Koch, 1998, Nawrot et al., 1999, Gerstner and Kistler, 2002, McNames, 2005]). Indeed, firing rates can be used as activity levels. Out of these, a particular methodological type will be discussed: filter methods [Kalman, 1960, Jazwinski, 1970, Tanizaki, 1996, Roweis and Ghahramani, 1999]. Central to these methods is a so called *kernel-* or *window-function* $k(\,\cdot\,)$, which is used to weight events in time differently, depending on when they occur. With a kernel-function $k : \mathbb{R} \to \mathbb{R}_{\geq 0}$ the activity level $a(t)$ at time t can be expressed by the following linear filter [Dayan and Abbott, 2005, p.13]:

$$a(t) = \int_{-\infty}^{+\infty} s(t-\tau)k(\tau)\ d\tau\ . \tag{4.30}$$

If and to which extent a spike contributes to the activity level depends on the kernel-function k: The weight of any spike depends on its occurrence relative to the time t for which the activity level is calculated. In order to see this, interpret the argument τ of the kernel as a time relative to time-point t where $\tau = 0$ stands for the present; positive and negative values of τ represent the past and the future, respectively. The larger the absolute value of τ, the more distant it is from the present. A spike that occurs at time $t - \tau$ is weighted by the value of the kernel-function $k(\tau)$, which is generally different for spikes occurring exactly at time t ($\tau = 0$), in the past ($\tau > 0$), or in the future ($\tau < 0$).

All times to which a kernel k assigns non-zero values make up its *support*:[4]

$$\operatorname{supp}(k) = \{x \in \mathbb{R} \mid k(x) > 0\} \ . \tag{4.31}$$

The support is helpful to characterise a kernel, which is said to be *causal*, if it does not use information gained in the future [Dayan and Abbott, 2005, p.14]. For such causal kernel the activity-level is determined by past and present spikes only; future spikes do not have any weight, i.e.

$$\operatorname{supp}(k) \cap \mathbb{R}_{<0} = \emptyset \ . \tag{4.32}$$

In order to calculate activity levels a causal kernel should be preferred in order to avoid any influence of spikes that did not occur so far. If this was the case, a non-causal kernel would render the model underlying the SSS non-causal, because neurons would know their future synaptic inputs; but this is biologically unreasonable. The following example, in which two causal and one non-causal kernel are presented, illustrates how the activity level differs depending on the kernel.

Example 7 (Causal and non-causal kernels) *Three different kernel functions are defined below and visualised in figure 4.3a. The first one is the causal kernel*

$$k_{steady\ decay}(x) = \begin{cases} \max\left\{0, \frac{2}{d} - \frac{2x}{d^2}\right\}, & x > 0, \\ 0, & otherwise, \end{cases} \qquad (d > 0) \tag{4.33}$$

with support $\operatorname{supp}(k_{steady\ decay}) = (0, d)$*. This kernel can be understood as the time-continuous version of the activity-level series defined by equation (3.1). The second causal kernel to consider is the so called α-function [Dayan and Abbott, 2005, p.14],*

$$k_{alpha}(x) = \max\left\{0, \alpha^2 x \exp(-\alpha x)\right\} \quad (\alpha > 0) \ , \tag{4.34}$$

for which any point in the past has (tiny but) positive value, since $\operatorname{supp}(k_{alpha}) = \mathbb{R}_{>0}$*. Present and future points in time are nil-weighted. Finally, the Gaussian kernel*

$$k_{Gaussian}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (\mu \in \mathbb{R},\ \sigma^2 > 0) \tag{4.35}$$

[4]Note that this definition of the support does not satisfy a strict topological [Narasimhan and Nievergelt, 2001, p.xiv] nor measure theoretic definition [Cohn, 1980, p.199]. However, this imprecision does not have an effect in practice where equation (4.30) reduces to a discrete sum (over time-bins).

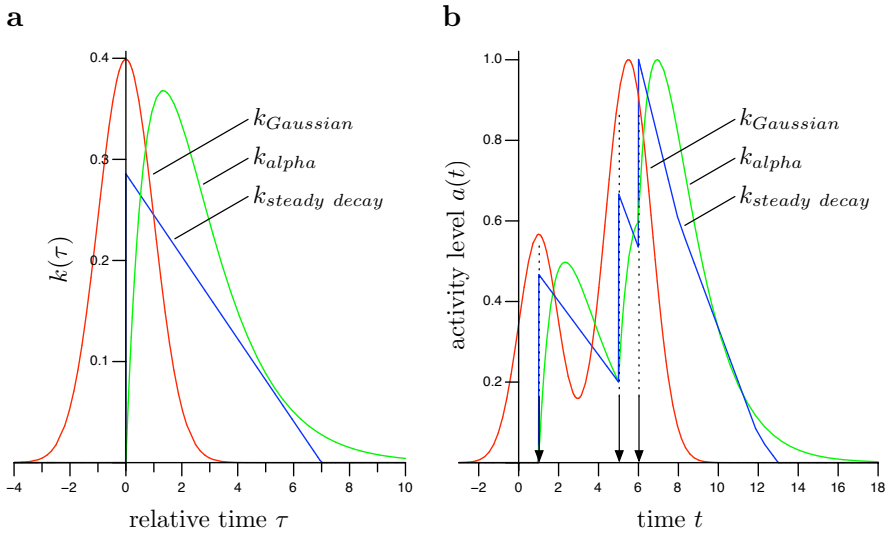


Figure 4.3: Different kernel functions (left) and resulting activity levels from a spike train (right). **a** Three kernel functions (defined in example 7) plotted over relative time τ (parameters $d = 2$; $\alpha = 0.75$; and $\mu = 0, \sigma = 1$). Positive values on time axis correspond to the past (the longer ago the larger the value); zero stands for the present; and negative values represent the future. The non-causal Gaussian kernel $k_{Gaussian}$ is positive for any argument, while causal kernels $k_{steady\ decay}$ and k_{alpha} are zero for all future times. **b** Activity levels resulting from application of kernel functions shown in (a) to spike train (with three spikes at time 1, 5, and 6; marked by arrows). The activity level curves differ in magnitude and time, which reflect each kernel's specific integration of spikes. Activity level calculated with non-causal Gaussian kernel peaks earlier than that of the causal kernels, because these do not assign positive weight to future spikes (a).

is non-causal. This is because any point in time is assigned positive value (although infinitesimally small for very large or small x): $\text{supp}(k_{Gaussian}) = \mathbb{R}$. *How these three kernel functions differ when applying them to calculate the activity level according to expression (4.30) can be seen in figure 4.3b. The activity level of both causal kernels lags that of the Gaussian one because they do not consider future spikes.*

Further kernel functions can be found in the literature (e.g. [Nawrot et al., 1999, Gabbianti and Koch, 1998]) but are not discussed here. Instead, different operations to join the activity level of multiple units are presented.

4.5.2 Alternative Definitions of Joined Activity

Once a suitable kernel to calculate the activity level has been chosen, configurations comprising only a single parent can be scored according to equation (4.29); however, in situations where a node has multiple parents their activity levels must be joined beforehand. A small selection of functions that are suitable for this purpose are suggested next. Therefore, let $a_i(t)$ $(i = 1, \ldots, m)$ denote m activity level functions that are to be joined. Their join $a_{(1,\ldots,m)}(t)$ could for instance be calculated using one of the following operations:

$$a_{(1,\ldots,m)}(t) = \begin{cases} \min\limits_{i=1,\ldots,m} a_i(t) & \textbf{minimum} \\ \max\limits_{i=1,\ldots,m} a_i(t) & \textbf{maximum} \\ \frac{1}{m}\sum\limits_{i=1}^{m} a_i(t) & \textbf{arithmetic mean} \\ \prod\limits_{i=1}^{m} a_i(t) & \textbf{multiplicative} \end{cases} \tag{4.36}$$

Figure 4.4a-e shows an example, which illustrates how joins of two channels differ when applying the four techniques above. Additionally, if activity levels are normalised $a_i(t) \in [0, 1]$, they can also be joined using the following expressions:

$$a_{(1,\ldots,m)}(t) = \begin{cases} \max\left\{0, 1 - m + \sum\limits_{i=1,\ldots,m} a_t^{(i)}\right\} & \textbf{piecewise linear} \\ \exp\left[-\left\{\sum\limits_i \left(-\ln a_t^{(i)}\right)^{\lambda}\right\}^{\frac{1}{\lambda}}\right] & \textbf{climactic}^5 \quad (\lambda \geq 1) \end{cases} \tag{4.37}$$

To see how these more complex operations differ in effect to other proposed ones consider figure 4.4fg. Further, any m-dimensional cumulative probability distribution function can be used in order to join multiple activity levels to one.

The examples above illustrate that countless possibilities for joining activity levels exist. In order to decide on one particular method, a biologically reasonable one might be chosen: Taking the perspective of the child neuron for which activity levels of parents are joined, the join reflects how synaptic inputs are integrated by the child. Models of synaptic integration (e.g. [Koch et al., 2003, Gerstner and Kistler, 2002, pp.51]) could be used to derive a corresponding join-function, but it needs to be questioned if this effort could pay off: Given the generally high density of neurons in neural tissue [Braitenberg and Schüz,

[5]Note that for $\lambda = 1$ the climactic term equals the multiplicative one. For $\lambda \to \infty$ the climactic term converges to that using the minimum.

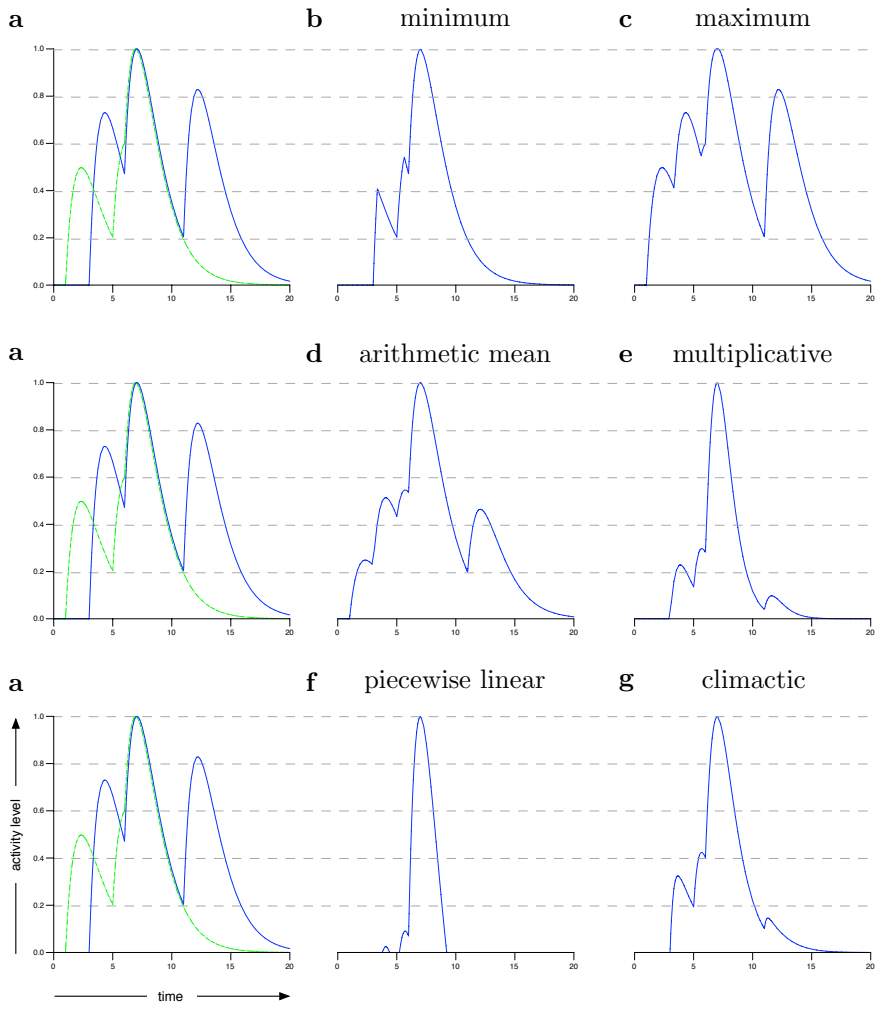


Figure 4.4: Continuous valued activity levels (left) joined with different operations (middle, right). **a** Two activity level functions (solid and dashed line) with partial overlap of activity. Identical plot shown in each row for comparison: Activity levels were joined using operations in equations (4.36) and (4.37). **b** Minimum: Both joined channels must show activity in order for the join to be non-zero. **c** Maximum: Activity on at least one of the joined channels is sufficient to evoke activity in the join. **d** Arithmetic mean: The joined activity is the average of that on input channels. **e** Multiplicative: Similar to the minimum operation (a), the join only shows activity if both joined channels are active. **f** Piecewise linear: Activity on both channels must exceed an implicit threshold before their join becomes active. **g** Climactic ($\lambda = 2$): Resulting join shares characteristics of both the minimum function (b) and the multiplicative joining of channels (e).

1998, chapter 4] and the comparatively coarse electrophysiological sampling of units, it is unlikely that the majority of data-channels is indeed collected from directly connected neurons. Therefore, the data might not match physiological assumptions on cell level and a corresponding join-function would be inadequate. Simple, pragmatic ideas can thus provide equally good or even better results, because they can be chosen for computational efficiency to benefit network inference (see appendices D and E). Additionally, the joining method is a good target in order implement and adjust Occam's razor (Section 1.4.2). This can be seen when comparing the effect different join operations (Fig. 4.4): Using the maximum of activity levels results in a high activity whenever at least one of the joined channels peaks (Fig. 4.4c). By this, sparse networks are strongly favoured, since the ratio in equation (4.29) is less likely to be large. This is because the denominator is large if parent activity is high; the numerator can only reach similar high values if all high parent activity is followed by child spikes. Hence, the more often the parents' activity is high, the more matching child spikes are needed for a good score. Considering a join operation that is more stringent about inputs leading to high joined activity shows a different picture. The multiplicative combination of activity levels, for example, results in zero activity even if only one of the inputs is zero (Fig. 4.4e). Compared to the maximum operation, fewer matching child spikes are needed for a high score, because parent activity peaks less frequently. Even further, activity of the join becomes less and less likely the more parents are joined; more complex configurations can thus be favoured over simpler ones, but which is not desirable. In conclusion, the impact of the join operation on sparseness of learned networks limits its adjustment according to biologically reasonable paradigms and further studies are needed to fully clarify these bounds; however, corresponding investigations are not within the scope of this thesis.

With the end of this chapter, the characterisation of the SSS is closed. As indicated in multiple foregoing sections, the understanding of the score is by no means complete; however, at this point the reader should be provided with sufficient information in order to address open questions of interest independently. While this chapter discussed the SSS itself, the following one relates it to the BD scores in order to work out differences and their implications for the analysis of spike train data.

Chapter 5

On the Relationship of the SSS to the BD Scores

In the foregoing chapters techniques for networks inference have been introduced; in particular, two types of scores were presented: the BD scores (Section 2.2) and the SSS (Section 3.1). In this chapter, a comparison of both techniques shows how they differ in concept. Interpretation of the scores themselves, as well as the meaning of learned networks will be discussed. Also, addressed are characteristics of spike train data and how these can complicate network inference with the BD scores. Suitable approaches to these problems are presented and related to the SSS.

5.1 A Brief View on the BD Scores and the SSS

The BD Scores are based on a set of mathematically convenient assumptions from which the scores can be derived elegantly [Cooper and Herskovits, 1992, Heckerman et al., 1995]. The fully Bayesian approach pays off by a wide applicability of these scores, which have undergone thorough theoretical investigations (e.g. [Cooper and Herskovits, 1992, Heckerman et al., 1995, Chickering, 1996, Heckerman, 1997, Yu, 2005]). These studies proved important features, for instance on learning complexity [Chickering, 1996] and consistency[1] [Heckerman et al., 1995], and have rendered the scores well studied tools for network inference. As outlined earlier, in section 2.2.6, BD scores can be used to infer DBNs, which are associated with a stochastic process [equation (1.21)] that gives precise information about the time-lag at which parents influence the child. In

[1] *Consistency* is a theoretical property which, with respect to the BDe score, means: If the dependencies in the data are generated by a BN and infinite data are available, the highest scoring structure is the generating one [Yu, 2005, p.56].

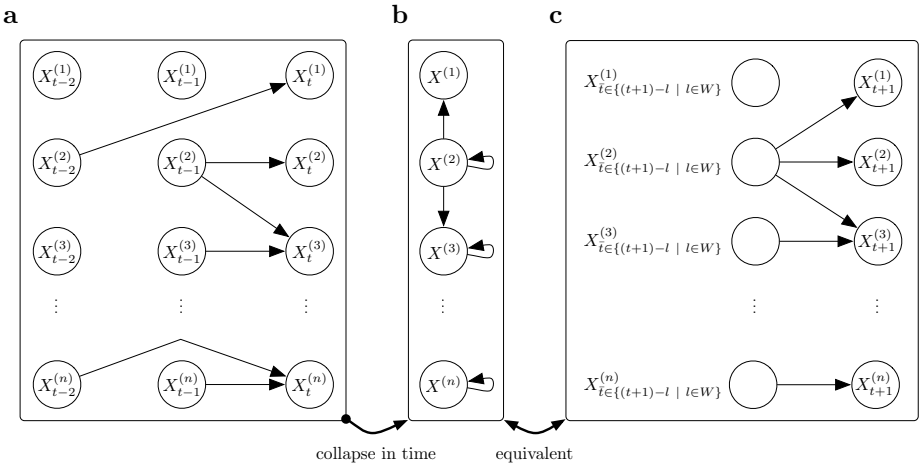


Figure 5.1: Schematic illustration of DBNs and networks learnt using the SSS. **a** Network illustrates a 2^{nd} order DBN over n variables. **b** Network in (a) collapsed in time. Information about the temporal offset of parents to the child is lost in the collapsed network. Links between variables correspond to time-lagged relations, but which are only known to be of lag 1, 2, or both. **c** Network that visualises relations as revealed by the SSS. Each blank node represents multiple instances in time of a single variable. This representation is equivalent to the one shown in (b), as each network can be unambiguously transformed into the other.

other words, the parent set $pa(X^{(i)})$ of each node i [equation (1.19)] only contains nodes with the correct time-offset to the child, but no others. Thus, in the DBN, only nodes corresponding to the precise time-lag are linked (Fig. 5.1a). As discussed next, this is different for the SSS.

In contrast to the BD scores, the SSS has been specifically designed for spike train data, and it also differs with respect to other aspects. The SSS has not been derived in a Bayesian manner, and it is constructed in order to detect a particular type of coordinated activity in a computationally highly efficient manner: excitatory relationships. This is achieved by its key component: the calculation of the activity level series. Spike trains are transformed by an enrichment of information about recent spiking activity. This more informative time-series eliminates the need for numerous point-to-point comparisons, which commonly accompany the detection of correlations over different time-lags, such that efficiency is largely increased. However, this advantage comes with a loss of precision: Although the SSS can reveal correlation over multiple time-lags (corresponding to the lag-window) resulting networks give no information about the precise lag of revealed relations. The SSS does not itemise the different lags of each variable, but treats them as one combined entity (Fig. 5.1c). With respect to the process [equation (3.8)] that corresponds to a learned network,

this means that parent-sets $pa(X^{(i)})$ either comprise all preceding instances $X^{(k)}_{\bar{t}\in\{(t+1)-l \mid l\in W\}}$ of a variable k or none of them. Different to DBNs, the time-lags are not specified more precisely than that, such that networks learned with the SSS can be understood as a DBN that has been collapsed in time (Fig. 5.1b).[2]

Despite their differences, both the BD scores and the SSS face the same fundamental obstacle: In general, spike train data is insufficient to reconstruct structural connectivity between units. This is because the neurons' activity is sampled with relatively low spatial density. But even if data could be collected from all neurons within a system or a part of it, their correlated activity (*functional connectivity*) does not convey enough information for an unambiguous decision on their causal interactions, due to observational equivalence (Section 1.4.2). Learning networks with either type of score is therefore not an attempt to suggest structural connectivity, but visualising functional relationships is thought to aid data analysis. As discussed in detail later (Section 5.2), interpretation of these networks differs for both scores; this is because their underlying models differ. The consequence is that links revealed by the BD scores represent probabilistic coupling between units, whereas those learned with the SSS represent cause-effect relationships. More precisely, they can be understood according to the integrate and fire paradigm (Section A.1.1) with links representing excitatory effects (*effective connectivity*) [Friston, 1994, Sporns et al., 2004].

DBNs can generally code information more precisely than those networks learned with the SSS; but in return the SSS is computationally very efficient, which makes it applicable to high dimensional data-sets, as required for practical application. For such analyses the SSS can be used in a Bayesian framework, as briefly outlined in the next section.

5.1.1 Bayes' Theorem and the SSS

The BD score of a network is proportional to the posterior probability of that network for some given data. The combination of prior information and likelihood (but not the evidence)[3] according to Bayes' theorem is comprised in the formula of the score. In contrast, this is not the case for the SSS, which does not constitute a posterior probability of the network.[4] Instead, the score should

[2]The analogy between networks learned with the SSS and time-collapsed DBNs assumes that suitable parameters have been chosen, such that the SSS's lag-window matches the lags considered in the DBN.

[3]See footnote 1 on page 28.

[4]It is possible that the SSS can be found to be a posterior probability for a particular set of assumptions. (I.e. different neuron models, which are assigned certain priors, could result

be understood as a likelihood measure, i.e. the probability that the given data could have been generated by the network to assess. This likelihood can then be used to calculate a posterior probability according to Bayes' theorem. Therefore, as outlined in section 1.4.1, each network structure to assess has to be assigned a prior probability, but which is not discussed here as it is specific to the problem under study. However, note the special case where the SSS is used for ranking networks without explicitely defining priors. Here, the preference of a network is purely determined by its likelihood, which means that an implicit uniform prior over all analysed networks is chosen. This is an appropriate choice if no prior information is available.

The foregoing considerations pointed out fundamental differences between BD scores and the SSS, especially concerning the temporal precision of revealed networks whose links carry different time-lag information for both scores. Additionally, network links differ in meaning, as the next section shows.

5.2 Translation of Data to Networks — a Question of Semantics

Networks can be found in a variety of domains [Bornholdt and Schuster, 2003, Sporns et al., 2004, Borgatti et al., 2009, Lazer et al., 2009, Hidalgo et al., 2009, Bullmore and Sporns, 2009] and the same structure possibly has a different meaning in each of these. This is because networks are used to encode different kinds of information: Train- or tube-networks, for example, generally represent the existence of direct physical connections between different stations. But connections can also be more abstract and encode references only, like for networks representing web-links from one web-site to another. Depending on the desired semantics of a network, different information is needed for its construction. Vice versa, the kind of information that is available determines which semantics a derived network can possibly have. In this section it will be discussed how the BD scores and the SSS use given data, i.e. which information is extracted for inference. This will show that both scores interpret the data differently, such that resulting networks do not share the same semantics.

in the score's formula when applying Bayes' theorem.) However, no attempt is made here in order to present the SSS as the outcome of such Bayesian concept, which is subject to future work.

5.2.1 Data Interpretation by BD Scores

As mentioned during the introduction of the BD scores (Section 2.2), these can handle any discrete data (with a finite number of states). No particular data-source was considered for their design and they are based on rather general assumptions; one of them is that states of variables do not possess any semantics. The ordering of the r_i states variable i can take is thus not important and any permutation of these results in the same score value. Therefore, the score cannot distinguish two complementary spike trains: If in one spike train all spiking time-bins are non-spiking ones in the other and vice versa, they can be exchanged without affecting the score value. In other words, the score cannot tell apart a spike train from a extremely active unit from one with very sparse firing, for example. For practical application this ambiguity is unlikely to have an effect when time-bins with high temporal resolution (1 msec) are chosen: If few or even single units are represented per channel, spike-events will be rare at common spike rates; channels that are active more than half of the time will probably not exist. Hence, it is unlikely that the data contain two time-series, which are complementary to each other. But the before mentioned rare occurrence of spikes can cause a different kind of problem, which is considered next.

The BD score assigns high score values to networks in which parents are good predictors of the child; more precisely, if the joint parent state and that of the child are reliably coupled. In a good network, particular firing patterns of parent nodes regularly evoke a certain response of their child. While there is nothing wrong with this criterion, practical network inference can be highly problematic when firing rates are low. In these cases spike events are clearly outnumbered by time-bins that code non-spiking. This enormous imbalance impacts on the assignment of scores: Structures are favoured for which parent- and child-states are well correlated; but when most time-bins code non-spiking, a reliable coupling between the parents and their child means that specifically their non-active states match well. Comparatively, occasional spikes are seen as minor disagreements, which have far less weight on the score value than the large number of matching non-spikes.[5] While the highest score value is assigned to the structure where both non-spiking and spiking activity of the parents and the child match, it is extremely difficult to identify that structure. The reason

[5]This effect is due to the Γ-function's increasing growth for larger numbers. This can be easily seen by its relationship to the factorial [equation (2.13)] for which *Stirling's formula* [Feller, 1950, pp.52] holds for large arguments x:

$$x! \approx \sqrt{2\pi} x^{x+\frac{1}{2}} e^{-x} \ . \tag{5.1}$$

As growth of $x!$ increases in x, parent-child-state combinations with high counts have higher weight for the score value than those for which counts are small.

for this is the similarity of scores for structures for which non-spiking bins match well. The score's undifferentiated assignment is, however, correct, which can be best understood in an extreme situation: Consider the m^{th} order Markov model

$$P\left(X_t^{(i)}\middle|\left(X_{\tilde{t}}^{(1)},\ldots,X_{\tilde{t}}^{(n)}\right)_{\tilde{t}\in\{t-1,\ldots,t-m\}}\right)=P\left(X_t^{(i)}\middle|\,pa(X^{(i)})\right) \tag{5.2}$$

for which the most appropriate parent set $pa(X^{(i)})$ is to be determined. If $X^{(i)}$ is in fact a constant, any combination of parents predicts its state equally well! Their scores should thus be similar or even equal. Neural data with rare spike events resemble a similar situation: Variables appear to be nearly constant, by which it becomes likely that several equally high scoring networks exist. When using search heuristics to recover these networks, their undifferentiated scores constitute different local maxima or even a plateau of solutions; this can lead to unstable results, such that connectivity of inferred networks differs substantially. This is problematic, since interpretation of highly diverse results is generally complicated, as model averaging techniques (Section B.3.1) might fail to identify any consensus between them. Analysing spike train data with the BD scores can thus be problematic; however, methods that can help to circumvent problems exist and will be discussed together with their drawbacks, later.

5.2.2 Data Interpretation by the SSS

The SSS has been designed with the aim to reveal excitatory relationships between neural entities. The score is particularly laid out for one kind of data — spike trains. With this specific adaptation it is possible to respect the semantics of these data: Unlike the BD scores, absolute variable states matter to the SSS, which treats spiking and non-spiking events very differently. Indeed, since the SSS is designed to detect time-lagged correlation of spiking activity, it is not concerned with periods without any activity; the score of parent configurations is only determined by the behaviour of the child after its parents have been active. Whether the child is silent or spiking at other times does not affect the score value. Networks learned with the SSS therefore do not represent close coupling of linked units, but rather uni-directional relations: A link indicates that parent spikes reliably trigger spike responses of the child. Revealed connections do not imply anything about the child's activity at times its parents are silent (unlike the BD scores).

The SSS's emphasis on periods where putative causal units are active is motivated by the relatively low spatial density and coverage at which spike train data can be collected nowadays. In detail, neurons for which spike trains are

recorded may show correlated activity with other recorded units; however, neurons for which no data is collected can also trigger firing in observed units. With such spike responses in mind, the SSS has been constructed: Child responses to hidden units do not affect the score value; the score only reflects the degree to which child spikes follow activity of observable parents. The SSS can thus be used to find plausible causal relations between observed units; these explanations of the data do not exclude the existence of external, hidden factors. The BD scores differ with respect to this aspect: While the SSS ignores *uncoordinated* spikes of the child, the BD scores weight activity of the child regardless whether its parents are active or not. Any lack of synchronisation of joint parent states and that of the child are interpreted as an indication for stochastic independence. Due to uncoordinated spikes, the child's activity might not be fully explainable by observed units, such that links corresponding to such *partial correlation* might not be learned with the BD scores.

Previous sections have explained how the BD scores and the SSS differ, as well as how learned networks are to be understood. It has also been explained why the application of BD scores to spike trains can be problematic, and ways to address these problems will be outlined shortly. Before that, however, it should be briefly considered whether or not it is generally useful to infer DBNs instead of temporally less precise networks corresponding to the SSS. This question is subject to the next section.

5.3 Side-Effects of High Precision

In comparison to networks that can be learned with the SSS, DBNs are generally more precise about the lags at which units are correlated; however, the higher precision comes at an over-proportionally high computational cost associated with learning these networks. This is due to the greatly increased number of potential models to evaluate (Section B.2). Additionally, in order to learn such a more complex model, more information is needed. In the following, this fact is illustrated for spike train data.

Neural responses to repeated stimulation can vary widely even if the stimulus itself is un-varied [Buonomano and Maass, 2009]; the relative timing of spikes between different neurons might jitter, for example (Fig. 5.2a). A DBN that can represent the different time offsets due to jittering is shown in figure 5.2b: Variables A and B are both represented for 4 different relative times in order to capture dependencies within the jitter-range. Given that the order of the DBN is high enough to accommodate jittering relations, learning corresponding links with BD scores can be problematic, as the score treats all variables

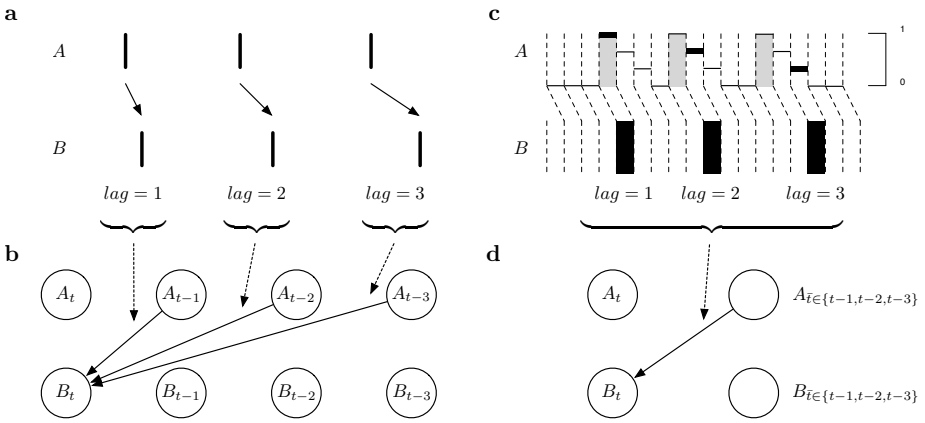


Figure 5.2: Correlated spiking of two units jittering in time and corresponding models. **a** Spike trains of two units A and B. Firing of neuron A triggers responses of B with different time-lags ($lag = 1, 2, 3$). **b** Graphical 3^{rd} order model that can represent the jittering responses of unit B to unit A. **c** Application of SSS to spike trains shown in (a): activity levels superimposed and snapshots triggered by unit B highlighted ($d = 3^{-1}$, $\Delta t = 1$). Jittering responses of unit B to unit A are interpreted jointly as indications for their relationship. **d** Graphical model that visualises relation as revealed by the SSS. Different time-lags between variables are represented collectively.

as separate entities and does not recognise multiple instances of the same variable at different times. Correlated firing that jitters in time is thus recognised as correlation between different variables that only appears sporadically. The coupling between the triggering and responding variables may thus appear too unreliable for a score value that is good enough to accept the corresponding link. Ultimately, it would be desirable, if the jittering relation was reflected in the model by linking all instances of variables that correspond to the jitter-range (Fig. 5.2b). This, however, requires the parent nodes and the child to show even stronger coupling: Joint-parent states and responses of the child would have to be closely locked in oder to counteract the BD scores' inherent tendency towards sparse networks. If neural interactions in the data are weak, jittering can cause them not to be inferred with these scores.

The same situation of jittering responses is now considered for the SSS, which pools variables with different time-lags (Fig. 5.1c). As mentioned earlier (Section 5.1), pooling causes a loss of information about exact time-lags, but with the advantage that indications for correlation are gathered together from all pooled units. Dependencies that are spread over different time-lags manifest in subtle evidence for each; these signs are combined when considering different time-lags jointly (Fig. 5.2c). Weak relations jittering in time can thereby become

evident, such that they can be detected and represented in learned networks (Fig. 5.2d). In such cases, the SSS's inability to determine precise time-lags can pay off by increased sensitivity.

Applying the BD scores to spike train data can be problematic, as outlined in the foregoing sections. But as DBNs can generally yield precise information about the lags at which units are correlated, it is worthwhile to elaborate on how the mentioned problems can be addressed. The following section therefore gives an overview of techniques that can facilitate the application of the BD scores to spike train data.

5.4 Using BD Scores on Spike Train Data

The first application of a BD score for network inference from electrophysiological data has been reported by Smith et al. [2006]. This pioneering study demonstrated successful recovery of information flow networks in the song-bird brain from which multi-unit activity has been collected and analysed with the BDe score. Learned networks matched anatomical knowledge very well and thereby confirmed the success of the approach. This work has shown that the BD scores are suitable in order to reveal neural interactions — at the same time it inspires how these scores could be used to analyse spike train data: Smith et al. collected multi-unit voltage traces, which were smoothed with a root mean square (RMS) method. The resulting data were then discretised (in 3 categories) such that the BDe score could be applied. In essence, smoothing and discretisation were key in pre-processing the data. The same processing steps can also be transferred to spike train data: The binary time-series need to be converted to continuous data, like smoothed voltage traces, and can then be re-discretised to different binary states or possibly more than two levels of activity. These steps can cure the problematic characteristics of spike train data: the enormous imbalance in the number of spiking- to non-spiking bins and temporal jitter. Smoothing techniques flatten abrupt and transient peaks in activity, by rounding of transitions between different levels of activity. This filtering can cause a spread of activity over more time-bins; the number of time-bins with activity can thereby become more balanced to those without any activity. Additionally, since transitions occur more slowly, minor differences in timing only cause minor changes in activity, such that temporal jitter is weakened in effect. The final (re-)discretisation step allows for additional fine-tuning: The number and boundaries of discretisation levels can be used to further even out the occurrence of different levels of activity, for example. Together, these preparatory steps can potentially transform spike train data suitably for the BD score; how-

ever, disadvantages come along, which are explained together with appropriate methods next.

5.4.1 Continuousation of Spike Trains

A natural conversion of spike trains to continuous valued data is the calculation of firing rates. A few of these methods are presented here, which, in general, consider all spike events over a certain time-interval. Weighting and relating the spikes to the length of the interval then yields a firing rate estimate.[6] The binary time-series, whose points reflect instantaneous activity, is thereby converted to a time-series in which points are estimated firing rates.

One of the simplest firing rate estimation techniques is know as the *instantaneous firing rate* [Dayan and Abbott, 2005, p.164]. For any inter-spike interval (ISI) of two directly succeeding spikes the firing rate during that period is defined as the inverse of its length $|ISI|$:

$$\text{rate}(ISI) = \frac{1}{|ISI|} \text{ Hz}, \tag{5.3}$$

under the assumption that $|ISI| > 0$. This coarse operation already smoothes the spike train significantly, which itself can be understood as coding only two firing rates: 0 Hz and (bin size)$^{-1}$ Hz. Other simple techniques to estimate firing rates exist; for instance, activity can be measured by splitting the spike train into equally sized sections and counting the number of spikes in each of them. The spike-counts can then be normalised by the width of the sections in order to yield firing rates (Fig. 5.3). Another possibility is to use a single window for which spikes within are counted. Sliding the window along the spike train yields a series of counts, which generally vary smoothly (Fig. 5.4). This *sliding-window* approach is the simplest of countless kernel methods than can also be used for firing rate estimation (e.g. [Gabbianti and Koch, 1998, Nawrot et al., 1999, Gerstner and Kistler, 2002, McNames, 2005]). Altogether, the presented examples indicate that firing rates can be estimated in a variety of ways. Adequate continuous valued data can thus be generated from spike trains. In order to be able to apply the BD scores, the next step is to suitably discretise the data. Again, different methods like interval- or quantile-discretisation can be used, but which are not presented here. (See Cios et al. [2007, Chapter 8],

[6] Shorter time-intervals facilitate a precise reflection of on- and off-set of spiking activity in the firing rate; however, firing rates may be over- and under-estimated, respectively. Longer intervals, on the other hand, yield better estimates of firing rates, but fail to precisely capture time-points at which changes in dynamics occur. Estimating firing rates is thus generally signified by a trade-off between preserving timing and quantifying dynamics. The inability to account for both the time and amplitude of dynamics simultaneously corresponds to *Heisenberg's Uncertainty Principle* [Heisenberg, 1927, 1983].

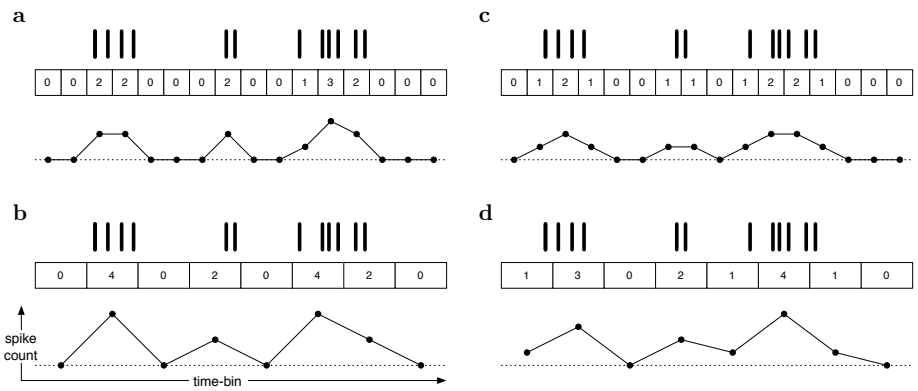


Figure 5.3: Variations in spike-count curves resulting from different section-width and time-offset. Spike trains (vertical bars) identical for all four parts of the figure in which spike-counts are determined for sections of different widths (a,c vs. b,d) and time-offset (a,b vs. c,d). Numbers in rectangles correspond to associated spike-counts, which are plotted as a spike-count curve below. **a** Small section-width and **b** large section-width with boundaries conveniently aligned to spikes, such that bursting and silent periods are suitably reflected in the activity profiles. Onsets of bursts are represented distinctively. **c** Small section-width and **d** large section-width aligned differently than in (a) and (b): Although the same spike train is used, differently aligned boundaries render spike-count curves less expressive. In comparison to curves in (a) and (b), contrast between bursting and silent phases is reduced. Additionally, burst onsets are represented delayed.

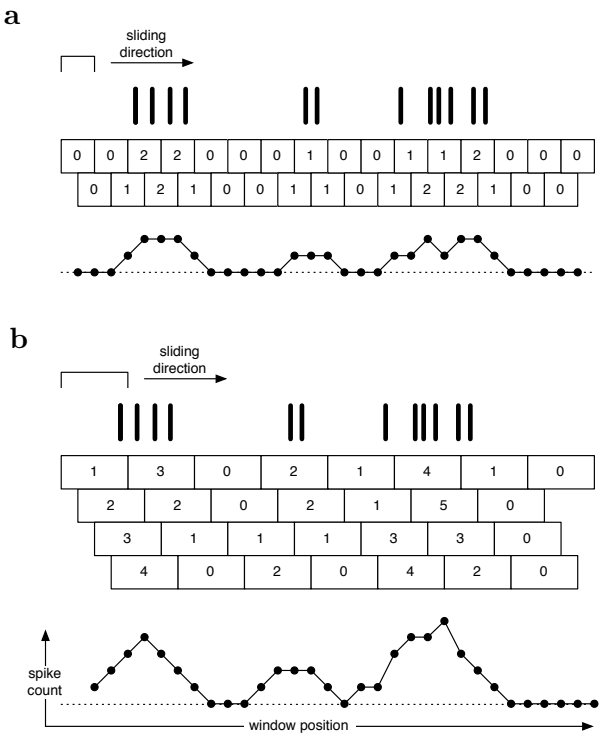


Figure 5.4: Firing rate estimation using a sliding window approach. Spike-counts resulting from successive window positions shown in rectangles below spike train (vertical bars). Resulting spike-count curves shown for two different window widths: **a** Small window width: Activity level changes reflect temporal on- and offsets of spiking well, but show low contrast concerning the actual magnitude of activity. **b** Large window width: High contrast in activity level, but exact on- and off-set of bursts cannot be recognised by a spike-count threshold.

for example.) It is instead noted, that generally both the continuousation and discretisation step inevitably involve a loss of information. This is obvious for a discretisation, where commonly whole ranges of values are reduced to relatively few categories, such that quantitative information is reduced. But transforming the spike train to a firing rate is also lossy, which is explained in the following. As indicated earlier, the spike train itself represents one of two firing rates, which instantaneously adopt to changes in dynamics. But this firing rate fails to inform about recent spikes, which might have occurred in a burst, for example. Multiple time-bins must be combined to a firing rate that codes this information. This means averaging over time by which information about the time-point of changes in dynamics degrades. (See also footnote 6.) Both steps (continuousation and discretisation) can thus lead to a loss of information and must be performed carefully: According to the data processing inequality [Borst and Theunissen, 1999, Quiroga and Panzeri, 2009], degraded information cannot be recovered at later stages. It is thus important to preserve as much valuable information about neural interplay as possible, in order to improve chances for successful network recovery.

As explained above, smoothing data can render inference of higher order models fruitless when required information is removed from the data. Since the SSS is based on smoothed spike trains, in form of the activity-level series, it must be questioned whether it is affected by a loss of information or not. Smoothing concepts discussed earlier and the activity level calculation differ, since the latter preserves all spike-events with full temporal precision. (Additionally to the precise onset of neural activity, the activity level's slow decay informs about how recently a spike occurred.[7]) The score also uses untransformed spike trains to trigger the snapshots of the activity level series. Here, again, the full temporal precision of spike trains is utilised. In conclusion, the intent behind using smoothing methods differs for both types of scores: For the BD scores these techniques may be required in order to match characteristics of the data to the score. In contrast, smoothing is a central part of the SSS for which a particular lossless low-pass filter is used, such that relations over different temporal lags can be efficiently determined.

Approaches for inference of information flow networks from spike train data need to take into account its special characteristics. Data transformations may be necessary in order to facilitate the application of existing methods like the BD scores. Alternatively, the SSS can be used, which utilises the semantics of

[7]Note that the activity level calculation is not a firing rate estimate. If it would be interpreted as such, firing rates could be underestimated: If spikes occur closely to each other, such that activity from a spike is not fully decayed at the time another spike follows, the *excessing* activity is ignored, such that the leading spike is not fully weighted.

the spike train and weights them accordingly. By clarifying these points, this chapter has (retroactively) motivated the need for a score that is specifically adopted to spike trains. The following chapter 6 now shows how the performance of these or any other network inference technique can be assessed with neural simulations.

Chapter 6

Assessing the SSS

The SSS has been introduced (Chapter 3) and shown to assign plausible score values to networks in toy examples 5 and 6 — examples of low dimensionality and very short data-sets. Such simplistic demonstrations can give a glimpse of the score's characteristics, but they do not illustrate its performance in practical situations. This requires a more realistic set-up dealing with complex data; i.e. data with dozens of channels, a longer length of the data in the order of minutes, and relations between data channels appearing to be rather stochastic than deterministic. Increasing the dimension, length, and randomness of examples prevents their discussion in the previous manner where relations between spike trains could be detected by eye and directly confirmed the score's rating of networks to be reasonable. Even a moderate increase in the number of data channels results in an over-proportional growth in the number of potential networks (Section B.2), such that these could neither be discussed separately nor even displayed appropriately all together. Therefore, a more general concept is needed, which facilitates the assessment of the SSS in realistic situations. In this chapter a suitable framework is presented, which combines neural simulations, network learning, and the evaluation of learned networks. New methods will be proposed in order to quantify relevant factors for successful network inference and to classify learned networks' links according to their plausibility, which facilitates an equitable performance assessment under partial observability conditions. The concepts to be presented are not specific to SSS, but they also suit application with alternative approaches to network inference.

6.1 A Performance Assessment-Framework for Neural Network Inference Techniques

Before utilising novel analysis methods, their capabilities and limitations must be understood in order to be able to review their results critically. New techniques should thus be examined by both mathematical analysis and, if possible, in simulation environments.

Exact mathematical studies are limited by the assumptions that enter the reasoning process. Using few or fairly weak assumptions may not be sufficient in order to derive strong statements about the method. Contrariwise, making several or strong presumptions can yield very explicit statements, but whose validity is limited to a special case (formed by the assumptions). Using such restricted result for the analysis of biological data requires extra care in order to assure that the assumed conditions are actually met; but whether or not these are fulfilled may be unknown due to insufficient knowledge about the studied system. For practical purposes it is thus important to know how a technique's results are affected if its presumptions are not accurately met: *"Do conclusions derived from the presumptions seem to hold in a slightly varied situation, too?"*, if not, *"How do they differ?"*, and so forth. A countless number of possibilities to violate assumptions exists out of which those that are practically relevant must be identified. This is where a simulation environment becomes useful in order to generate data under precisely controlled conditions: The technique to assess is applied to the artificial data and its results are compared to the expected outcome. Here the expected results can be derived because the circumstances under which the data are generated are known. Often, the next step is to vary the parameters of the simulation systematically to investigate changes in results of the method. This can reveal critical factors and their relative importance. Hence, by studying a new method using both a mathematical and a heuristic simulation approach one can get a comprehensive picture of the method's qualities. Finally, this characterisation of the technique can justify belief in its results for real data.

The SSS has been mathematically investigated in chapter 4 where several features of the score have been proven in an exact manner. Here, the SSS is tested in a simulation framework: Model neurons are connected according to a known network, which is then simulated in order to generate artificial spike train data; the SSS is applied to these data to learn a network, which is compared against the simulated network. This concept is motivated by the expectation that a good method will reveal the simulated network or at least a similar structure. In practice this concept involves 4 major steps (Fig. 6.1):

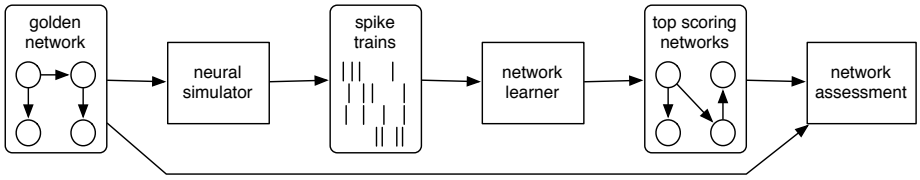


Figure 6.1: Generic assessment framework (work-flow from left to right). A golden network defines synaptic connections in a neuron model, which is simulated in order to generate spike train data. The artificial data are used by the network learner, which returns its results for the final assessment of the learned networks.

1. **Select a golden network.** The golden network represents the structural architecture of a neural network that will be simulated in the next step. Nodes in the golden network represent the neural entities (e.g. single neurons or populations of neurons) and links between nodes define causal interactions between them (e.g. excitation or inhibition). The network can either be constructed by hand or generated randomly.

2. **Simulate the network.** The golden network defines interactions between its nodes, which are associated with one neural model each. Which kind of model is used depends on what the neural entities are and on the desired level of model precision (Section A.1). The dynamics of the golden network are generated by simulating the neuron models, which yields artificial spike train data for each node.

3. **Learn networks.** The method, e.g. the SSS together with an appropriate learning algorithm (Appendix D), is applied to the simulated spike trains. The outcome of the network learning procedure is a set of networks, which represent the relations that were found in the data.

4. **Assess learned networks.** In the final step learned networks are related to the golden network in order to assess their quality. As will be discussed later, in general, a link-by-link comparison between the golden and a learned network is not sensible for assessing network quality. Instead the golden network is analysed first in order to determine links that are theoretically reasonable to be inferred from the data; learned networks are then assessed by their degree of plausibility. Different measures of how reasonable a learned network is yields a multi-dimensional vector expressing the learning performance of the method.

Before details on the actual implementation of the framework are given it is worthwhile to reflect on two general issues: First, how complicated is the task to be achieved? And secondly, how good are the results? The answers to these

trivially sounding questions are the cornerstones of the assessment framework. Obviously, without a proper measure for the quality of learned networks they cannot be assessed. But it is similarly important to regard the difficulty of obtaining the results. Both aspects determine the performance of a method, which is the quality of results given the severity of the challenge. In the following, the question of how to assess learned networks is addressed in section 6.3. The next section discusses the first question and a measure to rate the network learning task is proposed.

6.2 On the Degree of Learning-Difficulty

The assessment framework can be used to test the performance of a technique over a wide range of parameters, those of the neural simulation, for example. Learned networks might turn out to be satisfactory for some parameter combinations but not for others. If the neural simulation was realistic and at the same time parameters of a real system under study were known, one could test whether the technique delivers good networks in practical application or not. Unfortunately, both conditions will not be met most of the time: Computational constraints can impede neural network simulations on a realistic scale (Section A.1), and very few biological systems are known well enough in order to correctly parameterise detailed models of them. On the other hand, in much simplified models parameters cannot be related to the studied system; this is the case for the neuron models that will be used in this thesis. In order to relate simulation results to the biological system, an abstract measure to characterise the data is proposed, which assesses the difficulty of network inference. It is thereby possible to roughly estimate the expected performance of the SSS on real spike trains.

Inference from data is a common task whose difficulty can vary within a wide range. How complicated it is to draw conclusions from data depends on many factors, such as the particular conclusion and the amount of information about it in the data. Specific to the problem of network inference from spike trains this means: Learning a good network mainly depends on the complexity of the underlying system, the informativeness of the spike train data, and the budget of computing time. Hence, network inference from a simple two-neuron simulation poses a much easier problem than a large scale network simulation with realistic observability and noise levels. But what exactly determines the severity of the network learning task? How can the identified relevant factors be quantified? And to what extend does the performance of network inference depend on these? Two factors concerning these questions play a major role within the scope of the assessment framework: the complexity of the simulated system and data

quality with respect to information about network connectivity. Subsequent sections discuss these two aspects and measures are proposed, such that the dependency of learning performance on relevant factors can be quantified.

6.2.1 Complexity of the Simulation

The complexity of the simulated neural system depends on its components: the number of neurons, their connectivity, and the dynamics of each individual neuron. Neurons seem to exhibit a rather limited behavioural repertoire compared to the complex behaviour of higher vertebrates. Some models simplify neural dynamics even further down to a few key aspects. For example, leaky integrate and fire models can replicate certain observations of spiking neurons; however, they are unable to describe sub-threshold dynamics for which conductance based models are needed (Section A.1). The latter kind may thus show more complex behaviour, which is important when connecting several neurons to neural networks: The more complex the building blocks of the network are the more complex its overall dynamics can be. This means that increased model complexity can make interactions between elements appear to be more probabilistic than with simplistic neuron models. Because it is generally harder to detect probabilistic relations than deterministic ones, this of course affects network inference. The type of neuron model used to simulate neural networks can thus have an impact on network learning performance.

The brain is a good example of how network size influences system complexity: A single neuron seems to be capable of few computational operations only, but combining many such relatively simple units can yield a highly complex system with a huge variety of abilities. However, sheer network size (i.e. number of neurons and links between them) does not explain this arising complexity, but the actual connectivity patterns must also be taken into account. Such patterns, also called *motifs*, have been shown to be able to influence the dynamics of a neural system [Sporns et al., 2000, Sporns and Tononi, 2002, Galan, 2008, Bullmore and Sporns, 2009]. Reliably quantifying motifs with existing measures (see e.g. Strogatz [2001], Albert and Barabasi [2002], Sporns [2003], Costa et al. [2007]) commonly requires very large networks ($> 1,000$ nodes). These cannot be applied to the relatively small networks used in simulations within this framework; networks are therefore characterised in an ad hoc manner. For example, in a strict feed-forward network without any recurrent loops, activity is limited to flow into a predefined direction. Comparatively, a network consisting of interconnected clusters of nodes with recurrent connections within each cluster can exhibit a larger variety of fundamental states (e.g. combinations of neuron-groups that are active simultaneously) or different orders of sequential

cluster activation. The cluster-network can thus generate data with a larger variety in dynamics than the feed-forward network and it would be expected that network inference from these more complex data is a harder problem than for the feed-forward dynamics. The topology of the golden network can thus have an impact on the the performance of network learning.

6.2.2 Data Informativeness

Studying a system involves collection of corresponding data. Using these data for inference about the system can vary in difficulty, which depends on how much information the data conveys about the system: First of all, the data must be relevant with respect to aspects of interest. And additionally, quality and quantity of data affect how much information it can maximally convey. For example, a noise free data set with high temporal resolution can be more informative than one with high noise levels collected at a low sampling rate [Nyquist, 1928, Shannon, 1949]. Likewise, large data sets are more likely to contain several observations of a particular effect than just a few data-points in which the effect might only be observed once. Observing the effect repeatedly can strengthen belief in its existence by weakening the alternative of having simply observed an artefact.

Factors like data-length or noise level are easily controlled in a simulation environment; however, controlling the relevance of the data may not be straightforward. With respect to the assessment framework, it might seem unexpected why a spike train generated by a simulated neural network should be controlled and quantified with respect to its relevance; but recalling that causal interactions shall be inferred from the data shows why this makes sense: Consider that the neural simulation of the golden network was parameterised such that post-synaptic potentials induced by connected neurons would never suffice in order to evoke a spike in the receiving neuron. All model neurons were spontaneously active, but no other than these random spikes would be observed. Hence, spike trains would convey information about rates of spontaneous activity of the neurons, but this information is irrelevant with respect to causal interactions between modelled units. The lack of information about network connectivity in random spike trains makes them useless for network inference, and expecting that sensible relations could be recovered by any method is unreasonable. On the other hand, if the data contains sufficient indications about the interplay of units, these interactions should be expected to be revealed by network inference. In order to ensure that generated spike trains are enriched with relevant information about their underlying network, the simulation must be parametrised such that post-synaptic potentials are sufficiently excitatory.

Excitations can then initiate spikes that convey information about network connectivity through their temporal correlation.[1]

The difficulty of network inference varies depending on how distinct relations between units are reflected in the data. For a fair assessment of an analysis technique, the severity of the problems it is applied to needs to be taken into account. Therefore, the spike train data are quantified with respect to their relevant information concerning the neural connectivity.

Quantifying the Informative Value of Spike Trains

The neural simulation of the golden network yields spike trains, which shall be characterised with respect to their informativeness about the networks connectivity. Parameters of the neural simulation (spontaneous activity level, synaptic efficiency) control the ratio of uncorrelated spontaneous spikes and those which are evoked by post synaptic potentials. As explained earlier, this mixture of uncorrelated and correlated spikes determines the amount of information conveyed about the network. In order to quantify the degree of informativeness, each spike train is characterised by the proportion of spontaneous spikes and evoked ones in

Definition 5 (Impetus) *The impetus of a spike train is the relative increase in the number of spikes evoked by post-synaptic potentials to the number of spontaneous spikes:*

$$impetus = 100 \cdot \frac{\#evoked\ spikes}{\#spontaneous\ spikes}\ \% \ . \tag{6.1}$$

If, for example, *impetus*=0% then no spikes are evoked at all and the spike train only consists of spontaneous spikes, i.e. uncorrelated random spikes. For *impetus*=100% the spike train is a mixture of two halves: spontaneous spikes and evoked spikes. Thus, when the impetus is low the data is similar to the uncorrelated spike trains, representing inherent spontaneous activity of model neurons. A higher impetus indicates a more autonomous system with higher self-dynamics. In such systems, the spike trains are more informative about network connectivity, which is expected to improve network recovery.

Unfortunately, the impetus cannot be calculated for real data.[2] If the impetus was known, the quality of networks learned from recorded data could be

[1] Inhibition can also result in informative correlations, by impeding spontaneous activity, for example.

[2] Franziska Matthäus and William Heitler noted that certain experimental set-ups might facilitate the determination of the impetus: If chemical synaptic transmission is pharmaceutically blocked, observed activity corresponds to intrinsic spiking (neglecting influences of electrical synapses); comparing this base-line activity to recordings where synaptic transmission is unblocked can yield a reasonable estimate of the system's impetus.

appraised by results of the simulation with a similar impetus. However, the impetus may be basic enough in order to enable experimenters to roughly estimate the impetus for their data. For example, recordings within a feed-forward type structure are expected to exhibit a higher impetus than from an area with many converging external inputs.

Measuring the impetus of a spike train gives an indication for the informativeness of the data about network connectivity, which correlates with the severity of the network inference task. The initial position for learning networks can thus be rated; how it can be related to the quality of learned networks is discussed next.

6.3 Assessing Learned Networks

Previous sections discussed how the severity of the network learning problem can be quantified. Still, before the assessment framework can be applied, a decision needs to be made on how the quality of results shall be measured. Only when measures for both problem complexity and result quality are available, is a systematic investigation possible. This section addresses the question of a reasonable result-quality-meter in two parts: First, measures for the comparison of a learned network to a reference network are discussed. Thereafter, the focus is on how such reference network should actually appear. However, until then it is assumed that this decision has been already made and that a well defined network is given, which serves as a reference for any learned network.

6.3.1 Comparing Networks

Given a learned network and a reference network the concern now is to assess their similarity, which will be done by comparing the networks link by link. This matching can be measured from two perspectives: What percentage of links in the reference network can be found in the learned network, and, vice versa, to what extend do links of the learned network represent ones of the reference network? These percentages can easily be calculated from the adjacency matrices belonging to the networks (Section B.1). In the following let $A = (a_{ij})$ denote the adjacency matrix of the learned network and $B = (B_{ij})$ that of the reference

network. By comparing the learned network to the reference network we find:

$$\text{recovery rate} = \begin{cases} \frac{\sum_{i,j} a_{ij} b_{ij}}{\sum_{i,j} b_{ij}} , & \text{if } \sum_{i,j} b_{ij} > 0 \\ 0 , & \text{otherwise} \end{cases} , \tag{6.2}$$

$$\text{miss rate} = \begin{cases} \frac{\sum_{i,j} b_{ij} - a_{ij} b_{ij}}{\sum_{i,j} b_{ij}} , & \text{if } \sum_{i,j} b_{ij} > 0 \\ 0 , & \text{otherwise} \end{cases} , \text{ and} \tag{6.3}$$

$$\text{recovery rate} = 1 - \text{miss rate} . \tag{6.4}$$

By changing the perspective and comparing the reference network to the learned one, expressions become slightly different:

$$\text{precision} = \begin{cases} \frac{\sum_{i,j} a_{ij} b_{ij}}{\sum_{i,j} a_{ij}} , & \text{if } \sum_{i,j} a_{ij} > 0 \\ 0 , & \text{otherwise} \end{cases} , \tag{6.5}$$

$$\text{false positive rate} = \begin{cases} \frac{\sum_{i,j} a_{ij} - a_{ij} b_{ij}}{\sum_{i,j} a_{ij}} , & \text{if } \sum_{i,j} a_{ij} > 0 \\ 0 , & \text{otherwise} \end{cases} , \text{ and} \tag{6.6}$$

$$\text{precision} = 1 - \text{false positive rate} . \tag{6.7}$$

These basic entities can capture the quality of the learned network: Observing a high recovery rate[3] and a low rate of false positives at the same time represents the optimal case. In contrast, the quality of the learnt network is worst, if the rate of false positives is high while the percentage of recovered links is low. In other cases, i.e. where both percentages are high or low simultaneously, interpretation of results is not straightforward.

(ROC curve) The recovery rate and precision are somewhat dimensionless when analysed separately; they must either be seen in relation to each other or relativised by some other aspect. A common way to combine these rates for a series of networks are so called *receiver operating characteristic* (ROC) curves [Dayan and Abbott, 2005, Fawcett, 2006]. Therefore, networks are learned from the same data, but with different parameters of the score, such that their influence on the performance can be investigated. ROC curves display the false positives rates and precision of the different networks as points in a plane in order to visualise their relationship (Fig. 6.2).

(P-value) Another way to understand precision is by relativising it with respect to chance level. The so called *P-value* is the probability that a given

[3]Please note that the definition of the recovery rate corresponds to sensitivity [Fawcett, 2006]. In order to avoid misleading the reader, it is called differently, because, in contrast to sensitivity, it is not expected to reach 100%, as will be explained later (on page 98).

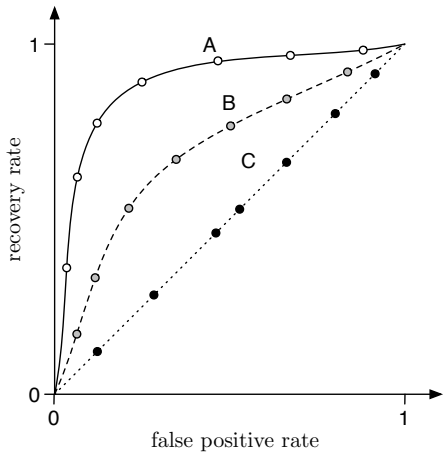


Figure 6.2: Schematic representation of three receiver operating characteristic (ROC) curves (A,B,C). Points represent rate pairs (false positives, recovery rate), which are connected by lines for illustrative purposes. Points on curve A show high recovery rates in combination with low rates of false positives. Performance shown by curve A is thus better than curves B and C, whose points do not show such good combination of coordinates: At any fixed rate of false positives, A has a higher recovery rate than B and C; and likewise, for any fixed recovery rate, curves B and C show more false positives. Note that in this representation curve C corresponds to chance level.

precision was reached by pure chance. The precision's P-value can be easily calculated when recognising the analogy to the following sampling problem: Given an urn of white and red coloured balls, a certain number of them is drawn randomly without replacement. The hypergeometric distribution [Feller, 1950, pp.43] can then be used to calculate the probability that there are at least so many red balls among the drawn ones. With respect to the networks, each of the balls in the urn represents one network link that could have been possibly learned. Balls corresponding to links in the reference network are red, all others are white. Then as many balls are drawn from the urn as there are links in the learned network and the probability of drawing at least as many red balls as there were hits in the learned network is calculated. More mathematically we find the chance level for at least h hits out of p plausible links out of N total links with k links learned as

$$P = \sum_{i=h}^{\min\{p,k\}} q_i \ , \quad \text{where } q_i = \frac{\binom{p}{i}\binom{N-p}{k-i}}{\binom{N}{k}} \ . \tag{6.8}$$

By considering that plausible links might be found by pure chance, the P-value of a learned network reflects the benefit of using the inference technique instead

of guessing networks. Likewise, it is possible to calculate P-Values for other rates (e.g. false positives), but most often the concern is on excluding the possibility that good performance (i.e. high precision) could have originated by chance.

The analysis of the learned network so far has assumed the existence of a reference network. As will be explained next, this is unlikely to be the golden network itself, but a structure that is constructed from it.

6.3.2 Reasonable Network Links to Infer

In the simulation step of the assessment framework (Fig. 6.1), network links define causal relations — signal transmission via synaptic connections — which can translate into correlation between spike trains. Learning a network from the data corresponds to detecting and representing these correlations. Unfortunately, similar correlation can be generated by different causal networks such that all of these become observationally equivalent (Fig. 1.4 on page 21). Hence, any of these potential networks should be considered *correct* although it might differ from the actual golden network used. A simple comparison between the golden network and the one that was inferred from correlations is thus often not a reasonable measure for the quality of the learned network. Instead, learned networks should be assessed by whether they give a plausible explanation for the data or not.

There exists also another situation where the golden network is not a good reference network for learned networks; that is when the studied system can only be partially observed. For example, electrophysiological recordings are limited with respect to spatial resolution at which data can be collected. This factor involves two, often related, aspects: sampling density and coverage of the studied system. Both data collection at low spatial density and observations covering only parts of the system result in incomplete anatomical coverage; collected data lacks information about unobserved elements either on a local or a global scale. For the sake of realism this lack of information is mimicked in the assessment framework: The network simulation imitates several neurons out of which only a sub-set is chosen to be observable; i.e. spike trains of all other units (and thus their existence) are hidden to the network learner. Learned networks consist of observable nodes and links between them only. These networks cannot be compared to the full golden network, because, in the golden network, signals can travel through chains of hidden units, which connect observable neurons with each other. Observable units can show correlation corresponding to such hidden connections and the network learner might infer appropriate links from

the data. Such connections correctly represent detected correlation in the data; however, they might not exist in the golden network. But by the given data the network learner cannot know about the hidden pathway; hence, links, which are plausible due to partial observability, must be acknowledged when assessing learned networks. It is, once again, insufficient to compare the network learned from partial data to the golden network. The next section shows how the beforementioned problems can be overcome by constructing a reference network from the golden network to which learned ones should be compared.

Plausibility Concept

In order to address the discussed problems caused by observational equivalence and partial observability the concept of (link) plausibility is introduced. Its conceptional idea is to analyse the known causal network in order to infer potential correlation between observable units; corresponding network links are considered to be plausible. The ultimate goal is to classify each possible link between observable nodes as either plausible or implausible; plausible links that are learned from the data will be regarded a positive result. The plausibility concept is not restricted to application to the SSS, but it can be used for the assessment of any other network learning technique as well.

Before the concept is defined mathematically, a simple example is considered for illustration. In the network shown in Fig. 6.3a the same connectivity pattern between 4 nodes is repeated in three cliques, which differ with respect to their observability. Depending on which nodes are observable and which ones are not, different links between observable ones appear to be sensible (Fig. 6.3b). For example, plausible links can be those which exist in the full network (e.g. $\underline{A1} \rightarrow \underline{A2}$, $\underline{A3} \rightarrow \underline{A4}$, and $\underline{B1} \rightarrow \underline{B2}$) or certain links for which a directed path from the link's starting node to its end node exists (e.g. $\underline{B1} \rightarrow \textit{(B3)} \rightarrow \underline{B4}$).[4] However, not every directed path might justify a plausible link between two units because it incorporates other observable nodes (e.g. unit $A3$ on the path $\underline{A1} \rightarrow \textit{A3} \rightarrow \underline{A4}$). A better alternative to a link *blocked* in this way would be two links, which connect units according to their causal order (e.g. $\underline{A1} \rightarrow \underline{A3}$, and $\underline{A3} \rightarrow \underline{A4}$). Other implausible links are those which contradict the full network by connecting nodes in the opposite direction of information flow (e.g. $\underline{A2} \rightarrow \underline{A1}$, $\underline{B4} \rightarrow \underline{B1}$, and $\underline{C4} \rightarrow \underline{C2}$). It is also possible that the system's partial observability leads to plausible links for which no directed path between units exists: A common trigger can cause co-ordinated firing between nodes, which are connected via the triggering node only (e.g. $\underline{C2} \leftarrow \textit{(C1)} \rightarrow \textit{(C3)} \rightarrow \underline{C4}$). A link connecting the two co-ordinated nodes (i.e. $C2 \rightarrow C4$) is thus plausible

[4] Start- and end-nodes underlined, nodes on path in full network (given for illustration) italic, hidden nodes in brackets.

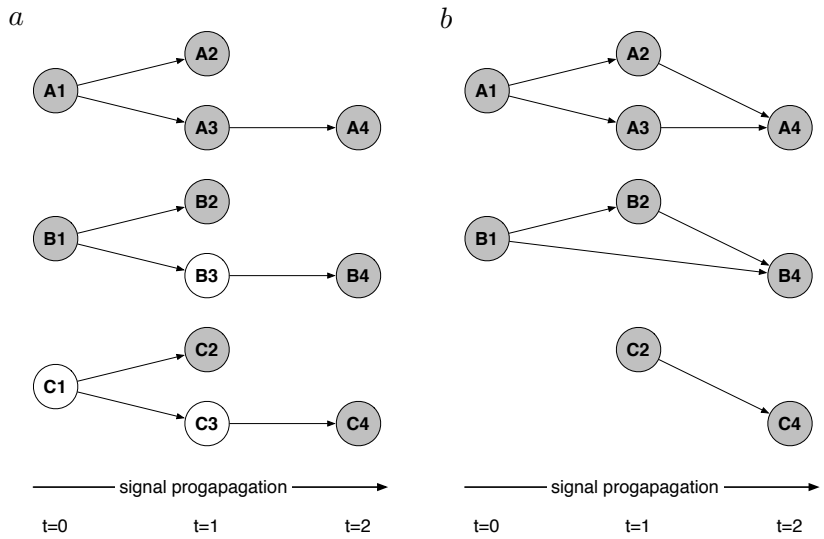


Figure 6.3: Network including observable and non-observable nodes (a) and corresponding plausible links between observable nodes (b). **a** Example network with three cliques (A,B,C) in which the same connectivity pattern is repeated. Cliques differ with respect to their observability: observable units shaded in grey and hidden units unshaded. **b** Observable nodes from network shown in (a) with all plausible links.

if one node could fire within the *plausible lag-window* $[l_{min}, l_{max}]$ of the other node (in the right order). Plausibility of a link thus depends on the full network, which nodes are observable, and parameters specifying the minimal and maximal plausible lag (l_{min}, l_{max}). This is formalised in the following

Definition 6 (plausibility) *Let a and b denote two observable nodes. Node a is called a plausible parent of b, if there exists a node s (in the full network) for which directed paths to both a and b exist such that:*

1. *their lengths*[5] *$l(s \to a)$ and $l(s \to b)$ fulfil*
   $l_{min} \leq l(s \to b) - l(s \to a) \leq l_{max}$, and
2. *if the path $s \to b$ includes a, none of the nodes visited after a fulfils the first condition.*

Node a is called an implausible parent of b otherwise. The link $a \to b$ is called (im)plausible whenever a is a(n im)plausible parent of b.

The first condition in the definition assures that correlation between nodes a and b can arise within the plausible lag-window. The second condition refines

[5]The length $l(a \to b)$ of directed path from node a to b is defined as number of links on the path (Section B.1). The length of a path $a \to b$ directly corresponds to the time-lag a signal needs to propagate from a to b. In the neural simulations shown later (Chapter 7), the time-lag in time-bins (1 msec) is equal to the length of a path.

the set of plausible parents by rejecting those for which correlation can only propagate through another plausible parent of b. In figure 6.3 for example, plausible lags were chosen $(l_{min}, l_{max}) = (1, 3)$, such that the link $\underline{A1} \rightarrow A3 \rightarrow \underline{A4}$ fulfils the first but not the second condition. Different from that, both conditions are met by the link $\underline{B1} \rightarrow (B3) \rightarrow \underline{B4}$. The two criteria plausible links have to fulfil render many links implausible; despite this selectivity, not all links that are classified as plausible are expected to be recovered by network inference. This is because, generally, some redundancy among them exists, which cannot be further reduced due to equivalence.[6] The set of plausible links is therefore a superset of those that are expected to be revealed. This must be kept in mind, when interpreting corresponding recovery rates (Section 6.3.1).

In order to determine the plausible links practically, the algorithmic implementation (Algorithm 1) of the definition can be used to analyse the full network. (In the example discussed so far (Fig. 6.3a), with plausible lags (l_{min}, l_{max})=(1,3), the algorithm yields the links shown in Fig. 6.3b.) The classification of links between observable nodes as plausible or implausible facilitates the construction of a reference network, which contains all plausible links. Learned networks can then be compared (see previous section) to this network in order to determine their degree of plausibility. The only parameters that must be specified for this approach are the plausible lags (l_{min}, l_{max}). These are determined by the assumptions about how short or long connections via hidden units can be, while resulting in significant correlation between observed units than can be detected. The length of a connecting path is thereby assumed to be proportional to the correlation's time-lag. This is based on the idea that each synapse on the path introduces a further delay of the signal to transmit.

The previous two sections addressed how the severity of the network learning problem can be rated and how to determine the quality of learned networks. The presented ideas take the informativeness of the data into account as well as the effects of observational equivalence and partial observability; thereby, learned networks can be evaluated fairly and their quality can be related to the level of difficulty. Through the quantification of the initial situation and a sensible analysis of the outcome it is possible to investigate the method's true performance for different degrees of severity. How this has been implemented in order to assess the SSS is outlined in the next section.

[6] As an example consider figure 1.4 on page 21: Plausible links would comprise all that are shown in Fig. 1.4a and c. Only one of the links connecting the right-most node is needed in order to explain the data, rendering the other one redundant. However, both are plausible, as any one of them is adequate.

Algorithm 1 Determination of all plausible links (Definition 6) for a partially observable network. The algorithm begins with preliminary determination of reachability sets for each node. These consist of all nodes to which a directed path exist. The main loop iterates over all observable nodes and implements the definition 6 for each of them in two steps: First, potential plausible parents are determined; these are observable nodes, which fulfil condition 1 in the definition. In the second step, condition 2 of the definition is checked and nodes violating the condition are removed as potential plausible parents. The remaining nodes fulfil both conditions of plausibility and are returned in the end.

Input: Full network, observable nodes
Parameters: minimal and maximal plausible lag (l_{min}, l_{max})

for all nodes s_i **do**
  determine the reachability set R_i of all nodes that can be reached from s_i via directed paths; $R_i = \{s_i\}$ if s_i isolated node without any links
end for
for all observable nodes o **do**
  (First determine all nodes ppa_o that fulfil condition 1 of definition 6)
  initialise its plausible parent set $ppa_o := \emptyset$
  for all nodes s_i where $o \in R_i$ **do**
    for all nodes $pa \in R_i$ **do**
      if paths $s_i \to o, s_i \to pa$ with $l_{min} \leq l(s_i \to o) - l(s_i \to pa) \leq l_{max}$ exist **then**
        $ppa_o := ppa_o \cup \{pa\}$
      end if
    end for
  end for
  (Remove all nodes from ppa_o that do not fulfil condition 2 of definition 6)
  for all parents $pa \in ppa_o$ of node o **do**
    if $\forall$ paths $pa \to o = (pa, x_1, \ldots, x_k, o) : ppa_o \cap \{x_1, \ldots, x_k\} \neq \emptyset$ **then**
      $ppa_o := ppa_o - \{pa\}$
    end if
  end for
end for
return all plausible parent sets ppa_o

6.4 Assessment-Framework Implementation

The generic assessment framework (Fig. 6.1) has been implemented by taking the foregoing sections' considerations into account. How the presented ideas have been combined is described here (Fig. 6.4) with details on the neural simulation, as well as the learning procedure of information flow networks.

Each node in the golden network is associated with one neuron whose dynamics are generated by an integrate and fire model (Section A.1.1). The time resolution of the simulation and the data are 1 msec time-bins. Each neuron shows spontaneous activity according to a homogeneous Poisson process [Feller,

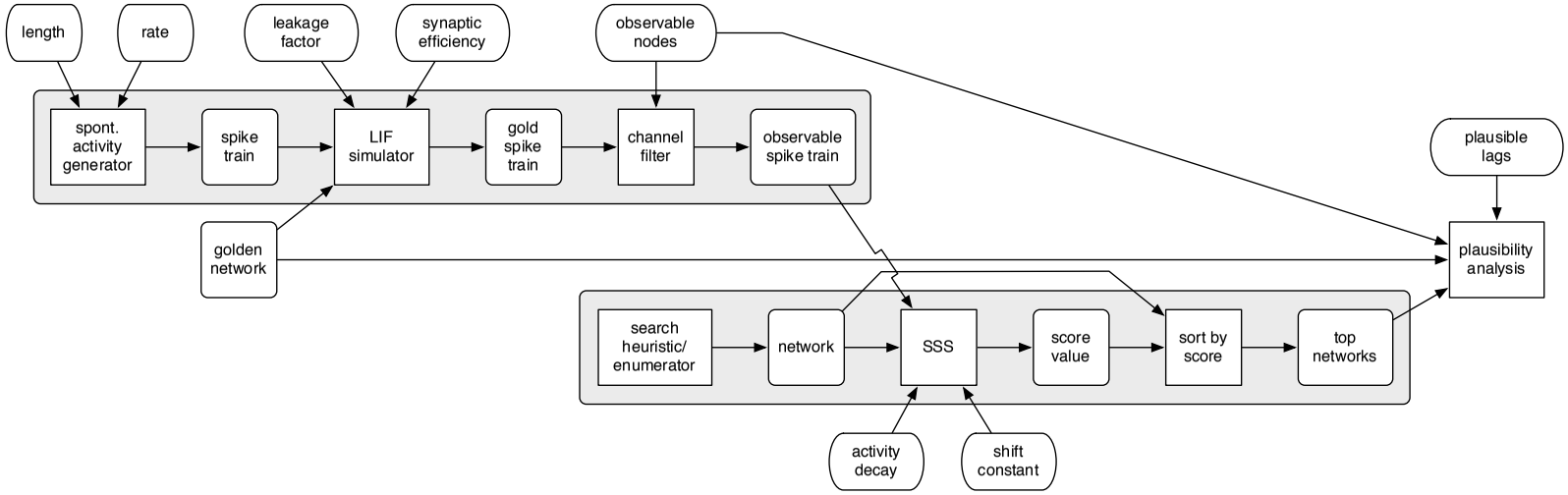


Figure 6.4: Assessment framework used for the SSS (work-flow from left to right). Processing stages represented by rectangles; elements with edges rounded off indicate objects and parameters. A golden network defines excitatory synaptic connections in a neural network, which is simulated by imposing stimulation on model neurons. The simulation generates several spike trains out of which a sub-set of observable channels is selected. These data are used for network inference. Learned networks are finally assessed according to their plausibility.

1950, pp.446]. The rate parameter of the process controls the level of spontaneous spiking activity (*spontaneous activity*), independently for each neuron. Spikes propagate in form of excitatory post-synaptic potentials; signal transmission occurs in the direction of network links with a delay of one time-bin (1 msec) per synapse. Neurons integrate received post-synaptic potentials over time and once a specified number (*synaptic efficiency*) of excitatory inputs is gathered, the neuron fires a spike. The golden network is simulated until enough data (*data length*) is generated.

Before the spike train data is passed on to the network learner it is filtered. In accordance with practical situations the simulated system is not completely observable, but only spike trains from a subset of units (*observable nodes*) can be collected and used for learning. Only these observable spike trains are passed to the network learner, which determines each node's best scoring parent configuration. Actually, it returns a list of all scored configurations, such that configurations of different or similar score can be compared to each other. Composing the parent configuration yields the recovered networks, which are analysed with respect to their plausibility.

Learned network links are classified as either *hit* or *miss* depending on whether they are plausible or not (Section 6.3.2). Statistics of the classification results (e.g. precision or false positive rate, Section 6.3.1) are then determined and related to the impetus of the data used for learning the network (Section 6.2.2).

The method under study can be tested in a variety of conditions by applying the framework for different parameter settings. Relating the quality of the outcome of each setting to the learning complexity, the framework can yield a comprehensive picture of the method's overall performance. Corresponding simulation results are presented later (Chapter 7), after an alternative assessment concept has been discussed.

6.5 An Alternative Assessment Method

A fundamental problem of network inference, which has been mentioned earlier, is that different causal models can generate the same or similar correlated output. In such cases it is impossible to reliably infer the causal network from mere stochastic relations. Unfortunately, correlated activity is the only information available to a network learner in order to infer an information flow network; revealed structures may therefore not match the causal connections. Since such discrepancy is not a failure of the inference method, it must be accounted for when evaluating results. In the presented framework (Fig. 6.4) this is done based on the concept of link plausibility, which allows sensibly tackling the problems of

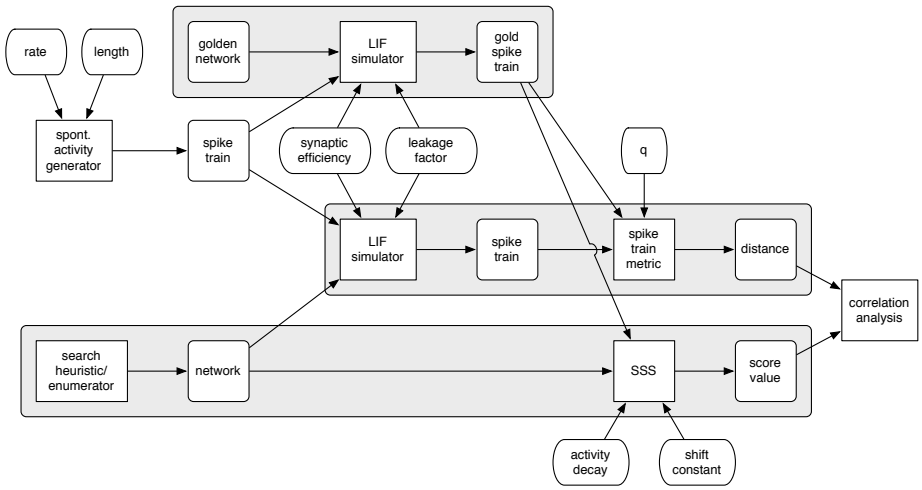


Figure 6.5: Alternative network inference assessment framework based on simulation of learned networks in order to compare their dynamics to those they represent (work-flow from left to right). Processing stages represented by rectangles; elements with edges rounded off indicate objects and parameters. A golden network defines excitatory synaptic connections in a neural network, which is simulated by imposing stimulation on model neurons. Potential networks are simulated and scored in parallel in order to yield their dynamics as a neural network and their score value. The dynamics of the potential network and those of the golden network are compared using a spike train metric. If low distances correlate with high network scores, i.e. networks that replicate the dynamics of the data used for learning, the scoring function imposes a sensible ranking on network structures.

network assessment caused by observational equivalence and partial observability. These issues can also be addressed differently, and one alternative approach is presented next.

Instead of inferring reasonable links from network connectivity, a more pragmaticly seeming idea is to test whether a learned network generates appropriate dynamics (Fig. 6.5): *Does a simulation of the inferred network result in spike trains that are similar to the data it was learned from?* In detail, like the golden network, the learned network can be simulated using neuron models in order to generate spike train data. The spike trains of both networks can then be compared to each other, by quantifying the matching of the dynamics with spike train metrics (Section A.2). If the dynamics of the inferred network are similar to those it was learned from, the structure correctly reflects functional relations in the data and is thus a good network. Finally, the similarity of dynamics is related to the score value of the learned network; high scores would optimally correlate with low distances, i.e. similar dynamics. If this is the case, the score rates networks suitably.

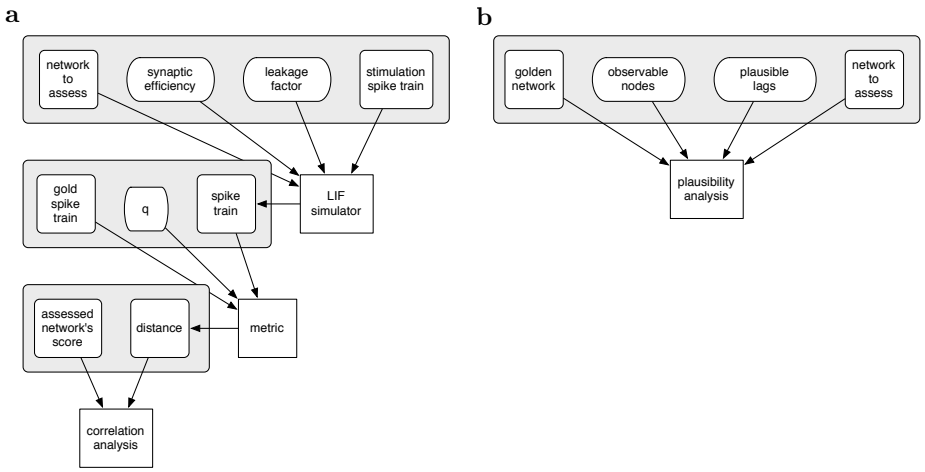


Figure 6.6: Comparison of parameters and processing steps needed in order to assess a network (work-flow from top to bottom). Processing stages represented by rectangles; elements with edges rounded off indicate objects and parameters. **a** Metric approach: The network to assess need to be simulated first in order to calculate the distance between its spike train and that of the golden network. The metric itself and the simulation must be parameterised. Note that additional parameters are needed in order to determine the stimulation activity (not shown). **b** Plausibility concept: Plausible lags must be specified in order to analyse the golden network. The learned network is then compared against the resulting reference network, which can be re-used for comparisons with further learned networks.

Like the plausibility concept, the alternative spike train metric based approach can be used under conditions of partial observability. Therefore channels of the golden network's spike trains would have to be filtered according to their observability (not shown in Fig. 6.5) before passing them on to the spike train metric and the scoring function. Either of the two presented concepts could thus be used to assess the SSS, but the plausibility concept is favoured because it is simpler. In fact, the two approaches differ widely in complexity: The metric based concept involves far more parameters and processing steps than that using link-plausibility (Fig. 6.6). Additionally, there is also a significant difference in the computational costs of the two approaches. With the alternative metric based approach every network is simulated and afterwards the distance between spike trains is calculated. The computational costs for this are determined by the size of the network and, even more critically, the length of the simulated spike train. Increasing data length results in non-critical linear growth of costs for the neural simulation, but computational demands for applying the spike train metric rise over-proportionally. Practical application is thereby limited to

relative short data segments.[7] In contrast, the plausibility analysis of a golden network does not depend on the length of the data but only on network size. The associated costs are comparatively low anyway and additionally, the golden network needs to be evaluated only once and can then serve as a reference for all networks that are learned from its dynamics. For these reasons the SSS has been assessed using the plausibility concept. Corresponding results are presented in the next chapter.

[7]Note that long data-sets should actually be favoured for assessments: Short data segments might match well or bad due to random effects. In contrast, observing (dis)similar dynamics over a long period yields higher confidence that artefacts are smoothed out; the measured distance is then a reflection to what degree the learned network can replicate features of the data is has been inferred from.

Chapter 7

Application of the Assessment Framework

The previous chapter motivated and described an assessment framework for information flow network inference techniques. Results from applying the framework to the SSS are presented in this chapter. Details on the chosen parameters are given first in order to complete the description of the set-up. Next, the outcomes of different parameter series are presented and thereafter, these will be reviewed critically. Based on the outcome of the discussion, further evaluation series are performed in order to address identified issues.

7.1 Set-up and Parameters

The first step in the work-flow of the assessment framework is to specify a network, which defines the number of neural entities and their interaction (Fig. 6.4 on page 100). For the following simulation series, a feed-forward type structure has been chosen whose nodes corresponds to one neuron each (Fig. 7.1a). For the neural simulation, the dynamics of each neuron are generated according to an integrate and fire model (Section A.1.1). As neurons according to this model do not show intrinsic spiking, baseline activity in the network must be induced by stimulation. This is done with Poisson processes, which are used to generate spontaneous spikes for each unit independently. The level of baseline activity is controlled by a constant parameter $\lambda = 10^{-1}, 15^{-1}, 25^{-1}, 30^{-1}, 40^{-1}$, or 50^{-1}; the rate parameter λ is negatively correlated with the level of *spontaneous spiking activity*. Any spike propagates according to network connectivity to directly linked units, where it results in an excitatory potential. Such transmission occurs with a delay of one time-bin (1 msec). Each neuron inte-

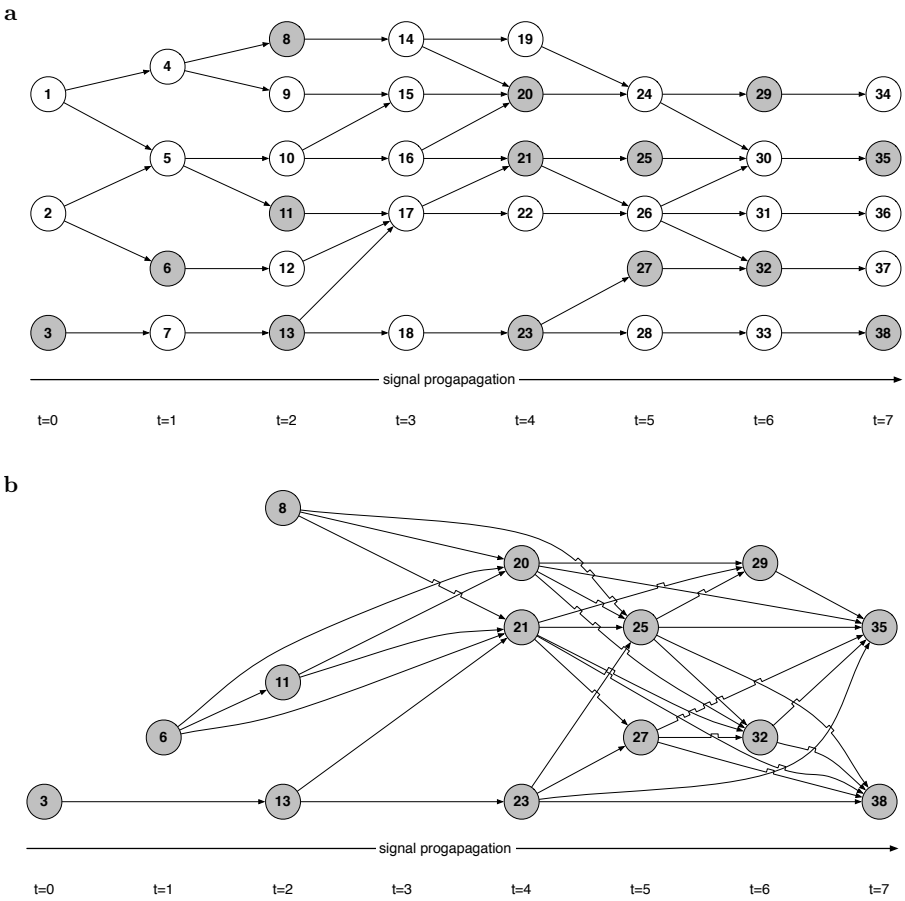


Figure 7.1: **a** Simulated feed-forward network (38 nodes, 47 links) with observable units (14 nodes ≈ 36.8%) shaded in grey and hidden units unshaded. **b** Observable nodes from network shown in (a) with all 34 plausible links (≈ 18.7% out of 182 possible links) for plausible lags $l_{min} = 1$ and $l_{max} = 3$. Examples of plausible links can be those which exist in the full network (e.g. $\underline{21} \rightarrow \underline{25}$, $\underline{23} \rightarrow \underline{27}$, and $\underline{27} \rightarrow \underline{32}$) or certain links for which a directed path from the link's starting node to its end node exists (e.g. $\underline{3} \rightarrow \textit{(7)} \rightarrow \underline{13}$, $\underline{13} \rightarrow \textit{(17)} \rightarrow \underline{21}$, and $\underline{23} \rightarrow \textit{(28)} \rightarrow \textit{(33)} \rightarrow \underline{38}$).[a] Links that contradict the full network by connecting nodes in the opposite direction of information flow in the feed-forward network (e.g. $\underline{13} \rightarrow \underline{3}$, $\underline{21} \rightarrow \underline{3}$, and $\underline{35} \rightarrow \underline{3}$) are implausible, for example. Also, a common trigger can cause co-ordinated firing between nodes, which are connected via the triggering node only (e.g. plausible link $\underline{6} \rightarrow \underline{11}$ due to common trigger *(2)*: $\underline{6} \leftarrow \textit{(2)} \rightarrow \textit{(5)} \rightarrow \underline{11}$). (For full details on the plausibility concept please see section 6.3.2.)

[a] Start- and end-nodes underlined, nodes on path in full network italic, hidden nodes in brackets.

grates synaptic inputs over time that are received from its parents (temporal summation). A parameter for *synaptic efficiency* determines whether $2, 3, 4$, or 5 excitatory inputs are necessary in order to evoke a spike in the receiving neuron. The parameters controlling the spontaneous activity and synaptic efficiency are equal for all neurons. As discussed in detail later (Section 7.3.4), neurons have been simulated without a leakage component, such that synaptic inputs would be integrated over infinite time. However, this does not happen in practice, because spontaneous spiking resets the membrane potential (just as activity evoked by synaptic inputs does). Since each simulated neuron exhibits spontaneous activity, its time-window for summing synaptic inputs is effectively limited.

In accordance with practical situations it is assumed that the simulated system is not completely observable, but that only spike trains from a subset of units (*observable nodes*) can be collected and used for learning networks (grey shaded nodes in Fig. 7.1a). In order to assess inferred networks, the effects of partial observability are taken into account by applying the plausibility concept (Section 6.3.2). The algorithmic implementation (Algorithm 1) of the plausibility definition 6 was applied to the full network (Fig. 7.1a) with plausible lags $(l_{min}, l_{max}) = (1, 3)$ in order to determine all plausible links (Fig. 7.1b). The minimal plausible lag l_{min} has been chosen according to the smallest lag observable nodes can show non-spurious correlation and l_{max} such that a modest percentage of links will be plausible.[1]

Two different series were run in which both the parameters of the neural simulation and the score were altered systematically in order to assess their influence on the quality of the recovered networks (Fig. 7.2). Each combination of parameters is simulated 10 times for different *data lengths* ($5, 10$, and 30 seconds, $1, 5$, and 10 minutes), whereby the full network, observable nodes, and plausible links are left unaltered. Simulating the full network yields spike train data from which observable channels are then used to determine each node's best scoring parent configuration with up to 3 parents. (Unless noted otherwise, the parameters of the SSS were $d = 3^{-1}$ and $\Delta t = 1$. The LAT hat been determined from configurations with 3 parents.) Composing parent configuration yields the recovered network, which is analysed with respect to its plausibility: Links of learned networks are classified as either *hit* or *miss* (depending on whether they are plausible or not). The *recovery rate* and *precision* were determined together with the corresponding P-value in order to account for the fact that the probability of finding plausible links by chance depends on their percentage (Section 6.3.1).

[1] For $l_{min} = 1$ there exist 12 ($\approx 6.6\%$ out of 182 possible links, $l_{max} = 1$), 27 ($\approx 14.8\%$, $l_{max} = 2$), 34 ($\approx 18.7\%$, $l_{max} = 3$), 37 ($\approx 20.0\%$, $l_{max} = 4$) plausible links.

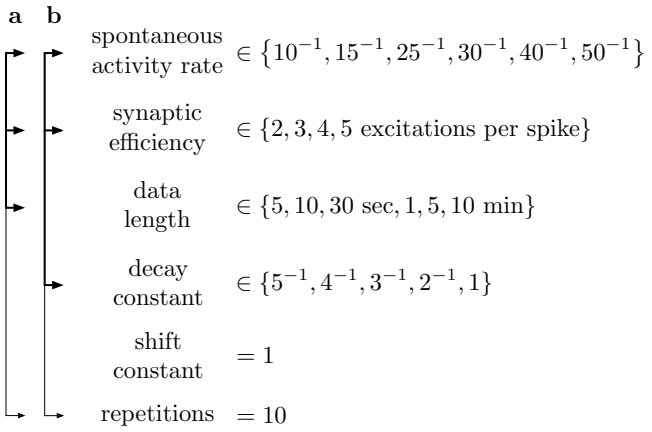


Figure 7.2: Parameter overview for the assessment framework. Two series **a** and **b** were run in which the marked parameters have been combined in all possible ways. For each of the combinations the neural simulation and the network learning step has been repeated 10 times in order to average out random effects.

Note that parameters of the neural simulation (spontaneous activity level, synaptic efficiency) do not have any physical correspondence in this sparsely connected network; their influence on network inference is thus not shown individually as their combined effect is fully reflected in the simulated spike train. Instead, the simulation output, the spike train, is characterised by the amount of information it contains about the structure of the network by calculating its *impetus* (Section 6.2.2). Determining the impetus for every data set that is used for network learning and relating it to the corresponding success rates, recovery rate and precision, shows its relationship to the performance of the SSS. The results of this investigation are presented in the next section.

7.2 Simulation Results

In the first series (Fig. 7.2a), where the parameters of the score are fixed ($d = 3^{-1}, \Delta t = 1$), an expected relation between quality and amount of data (impetus, length of spike train) and the grade (recovery rate, precision) of networks learned from the data can be observed (Fig. 7.3): In order to learn networks of a certain grade, a lower impetus can be compensated by longer recordings; longer recordings at a fixed impetus generally improve the quality of the learned networks; and for a particular data length, recovered networks are generally better for higher impetuses, i.e. more informative data.

For any learned network the recovery rate does not exceed 35% (Fig. 7.3a),

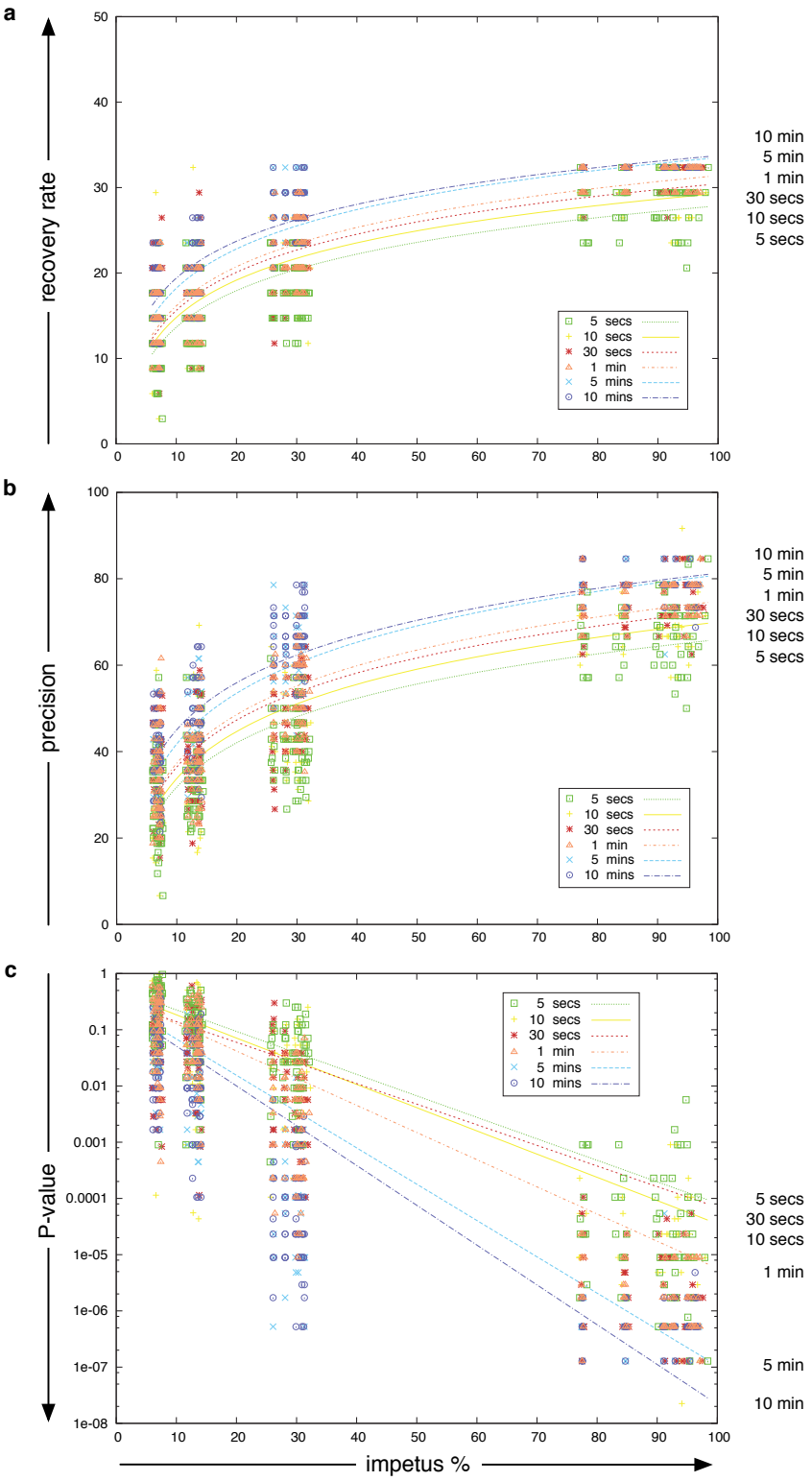


Figure 7.3: Legend in separate box on page 110.

Fig. 7.3 legend: Quality of recovered networks depending on impetus and recording length ($d = 3^{-1}$, $\Delta t = 1$ fixed). Data points correspond to one learned network each (learned from a separate simulation); trend lines fitted to points of each data-length. **a** Recovery rate of all plausible links for different impetuses and different lengths of data. Higher impetuses and longer data sets result in better recovery rates. Longer data sets can compensate for lower impetuses. **b** Precision of recovered links for different impetuses and different lengths of data. High impetuses and longer recordings improve precision. Longer data sets can compensate for lower impetuses. **c** P-values of precision shown in (b) on logarithmic scale. Networks with a higher precision are less likely to be revealed by chance. Lower P-values thus indicate better performance.

but recall that not all of the links that are classified as plausible are expected to be recovered (Section 6.3.2, page 98 especially). This is because, generally, some redundancy among them exists; for example, in figure 7.1b links $20 \rightarrow 29$, $21 \rightarrow 29$ are both plausible, but given the joint excitation of nodes 20 and 21 by unit 16, they might be similar enough, such that one of them is sufficient to explain spiking of unit 29. The set of plausible links is therefore a superset of those that are expected to be revealed by network inference. Out of this superset, the SSS predominantly recovers links that connect nodes and their closest plausible parent, but not more distant ones. (This can be explicitly seen in Fig. 7.6, which will be discussed later.) Additional plausible links that do not sufficiently increase the explanatory quality of the network are omitted due to the SSS's preference for simpler networks.

In contrast, precision would optimally reach 100% such that all recovered links are indeed plausible. Fig. 7.3b shows the relationship between impetus and precision, which are positively correlated. The corresponding P-values show that high precision is extremely unlike to be reached by chance (Fig. 7.3c). At lower impetuses, the percentage of learned links that are implausible increases and causes precision to drop down. This is to be expected, because the lower the impetus, the more similar spike trains are to the spike patterns from the random processes that induce the baseline activity into the neurons. These spikes do not convey any connectivity information about the network, such that links learned from these data are more likely to correspond to spurious correlation than to functional connectivity between neurons.

In the second series (Fig. 7.2b), the framework is used to investigate the robustness of learning performance with respect to different parameter settings of the SSS. At a fixed data length (30 seconds), different decay constants are used to reveal networks. The corresponding results are depicted in Fig. 7.4. Exclud-

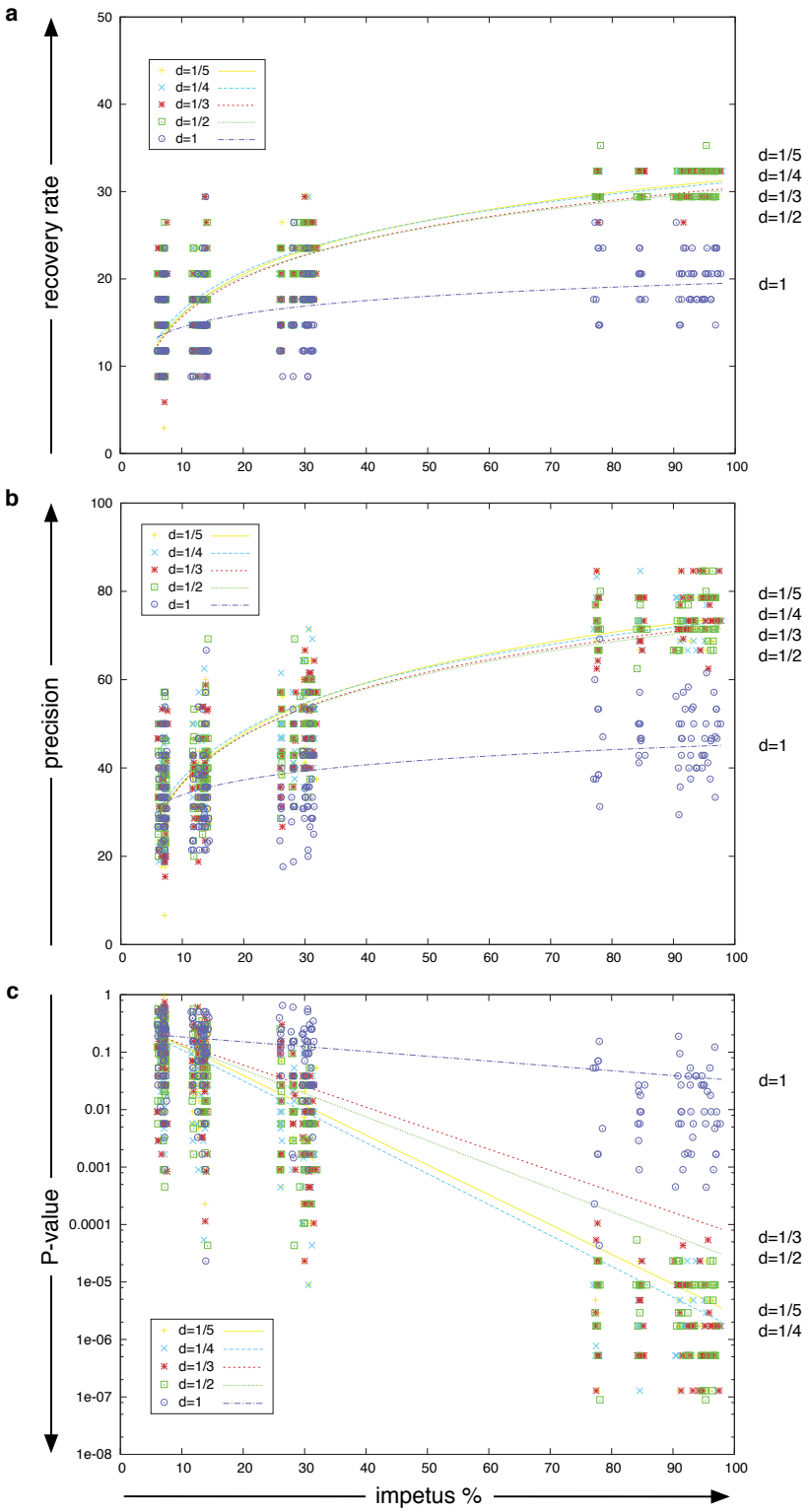


Figure 7.4: Legend in separate box on page 112.

Fig. 7.4 legend: Quality of recovered networks depending on impetus and decay constant (recording length 30 seconds, $\Delta t = 1$ fixed). Data points correspond to one learned network each (learned from a separate simulation); trend lines fitted to points of each choice of decay constant. **a** Recovery rate of plausible links for different impetuses and choices of decay constant d. Decay constants $d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}\}$ result in similar good recovery rate. Differently, setting the decay constant $d = 1$ results in significantly worse recovery rates due to correlation over multiple time-lags, which is not considered in this case. **b** Corresponding precision for impetuses and decay constants shown in (a). As in (a) performance is similar for decay constant $d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}\}$, but worse for $d = 1$ (for the same reason). **c** P-values of precision shown in (b) on logarithmic scale.

ing the extreme case $d = 1$, all settings lead to similarly good performance over a range of impetuses.[2] Setting $d = 1$ results in a significantly worse performance, as the resulting lag-window $[1, 1]_{\mathbb{Z}}$ causes the SSS to score correlation of lag 1 only. Links with larger time-lags (e.g. $\underline{3} \to \textit{(7)} \to \underline{13}$, $\underline{13} \to \textit{(18)} \to \underline{32}$, and $\underline{23} \to \textit{(28)} \to \textit{(33)} \to \underline{38}$) can thus only be found through spurious correlation of lag 1. This limitation accounts for the comparatively low performance of this setting.

The previous two parameter series explored the performance of the SSS over wide parameter ranges. The results show the performance of the score for many data-sets. In realistic situations data collection is often limited, such that only few recordings under similar conditions of the studied system can be made. It is thus important to investigate how the SSS can be applied on a small number of data sets. This is done next, where the practical limit of data acquisition is mimicked by simulating 10 data sets of 30 seconds length each; for all of them the same simulation parameters were used, such that all spike trains exhibit about the same 30% impetus (Fig. 7.5).[3] For each of the 10 data sets networks are learned (one per data-set for each decay constant $d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}, 1\}$, $\Delta t = 1$) and averaged for each decay constant. Observations made earlier (Fig. 7.4) are found to be confirmed: On average, networks learned with decay constant $d = 1$ contain fewer plausible links than

[2]Note that results from such parameter series of the score could be displayed as an ROC-curve (Section 6.3.1), but there are two reasons for not using this kind of result display here: (1) The curve would look misleadingly bad, because, unlike the for common use of this plot, the recovery rate is not expected to reach 100%, such that points could fall below the diagonal, which generally corresponds to chance level. (2) The high similarity in performance for all but one of the parameters (Fig. 7.4) would result in basically two points in the ROC-plot, which would be less informative than the chosen style to present results.

[3]The data sets sum up to a total recording time of 5 minutes. The impetus was found to be between 29.7% and 30.0%. For an impetus of 30% the simulation output consists of 3 evoked spikes per 10 uncorrelated stimulation spikes (on average), i.e. only 3 out of 10 spikes are informative about network structure (Section 6.2.2).

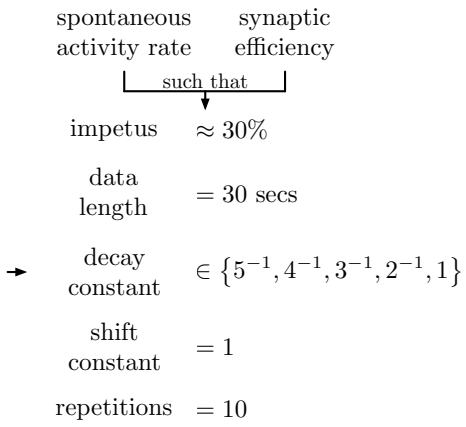


Figure 7.5: Parameter overview for the assessment framework. Series with constant parameters of neural simulation and varying decay constant. The neural simulation and the network learning step have been repeated 10 times in order to mimic data sampling under similar conditions.

those learned with smaller decay constants (Fig. 7.6a vs. 7.6b-e). Independent of the decay constant, implausible links show lower percentages of recovery than plausible ones (Fig. 7.6a-e). Furthermore, most plausible links are recovered consistently while implausible ones vary for different decay constants (Fig. 7.6a-e). In order to account for links learned from spurious correlations, all learned networks were averaged ($d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}, 1\}$); this reduces the frequency of implausible links significantly while preserving plausible ones with high percentages (Fig. 7.6f).

Finally, a parameter series varying the shift constant Δt would be desirable. For reasons given later (in the following review section 7.3), such series needs additional care when run with the chosen network (Fig. 7.1a). Corresponding results are thus shown after related issues have been discussed.

The results presented in the foregoing section are promising and show that the SSS can successfully reveal functional connectivity from simulated spike train data. The parameter series give insights that are valuable for the SSS's application to real data; however, before drawing conclusions, the complexity of the used framework necessitates thorough questioning of its outcome. This includes a detailed review of the set-up used to generate spike train data, features of the simulated network, as well as the network learning procedure and the evaluation of its results. The following section addresses these issues and tries to identify potential weaknesses that could affect the analysis of real data.

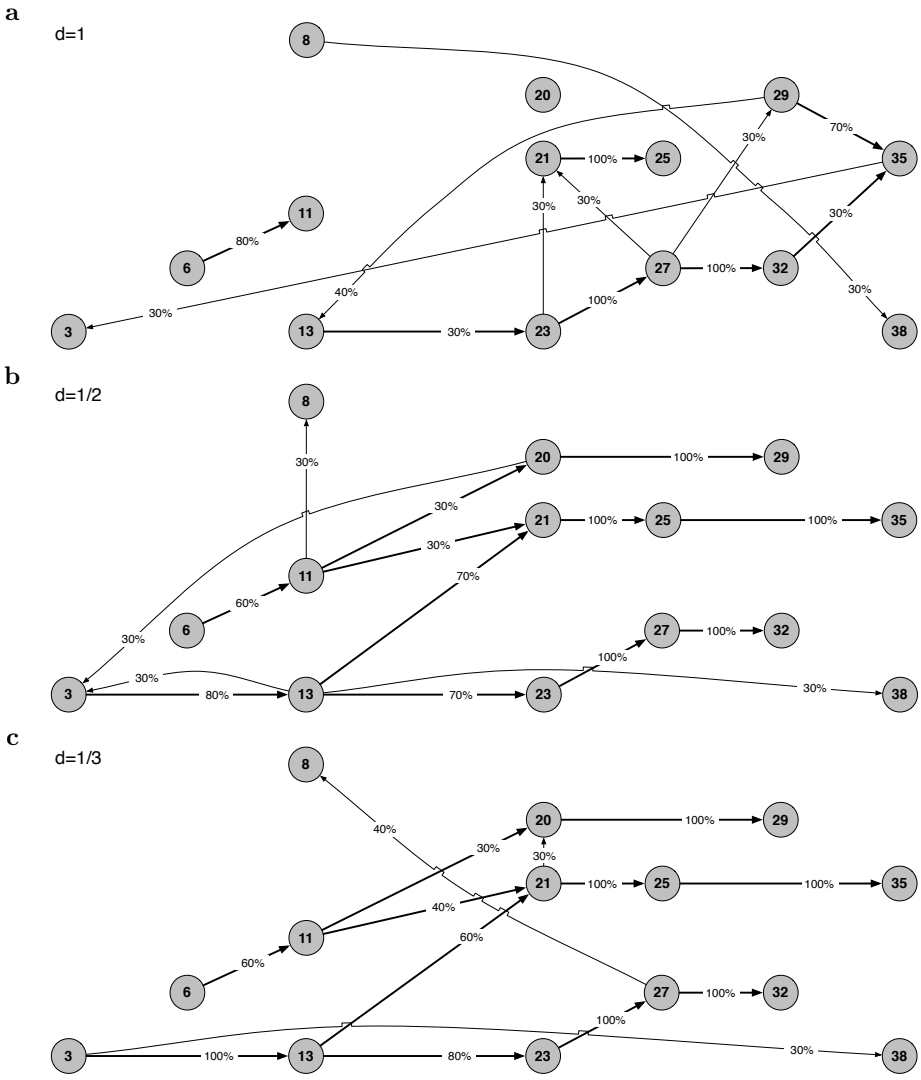


Figure 7.6: **(part 1)** Average of networks learned for data of fixed length (30 seconds) and similar impetuses ($\approx 30\%$) using different decay constants ($d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}, 1\}$). Links that can be found in at least 30% of learned networks (for each choice of d) shown with percental frequencies next to link. Plausible links are marked bold (Fig. 7.1b). **a** ($d = 1$) Except for one link ($\underline{13} \rightarrow$ *(18)* $\rightarrow \underline{23}$), only plausible links with lag 1 are recovered. The decay constant is not small enough to capture dependence over larger time-lags. **b** ($d = 2^{-1}$) Good recovery of plausible links with lags 1 and 2. Link $\underline{13} \rightarrow$ *(18)* $\rightarrow \underline{23}$ recovered with higher percentage than in (a). **c** ($d = 3^{-1}$) Similar recovery of plausible links as in (b), but different implausible links (recovered in $\geq 30\%$ of networks) due to wider lag-window.

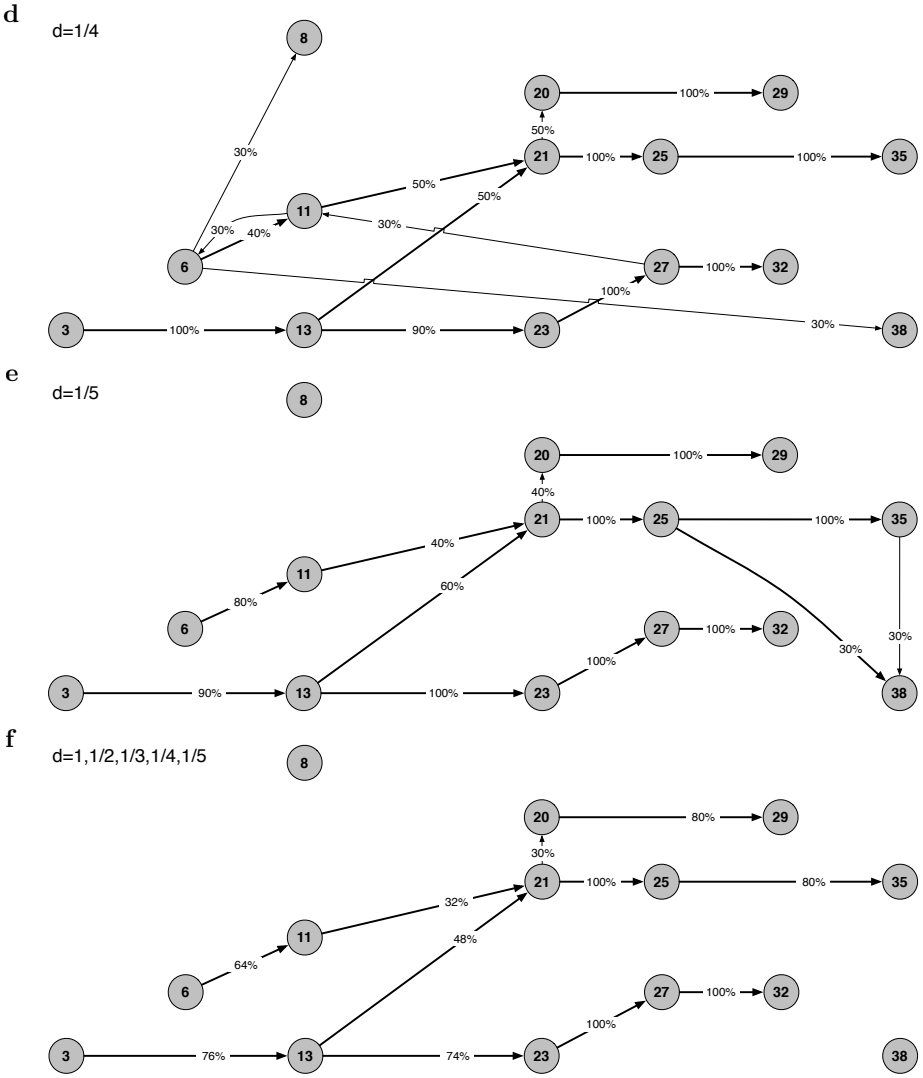


Figure 7.6: **(part 2) d** ($d = 4^{-1}$) Similar recovery of plausible links as in (b) and (c), but more implausible links (recovered in $\geq$ 30% of networks) than in (b) and (c). **e** ($d = 5^{-1}$) Plausible links recovered as in (b-d), plus link $\underline{25} \rightarrow \underline{38}$, but different implausible links ($\geq$ 30%) than in (a-d). The differences among implausible links in (a-e) indicate that they are learned from spurious correlation over different time-lags. **f** ($d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}, 1\}$) Average over all networks (a-e) shows good recovery of plausible links. Several implausible links learned from spurious relationships average out (frequency below 30%).

7.3 A Critical Result Review

The SSS is a new analysis tool for neural spike train data, which needs to be thoroughly tested before applying it to real data. This chapter complements the theoretical examination of the score (Chapter 4) by applying the assessment framework presented earlier (Chapter 6). Unfortunately, it is not possible to simulate a fully realistic environment, such that testing the score is limited to few relatively simplistic scenarios. Results gained under these idealised circumstances may not generalise to real data and that is why it is important to understand what exactly the unrealistic components of the framework are. Identified issues must be investigated with respect to their influence on results, which might require further simulation experiments — slightly differing from the ones before. Careful dissection can then indicate how to interpret results gained with the SSS on real data.

In order to understand the presented results and their practical relevance, all relevant steps in the assessment procedure and associated parameter choices are analysed. This is done by discussing the components of the assessment framework in separate sections, which are ordered according to its work-flow (Fig. 6.4, p. 100). Each of the sections starts with a brief reminder on the actual implementation of particular points to discuss. It is then explained why and to what extent elements of the framework can cause potential problems and whether they can be addressed or not. Further simulation results will be shown where appropriate.

7.3.1 Network Structure and Observability

Starting point of the assessment framework is the network structure that defines the number of neurons and their interactions. Alongside with the network some of its nodes are chosen to be observable, i.e. data can only be collected from these units. When these decisions are made, the size, topology, and observability of the artificial neural circuit are defined. The neural simulation is influenced by each of these three factors, which are therefore discussed in the following.

(Overall complexity) Compared to the enormous number of neurons in the brain (e.g. 10^{11} in humans [Kandel et al., 2000, p.19]) and their manifold of dendritic wirings, it is obvious that the simulated network's complexity is probably the most unrealistic component in the set-up. It only consists of a small number of neurons, which are sparsely interconnected compared to $10,000$–$150,000$ synaptic contacts a real neuron makes [Kandel et al., 2000, p.25]. Further, although only less than half of the simulated neurons have been chosen to be observable, this level of observability is chosen too high, compared to techno-

logical limits on data-collection, which do not facilitate such high coverage in monitoring. The constructed network and data-sampling capabilities thus do not correspond to a realistic neural circuit in three points: size, connectivity, and observability of the simulated system. The reason for using a comparatively small network with sparse connectivity is that even rebuilding only small parts of the nervous system in a nearly realistic manner is accompanied by two problematic demands: A severe amount of data to build and parameterise such model, and extremely powerful computational resources are needed for its simulation (e.g. [Markram, 2006, Bhalla, 2008]). Existing complex simulators are not freely accessible (yet) and setting such up is clearly beyond the scope of this PhD project. Investigations of the SSS therefore have to be done with highly simplified neural networks. Whether resulting observations also hold in more complex situations will therefore remain subject to speculations.

(Impact of Topology on Dynamics) Studies have shown that depending on how neurons are interconnected with each other different pattern of activity can emerge [Sporns et al., 2000, Sporns and Tononi, 2002, Galan, 2008, Bullmore and Sporns, 2009]. The modelled neural circuitry used here is not complex enough to observe complex patterns, but network topology has still an effect on the activity of modelled neurons and especially on the impetus of their data. Before this is explained in detail, the practical implication of the relationship between network structure and impetus is mentioned: The SSS's performance is positively correlated with the impetus and the length data; analysis of real recordings might thus depend on the origin of the data. Data from structures with feed-forward connections (like the visual system [Ganis and Kosslyn, 2007]) might show a lower impetus than those from circuits with significant recurrent connections (e.g. hippocampus [Siegel and Sapru, 2007, pp.447]); different amounts of data may thus be required for successful network recovery. But how can network structure influence the impetus in the data? This is explained next for the simulated network, where links represent excitatory synaptic connections.

The impetus of a node is affected by two factors: the number of converging connections in that node and its relative position in the network. The first point is obvious by noticing that the number of parents of a node correlates with the number of excitatory inputs this node can receive simultaneously. More spikes can thus be evoked in nodes with many parents, and it can therefore have a higher impetus than a node with fewer incoming connections. For one part the impetus of a node is thus influenced by local network properties like its parents, but the position of that node within the network (a rather global property) can also affect its impetus. This can be seen by considering the feed-forward

network used in the simulations (Fig. 7.1), where activity propagating through the network has a tendency to increase. In more detail, neurons on the input-side of the network (Fig. 7.1a, left) do not receive any or just few incoming connections. They do thus not receive significant excitatory inputs, but only the spontaneous baseline stimulation. Differently, nodes in the centre of the network (Fig. 7.1a, middle) exhibit more incoming connections through which excitatory potentials are received from neurons nearer to the input side. These centre-neurons are thus more likely to show spiking activity because, additional to stimulation spikes, they also spike due to excitations from neurons further upstream in the network. The activity and impetus of the centre neurons is therefore higher than that of neurons at the input side of the network. Moving further downstream towards the output side of the network (Fig. 7.1a, right), neurons receive excitatory inputs at even higher rates than the centre neurons before them. In this set-up, nodes on the output side of the network are thus likely to exhibit the highest activity and impetus, simply because activity slowly accumulates over neuron-layers. Whether the inclining impetus throughout the structure actually had an effect in the simulated network cannot be reliably assessed: Since most observable nodes are located in the output half of the network, only few plausible links exist within the other half. Although some of these are reliably inferred (Fig. 7.6), their small number prevents a significant comparison in order to show whether these are less likely to be revealed than those towards the output side of the network. Investigating this aspect would require varying observability of the network to ensure equal distribution of plausible links; however, this is subject to future work. Here, observable nodes are left unchanged in order to facilitate comparisons between different simulations shown (later).

7.3.2 Data Length

The complexity of the network structure is fundamental to the neural dynamics that are generated from it. The parameter that controls the length of that simulated data is another important factor, which determines the severity of the learning problem. In the presented series, the amount of data available for the analysis with the SSS has been varied in the range of seconds to minutes. Data lengths in the simulations thus correspond to realistic data gathering lengths with electrophysiological recordings, which can be made on the order of seconds to hours (or even longer with chronic implants), depending on the experimental set-up.

(Limit Behaviour) From a theoretical standpoint it is interesting to see how unrealistically long data-lengths affect the performance of the method. This investigation can show whether, in practice, additional effort involved in longer recordings would pay off in terms of improved results from network inference. A few experiments have thus been performed, where extremely long simulations of 30 and 60 minutes data lengths were analysed using the SSS. The results show further increase in performance for these data-sets. To investigate the relationship between the volume of data and the quality of recovered networks both entities have been plotted against each other (Fig. 7.7). The length of the data correlates with the quality of networks in a (sub-)logarithmic manner, such that any further quality improvement requires a rapidly growing amount of data. The analysis of extremely long practical recordings could benefit the same way, but since increasing the quality of learned networks becomes less and less economic, a sensible trade-off between effort to collect the data and the potential benefit must be found.

7.3.3 Spontaneous Activity

The integrate and fire neurons in the neural simulation do not posses any intrinsic spontaneous activity. They do not generate any spikes unless stimulated externally. Therefore, a *spontaneous activity generator* produces spike trains, which are composed of uncorrelated spikes. These random spike trains are used to induce spiking activity into the simulation: Each neuron receives one of those spike trains and exhibits spiking activity on every fed-in spike. (These spikes propagate according to network connectivity and can evoke further spikes in receiving neurons.) The stimulation activity is thus mirrored in the output spike train of the simulation.

(Memoryless Random Process) The spontaneous spikes are generated by a homogeneous Poisson process, i.e. the rate parameter of the process stays unchanged throughout the simulation. There are thus no parameterised variations in activity; however, variations naturally arise in Poisson processes whose exponentially distributed inter-spike interval lengths are independent of each other [Feller, 1950, pp.446]. This renders the process *memoryless*, such that spikes can occur arbitrarily close to each other; even in directly succeeding time bins. The rate of spiking activity is thus unlimited. The Poisson process is thereby a potential violator of a realistic constraint, since real neurons posses an upper limit on spiking rate, induced by their refractory period [Kandel et al., 2000, p.157]. However, it turns out that for the chosen rates of spontaneous activity the process is extremely unlikely to generate bursting behaviour: The

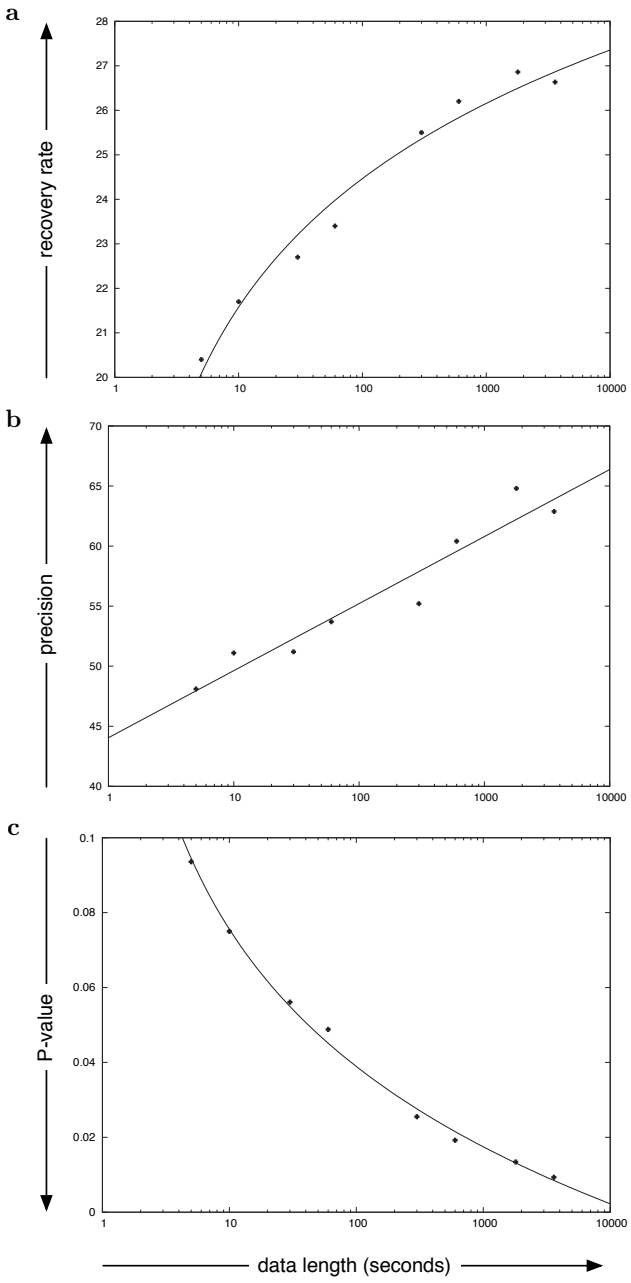


Figure 7.7: Relationship between data-length and quality of recovered networks. The curves fitted to the data in figure 7.3 were evaluated at an impetus of 30% for different data lengths, which are shown on a logarithmic scale on the horizontal axis. Logarithmic functions fitted for illustrative purposes. **a** Recovery rate dependent on data-length in double-logarithmic manner, such that any further improvement requires rapidly growing amounts of data. **b** Precision of recovered networks shows logarithmic increase, i.e. it improves faster than the recovery rate. **c** P-values of the precision show double-logarithmic dependence on data-length.

probability that k spikes occur within a time-interval of length t is [Feller, 1950, p.447]

$$p_{k,t}(\lambda) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \ . \tag{7.1}$$

For the rate parameters used in the simulations ($\lambda = 10^{-1}, 15^{-1}, 25^{-1}, 30^{-1}, 40^{-1}, 50^{-1}$) the probability of two spikes within a time interval of length 2 is low even for the highest rate parameter ($p_{2,2} \approx 0.016, 0.008, 0.003, 0.002, 0.001, 0.001$). Observing larger numbers of succeeding spikes is less likely, e.g. $p_{3,3} \in [0.000033, 0.003334]$, such that long, unrealistically high frequency bursts of spikes are very unlikely to occur. Additionally, the calculations ignores that spikes are binned with 1 msec resolution; two spikes that occur within one time-bin appear as only one event in the spike train. Thus, in order to observe a burst of k spikes in the data all spikes need to occur within k different succeeding time-bins. The corresponding probability for k succeeding spikes in the spike train is thus

$$\bar{p}_k(\lambda) = \left[1 - p_{0,1}(\lambda)\right]^k = \left[1 - e^{-\lambda}\right]^k \ . \tag{7.2}$$

The probability of a burst in the data is thus even less than the probabilities calculated above: We find $\bar{p}_2 \in [0.00039, 0.00906]$ and $\bar{p}_3 \in [0.000007, 0.000862]$. In conclusion, unrealistic bursting effects due to the lack of memory of the Poisson process can be excluded having caused significant artefacts, because of their expectedly low rate of occurrence. (An actual test of the data will be presented in another context in the next section 7.3.4.)

(Homogenous Activity Rates) Another aspect concerning the spontaneous activity is that the spikes of each neuron are generated independently, but at the same rate for all channels. Such homogeneous baseline-activity can be considered unrealistic, as neurons have been observed to exhibit a whole range of spontaneous activity even within a single section of the brain (e.g. [Yamaoka and Hagino, 1974, Abeles, 1982, Legendy and Salcman, 1985, Tsodyks et al., 1999]). Such diversity can be easily replicated by using different rate parameters for each neuron. But together with the remaining parameters of interest (Fig. 7.2) the number of combinations to explore is extremely large, which causes enormous computational demands when evaluating all of them. It is questionable whether such effort would be worthwhile, due to the influence of network connectivity on the activity of individual units (as discussed earlier). Using more than one rate parameter and investigating the resulting effects requires a network topology with minimal influence on activity rates. Such network would be limited in diversity of connectivity patterns, since only very regular structures could guarantee that any observed effects are indeed caused by the different

rates of activity. In essence, a homogenous baseline activity becomes diversified by network topology and tight control of this diversification would be needed in order to investigate influences of individual neurons. Within the scope of the assessment framework, it therefore seems reasonable to use a single parameter to control the spontaneous activity of all neurons.

7.3.4 Neural Simulation

The basis of the neural simulation is a leaky integrate and fire model (Section A.1.1). In the simulations leakage was set to zero, which is a crude approximation compared to real physiology where leaky membrane currents are commonly observed [Kandel et al., 2000, Chapter 8]. Similarly, synapses have been been modelled to be excitatory exclusively, which does not match realistic diversity: Synapses can also be inhibitory, or non-chemical, like electrical synapses, which can tightly couple two cells [Kandel et al., 2000, Chapter 10].

(Lack of Leakage Current) The membrane of a neuron is not a perfect isolator such that ionic currents occur whenever the membrane potential deviates from its resting potential [Kandel et al., 2000, Chapter 8]. The neuron model can account for such leakage currents either by a constant leakage or, more sophisticated, a dynamic, voltage dependent function. However, this has not been done, because this extension of the model is associated with at least one additional parameter. This extra variable causes two problems: One of them is that it increases the number of parameter combinations to explore; unless another parameter is left un-varied throughout the series, the computation time for its evaluation increases significantly. The second problem is more subtle: Varying leakage can affect the spike output of the modelled cell, which in turn can impact on the impetus of the data. The impetus must not be too low since sensible inference of functional connections becomes impossible otherwise (Section 6.2.2). Thus, very careful adjustments of the neural simulation are needed in order to guarantee a reasonable impetus in the spike trains. In the simulations shown this has been ensured by tuning the parameters for spontaneous activity and synaptic efficiency with respect to each other. Including a leakage component in the model would require calibrating the corresponding parameter(s) in combination with the before-mentioned ones. Naturally, an increase in number of parameters that are dependent on each other makes their adjustment more complicated and should thus be avoided if unnecessary. In order to ensure that omitting a leakage component in the model has no unforeseen effects, piloting runs were performed in which model neurons were leaky (constant leakage current). Networks learned from the resulting spike trains were not substan-

tially changed (not shown). Within the explored range of parameters leakage currents thus did not seem to have an impact on network learning performance; the impetus of the data presented itself as the crucial factor, no matter which parameter settings of the simulation lead to the corresponding data. Based on these experiments and in order to reduce computation time, the parameter series have been chosen to contain key parameters only and a leakage component has been omitted.

(Lack of Refractory Period) The neural simulation suffers from another unrealistic inaccuracy, which is the neurons' lack of a refractory period [Kandel et al., 2000, p.157]. Modelled neurons can spike at any time and do not possess the typically observed time-window following an action potential in which synaptic inputs cannot evoke any further spike. Whether this simplification results in a noticeable effect or not depends on the rate at which excitatory synaptic inputs occur. If this rate is sufficiently low, excitatory inputs do not accumulate fast enough to evoke directly succeeding spikes. In such case, there is no difference between a model including a refractory period and one without this property. Differently, if the dynamics in the neural network are very high, lacking a refractory period can lead to unrealistically high firing rates of individual neurons due to their persistent activity induced by excitatory inputs. In the parameter series overall activity has been fairly low, such that such an overflow of activity it quite unlikely to appear. The model's lack of refractory period should thus not have caused any significant artefacts. But before rushing through this issue too quickly, there is another reason why the data should be inspected with respect to spikes occurring extremely close to each other: The spontaneous activity of each neuron occurs on top of its synaptically evoked spikes. Spiking can thus be caused by two independent mechanisms each of which has a fairly low probability of generating artefacts; however, their combined effect might be significant. But an inspection of the data shows that dynamics of the network are within reasonable bounds (Fig. 7.8): Plotting the inter-spike interval (ISI) distributions shows that activity for low impetuses basically corresponds to the spontaneous spiking activity; for these data close spikes are unlikely, as theoretically determined (Section 7.3.3). For higher impetuses, however, the influence of evokes spikes becomes predominant and higher firing rates are observed. For some neurons, the increased activity is also accompanied with a changed ISI distribution. These change from exponential type to Poissonian, such that it becomes more likely that spikes occur close to each other; however, these characteristics match observations from experimental data, for example in the visual cortex [Bair et al., 1994]. In conclusion, despite lacking a refractory period the simulation seems to generate reasonable spike trains.

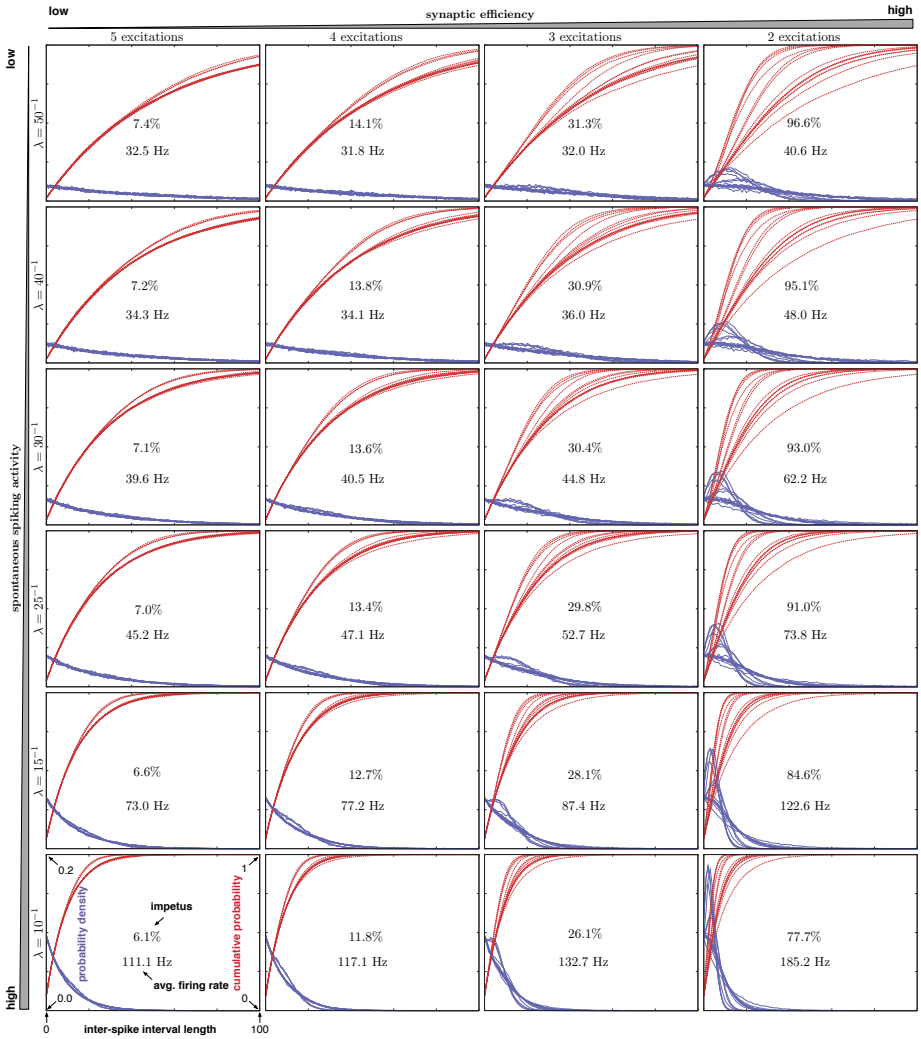


Figure 7.8: Inter-spike interval (ISI) distribution plots of observable nodes (Fig. 7.1). Sub-plots correspond to different parameters of the neural simulation (data length 10 min, parameters above and left of each column and row, scales on bottom left). Each sub-plot shows one data-set with functions of two types for all observable nodes overlayed: (Type I, blue) Probability density functions (range of left vertical axis $[0, 0.2]$) over different inter-spike interval lengths (range of horizontal axis 0–100 msec). These functions approach zero for increasing ISI-length. (Type II, red) Corresponding cumulative probability distribution (range of right vertical axis $[0, 1]$). These functions approach value one for increasing ISI-length. The impetus of the corresponding data is given in the centre of the plot together with the average firing rate. For low impetus ranges probability density functions resemble that of an exponential distributions, which correspond to the ISIs that originate from the baseline stimulation using Poisson processes. As the impetus increases, they become more similar to Poisson-type distributions, which correctly reflects the higher percentage of spikes evoked by excitatory potentials (see text).

(Synaptic Matters) In the current simulation set-up, a single parameter controls the synaptic efficiency of all connections; it determines the size of the excitatory post-synaptic potential. The synaptic efficiency was chosen to be equal for all synapses, which does not match physiological data: Post-synaptic potentials can differ, even within a single cell, depending on the location of the synapse [Kandel et al., 2000, pp.224]. Where the synaptic input occurs can also affect the time-lag of synaptic transmission [Kandel et al., 2000, pp.143]; this has been uniformly modelled as a lag of 1 time-bin. Similar to previously discussed options, it is not difficult to extend the neural model such that efficiency and time-lag can be controlled separately for every synaptic connection; however, such diversification complicates the investigation of the score: Considering different synaptic strengths, for example, makes it difficult to confirm that plausible links over shorter time-lags are recovered with higher preference than over long ones (Fig. 7.6). Such conclusion would have to take into account the synaptic strengths of the neural path underlying the links, in order to facilitate a comparison between recovery rates; otherwise it cannot be excluded that certain links are simply recovered more often, because corresponding synaptic efficiencies are higher. Investigations at later stages could address the effects of different synaptic properties on network inference; however, this is not within the scope of pioneering studies presented here.

Potential aspects to model further include different types of synapses. In the presented set-up all modelled synaptic connections are excitatory. This does not match practical data, which suggests that neurons generally receive both excitatory and inhibitory inputs [Kandel et al., 2000, Chapter 12]; however, inhibitory ones have not been considered in the model. The same applies to uni- or bidirectional electrical synapses, which can lead to (nearly) instantaneous interactions between neurons [Kandel et al., 2000, p.175]. Modelling inhibitory synapses simply requires to reverse the effect of the post-synaptic potentials, by which a mixture of excitatory and inhibitory interactions can be easily achieved. Electrical synapses could be implemented through (quasi-)instantaneous coupling between two neurons' membrane potential. Including these different types of synapses in the model can increase the complexity of the dynamics whose effects on network recovery are worthwhile to be studied. Again, this has not been done, not only to reduce the parameter-space, but also to avoid problems associated with the careful adjustments of synaptic efficiencies with respect to each other. Such reciprocal balancing is needed in order to guarantee a sufficiently high impetus of the data, because otherwise, networks cannot be revealed successfully due to a lack of information (Section 6.2.2). Even if excitatory synapses are considered only, network dynamics are very sensitive to parameter changes

and it can be tricky to find settings that result in a satisfactory impetus.[4] The parameter adjustment problem becomes more complicated when inhibition is included in the model. This is because two counteracting forces must be tuned such that the dynamics of the network are reasonable; i.e., they must neither be extinguished after periods of low spontaneous activity nor show continuous bursting, for example. Future simulations, however, could investigate the effect of inhibitory connections on network inference.

7.3.5 Selection of Networks to Score

For a thorough investigation of the SSS it would be helpful to test its assessment of any possible network for a given data-set. Unfortunately, often too many networks exist (Section B.2), such that the analysis must be limited to a manageable sub-set. For the shown results this sub-set has been chosen to span all networks where none of the nodes has more than 3 parents. Therefore, for each node, all parent configurations with up to 3 parents were evaluated exhaustively, such that it is for sure that the best network within that sub-space has been found. Despite the SSS's tendency towards sparse networks, a better scoring one with nodes having more parents might exist; in other words, it is possible that the best scoring configurations have been excluded from the analysis a priori, by limiting the number of parents. However, leaving the configuration space unrestricted and using search heuristics or MCMC sampling methods (Appendix D) in order to select promising networks to score does not circumvent this problem. These optimisation techniques cannot score all configurations either, but they use actual score values to determine deterministic paths or random walks in the configuration space. In sufficient computation time and if the assumptions of the method match the characteristics of the score, these methods can thereby find local and global maxima even without having to score all configurations. However, many of these approaches involve at least one random component by which results cannot be guaranteed to be replicated. This side effect depreciates the value of these methods for the assessment framework: Although optimisation methods might reveal higher scoring configurations from all configurations than an exhaustive evaluation of a limited sub-space, results must be replicable in order to compare results. But random effects can lead to inference of different networks from the same data. Such variation is entirely due to the search method, but not the SSS. Since the purpose of the framework is to assess the SSS and not the search method, side effects

[4] Changing parameters that control the rate of spontaneous activity or synaptic efficiency can have a non-linear effect on the impetus. As an example consider figures 7.3 and 7.4: Parameters of the neural simulation were changed in equidistant steps, but the impetus of the data exhibits irregular jumps.

that might blur results must be minimised. The computationally expensive solution to evaluate sparse network exhaustively ensures that performance of the SSS is not influenced by un-foreseeable random effects. Uncertainty about the existence of higher scoring networks is thereby limited to that arising from incomplete evaluation, which must be accepted for any selection procedure of networks. Future work could compare different optimisation methods, which will be needed in higher dimensions where an exhaustive search (even for limited number of parents) is infeasible. However, such investigations are beyond the scope of this work, which aims at suggesting a practical and reproducible assessment-procedure in the first place.

7.3.6 Score Parameters

Learning networks using the SSS requires two parameters to be set: the decay constant d and the shift constant Δt. In the presented series, the shift constant remained fixed while the decay constant has been altered in order to assess its importance for successful network recovery. The inferred networks were found to be stable across different decays of the activity level (Fig. 7.6b-e), which has been varied within bounds that are reasonable for the size of the simulated network.

Additional to the decay constant, different choices for the shift constant should also be tested. However, due to the small size of the simulated network, here, the shift constant can only be varied within a very small range. Otherwise it is foreseeable that performance would drop, because no plausible links exist that span over more than three time-lags. (A larger network would thus have to be chosen in order to run extensive series varying the shift constant; however, results could not be directly compared to those gained from the network used before.) Thus, instead of testing numerous different shift constants, only one additional setting has been evaluated: The same parameter series as before was run (Fig. 7.2) except that the shift constant was chosen $\Delta t = 2$. As to be expected, the resulting recovery rates and precision are generally lower than with the optimal setting of the shift constant, which is especially true when the impetus is low (Fig. 7.3 vs. 7.9). This is to be explained by the score's lag-window $[2, 4]_{\mathbb{Z}}$ when using this shift constant ($\Delta t = 2$): Correlation with lag one is not accounted for in this setting, such that corresponding plausible links cannot be learned; but these are 12 out of 34 plausible links (Fig. 7.1b). However, remaining links to recover are found at high impetuses, such that both settings of the shift constant result in similar performance. The SSS thus shows a certain tolerance for a sub-optimal setting of this parameter.

The second simulation series (Fig 7.2b), in which the data-length was fixed

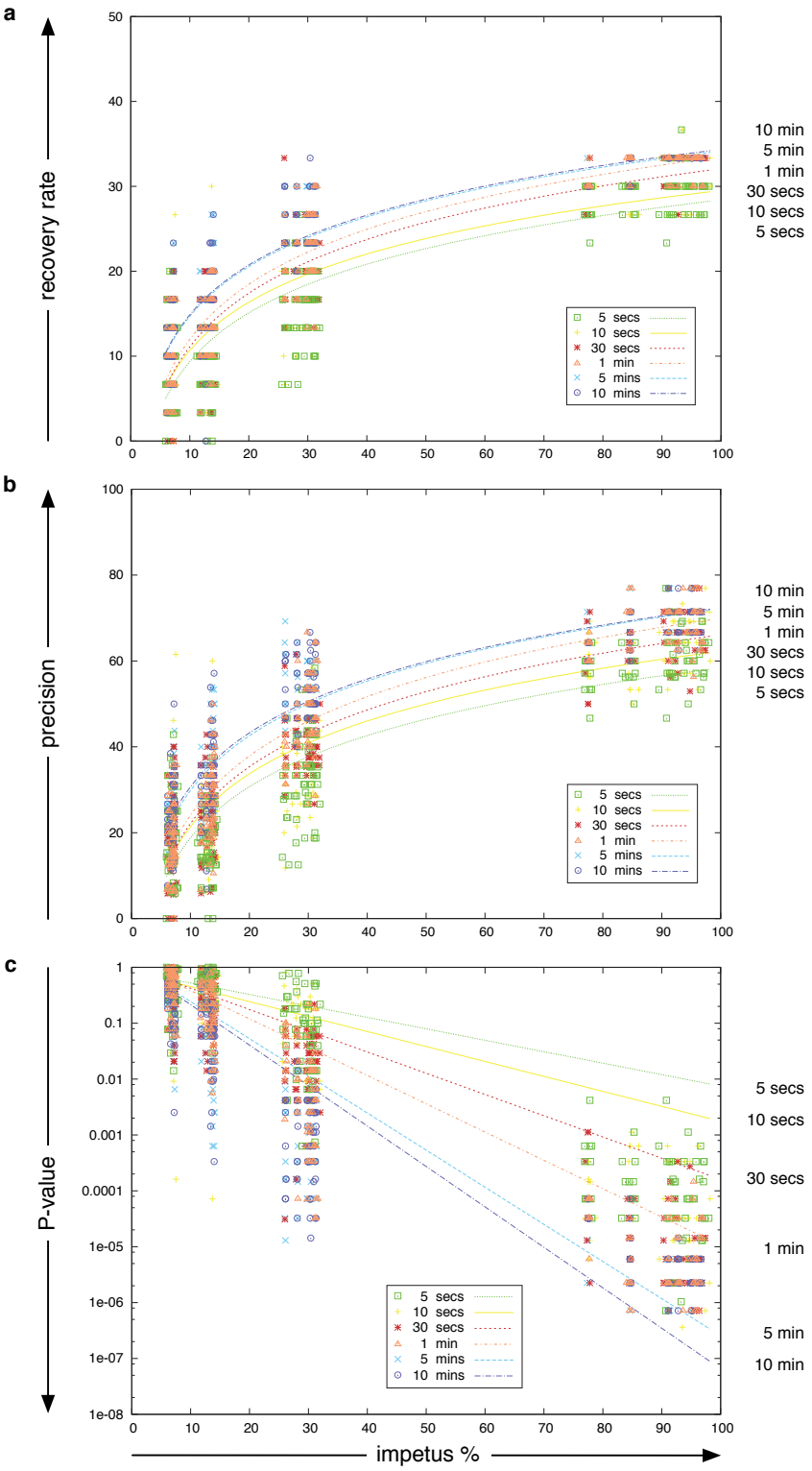


Figure 7.9: Legend in separate box on page 129.

Fig. 7.9 legend: Quality of recovered networks depending on impetus and recording length ($d = 3^{-1}$, $\Delta t = 2$ fixed). Data points correspond to one learned network each (learned from a separate simulation); trend lines fitted to points of each data-length. **a** Recovery rate of all plausible links for different impetuses and different lengths of data. **b** Precision of recovered links for different impetuses and different lengths of data. **c** P-values of precision shown in (b) on logarithmic scale. At low impetuses performance is found to be worse than with shift parameter $t = 1$ (Fig. 7.3). Both settings of the shift constant result in similar performance when the impetus is high.

Fig. 7.10 legend: Quality of recovered networks depending on impetus and decay constant (recording length 30 seconds, $\Delta t = 2$ fixed). Data points correspond to one learned network each (learned from a separate simulation); trend lines fitted to points of each choice of decay constant. **a** Recovery rate of plausible links for different impetuses and choices of decay constant d. All decay constants $d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}\}$ result in similar good recovery rate. Minimally worse recovery rate for decay constant $d = 1$, due to correlation with time-lags other than 2, which are not considered in this case. **b** Corresponding precision for impetuses and decay constants shown in (a). As in (a) performance is similar for decay constant $d \in \{5^{-1}, 4^{-1}, 3^{-1}, 2^{-1}\}$, but worse for $d = 1$ (for the same reason). **c** P-values of precision shown in (b) on logarithmic scale. Precision reached with $d = 1$ is least likely.

and the decay constant was varied, has also been repeated (with shift constant $\Delta t = 2$) and similar effects as before can be observed (Fig. 7.4 vs. 7.10). The performance reached by using different activity decays is less separated than with shift constant $t = 1$ (Fig. 7.4). This is probably due to the fact that no plausible links spanning over more than three time-layers exist, such that any decay constant smaller than 2^{-1} widens the score's lag-window without benefit. This hypothesis would be supported by the worsening of P-values, as the decay becomes smaller.

For an extended analysis of the shift parameter a larger network would have to be simulated, as only when provided long enough paths, larger shift constants can be reliably tested. This, however, is subject to future work.

7.3.7 Performance Measurement

In the final step of the assessment framework learned networks are evaluated. Performance is determined by a comparison of inferred networks to a reference network using different fundamental measures (Section 6.3.1). It is presumed

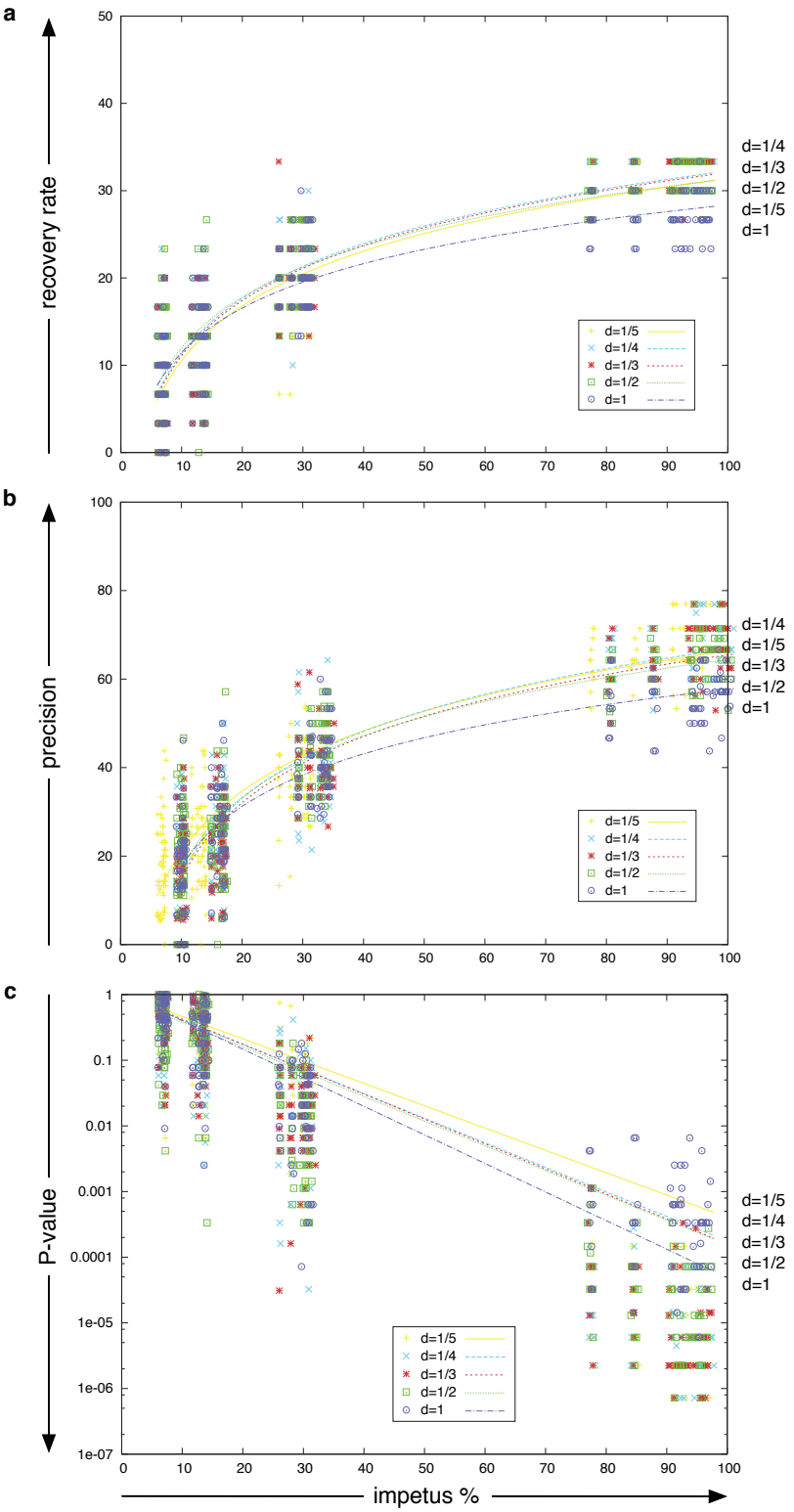


Figure 7.10: Legend in separate box on page 129.

that learned networks that are good share elements of the reference network, which is derived using the plausibility concept.

The basic measures of how well networks match, recovery rate and precision, provide a good overview of the overall agreement between networks. They fail to inform about which parts of the network match. Fortunately, the simulated network is small enough to identify such particular sites by eye. Even larger numbers of networks can be inspected with this manual approach by averaging them (e.g. Fig. 7.6). Comparing (average) networks to the reference network, measuring the two success rates, and calculating the precision's P-value can thus give a good overall picture of quantity and quality for inferred networks — relative to the reference network.

Classifying links according to their plausibility is central to the assessment of results: The reference network, to which all learned networks are compared, is composed of all plausible links. The plausibility of each link is derived from the known, simulated network, based on whether two model neurons can show correlation within specified time-lags based on their connectivity. Network inference techniques have previously been tested by simple link-by-link comparisons with the simulated network; given that the simulated system was fully observable [Chornoboy et al., 1988, Baccalá and Sameshima, 2001, Nykamp, 2005, Makarov et al., 2005, Okatan et al., 2005, Lindsey and Gerstein, 2006, Astolfi et al., 2006, Eichler, 2006] or under partial observability, but restricted to the evaluation of few hand-chosen paths in the network, which were expected to be learned [Cadotte et al., 2008]. These procedures may not be appropriate under partial observability conditions, or they might not be suitable for automated analysis of many result networks, which is needed for extensive series, as presented here. Differently, the algorithmic implementation of the plausibility concept facilitates automatic analysis of many networks.[5] Further, in contrast to manual approaches, which depend on individual preferences, the proposed concept is more objective since it provides a reproducible classification of links.

Lacking alternatives to the plausibility concept in the literature, a second, different approach has been presented in chapter 6. This second framework compares networks indirectly by the neural dynamics they produce; these are compared with spike train metrics (Section 6.5). The similarity of the spike trains produced by the original and the learned network is reflected in the distance, which thereby provides a measure of how well the two networks match functionally. Using the distance as a measure of performance could complement those that were used in connection with the plausibility concept, such that it

[5]The plausibility concept is indeed suitable for high throughput evaluation: For example, for result figures 7.3 and 7.4 a total of 2,640 networks has been assessed. In these cases the simulated network has been the same, but with the plausibility approach, a different network could have been used in each simulation without rising computational costs noticeably.

would be best to apply both frameworks simultaneously in order to get a more comprehensive picture. However, this has been hindered by computational reasons. Only one approach could be used and preference has been given to the plausibility approach, because it requires far less parameters to be specified (Section 6.5, Fig. 6.6).

(Choice of Plausible Time-lags) The two parameters of the plausibility concept (minimal and maximal plausible lag) define the plausible lag-window. This window determines which links can become classified plausible and it can be argued that choosing it to be equal to the SSS's lag-window $[1, 3]_{\mathbb{Z}}$ is not optimal: This matching is expected to yield the best inference results because links in learned networks only connect nodes with appropriate time-lags — those within the SSS lag-window. As the lag-windows are identical all links that are plausible could be learned, but not those spanning over time-lags outside the plausible lag-window. By this restriction, network inference can be seen as being guided towards correct solutions. However, it has been shown that the performance of the SSS does not depend on this matching by parameter series in which the decay constant d (and thus the lag-window) has been varied (Fig. 7.2b). (The lag windows also do not match in the series run with shift constant $\Delta t = 2$, for which results are shown in Figs. 7.9 and 7.10.) The results of the series do not indicate any benefit from matching of the two lag-windows (Fig. 7.4); indeed, decay constants $d = 4^{-1}$ and 5^{-1} perform slightly better (on average), although corresponding lag-windows do not match the plausible one.

7.3.8 Comparisons to Other Methods

Using the simulation framework, the SSS has been assessed under a variety of conditions. These investigations showed that the SSS can reveal plausible networks if the impetus of the data is sufficiently large. This is an important observation, but which is somewhat dimensionless, since the impetus has not been used before in order to specify the dependence of other techniques on it. It would thus be desirable to see how other network inference methods (e.g. those in Table 2.1) perform under similar conditions. This would allow for a direct comparison to the new SSS; however, within the scope of this work not all techniques can be tested, but one has been chosen: cross-correlation [Perkel et al., 1967], which results in comparable computational costs. Details on how this technique has been applied are given next.

Cross-correlation is calculated between two time-series; in order to account for different time-lags, the correlation between any two channels A and B has been evaluated for time-shifted data: shifting B's data forward (relative to A)

Table 7.1: Quality comparison of networks inferred with the SSS and cross-correlation for different ranges of impetus. The data of the first simulation series (Fig. 7.2a) were analysed with both techniques. Resulting networks were categorised according to the impetus of the data (low, medium, high) before evaluating them; for each category values (recovery rate, precision, P-value) were averaged over all data-lengths and rounded.

impetus i		recovery-rate		precision		P-value[a]	
		SSS	cor	SSS	cor	SSS	cor
low	$(5\% \leq i \leq 20\%)$	16%	12%	37%	44%	0.14	0.26
medium	$(25\% \leq i \leq 35\%)$	23%	13%	53%	53%	0.02	0.16
high	$(75\% \leq i \leq 100\%)$	31%	16%	74%	58%	10^{-4}	0.09

[a] Note that a comparison regarding precision does not render the comparison of P-values superfluous: Precision corresponds to percentage of recovered links that are plausible, but the P-value also takes into account the total number of recovered links. Learning all possible links can yield a moderate precision, if many plausible links exist, but the corresponding P-value one will then indicate the irrelevance of this achievement.

by $1, 2$, or 3 time-bins. The maximal correlation between channels was then assigned to the corresponding link $A \rightarrow B$. Links with maximum correlation equal or above a threshold α are *learned*; all remaining ones are not. In practice, the threshold α would have to be chosen by the user, which is not possible for the large number of simulations performed here. Instead, for each analysed data-set, the threshold has been chosen to yield optimal performance: According to the *Neyman-Pearson Lemma* (Neyman and Pearson [1933] or [Dayan and Abbott, 2005, pp.119]), no better choice for α exists than the one which yields the highest *likelihood-ratio* (i.e. recovery rate / [100 - precision]). The threshold α is set according to this optimal trade-off and thus yields the best performance that can be reached with this technique.

The cross-correlation has been calculated for the same data the SSS has been applied to in the first series (Fig. 7.2a). Like for the SSS, learned networks were assessed regarding their plausibility. In order to compare both approaches, the average performance of each has been determined for three ranges of impetus (Table 7.1), which show that results of the SSS are generally better. Only in two cases, for low and medium impetus, the precision of the cross-correlation reaches or exceeds that of the SSS.

For practical application it is of interest how the performance of either technique depends on the amount of data that are available for analysis. To show this, the results have been clustered according to the length of the spike trains, but also regarding the impetus of the data (Fig. 7.11). Both more data and

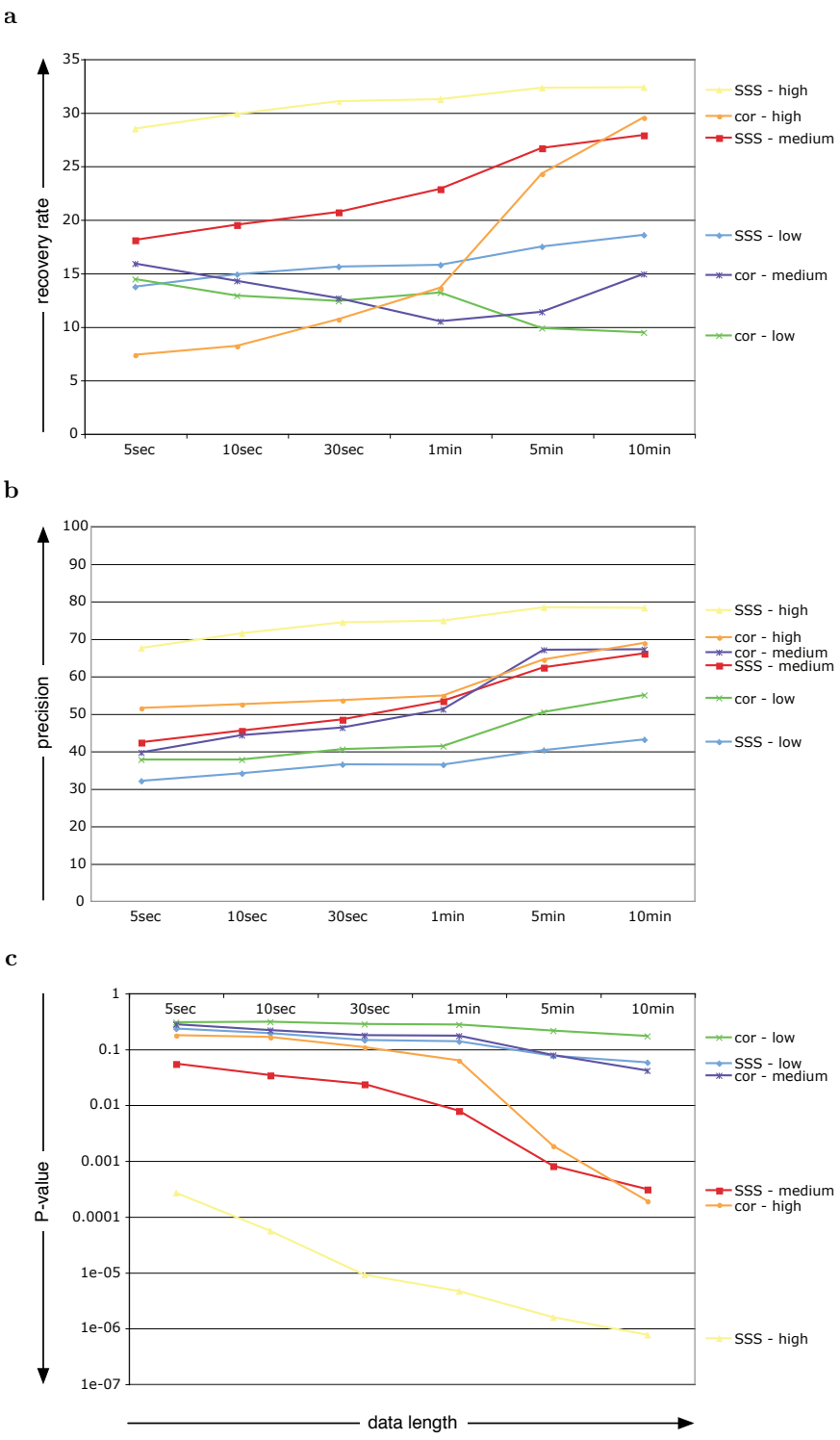


Figure 7.11: Legend in separate box on page 135.

Fig. 7.11 legend: Quality comparison of networks inferred with the SSS and cross-correlation for data of the first simulation series (Fig. 7.2a). Resulting networks were categorised according to the length and impetus of the data before evaluating them; for each category values (recovery rate, precision, P-value) were averaged. (Impetus ranges low, medium, and high defined according to table 7.1.) The curves illustrate the performance of both methods for different data-lengths and impetus: **a** recovery rates, **b** precision, and **c** P-values of precision shown in (b) on logarithmic scale. (See text for interpretation.)

higher impetus generally benefit either technique; however, there are two exceptions concerning the recovery rate of the cross-correlation method: For both low and medium impetuses an increase of data length does not improve but decrease recovery rates. For medium impetus data this seems to be a transient drop, since rates increase as data length reaches the order of several minutes, but no such recovery is observed at low impetus. This might be an effect of spurious correlations, which average out as amount of data exceeds a critical length. The significant improvement in recovery rate at high impetus (between 1 and 5 minutes) supports this hypothesis, which, in order to be tested, would require much longer data-sets; however, these experiments have not been performed. In contrast to cross-correlation, the SSS shows a steady improvement in performance over the explored parameter range and it seems to be less affected by spurious correlation in short data-sets.

In conclusion, the SSS yields comparatively good results, especially since cross-correlation was applied with optimal threshold α: Knowledge used in order to determine this threshold is generally not available in practical application; performance of the cross-correlation method is thus likely to be worse than shown here. However, the SSS can successfully compete against this technique under best case performance. The reason for this is the score's interpretation of the data (Section 5.2.2), which focuses on excitatory relationships, as simulated here. This practically demonstrates the advantage of data-specific adaptation for network inference, as discussed earlier (Chapter 5).

This ends the neural simulation series and their discussion. Corresponding results are now briefly summarised and analysed with respect to their implication on real data analysis.

7.4 Conclusions From Simulation Work

In the preceeding sections the SSS has been assessed with numerous simulations using a wide range of parameters, which resulted in various grades of difficulty to infer networks. The score has been found to successfully reveal good networks irrespective of (non-extreme) choices for the decay constant, although, even at the highest impetus considered (100%), only half of the spikes are informative about network structure. Further, the SSS has been found to deliver superior results to that from a cross-correlation analysis over multiple time-lags for which parameters were chosen optimally.

The different findings give valuable indications for practical application where doubts about parameter settings exists. In such situations the simulation results suggest a potential approach for analysis: A fixed shift-constant Δt needs to be chosen and should then be used together with a series of decay constants d. Therefore, the investigator's expectation about the minimal time-lag at which relationships may occur is needed: Electrode placement (close to each other / in different brain regions) during data recording may suggest smaller or larger time-lags; expressing the expected time-lag as number of corresponding time-bins in the spike train yields the optimal shift constant. Once Δt is chosen, networks need to be learned for gradually decreasing decay constants, until a further decrease does not lead to substantial differences in learned networks. This result stabilisation indicates that the decay constant is sufficiently small, such that the resulting lag-window is wide enough to capture all relevant dependencies. Practically, the average of learned networks can indicate which links are consistently found (Fig. 7.6f). The largest decay constant for which (most of) these links are learned is optimal, as the lag-window is just as wide as it needs to be. (A decay of $d = 1/2$ would thus be a reasonable choice for Fig. 7.6.) As discussed earlier (Sections 3.1.1 and 5.1), the shift- and decay-constant determine the interpretation of learned networks.

Practical analysis might also benefit from the insights concerning the importance of data-length for network inference (Section 7.3.2). Due to the (at most) logarithmic improvement of recovered networks when increasing the amount of data, it seems reasonable to partition recordings into adequate lengths before learning networks from them. Instead of using all the data at once, networks learned from different sections can then be averaged (Section B.3.1) in order to remove false positive links (as seen in Fig. 7.6). But partitioning a long data set and processing it piecewise can also be beneficial in order to detect changes in dynamics: As will be demonstrated in the next chapter, different dynamics can manifest in substantially different networks. Thus, comparing networks learned from different data sections can help to detect such changes.

The presented work on simulated data has confirmed the SSS's ability to reveal neural interactions from simulated spike trains. Together with the theoretical exploration of the SSS (Chapter 4), the investigations have given broad insights to the score's working principle and guidance to practical application. However, it remains to be shown that the SSS delivers adequate results for real data in which, for example, neural interactions might be significantly weaker than simulated here; but this is the subject of the next chapter.

Chapter 8

Real Data Applications

Throughout the preceding chapters a new analysis technique, the Snap Shot Score, has been introduced (Chapter 3) and investigated both theoretically (Chapter 4) and in simulations (Chapter 7). These studies have led to a comprehensive understanding of the score, which is preliminary to its practical application. In this chapter, the SSS is applied to two types of real data, which originate from the retina and the hippocampus, respectively. The data are not analysed in order to reveal novel biological insights, but to validate that the score leads to sensible results when used on real data. Separate sections for each data type begin with a brief introduction in which the origin and known characteristics of the data are reviewed. Thereafter, results gained from applying the SSS are presented and confirmed to agree with expected outcomes.

8.1 Retinal Data

The first type of real data to which the SSS is applied has been collected from the retina. The retina is located at the back of the eye and it is probably the most accessible part of the brain (Fig. 8.1).[1] Vision starts at the retina, as light enters the eye through the transparent *cornea* and hits the retina, which acts as a photon-receptor. Counter intuitively, light has to pass through all layers of the retina before it reaches the deepest layer consisting of photo-receptors (*rods* and *cones*), which convert photons into electrical signals [Kandel et al., 2000, pp.507].[2] These are then transmitted via *bipolar cells* to the top most level of

[1] The retina is considered to be part of the brain, because, during embryonic development, it emerges from the neural tube [Dowling, 1987, p.8]: the same type of cells principal components of the nervous system originate from, like the (rest of the) brain and the spinal cord [Oyster, 1999, pp.62].

[2] It has been discovered recently that other cells, namely ganglion cells, are also photon sensitive (see [Hankins et al., 2008] for a review). However, their sensitivity is much less prominent than that of photo-receptors (by a factor of about 10^4) [Do et al., 2009].

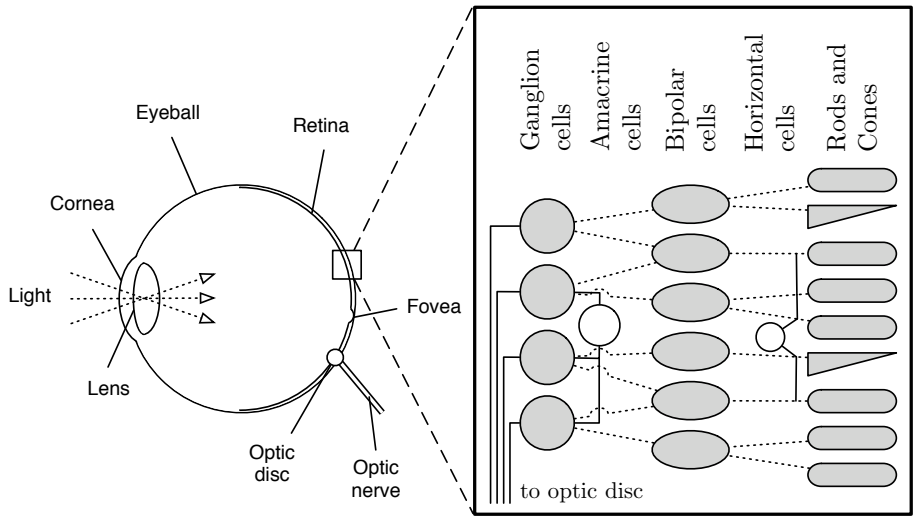


Figure 8.1: Adopted from Hubel [1995]: Schematic illustration of retina location within eyeball (left) and close-up view on layers of retina (right). Prominent cell types and their connectivity in different depth of the retina define several distinct layers of it. Photo-receptors (rods and cones) are depolarised when hit by entering light [Kandel et al., 2000, pp.510]. Signals of many receptors converge while being forwarded to ganglion cells, which generate action potentials that leave the retina via the optic nerve. (For none of the other retinal cell types, except for amacrine cells, has spiking been reported. Generally, except for ganglion cells, transient signalling is common.) [Dowling, 1987, pp.82].

ganglion cells whose projections leave the retina through the *optic disc*, forming the *optic nerve*. Visual information is thereby forwarded to the thalamus (lateral geniculate nucleus, LGN) and further to the visual cortex where different features are extracted for further processing [Kandel et al., 2000, pp.523]. However, before all that, i.e. in the retina itself, incoming light actuates a cascade of reactions and corresponding local computations: Signals from photo-receptors are processed, such that firing of ganglion cells shows distinctive selectivity for particular visual features in their *receptive field* [Kandel et al., 2000, pp.507]. Further, the visual information transmitted by the ganglion cells has undergone contrast exaggeration before leaving the retina [Oyster, 1999, Ch.14, p.635]. The local computations performed in the retina distinguish it from a group of simple receptors (like for touch, temperature, or pain) that simply translate stimuli information into the electrical domain without modifying it afterwards [Kandel et al., 2000, pp.430].

Intensive studies of the retina make it a well known neural structure whose function is understood in large parts. One particular aspect that is still under investigation is the role the retina plays in development of the visual system, where the retina shows specific activity patterns. Such activity has been recorded from blind adolescent mice and is used here in order to verify the SSS's ability to analyse real data. Details on characteristics of these data follow next.

8.1.1 Retinal Waves

Several studies have reported a distinct kind of spontaneous spiking activity in the retina that occurs during development: So called *retinal waves* occur in many species before birth [Wong, 1999]. This form of activity can exhibit high spatial coherence while travelling across the retina (Fig. 8.2) [Maffei and Galli-Resta, 1990, Wong et al., 1995]. As the animal matures, the activity patterns change from distinctive waves to chaotic [Demas et al., 2003] and they completely disappear shortly after birth [Sernagor et al., 2006, pp.265]. Until then, the activity can be observed by mounting the retina onto a multi-electrode array (MEA) [Meister et al., 1991, Wong et al., 1993], for example, and differences both over time and across different species can be identified [Wong, 1999]. In the following, these distinct wave-like activity patterns are used for an ad hoc validation of the SSS's ability to handle multi-unit data.

8.1.2 The SSS Applied to Retinal Wave Data

Evelyne Sernagor kindly provided multi-unit spike train data, which have been collected from a neonatal retina of a Crx-knockout mouse [Furukawa et al., 1997, Adams et al., 2008]. Spikes were extracted from the raw data with a voltage-

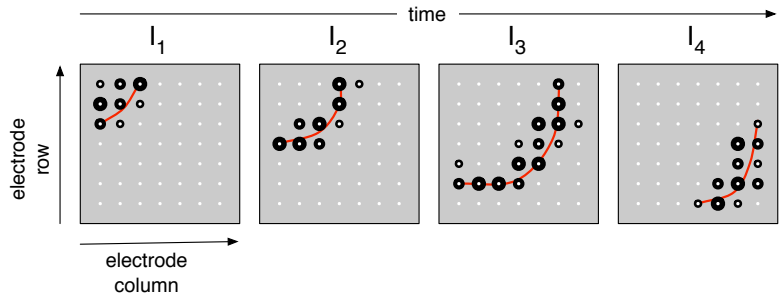


Figure 8.2: Adopted from Demas et al. [2003]: Schematic visualisation of neural activity recorded from a multi-electrode array during four succeeding, non-overlapping time-intervals (I_1, I_2, I_3, I_4) of equal length. Radius of circles proportional to average firing rates of cells recorded with electrodes (white dots). Line indicating wavefront in order to emphasise wavelike propagation of neural activity.

threshold and binned with 1 msec resolution.[3] No further modifications were applied to the data before using them for network inference with the SSS.

In order to validate that the SSS reveals adequate networks, 3 succeeding (5 second long) sections were extracted from the recording. For each of these, networks have been learned by evaluating all parent configurations with up to 3 parents.[4] Superimposing the resulting networks suitably onto the MEA shows that revealed links predominantly connect nodes that correspond to highly active electrodes (Fig. 8.3). As time progresses, the retinal wave travels across the array; according to this, nodes in learned networks are linked differently, such that their connectivity matches the centre of neural activity very well for all three data-sections. Although none of the less active channels is completely silent, no spurious links to or between them occur. These observations have been found nearly identical for all choices for the shift constant ($\Delta t = 1, \ldots, 5$), such that results are only shown for the setting $\Delta t = 5$. Unfortunately, factual information about the retina is insufficient in order to predict the functional connectivity that would be expected to be revealed from the data. Therefore, it is not possible to evaluate the plausibility of learned networks in a quantitative manner like in simulations (Chapter 7). For this real data-set the only certain clues are that dynamics are local and that they obviously change over time; learned networks should therefore also be local and changing over time. This

[3]Evelyne Sernagor and Chris Adams used a high density MEA for recording from a Crx-knockout mouse [Adams et al., 2008]. The data were collected on postnatal day 7 (P7) and Evelyne Sernagor applied a voltage threshold in order to extract the spikes from the raw data. Note that photo-receptors in these knockout mice lack the outer segment, which contain the photo-pigment; animals are therefore blind, so that abnormal effects can be observed in the data.

[4]The SSS's parameters were chosen as follows: decay constant $d = 3^{-1}$ and shift constant $\Delta t = 1, \ldots, 5$. The LAT has been determined from configurations with 3 parents.

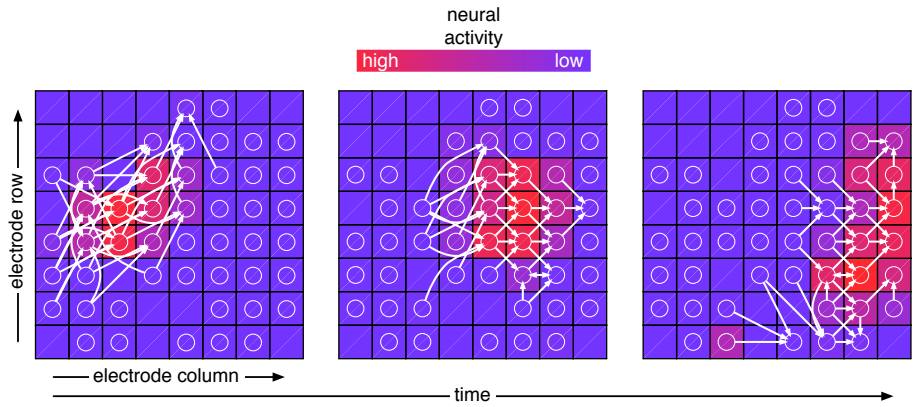


Figure 8.3: Retinal data provided by Evelyne Sernagor: Three heat-maps of a retinal activity wave travelling from left to right. Each heat-map shows activity during a 5 second period that has been extracted from a recording made with a MEA from a gene-modified mouse (see text for details). Squares correspond to one electrode each and are arranged according to their position on the MEA. Dependency structures revealed with the SSS are superimposed as networks such that nodes match the corresponding activity-field of the electrode. (Squares without nodes correspond to *dead electrodes* for which no data is available.) Good agreement of network links with centre of neural activity is observed throughout the time course.

is indeed what the results show: For each data-section, the interconnectivity of learned networks closely matches the high spatial coherence of the dynamics; and hence, as neural activity progresses, this change is reflected in the structural differences.

The results so far show that differences in the data are plausibly reflected in inferred networks. Likewise, one would expect that networks learned from similar data are structurally related. That this is indeed the case can be shown by extracting overlapping sections from the data: Some of the data are thereby shared between different sections, such that networks inferred from them are expected to be more alike than those learnt from sections without any overlap. In order to test this for the retinal wave data, it was sectioned in six 10 second long pieces with an overlap of 5 seconds for directly succeeding ones. Networks were learned for all data-sections[5] and compared to each other in a pairwise manner. Similarity has thereby been assessed by *edit distance*, i.e. the number of changes that have to be made to a network in order to transform it into another. Given the adjacency matrices of both networks (Section B.1), the

[5]All parent configurations with up to 5 parents were evaluated exhaustively for all nodes. The SSS's parameters were chosen as follows: decay constant $d = 3^{-1}$ and shift constant $\Delta t = 5$. The LAT has been determined from configurations with 5 parents. Learned networks were composed of all links in the 10 best scoring parent configurations of each node.

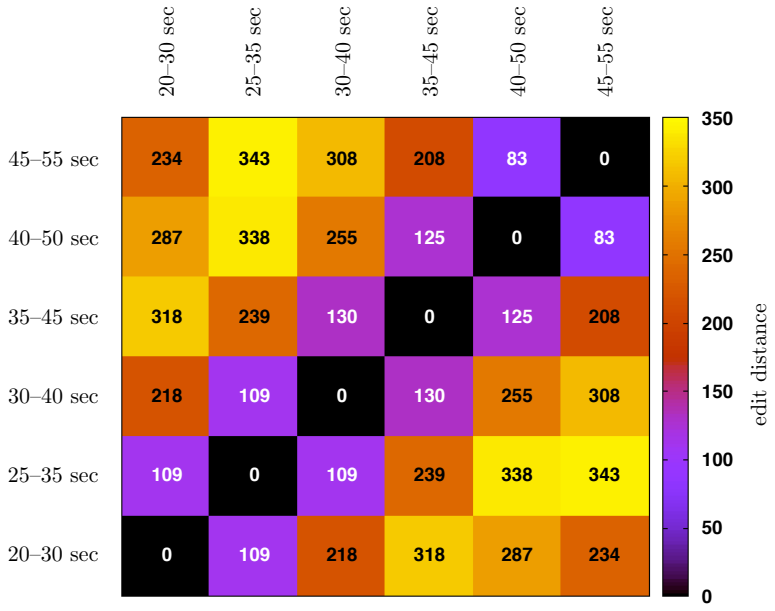


Figure 8.4: Edit distance of networks learned from retinal wave data split in six overlapping sections (10 seconds long, 5 seconds overlap). Data-sections given next to rows and columns, edit distance for each pair within corresponding rectangle. Distances between networks learned from overlapping sections (mean ≈ 111) are lower than to those inferred from data without any overlap (mean ≈ 275). Similar data thus results in alike networks.

edit distance thus corresponds to the number of differing entries between the two and thereby reflects their similarity: Unlike structures have have a high distance, while identical networks have distance zero. Applying this measure to all pairs of networks that were learned from the data sections shows the expected outcome (Fig. 8.4): Networks show similarity to those that were inferred from an overlapping section, but not others. This shows that the SSS performs a consistent assignment of networks, since gradual changes in the data lead to corresponding alterations in structure, but no abrupt variations.

In conclusion, the initial application of the SSS to real data yields results, which are in good agreement with expected outcomes. On the one hand, this is reassuring for the good results from earlier theoretical and simulation-based investigations, and also, it is a positive indication for the score' practical suitability.

The initial test of the SSS on real data has been successful: Recovered networks suitably reflect the wave-like dynamics in the data, which consisted of multi-unit spike trains. Next, the SSS is applied to single-unit data from a different brain

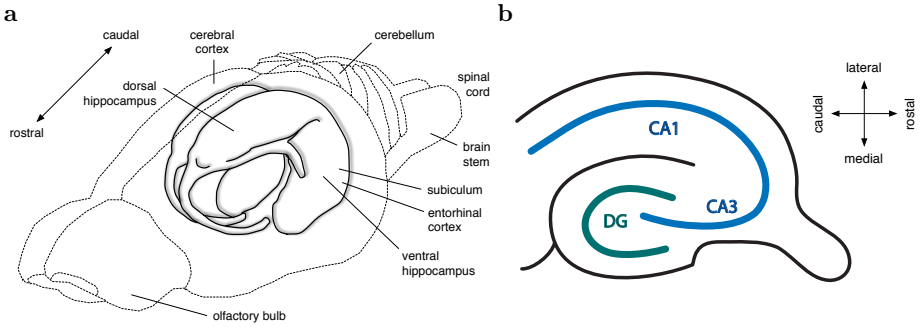


Figure 8.5: Adopted from Paxinos [1995]: Schematic illustration of rat brain and hippocampal formations within (a), as well as distinctive cell organisation within the hippocampus (b). **a** Whole rat brain schematics with highlighted hippocampal formations. **b** Horizontal mid-hippocampus section showing fields CA1 and CA3 (CA = cornu ammonis), and dentate gyrus (DG). Together with the field CA2 (not shown) these are the principal components of the mammalian hippocampus, where pyramidal cells are the primary neuron-type. Pyramidal cells in the regions CA3 and CA2 are clearly larger than those in CA1 [Paxinos, 1995] and axons of cells in both CA3 and CA1 branch extensively within all the hippocampus. Relatively cell-sparse layers lie above and below a single, densely-packed layer of neuronal somata (labelled lines) in both the hippocampus and dentate gyrus [Cooper and Lowenstein, 2002].

region: the hippocampus. In detail, hippocampal place cell data will be used in order to demonstrate that the score consistently captures functional connections between single neurons as well. This is shown in the next section.

8.2 Hippocampal Data

Phylogenetically, the hippocampus[6] is one of the oldest parts of the brain. Together with adjacent brain regions, it forms the *hippocampal formation*. In mammals, this comprises the tube-like hippocampus, the dentate gyrus, the subiculum and the entorhinal cortex (Fig. 8.5) [Cooper and Lowenstein, 2002]. The traditional concept of the hippocampus' functional organisation in mammals is the tri-synaptic circuit, which assumes the formation to be a closed feed-forward loop of excitatory synapses. Current knowledge indicates that this strictly serial design has to be extended by integration of diverse inputs from other brain regions [Cooper and Lowenstein, 2002]. But despite open questions concerning its functional organisation, it is widely accepted that the hippocampus plays an important part in memory; especially its spatial aspects, as used for navigation [Hanser, 2005].

[6]Greek origin hippokampus, meaning sea horse, referring to the shape of this brain region [Cooper and Lowenstein, 2002].

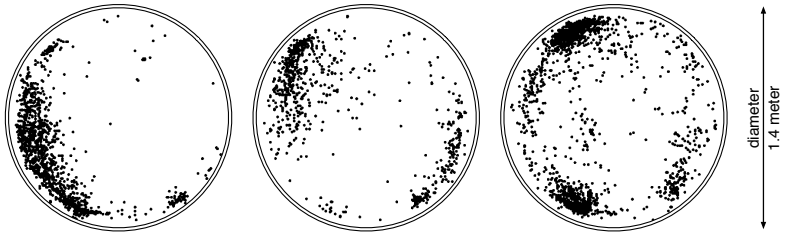


Figure 8.6: Hippocampal place cell data provided by Tom V. Smulders: Three top views on cylindrical arena, each showing place fields of one place cell (recordings from rat). For each cell the animals location is marked with a dot whenever the cell spiked (no filtering or normalisation by time spent in each particular location has been applied). The three cells show selective firing: In their place field(s) firing rates are high and lead to a dense accumulation of dots. Spontaneous activity can be observed outside the place fields as well, but spikes occur far less frequent than within.

8.2.1 Characterising Hippocampal Place Cell Data

During investigations of the hippocampus of a freely moving rat, O'Keefe and Dostrovsky [1971] have discovered cells that showed spiking activity only when the rat was in a certain location of the maze. These cells have therefore been termed *place cells* (Fig. 8.6). Since then, many researches have gathered information about these cells: During the first exposure to a new environment place cells establish their preferred spiking location, named *place fields*, within a few minutes; these usually remain stable during multiple exposures to the same environment. Also, once the place fields are stable, an extension of the maze does not affect the place fields in the original part of the maze significantly [Wilson and McNaughton, 1993]. When the rat is exposed to a different environment, even if it is equally shaped, previously active cells may become silent, or have different firing preferences [Leutgeb et al., 2004]. Even though place fields usually remain stable during repeated visits to the same environment, their shape can change or the preferred firing location can *re-map*, even if the environment remains physically un-varied. This can, for instance, be triggered by fear, and it has been suggested that re-mapping is a mechanism for providing multiple representations of a single environment, dependent on its current state [Moita et al., 2003]. Other findings concerning place cells are that a local field-potential oscillation in the hippocampus, called the *theta rhythm*,[7] is a prerequisite for place cell activity [Foster et al., 1989]. This oscillation's

[7] Frequencies in the band of approximately 4-12 Hz are termed *theta frequencies* [Buzsaki, 2002, Vertes, 2005].

frequency is unaffected by the animal's running speed [Czurko et al., 1999]; in contrast, the firing frequency of a place cell depends positively on the speed at which the rat traverses the dedicated place field [Zhang et al., 1998]. Place fields that are established while visual cues are provided even stay stable when these landmarks are removed [Muller et al., 1994]. Even more, they are unaffected by complete darkness, and thus, they do not solely depend on visual information [Quirk et al., 1990]. Some place cells have been reported to show directionality, i.e. the traversing of the place field alone is not sufficient to cause the place cell spiking, but also the direction of passing through it has an effect. This has, for instance, been observed on linear tracks and on arms of an 8-arm-maze[8]. Directionality is explained to be induced by the shape of the maze, which retards the rat from moving and looking in certain directions. This is in accordance with experiments in a circular arena, showing directionality only for place fields near the wall, but not centred ones; directionality is thus thought to be an intrinsic property of edge fields. Observations of cells that are omnidirectional in a circular arena, but directional in the 8-arm-maze, further support the conclusion that directionality is not a cell specific feature [Muller et al., 1994].

The precise role and function of place cells is still under investigation. However, early results already suggested that the hippocampus provides a cognitive map, which serves spatial orientation purposes [O'Keefe and Nadel, 1979]. This hypothesis had to be broadened due to findings in recent years. For example, experiments showed that place cells are not solely dependent on location, but also on different types of memory episodes[9] [Wood et al., 2000]. Also, it is still unclear if place cells code for the present position of the rat, the recently past position (retrospective coding), or a future position (prospective coding).[10] The interesting properties of place cells raise the question of how their distinct characteristics emerge and the recent discovery of so called *grid cells* [Hafting et al., 2005, Heyman, 2006] provides some possible explanations: Like place cells, the firing of grid cells is associated with particular spatial fields, but differently, grid cells have several firing fields, which are arranged in a highly regular grid-like fashion. Due to their similarity, grid cells have been investigated in experiments analogous to place cells [Fyhn et al., 2007, Giocomo et al., 2007, Barry et al., 2007, Hafting et al., 2008] and since the entorhinal cortex, in which grid cells were found, is anatomically adjacent and functionally interconnected with the hippocampus [Brun et al., 2002], models have been suggested in order to explain

[8] An *8-arm-maze* is a regular starlike arrangement of straight branchings from a connecting centre [Olton and Samuelson, 1976].

[9] Wood et al. [2000] observed that the place cell activity in the central stem of a connected T-maze was dependent on the previous turning direction at the T junction.

[10] Modelling results presented by Barbieri et al. [2005] suggest that retrospective coding is performed; however, the determined time-lag has to be verified and possible influence factors on this, such as running speed or distance of visual cues are not yet accounted for.

the emergence of place-cells from grid-cells [Franzius et al., 2007, Molter and Yamaguchi, 2008, de Almeida et al., 2009]. Finally, it should be noted that, although most experimental work is done in mice and rats, place cells have also been identified in the human hippocampus. But due to a very small experimental basis, not much can be said about them; anyway, the few performed investigations showed that human place cells basically match the properties that have been identified in animals [Ekstrom et al., 2003].

8.2.2 The SSS Applied to Place Cell Data

Tom V. Smulders kindly provided hippocampal place cell data. The 40 minute long data-set consists of 25 single-unit spike trains, which were recorded with a MEA from hippocampus in a freely moving rat. When collecting the data, video-tracking (with a temporal resolution of 50 Hz) has been used in order to determine the animal's position, which can therefore be linked to the neural dynamics. Before using the data, the spike trains have been examined in order to select 10 channels for which cells had designated place-fields. These data were used at full resolution (1 msec bin-size) in the experiment described next.

Place cells exhibit broadly varying dynamics, which are associated with their place fields (Fig. 8.6). As the animal traverses the arena different locations are visited, which evoke responses of corresponding place cells. This relationship between spatial position and neural activity can be used for a validation of the SSS on real single-unit data: Depending on the rat's location, different dynamics are observed, which are expected to impact on inferred networks. In order to test this hypothesis, the data are partitioned according to the rat's position. Therefore, twelve different sections are defined near the wall of the arena (Fig. 8.7); every pair of directly adjacent sections (e.g. 1&2, 2&3, 12&1) is then used to partition the data: As long as the animal is in any one of the paired-sections, the data of all channels is extracted. These data snippets are then concatenated to one spike train data-set, for any section-pair.[11] By construction, every of the resulting twelve data-sets exhibits some overlap with two others. Networks learned from these data are therefore expected to exhibit a certain degree of similarity; for all others, in contrast, their structure should be different. Indeed, analysing the data-sets with the SSS,[12] shows that resulting networks are more alike when corresponding to overlapping sector pairs (Fig. 8.8). This is basically true for

[11] In order to avoid for correlation artefacts when concatenating data snippets, these have been separated by 20 time-bins of non-spiking activity on each channel.

[12] For learning networks, the parameters of the SSS were chosen as follows: decay constant $d = 0.15$ and shift constant $\Delta t = 5$. All parent configurations with up to 7 parents were evaluated exhaustively. The LAT has been determined from configurations with 7 parents. Learned networks were composed of all links in the 10 best scoring parent configurations of each node.

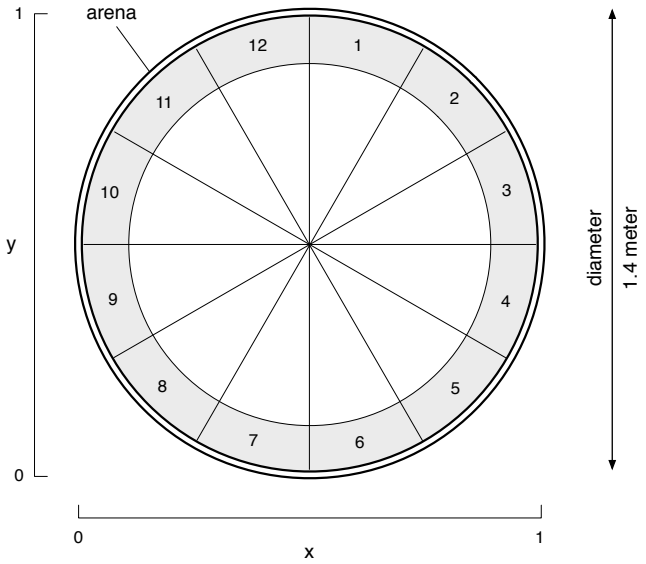


Figure 8.7: Top view on cylindrical arena separated into twelve equally sized sections (highlighted gray, labelled 1–12). The sections are defined by directly adjacent sectors ($\angle 30°$) of the arena that lie outside the white circle (with centre $x = 0.47$, $y = 0.5$, radius $r = 0.42$). In order to account for tracking artefacts, data points corresponding to positions outside the arena have been omitted. The sections are defined such that the position of the animal is in one of them most of the time; as will be shown later (Fig. 8.9), only few trajectories cover the arena central (white circle).

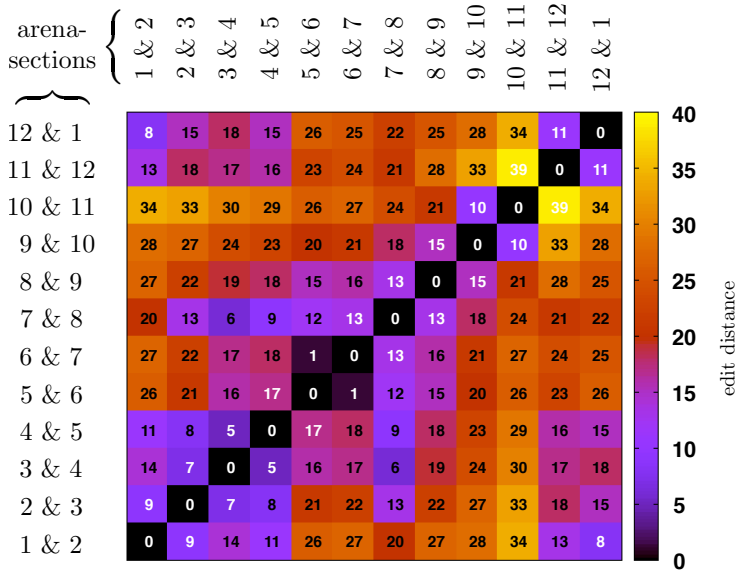


Figure 8.8: Edit distance of networks learned from hippocampal place cell data split in twelve overlapping data-sections. Arena-sections (Fig. 8.7) that define the data given next to rows and columns, edit distance (Section 8.1.2) for each pair within corresponding rectangle. All except one pair of networks (corresponding to data from sectors 10&11 and 11&12) that are expected to be similar (white font) show relatively low distances. Minor deviations can be explained by the various factors that might have influenced the dynamics of the place cells (Section 8.2.1).

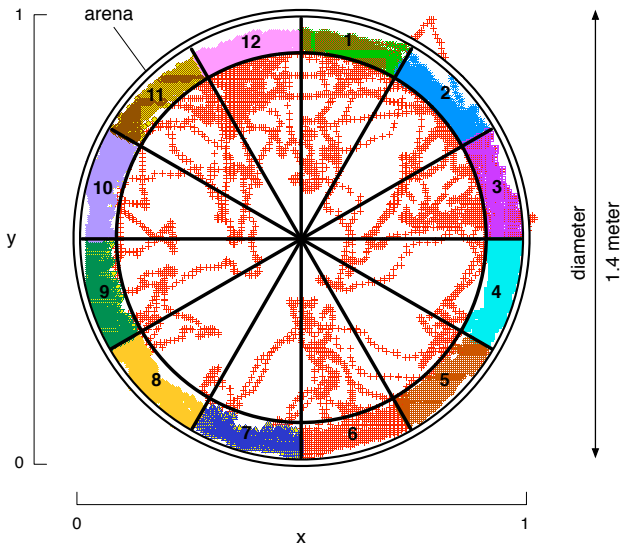


Figure 8.9: Top view on cylindrical arena with tracked locations of the rat during the whole recording (40 min). Locations corresponding to twelve sections (Fig. 8.7) indicated in different colours. As the dense accumulation of dots along the boundary of the arena shows, the animal generally avoids the open space in the middle. Short excursions away from the wall may be explained by chasing for food pellets, which have been regularly dropped (every 20 seconds) in a random location of the arena. However, behaviour is different for the inner sector corresponding to section 12 (and partly overlapping to sector 11): Here, numerous locations have been tracked further away from the arena-bound. The reasons for the animal's affinity to that particular location are unknown, as well as its behaviour in the place (e.g. chasing, eating, or grooming).

all but one data-pair: Unlike for data from other overlapping sections, networks corresponding to sections 10&11 and 11&12 show strong dissimilarities, which need to be explained. The animal's behaviour might provide a clue why networks differ strongly at these locations (Fig. 8.9): For most sectors the rat was predominantly located near the arena-wall; for sector 12 (and partly sector 11), however, the animal was observed more towards the middle. The place cell dynamics of these sections will therefore be influenced by movements towards and away from the centre of the arena. This contrasts remaining sections, where the animal's trajectories mainly followed the boundary of the arena. Further, direct inspection of the spike trains can explain the irregularities in the upper left quadrant: Analysing all twelve data-sets shows that each of them contains 277–823 spikes with firing rates (averaged over time spent in section, as well as all units) varying between 10–34Hz. Data from sections 11&12 only contains 277 spikes and it shows the lowest average firing rate; for sections 10&11 spike rates are minimally higher ($\approx$ 12Hz), but contain nearly three times as many

spikes (787 in total). The differences in spike-counts suggest that the network for sections 10&11 has been more influenced by the dynamics in section 10 (678 spikes) than by those in section 11 (109 spikes). Similarly, about 70% more spikes can be observed in section 12 (168 spikes) than in section 11; the inferred network for sections 11&12 might therefore represent dynamics in section 12 more than those in section 11. Together, although the two networks were inferred from data with overlap in section 11, they both might (over-)represent the dynamics from non-overlapping sections 10 and 12, for which similarity is not expected.

In conclusion, the observations from hippocampal data generally confirm two expectations: (1) Different dynamics result in unlike structures, and (2) similar dynamics also lead to comparable networks. The SSS thus assigns steady scores, such that the dissimilarity of results across data-sets is known to reflect differences in functional connectivity; these were secured in the construction of the data and learned networks reflect them correctly. Like for the retinal data (Section 8.1), the results from single-unit data give positive indications for the SSS's applicability to real data.

With the closing of this chapter, the investigation of the SSS comes to an end. The new score has been examined theoretically (Chapter 4) before assessing it in several simulations (Chapter 7). Finally, in this chapter, the SSS has been successfully tested on real data of two different kinds. In the next chapter, the results of these comprehensive studies are summarised and reviewed.

Chapter 9

Conclusions

This thesis contributes to the field of biological data analysis techniques using graphical models (Chapter 1). Existing methods were discussed (Chapter 2) and a new concept has been introduced: the Snap Shot Score (Chapter 3). The characteristics of the novel score were both illustrated in examples and investigated theoretically (Chapter 4). Thereafter, the SSS was related to the BD scores in order to further improve understanding of its potential strengths (Chapter 5). Considerations about how these can be validated and quantified resulted in a neural simulation framework (Chapter 6) with which the SSS was assessed (Chapter 7). Following the good performance for simulated data, the score was applied to real single- and multi-unit spike trains for which results are similarly encouraging (Chapter 8). Finally, in this chapter, key points of these comprehensive studies are summarised briefly.

9.1 Contributions of This Work

In this thesis two novel ideas have been presented: The SSS itself and the plausibility concept, which was applied for the assessment of the score. Throughout this work, the SSS has been investigated and simulations showed its good performance in a broad range of situations. Piloting applications of the score to real data were also positive, such that, despite its simple concept, the SSS seems suitable for the analysis of neural spike train data.

The new SSS has been compared to the BD scores with which neural relationships were successfully revealed earlier. In contrast to the SSS, the BD scores can be used to infer higher order DBNs, which can represent different time-lags precisely; spike train data, however, can be problematic for these scores and careful transformations might be needed in order to make the scores applicable to them. Additionally, inference of higher order models is computationally

far more challenging than that of 1st order models, whose demands correspond to that of the SSS; but the SSS can reveal excitatory relations over multiple time-lags. Altogether, the SSS does not substitute, but complements existing methods: It is a computationally highly efficient analysis technique to reveal information flow networks, which, by their causal excitatory semantics, characterise the data from a new perspective.

Besides the SSS, a new approach to assess neural network inference techniques under conditions of partial observability has been introduced — the plausibility concept. This novel idea makes a small but crucial contribution to the assessment framework, as it allows accounting for observational equivalence. Despite its simplicity, I am not aware of similar concepts being used for the systematic evaluation of methods for network learning; these have so far either been investigated under full observability (ignoring potential observational equivalence) or under partial observability by checking the quality of revealed networks manually. With the plausibility approach, techniques can now be assessed both more accurately and automatically; it allows for comprehensive neural simulation series, which can yield valuable insights that could hardly be gained before.

It is inherent to research that any resolved issue entails at least a dozen new questions; each of them requiring substantial work in order to find an answer. Results shown in this thesis are not any different. The following section therefore points to some open questions and interesting ideas for future research.

9.2 Directions for Future Research

As part of the characterisation of the SSS, it has been outlined how the score can be altered and generalised (Section 4.5). Suitable modifications might render the score applicable to other neural time-series data besides spike trains: e.g. from calcium imaging [Stosiek et al., 2003], fMRI [Jezzard et al., 2001, Matthews and Jezzard, 2004], or EEG [Nunez and Srinivasan, 2007]. Also, regardless of the data-source, the snapshot-concept might prove helpful for synchrony detection (by choosing an extremely small shift constant or even setting it to zero). Anyway, using the SSS for data analysis — whether as defined in this work or a modification of it — can only be beneficial if the semantics of learned networks are clear. The understanding of networks conveyed in this thesis might be greatly improved by explaining the SSS in a fully Bayesian approach. The simplistic neuron model suggested in section 3.1.1 could be a starting point for such work.

Future simulation-based work to assess network inference techniques will

involve large networks, many different topologies, and parameter series; corresponding results will need to be automatically evaluated with a minimum of human interaction. Therefore, the new approach to assess learned networks by their plausibility is a potential solution. This concept could, for instance, be refined by not only classifying links as plausible or not, but by gradually weighting them. For example, a link spanning over several time-lags could be a regarded as *less plausible* than a path composed of short links that explain the same dependency. Sophistications like this would make it possible to exactly specify the desired outcome of an analysis; whether results of the technique under question match the wanted ones can then be automatically determined.

Finally, the SSS should be applied in order to address biological questions. Learning networks in practice often involves heavy computations, but the waiting-time for these to finish may be highly appreciated in order to realise what learned networks mean in the specific context of the data. Once it has been understood how to query the results, the SSS serves as a powerful tool that can help to gain novel insights.

Appendices

Appendix A

Modelling and Comparing Neural Dynamics

Models of neural systems are simplified descriptions, which picture nature as idealised prototypes. With secondary aspects omitted, key features of the system they describe can be studied effectively. Often models can be used as generators of artifical data, such that extensive data sampling becomes possible in silico. This potential is utilised in the assessment framework for the SSS (Chapter 6) where spike train data is generated with neuron models, which are presented next. Additionally, the comparison of neural dynamics by using spike train metrics will be discussed later (Section A.2). Such metric is key to the alternative assessment framework presented in section 6.5.

A.1 Models of Neural Dynamics

Neurons have a diverse and complex morphology, which attributes to their wide range of dynamics [Yuste and Tank, 1996, Magee, 2000]. Due to this enormous diversity neuron models are characterised by balance of detail and abstraction, depending on which features are regarded essential. This can, for instance, be seen by the different spatial scales at which neural circuits are modelled: Starting from population models, which are used to describe the dynamics of a neural assembly without considering individual neurons, down to multi-compartment models of single cells. (See Herz et al. [2006] for a review.) While population models can replicate the dynamics of several brain regions they give no insights into the role each individual neuron plays. Vice versa, models which give a reasonable close-up description of a single cell are generally to complex to build large neural networks. It is thus not surprising that until now very few attempts

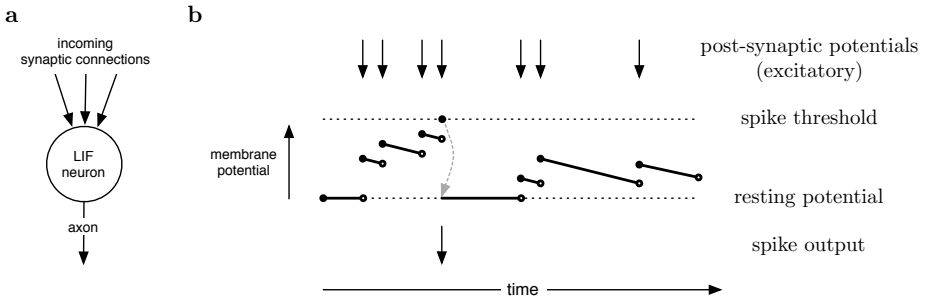


Figure A.1: Schematic representation of a point neuron model and its dynamics according to the LIF-model. **a** The model neuron receives synaptic inputs from connected neurons and outputs to one or more units. A point neuron model does not include a functional description of dendrites or axons but the cell body only. **b** LIF neuron receiving excitatory post-synaptic potentials, which rise its membrane potential. Once the neuron's spike threshold is reached it fires a spike and resets the membrane potential to resting state. At any time when the membrane potential is different from the resting potential, a leakage current causes a gradual convergence towards it. This leakage determines the time-span over which temporal summation of synaptic inputs occurs.

have been made in order to model large neural ensembles at close to reality, sub-cell level [Markram, 2006]. Regarding the needs of this thesis, single cell models, which can be used to generate adequate neural dynamics for small networks, are sufficient. Two important model types, which are suitable for this task, are presented next: the leaky integrate and fire model and the Hodgkin-Huxley model.

A.1.1 The Leaky Integrate and Fire Neuron Model

Probably the most prominent representative for single cell models is the *leaky integrate and fire (LIF)* model (e.g. [Stein, 1965] or [Dayan and Abbott, 2005, pp.162]). A LIF neuron is basically an accumulator of synaptic inputs, which shift its membrane potential, and whenever the potential reaches a threshold a spike is emitted. Additionally, the neuron is leaking in the sense that it tends to slowly approach its resting potential (Fig. A.1, Algorithm 2).

The LIF neuron model is often attributed to Lapicque's work back in 1907 [Lapicque, 1907] (e.g. [Abbott, 1999, Herz et al., 2006, Richardson, 2007]), but see Brunel and van Rossum [2007] for a recent clarification. Although the model is a rather phenomenological description, which has been overcome by insights from work done by Hodgkin and Huxley [1952], LIF models are still popular today as ongoing work on variants of the model show: for example, the quadratic (QIF) [Ermentrout and Kopell, 1986], exponential (EIF) [Fourcaud-Trocme et al., 2003], or non-linear integrate and fire models [Richardson, 2007].

Algorithm 2 Simulation of a leaky integrate and fire (LIF) neuron model. The membrane potential of each neuron is updated successively based on incoming potentials and its leakage. For simplicity the leakage function $leakage(potential)$ can be chosen as a constant.

loop
  for all neurons **do**
    determine new membrane potential:

$$potential(t+1) := potential(t) + \sum synaptic\ inputs - leakage\,(potential\,(t))$$

    if $(potential(t+1) \geq spike\ threshold)$ **then**
      emit spike and
      set $potential(t+1) := resting\ potential$
    end if
  end for
end loop

The striking simplicity of the LIF model leads to low computational costs of neural simulations using these models [Herz et al., 2006], which makes them suitable even for large network simulations [Vogels et al., 2005]. Unfortunately, they may fail to explain certain phenomena, which can only be described by more detailed models [Traub et al., 2005]. But large scale network simulations at a high level of detail require enormous computational resources [Markram, 2006], which may not be accessible. Additionally, apart from computational aspects, determining adequate parameter settings for complex models can pose a major problem.

As already mentioned above, Hodgkin and Huxley [1952] presented groundbreaking work in which they suggest a neuron model that is far more precise than the LIF model. Although their model has not been used for simulations presented in this thesis, the model is briefly introduced next; afterwards, it will be explained why preference has been given to the simpler LIF model.

A.1.2 The Hodgkin-Huxley Model

In 1952, Hodgkin and Huxley presented their work on the squid giant axon [Hodgkin and Huxley, 1952], which included a model that gave a detailed explanation of the mechanisms involved in generation of an action potential. Partly for this important element of their work, in 1963, they have been awarded the Nobel Prize *"for their discoveries concerning the ionic mechanisms involved in excitation and inhibition in the peripheral and central portions of the nerve cell*

membrane".[1]

The model proposed by Hodgkin and Huxley (nowadays known as *HH-model*) describes the dynamics of the membrane potential in a squid's giant axon. Changes in voltage across the membrane are due to in- and out-flux of different ions (Sodium, Potassium, Chloride, etc.). The HH-model not only accounts for the actual membrane conductance for the major ions involved, but it even includes the corresponding causes: the probabilistic opening and closing of gates of particular ion channels, which are described jointly for a large number of channels. These details can be found in the literature ([Hodgkin and Huxley, 1952] or [Dayan and Abbott, 2005, pp.173], for example), but in its essence the HH-model reads as:

$$\frac{dV}{dt} = -\frac{1}{C}\left[\underbrace{g_{\mathrm{Na}}(V,t)\cdot(V-E_{\mathrm{Na}})}_{\text{Sodium current}} + \underbrace{g_{\mathrm{K}}(V,t)\cdot(V-E_{\mathrm{K}})}_{\text{Potassium current}} + \underbrace{g_{\mathrm{l}}\cdot(V-E_{\mathrm{l}})}_{\text{leakage current}}\right] . \tag{A.1}$$

Changes in membrane potential V are dependent on the capacitance of the membrane C, as well as a current composed of three components: Sodium, Potassium, and a leakage current. These currents are determined by two parts, which are the conductances g_{Na}, g_{K}, and g_{l} for the particular ions and their corresponding driving force, i.e. the difference between the current membrane potential V and the equilibrium potentials E_{Na}, E_{K}, and E_{l}. Note that the leakage conductance g_{l} is a constant while the other two, g_{Na} and g_{K}, are functions of both voltage and time. In the original publication [Hodgkin and Huxley, 1952] the conductance functions are characterised by additional differential equations, which describe the large scale effects of the probabilistically opening and closing gates of involved ion channels.

Compared to the phenomenological description a LIF model gives about a spike generating neuron, the HH-model is obviously much more sophisticated. By accounting for many details it can for instance explain the refractory period, which needs to be explicitely added to a LIF model, if desired. However, the HH-model shows that the LIF model is not fundamentally wrong in the first place. And of course the level of detail in the HH-model comes with a computational cost when simulating it. This is due to the numerical integration of the differential equations, which must be updated in two domains: voltage and time. Simulations with the HH-model are therefore generally restricted to relatively small neural ensembles. The output of such simulation are voltage traces of each neuron's membrane potential.

The techniques discussed in this thesis cannot be directly applied to volt-

[1] Quotation from the official web site of the Nobel Foundation:
`http://nobelprize.org/nobel_prizes/medicine/laureates/1963/index.html`

age traces, but to discrete data only, like spike trains. In order to apply these methods, the voltage trace data has to be converted into spike trains by discretising time and choosing a voltage-threshold, such that each neuron's activity can be classified as spiking or non-spiking depending on whether the threshold is exceeded or not. Obviously, all sub-spike-threshold dynamics are lost by this conversion; the main benefit of the HH-model — its adequate description of exactly these non-spiking dynamics — would be lost. Using the model anyway would result in high computational costs, which is problematic for the large numbers of simulations that had to be performed for the presented work (Chapter 7). Except for prototyping, the HH-model has thus not been utilised. Instead, a LIF model has been chosen as a manageable and satisfying compromise of complexity and computational costs. Data generated with this model is sufficient for the assessments framework, which requires reasonable spike trains of multiple correlated units.

In section 6.5 an alternative assessment framework for neural network inference techniques is presented. One of its components uses spike train metrics and the following section is a brief primer on this concept.

A.2 Spike Train Metrics

Two spike trains can differ in a variety of ways, for example by their maximum, average, and minimum spike rates or by their spike patterns, such as bursting or regular spiking. But even two similar spike trains, both showing regular spiking at the same rate, will most likely differ in the exact number and timing of spikes. Identifying such subtle differences between two spike trains is easily done by a cheap comparison between the two; however, the resulting list of differences is likely to be long and more confusing than helpful. In contrast, if the difference between the spike trains was summarised by a number that actually reflects their (dis)similarity it would be much easier process and understand large volumes of data. Such measure of alikeness is provided by *spike train metrics* [Victor, 2005].

Spike train metrics measure the distance between two spike trains by transforming one into the other (Fig. A.2). Therefore, individual spikes can be inserted or deleted, as well as shifted in time. Each of these operations has a particular cost; commonly the costs are 1 for the insertion and deletion of a spike. The costs for shifting a spike are proportional (by a constant factor q) to the time difference of the shift. Generally a large number of possible combinations to transform one spike train into the other exist; the cheapest of these conversions defines the actual distance as its total cost. Very different spike

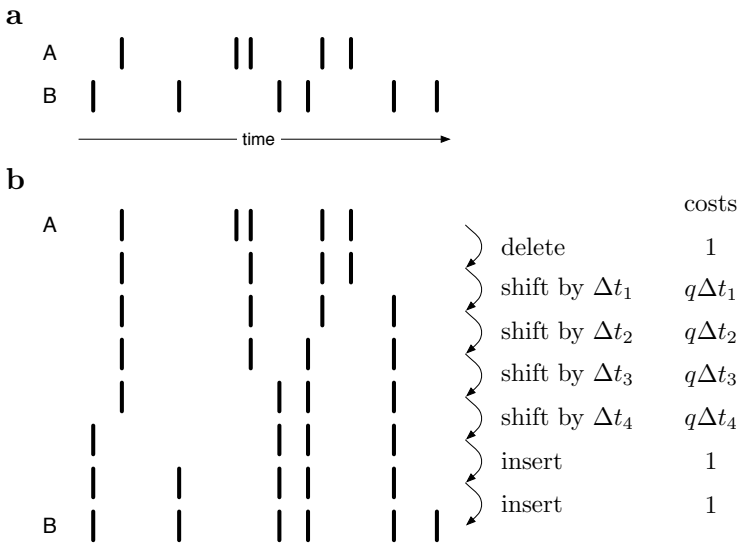


Figure A.2: Adopted from Victor [2005] and extended: Working principle of a spike train metric illustrated schematically. **a** Two different spike trains A and B. **b** Transformation of spike train A to spike train B with 7 elementary steps. The total cost of the transformation is the sum of the conversion steps involved. The distance between A and B is the cost of the cheapest possible transformation.

trains thus have large distances whereas identical spike trains have distance zero.

The basic idea of spike train metrics sounds relatively simple, but it can actually be quite complicated to calculate them. This is due to the large number of possibilities to combine operations that convert one spike train into the other. Evaluating all transformation in order to find the cheapest one can be intractable for computational reasons, especially for long spike trains with many spikes. Sequence alignments in genetics face a similar problem: In order to derive evolutionary distances DNA sequences are analysed with respect to their similarity [Eddy, 2004]. These sequences are defined over an alphabet of 4 letters, which correspond to the different nucleotides.[2] A successful mechanism for their distance evaluation could thus be adopted to spike trains, which only consist of 2 letters corresponding to spiking- and non-spiking events. Indeed, an efficient algorithmic implementation has been proposed [Sellers, 1974] and adopted to evaluate spike train metrics [Victor and Purpura, 1996]: Let $S_a = (a_1, \ldots, a_n)$ and $S_b = (b_1, \ldots, b_m)$ denote the spike times of two spike trains with n and m spikes respectively. Their spike time distance $G_{n,m}$ is then

[2]The four nucleotides adenine, guanine, cytosine and thymine are commonly abbreviated with letters A, G, C, and T respectively [Mount, 2004].

given by the by the recursion

$$G_{i,j} = \min \left\{ \begin{array}{lll} \underbrace{G_{i-1,j} + 1}_{\text{costs if spike } a_i \text{ deleted}}, & \underbrace{G_{i,j-1} + 1}_{\text{costs if spike inserted to match } b_j}, & \underbrace{G_{i-1,j-1} + q|a_i - b_j|}_{\text{costs if spikes } a_i \text{ and } b_j \text{ shifted to match}} \end{array} \right\} , \qquad \text{(A.2)}$$

$$\text{where} \quad G_{i,j} = 0 \quad \text{if} \quad \min\{i,j\} \leq 0 \; .$$

This equation shows the role of the cost parameter q for shifting spikes in time: It balances the sensitivity between spike-count and relative timing; shifting a spike by $1/q$ is as costly as inserting or deleting one spike [Victor, 2005].

So far discussed were spike *time* metrics, which use the absolute time-points of each spike train; another type of metric, so called spike *interval* metrics, considers the times in between spikes [Victor and Purpura, 1996, 1997]. In detail, given the spike time-series $S_a = (a_1, \ldots, a_n)$ and $S_b = (b_1, \ldots, b_m)$ of two spike trains, interval metrics do insert, remove, or shift spikes; however, this is not done in order to match time-points of both series. Instead such metric considers series that are derived from the spike times: their inter-spike intervals (ISI), i.e.

$$\begin{aligned} ISI_a &= (a_2 - a_1, a_3 - a_2, \ldots, a_n - a_{n-1}), \quad \text{and} \\ ISI_b &= (b_2 - b_1, b_3 - b_2, \ldots, b_m - b_{m-1}) \; . \end{aligned} \qquad \text{(A.3)}$$

Spikes are added, deleted, or moved at a particular cost in order to change the corresponding inter-spike intervals, which are ultimately matched to each other. The algorithmic implementation of the distance calculation is analogous to that of spike-time metrics [Victor, 2005].

Each of the two types of metrics has advantages in particular situations. For example, spike time metrics can be used in order to analyse data collected during repeated stimulus presentations. Aligning spike trains with the stimulus facilitates to measure absolute time response variability between different instances of the stimulus. In situations where spike trains cannot be precisely aligned, these metrics could result in high distances due to relative shifts of spike trains. Under these circumstances spike interval metrics can be used in order to compensate for the misalignments.

Appendix B

Graphs and Their Number

Networks play a central role in this thesis; therefore, this chapter gives a brief introduction to the mathematical concept of graphs, which can be used to formally represent networks. For practical application of network inference techniques, it is important to know the number of potential networks in order to decide on appropriate methods for learning them. Therefore, classes of graphs that are related to techniques discussed in this thesis will be analysed with respect to their size and characteristics of their elements.

B.1 Graphs

Networks can be found in a variety of domains and are increasingly becoming subject to investigations [Albert and Barabasi, 2002, Bornholdt and Schuster, 2003, Sporns et al., 2004, Borgatti et al., 2009, Lazer et al., 2009, Hidalgo et al., 2009, Bullmore and Sporns, 2009]. Mathematical *graph theory* provides some of the essential framework in order to formalise and perform these studies: An abstract, non-visual notion of networks, which will be introduced here. As graph theory is a field on its own [Bollobs, 1998, Diestel, 2006], the following definitions and comments are restricted to network types discussed in this thesis; they thus do not cover undirected graphs or multi-graphs (where multiple links between any two nodes can exist) in particular. The focus is instead on simple directed graphs according to the following

Definition 7 (Graph) *A graph is a pair of two sets* $G = (V, E)$*. The elements of* $V = \{v_1, \ldots, v_n\}$ *are called vertices and those of* $E = \{e_1, \ldots, e_m\}$ *are called edges. Each edge is a tuple of two vertices* $e_i = (v_a, v_b)$ *and represents a connection from vertex* v_a *to* v_b*; sometimes denoted as* $v_a \rightarrow v_b$*.*

Common terminology calls the starting node of a link a *parent* of the destination node (*child*). The *family* of a node comprises the node itself together with its parents.

Similar to particular node-sub-groups, connections between different nodes or those linking a node with itself have specific names: A link connecting a node with itself is called a *loop*. Multiple links can be combined in order to form a *path*, which is a series of (directed) links in which the origin of all links equals their predecessors' destination. (Single links are considered paths themselves.) The *length* of a path is defined as the number of chained links. Just as a loop links a node to itself it is also possible that a path starts and ends at the same node. Such paths and loops are both called *cycles*. Bayesian networks (Section 1.2.1) are associated with graphs in which no cycle exists. These graphs are therefore called *directed acyclic graphs* (*DAG*s).

The abstract description of a graph as two sets, vertices and edges (Def. 7), can be easily represented as a network and vice versa: Each vertex is represented by a node and directed links connect nodes according to edges in the graph. Another way to represent graphs uses a matrix and is given in the following

Definition 8 *Let $G = (V, E)$ be a graph with n vertices $V = \{v_1, \ldots, v_n\}$. The adjacency matrix $A = (a_{ij})_{i,j\in\{1,\ldots,n\}}$ of the graph is defined by*

$$a_{ij} = \begin{cases} 1, & \exists\ e = (v_i, v_j) \in E,\ \textit{i.e. a link } v_i \to v_j \textit{ exists}, \\ 0, & \textit{otherwise}\ . \end{cases} \tag{B.1}$$

In order to simplify notation, nodes v_i are often identified by their index number i. The adjacency matrix can then be read as: The parents of node j are those rows i where $a_{ij} = 1$. Note that the adjacency matrix fully specifies the size and connectivity of a graph. And vice versa, every matrix with entries $\in \{0, 1\}$ defines a graph. There is thus a one-to-one relationship between graphs and matrices whose usefulness will become evident in the following section. Another aspect of adjacency matrices will be utilised for argumentation as well: Re-ordering nodes v_i with respect to their index i shuffles rows and columns in the adjacency matrix. This does not affect the structure of the graph, but only the position at which a link is indicated in the matrix. By sorting nodes appropriately the adjacency matrix can often be formatted such that essential features of the graph become clearly visible.

The basic formalism in order to describe graphs will be used to characterise and count structures that belong to a particular class. Classes of graphs that will be discussed are associated with the different kinds of graphical models presented in this thesis. Such knowledge is needed for practical runtime estimates. It

further helps to identify model-specific optimisation potential of methods used for learning networks.

B.2 Classes of Graphs and Their Number

The graphical models discussed in this thesis differ with respect to their particular approach to modelling stochastic dependencies. For example, static BNs encode the factorisation of a probability distribution that describes *instantaneous* interactions between variables; dynamic BNs represent time-lagged correlation. Conclusively, even if the same variables are represented in either model, the BN and the DBN differ substantially: First, the number of network nodes differs as each variable is represented by one node in the BN, but multiple times in the DBN — once per time-layer (e.g. Fig. 1.3bc on page 15). Further, in a BN nodes can be connected in any way, as long as the resulting graph remains acyclic; in DBNs only links from nodes in previous time-layer to the ones in the current time-layer are allowed. Each model type imposes different restrictions on the graph's structure. Hence, each model can be associated with a distinct class of graphs, i.e. the set of all structures that are consistent with the model's restrictions. Apart from theoretical curiosity, these classes are important in practical applications. The high dimensionality of realistic data causes high computational demands of network inference, which should not be increased unnecessarily by evaluations of inconsistent network structures, for example. In order to ensure that only sound structures are considered the characteristics of potential graphs must be known. It is also helpful to know their number, which can be used for run-time estimates of the analysis. These issues are therefore addressed in the following sections in which classes related to different model types are discussed separately.

B.2.1 The Snap Shot Score's Graph-Classes

The SSS value of a parent configuration reflects the degree to which high activity of parent nodes is followed by spikes of the child; child spikes that occur within the lag-window of the score (Section 3.1.1). Correlated activity between the parents and the child is thus required to be time-lagged, such that past activity of the parents can be interpreted as determining the present activity of the child [within the limits imposed by observational equivalence (Section 1.4.2)]. The time-lag of the child's response to the parent(s) in not precisely known, but it can only be specified to be within the score's lag-window, which generally

comprises multiple time-lags (Section 5.1).[1] Parents of a node are represented as entities from within a common past — ranging back according to the lag-window. Graphs associated with the SSS thus represent each node twice: once in the past and once in the present. For each of n nodes in the present there are n potential parents from the past, which can be combined to 2^n different parent configuration. The parent configurations of different nodes can be combined in any way, such that there are $2^{n \cdot n}$ possible graphs. Inspecting the adjacency matrices associated with the graphs shows why these numbers are correct:

> The set of vertices V comprises $2 \cdot n$ elements since any of the n variables is represented twice: in the past and in the present. Any link between these nodes which is directed forward in time is valid; such would correspond to an entry 1 in the adjacency matrix, which has $2 \cdot n$ rows and columns. Out of its $4 \cdot n^2$ elements only n^2 correspond to appropriate links; links within either time-layer or those directed backwards in time cannot take the value 1. The vertices of the graph can thus be re-ordered such that its adjacency matrix A has the following form:
>
> $$A = \begin{pmatrix} 0 & 0 \\ A^* & 0 \end{pmatrix} \in \mathbb{M}\left[2n \cdot \times 2n, \{0,1\}\right] \ , \quad \text{where } A^* \in \mathbb{M}\left[n \times n, \{0,1\}\right] \ . \tag{B.2}$$
>
> Hence, since any combination of parents is valid for each node and because each node's configuration can be combined with that of all other nodes, the n^2 entries in the matrix A^* can take any combination of values $\{0,1\}$. Simple combinatorics then shows that $[\#\{0,1\}]^{n^2}$ different combinations exist, each of which corresponds to a different graph. The number of parent configurations of any node is the number of combinations that can be found in its particular column in the adjacency matrix. There are n elements in the column vector which can be non-zero in any combination; hence, $[\#\{0,1\}]^n$ vectors corresponding to valid parent configurations exist.

The size of the graph-class can be used to visualise its enormous growth, which we find to be [Bronstein et al., 1999, pp.373]:

$$\frac{d}{dn} 2^{n^2} = \ln(2) \cdot 2n \cdot 2^{n^2} \ . \tag{B.3}$$

[1] Alternatively, parents can be understood to be in the present time-layer from which they determine the child's future. In this equivalent view the precise timing of the parents is known whereas the child's response can vary within the SSS's lag-window.

Table B.1: Graph-class sizes summary for the SSS. (Number of variables n, `NA` not applicable.)

class description	**number of parent configurations**	**number of graphs**
all possible	2^n	2^{n^2}
excluding self-excitation	2^{n-1}	$2^{n(n-1)}$
$\leq k$ links	`NA`	$\sum_{i=0,\dots,k} \binom{n^2}{i}$
$\leq k$ links, no self-excitation	`NA`	$\sum_{i=0,\dots,k} \binom{n^2-n}{i}$
$\leq p_{max}$ parents per node	$\sum_{i=0,\dots,p_{max}} \binom{n}{i}$	$\left[\sum_{i=0,\dots,p_{max}} \binom{n}{i}\right]^n$
$\leq p_{max}$ parents per node, no self-excitation	$\sum_{i=0,\dots,p_{max}} \binom{n-1}{i}$	$\left[\sum_{i=0,\dots,p_{max}} \binom{n-1}{i}\right]^{n-1}$

The immense size and rapid growth of the graph-class calls for sensible restrictions in practical application. Such might be the exclusion of self-exciting configurations (where a node is its own parent). This imposes a restriction on the adjacency matrix A^* (whose diagonal elements must be zero in this case) and leads to a reduction in potential parent configurations and networks. Analogous to the combinatorial reasoning outlined in detail earlier the resulting numbers were derived; these are summarised in Table B.1.

Further restrictions on graph structure could be a limit on the maximal number total links or parents per node, for example. Such limitations might be reasonable when relations in the studied system are known to be sparse. However, they might also be purely enforced by computational constraints affecting high-dimensional data-analysis; limiting the number of links can then be used in order to reduce the number of potential networks (Table B.1).

As the next section will show, graph-class sizes of the SSS are equal to that of 1$^{\text{st}}$ order DBNs. But, for higher orders the class of DBNs is much larger.

B.2.2 Graph-Classes of Dynamic Bayesian Networks

Links in DBNs describe time-lagged influences of variables on each other. Thereby the consensus is that causes must precede effects; links are directed forward in time such that they end in the time-layer of the presence (Fig. 1.3bc on page 15). Variables can be cause and effect at the same time; effects can even be their

own cause. There are no restrictions on which links can occur jointly in the network. The special structure of these graphs can be recognised if the vertices are ordered such that nodes in the present time-layer have lowest indices; followed by nodes in the previous past layer, and so forth:

> Consider a m^{th}-order DBN with n variables. The corresponding graph has $n \cdot (1+m)$ vertices $V = \{v_1, \ldots, v_{n\cdot(1+m)}\}$, which can be ordered such that
>
> $$\text{vertex } v_i \text{ is in time-layer } t - \left\lfloor \frac{i-1}{n} \right\rfloor . \tag{B.4}$$
>
> In other words, the first n vertices $v_1, \ldots, v_n$ correspond to the present time-layer t; the next n vertices $v_{n+1}, \ldots, v_{2n}$ are in layer $t-1$; and so forth. Using this ordering, the structure of the adjacency matrix A can be deduced from the characteristics of the graph: All links start in past time-layers and terminate in the present time-layer t only. Since no other links are possible, all elements in A corresponding to links starting in layer t or ending in a layer different than t must be zero. These are the first n rows and all except the first n columns; the adjacency matrix A therefore has the following form:
>
> $$A = \begin{pmatrix} 0 & 0 \\ A^* & 0 \end{pmatrix} \in \mathbb{M}\left[n \cdot (1+m) \times n \cdot (1+m), \{0,1\}\right] , \quad \text{where } A^* \in \mathbb{M}\left[n \cdot m \times n, \{0,1\}\right] . \tag{B.5}$$
>
> The adjacency matrix A of a DBN is fully determined by the rectangular sub-matrix A^*. All elements of A^* can take two values $\{0,1\}$ independent of each other such that the number of different DBNs corresponds to the number of possible matrices A^*.

This analysis yields a total of $2^{m\cdot n^2}$ different graph structures (i.e. $2^{m\cdot n}$ parent configuration per node). In comparison, there are at least as many DBNs as graphs in the SSS's class. The same is true for the growth of this class with respect to the number of nodes n; this is given by [Bronstein et al., 1999, pp.373]:

$$\frac{d}{dn} 2^{m\cdot n^2} = \ln(2) \cdot 2nm \cdot 2^{n^2} \tag{B.6}$$

and equals that of the SSS [equation (B.3)] for order $m = 1$. For higher orders, however, the number of DBNs grows much faster, such that DBNs clearly outnumber graphs of the SSS. This is due to the fact that DBNs, in contrast to the SSS, can capture the exact time-lag of a parent to its child (Section 5.1).

Table B.2: Graph-class sizes associated with DBNs. (Number of variables n, model order m, `NA` not applicable.)

class description	**number of parent configurations**	**number of graphs**
all DBNs	$2^{m \cdot n}$	$2^{m \cdot n^2}$
$\leq k$ links	`NA`	$\sum_{i=0}^{\min(k, mn^2)} \binom{mn^2}{i}$
$\leq p_{max}$ parents per node	$\sum_{i=0}^{\min(p_{max}, mn)} \binom{mn}{i}$	$\left\{\sum_{i=0}^{\min(p_{max}, mn)} \binom{mn}{i}\right\}^n$

As the equation above already shows, this precision comes at the cost of many potential networks. Investigating the dependence of the number of DBNs with respect to the order m,

$$\frac{d}{dm} 2^{m \cdot n^2} = \ln(2) \cdot n^2 \cdot 2^{m \cdot n^2} \ , \tag{B.7}$$

we find even larger growth although the number of variables n stays unchanged. The comparison to the class of the SSS shows that, even for moderate numbers of nodes and small orders, the number of DBNs can be extremely large. In practical application it is therefore common to restrict the graphs by the number of total links or by a maximum number of parents per node. The sizes of these classes are summarised in table B.2.

Graph classes for the SSS and DBNs could be easily characterised by analyses of their adjacency matrices. These could then be used with combinatorial arguments in order to determine the size of the class. The next section illustrates that such simple solution is not always possible: Counting static BNs is significantly more complicated.

B.2.3 Graph-Class of Bayesian Networks

Non-dynamic or static BNs have been introduced as a simple illustration of probabilistic graphical models (Section 1.2.1). BNs do further get used by the BD scores: When learning DBNs with these scores they are implicitly converted to BNs (Section 2.2.6). Here their associated graph-class is discussed not only for completeness reasons, but also because BNs give a good example of how a relatively simple restrictions on the structure of a graph can render counting its elements highly complicated.

As a BN represents a factorisation of a probability distribution its graph

structure will always be acyclic;[2] i.e. it is a directed acyclic graph (DAG) (Section B.1). In order to determine whether a given graph is acyclic or not it is not sufficient to determine properties of each node's parents separately. Instead, the full network, i.e. its entire adjacency matrix, must be analysed at once to ensure acyclicity. This is because a cycle can be up to $n-1$ links long, when each node is visited once. Inspecting parent configurations can only reveal loops, but no cycles involving 2 or more links. In order to identify the latter all possible paths in the graph must be analysed for which the full adjacency matrix needs to be considered. As a consequence of this, the class of DAGs contrasts those of the SSS and DBNs for which the number of each nodes' parent configurations was sufficient to derive the number of graphs; DAGs cannot be counted in a simple way. However, in 1973, Robinson [1973] and Stanley [2006] independently derived the corresponding formula: The number of DAGs a_n with n vertices is given by the recursion

$$a_0 = 1, \quad a_n = \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} 2^{k(n-k)} a_{n-k} \quad \text{for} \quad n \in \mathbb{N} \,. \tag{B.8}$$

This formula does not convey an intuitive understanding of the actual number of DAGs. However, theoretical investigations about its asymptotic behaviour are easier to comprehend [Bender et al., 1986, Bender and Robinson, 1988]:

$$a_n \sim n! \frac{2^{\binom{n}{2}}}{M p^n} \quad \text{for} \quad p \approx 1.488\ldots, M \approx 0.474\ldots \,. \tag{B.9}$$

The value of the fraction increases rapidly, as can be seen by re-writing the numerator to $2^{2^{-1}(n^2-n)}$: This super-exponential growth exceeds the exponential growth of the denominator. In total the number of DAGs is thus quickly growing with respect to the number of nodes. The complications involved in counting DAGs give an idea of the complexity to number graphs that arise from constraints additional to acyclicity, e.g. a maximum on the number of links or parents per node. Corresponding results can probably be found in the mathematical literature, but no attempt is made here in order to summarise them.

Additional to the size of the DAG-class, it would be interesting to understand the structure of its elements. As already mentioned, a simple and specific characterisation is not possible; however, the adjacency matrix of any DAG has interesting features, which at least facilitate a glimpse on the structure of this class. This is outlined in the following sections in which two approaches

[2] This follows directly from the factorisation via the chain rule in equation (1.10) (page 11) by which dependencies are never dispersed in a cyclic fashion.

to identify an acyclic graph are presented. The considerations have practical importance because — as motivated in the introduction to section B.2 — networks whose structure is inconsistent with the model should not be scored. For BNs this means that graphs need to be acyclic, which can be ensured with the following two concepts.

Acyclicity-Check Using Node-re-ordering

In general it is not possible to determine whether a graph is cyclic or not by checking whether its adjacency matrix has a particular form. In contrary, it is possible to tell that certain adjacency matrices correspond to acyclic graphs. These matrices are strict triangular, i.e. all non-zero elements can be found on one side of the diagonal. For example, an strict upper triangular matrix has the following form

$$A = \begin{pmatrix} 0 & * & * \\ 0 & \ddots & * \\ 0 & 0 & 0 \end{pmatrix} \tag{B.10}$$

where each $*$ indicates an arbitrary value. Any adjacency matrix of this form can be verified to describe an acyclic graph by applying its definition. But DAGs can have non-triangular adjacency matrices as well; however, re-ordering nodes can always yield a triangular matrix. In contrary, for cyclic graphs no arrangement of nodes exists, which yields a triangular matrix.

It is questionable whether this concept of node re-ordering can yield an efficient implementation of an acyclicity test; however, note that the form of the adjacency matrix (B.10) implies that there must be at least one node without any links starting from it: A's last row has zero entries only. Likewise, A's first column implies that there must be at least one node without any parents. These are two necessary conditions for acyclicity, which can serve as a computationally cheap preliminary cyclicity check.

Another, more practicable approach to test the acyclicity of a graph is presented next. Instead of re-ordering nodes this approach uses matrix multiplication and can be easily be implemented. The outlined procedure is known as the *Floyd-Warshall algorithm* ([Floyd, 1962, Warshall, 1962] or [Cormen et al., 2001, pp.629]).

Acyclicity-Check Using Matrix-Multiplication

This section discusses the theoretical foundation of the Floyd-Warshall algorithm [Warshall, 1962]. Its implementation according to a dynamic program-

ming approach can be found in the literature (e.g. [Floyd, 1962] or [Cormen et al., 2001, pp.629]). Warshall's essential idea lies in the understanding of the adjacency matrix: It describes direct connections between nodes, i.e. paths of length 1. He then showed that, if a boolean product[3] of matrices is used, the powers of the adjacency matrix A^r express the existence or non-existence of a path of length r between any two nodes. Floyd utilised this result to formulate an algorithm, which determines the shortest path between any two nodes [Floyd, 1962]. Here, the interest is only on whether a path from a node to itself exists at all, which would mean that the graph was cyclic. (Otherwise, if no cycle exists for any node, the graph is acyclic.) A DAG thus fulfils the necessary and sufficient condition:

$$\forall\, r \in \{1, \ldots, n-1\} : \; \mathrm{diag}(A^r) \overset{A^r=\left(a_{i,j}^{(r)}\right)}{=} \left(a_{1,1}^{(r)}, \ldots, a_{n,n}^{(r)}\right) \overset{!}{=} (0, \ldots, 0) \;, \quad \text{(B.11)}$$

which simply states that for none of the nodes a cycle of any length r exists.[4]

This completes the discussion of different graph-classes, and a problem that is shared between all graphical models is considered next: The joint representation of multiple networks as one. This is an important practical aspect of network inference, which can result in a whole set of networks rather than a single structure that is superior to all others. Different kinds of equivalence have been discussed (Section 1.4.2), which can cause ambiguities and hence multiple solutions. But such can also arise volitionally when sampling methods are used for network learning (Appendix D). Regardless of the reason for multiple solutions, situations in which too many results exist in order to inspect them separately require adequate techniques to understand them. This is why the following section introduces suitable methods to compact results. Such post-processing methods might reveal commonalities between recovered networks, for example, and format shared features such that they can be perceived.

[3] In ordinary matrix multiplication, elements are multiplied and added. The boolean product of two binary matrices $A = (a_{i,j})$ and $B = (b_{i,j})$ uses a logical-and $\wedge$ and logical-or $\vee$ instead, i.e. $(AB)_{i,j} = \bigvee_{k=1}^{n} a_{i,k} \wedge b_{k,j}$. The resulting matrix is hence a binary matrix as well.

[4] Note that paths of length n or greater must be cycles; in such cyclic path, at least one node is visited (at least) twice. The full cycle must thus contain a sub-cycle of length smaller than n, which causes a violation of condition (B.11).

B.3 Compact Representation of Graphical Models

This section is concerned with techniques that merge multiple networks to one. It is assumed that a list of networks is given, all of which are defined over the same set of nodes; this list is denoted by $L = (G_1, \ldots, G_m)$. Two different approaches to join the given networks will be introduced: averaging and comparison. Averaging is based on probability theory while comparison is a purely graphical technique. Both methods are complementary and can help to gain an overview about shared characteristics and the level of diversity of processed networks.

B.3.1 Model Averaging

Two model averaging methods are discussed in this section: a bootstrap approach [Pe'er, 2005, Friedman et al., 1999a] and Bayesian model averaging [Friedman and Koller, 2003]. These concepts are not restricted to network averaging, but they can be used in order to determine the probability of any associated feature. In the simplest case the feature would be the existence of a specific link; determining the probability of all potential links yields the average network. But features can be more complex, like the existence of a cycle. In the following the feature of interest will be denoted by f, which is associated with an indicator function $f(\,\cdot\,)$. When this function is applied to a graph G it either takes value 1 or 0, depending on whether the feature f is present in the graph or not:

$$f(G) = \begin{cases} 1, & G \text{ has feature } f\ , \\ 0, & G \text{ does not have feature } f\ . \end{cases} \tag{B.12}$$

The following two approaches can help to deduce the probability of the feature $P(f|L)$ for the given list of networks L.

Bootstrap Averaging

Confronted with a list of networks L and the question whether a particular feature f is common to them or not, the straightforward approach to prepare an answer to that question would be to determine the relative frequency of the feature. This is simply because the rate of the feature's occurrence reflects our *confidence* in its commonness. I.e. the higher the value

$$\text{confidence}(f|L) = \frac{1}{m}\sum_{i=1}^{m} f(G_i) \ \in [0,1] \tag{B.13}$$

the higher the probability $P(f|L)$ that is assigned to the feature. The reason to distinguish between the confidence and the actual probability assigned to f is that, when the list of networks is finite, the bootstrap approach (B.13) can yield a biased reflection of the feature's actual commonness. An aware applicant of that method might thus decide to base the probability assignment not only on the confidence, but also on other background knowledge in order to correct for a (potential) sampling bias (Section 1.1). However, simulation experiments have shown that features with high confidence values are rarely false positives [Jones and Hobert, 2001, Pe'er, 2005, 2003]. The good performance of this simple approach can be explained by its relatedness to Bayesian model averaging, which is discussed next.

Bayesian Model Averaging

The discussed bootstrap approach determines the confidence in a feature f from the given list of networks L, only. Bayesian model averaging takes additional information into account in order to assess the commonness of the feature: the probability of each network. This leads to a weighted relative frequency in which un-probable networks contribute less to the *belief* in the feature than highly probable ones. Assuming that some data D are available in order to assign conditional probabilities $P(G|D)$ to each network G, the belief in a feature f can be formally expressed as

$$\text{belief}(f|L,D) = \; c \cdot \sum_{i=1}^{m} P(G_i|D) f(G_i) \tag{B.14}$$

with normalising constant $c = \left[\sum_{i=1}^{m} P(G_i|D)\right]^{-1}$. Like the confidence, the belief in f reaches its maximum value 1 only if the feature is present in all graphs G_i of the list L; if some graphs lack the feature, the confidence and the belief will generally differ. The possibility of a sampling bias due to the finiteness of L makes it again necessary to distinguish between the belief in the feature and its actually assigned probability $P(f|L,D)$. However, if the list of networks consists of all possible networks $\mathbb{L}$ we find

$$\text{belief}(f|\mathbb{L},D) = \sum_{G \in \mathbb{L}} P(G|D) f(G) \;\hat{=}\; P(f|D) \;, \tag{B.15}$$

i.e. the features probability for the given data. The same result can be gained with the bootstrap approach, if networks in L are drawn from the target distribution $P(G|D)$ using MCMC methods (Appendix D), for example. As the number of samples in L grows to infinity, i.e. $m \to \infty$, the bootstrap approach converges to Bayesian model averaging [Andrieu et al., 2003, Neal, 1993, Hoeting

et al., 1999].

In practical applications, however, the number of networks to average is often limited by computational constraints. But although not all possible networks $\mathbb{L}$ can be considered, the belief in a feature can be a good indicator of $P(f|D)$. This is especially the case when L corresponds to *Occam's window* [Hoeting et al., 1999, Madigan and Raftery, 1994], i.e. it contains the most probable networks. In this situation the following approximation holds:

$$\text{belief}(f|L,D) = c \cdot \sum_{G:P(G|D)\text{ high}} P(G|D) f(G) \approx P(f|D) \; . \tag{B.16}$$

In case of uncertainty about whether L corresponds to Occam's window or not the normalising constant c can be changed in order to derive a lower bound for $P(f|D)$. Setting $c = [\#L]^{-1}$, where $\#L$ denotes the number of averaged networks, renders the belief a conservative indicator of the feature's commonality.

The two averaging approaches discussed above can bring out shared features of different networks by smoothing out minor variations. When subtle differences between networks are actually important, the following graphical technique can be used in order to identify and visualise them.

B.3.2 The Consensus Network

Model averaging techniques discussed earlier can be used to reduce the list of networks L to a single average network. These methods merge links of all networks in L (weighted appropriately); especially those which connect two nodes a and b in either direction, i.e. $a \to b$ or $b \to a$. In causal networks, where links describe cause-effect relationships, such discrepancy should not go unnoticed, as causal relations are generally assumed to be unidirectional. Therefore, the list L can be combined to a new network in which contrariwise directed relations are highlighted by undirected links: the *consensus network*. This network has both directed and undirected links: Directed links indicate a consensus over link direction in all networks, whereas undirected ones reflect a contradiction between at least two networks in the list. The following rules describe the construction of the consensus network G_c:

R1 Any link that is present in any member G_i of the list L is added to G_c, unless it is already present.

R2 After all links have been added according to R1, contradicting links in G_c are replaced with undirected ones: Any pair of nodes that is connected by

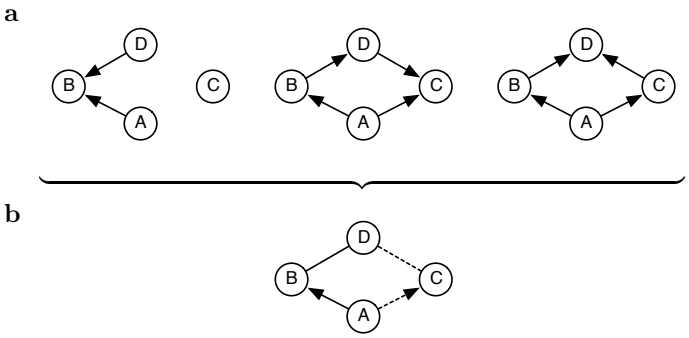


Figure B.1: Illustration of consensus network (b) constructed from three networks (a). **a** Three different networks constituting the list of networks L. **b** Consensus network G_c of networks in L. Links $A \rightarrow B$ and $A \rightarrow C$ are directed, since none of the networks in L contains an oppositely directed link $B \rightarrow A$ or $C \rightarrow A$. In contrast, networks in L connect nodes B and D as well as C and D in different directions; the corresponding links in G_c are thus undirected. Nodes A and B as well as B and D are linked in all of L's networks; links between them in G_c are therefore solid lines. Dashed lines between nodes A and C as well as C and D indicate that at least one network in L does not link them in any direction. None of the networks in L connects nodes A and D and so they are not linked in G_c either.

two (oppositely directed) links is instead undirectedly linked.

The resulting graph G_c immediately reveals which links are consistent with members of the list and which ones are opposing each other. The consensus graph G_c may however contain links, which are only present in few or even a single network in L. In order to indicate such potentially rare connections the links of G_c can be formatted according to the following rule:

R3 A link in G_c connecting two nodes is drawn as a full line, if these nodes are linked (in any direction) in all members G_i of the list L; and dashed otherwise.

With this refinement G_c not only visualises consensus and disagreement, but also communicates whether these are absolute or sporadic (Fig. B.1). Other format parameters like thickness or colour of links can be used to convey additional information if needed, but are not discussed here.

Appendix C

Proof of the Limit on Factors of the K2 Score

Here a proof is given for equation (2.16) from chapter 2 [page 34]. The equation claims that with the sufficient statistics of the K2 score we find

$$\prod_{j=1}^{q_i} \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} \prod_{k=1}^{r_i} N_{ijk}! \;\leq\; 1 \;. \tag{C.1}$$

It will be shown that in fact every factor of the outer product (over variable j) fulfils the inequality. Since the product of semi-positive numbers that are less or equal to 1 cannot yield a number greater than one, this also proves the inequality (C.1). For clarity, unnecessary indices are omitted in the following.

Claim:

$$\begin{aligned} &\frac{(r-1)!}{(S+r-1)!} \prod_{k=1}^{r} S_k! \;\leq\; 1 \;, \\ &\text{where} \;\; 1 \leq r, \;\forall\; k : 0 \leq S_k, \;\text{and} \;\; S = \sum_{k=1}^{r} S_k \;. \end{aligned} \tag{C.2}$$

Proof: The claim is proven by re-writing the expression, cancelling duplicate terms, and comparing the resulting fractions. Consider the special case $r = 1$ first, where both sides of the inequality (C.2) are equal to one. (This is because $0! = 1$ and $S = S_1$ per definition.) For all other cases, where $r > 1$, both sides will not be equal, as is shown next. Expanding the denominator we find the left

hand side of equation (C.2) equivalent to

$$\frac{(r-1)!}{[(S+r-1)!/(r-1)!]\cdot(r-1)!}\prod_{k=1}^{r}S_k! \iff \frac{S_1!\cdot S_2!\cdot\ldots\cdot S_r!}{(S+r-1)!/(r-1)!} \tag{C.3}$$

In order to simplify the denominator it is written as a product of S terms d_i:

$$\begin{aligned}\frac{(S+r-1)!}{(r-1)!} &= \frac{(S+r-1)\cdot(S+r-2)\cdot\ldots\cdot\overbrace{(S+r-S)}^{=r}\cdot\ldots\cdot 1}{(r-1)\cdot\ldots\cdot 1} \\ &= \underbrace{(S+r-1)}_{=:d_S}\cdot\underbrace{(S+r-2)}_{=:d_{S-1}}\cdot\ldots\cdot\underbrace{(S+r-S)}_{=:d_1}\ .\end{aligned} \tag{C.4}$$

The terms d_i will be compared to those in the numerator of equation (C.3) where each of the factorials yields S_k factors

$$\underbrace{S_k\cdot(S_k-1)\cdot\ldots\cdot 2\cdot 1}_{S_k\text{ factors}}\ . \tag{C.5}$$

Each these factors in the nominator is out-valued by a factor d_i in the denominator; starting with the factors of the first factorial ($k=1$) yields

$$\begin{aligned} 1 &< d_1 &&= r \\ 2 &< d_2 &&= r+1 \\ &\ \vdots \\ S_1-1 &< d_{S_1-1} &&= r+(S_1-2) \\ S_1 &< d_{S_1} &&= r+(S_1-1)\ . \end{aligned} \tag{C.6}$$

It is easy to see that all the remaining factors will be out-valued as well. For the k^{th} factorial we find:

$$\begin{aligned} 1 &< d_{1+\sum_{j=1}^{k-1}S_j} &&= r+\textstyle\sum_{j=1}^{k-1}S_j \\ 2 &< d_{2+\sum_{j=1}^{k-1}S_j} &&= r+1+\textstyle\sum_{j=1}^{k-1}S_j \\ &\ \vdots \\ S_k-1 &< d_{(S_k-1)+\sum_{j=1}^{k-1}S_j} &&= r+(S_k-2)+\textstyle\sum_{j=1}^{k-1}S_j \\ S_k &< d_{S_k+\sum_{j=1}^{k-1}S_j} &&= r+(S_k-1)+\textstyle\sum_{j=1}^{k-1}S_j\ . \end{aligned} \tag{C.7}$$

Finally, to verify that the number of factors matches recall that $\sum_{k=1}^{r}S_k=S$. (For $k=n$ the last term in equation (C.7) is thus $r+S-1$.) All factors in the numerator of equation (C.3) are out-valued by a factor in the denominator; their ratio is therefore smaller than 1. The proof is thus complete. ∎

Appendix D

Search Heuristics and Sampling Methods

Analysing data and representing essential results as a graphical model is one way of network inference. Another, rather data driven approach is to use scoring functions (Chapters 2 and 3) to assess large numbers of networks in order to find the best structure. This concept basically turns network inference into an optimisation problem, which becomes challenging in practical dimensions where extremely large numbers of potential networks exist (Section B.2): These cannot all be scored within reasonable time. Indeed, the problem of inferring BNs has been shown to be NP-hard in general [Chickering et al., 2004], and it is even NP-complete when the BDe score is used [Chickering, 1996]. Thus, no methodological nor technical advances (in foreseeable future) can overcome the incomplete model space investigation. Hence, within reasonable time, generally, only a small fraction of networks can be scored, which leaves doubts about having found the best scoring among all networks. Computation time should therefore be spent wisely by selecting promising networks to assess. Optimisation techniques address this point and this chapter is about how these methods can be used to assist inference of networks from data. The presented methods are generic and not bound to a particular network scoring function, in fact, these techniques are not even specific to network inference, but they rather constitute universal optimisation principles. How these principles can be applied to network learning is discussed next.

Fundamental Ideas and Terminology

Different techniques can be used in order to optimise the structure of a network with respect to a scoring function; for example, search heuristics or sampling

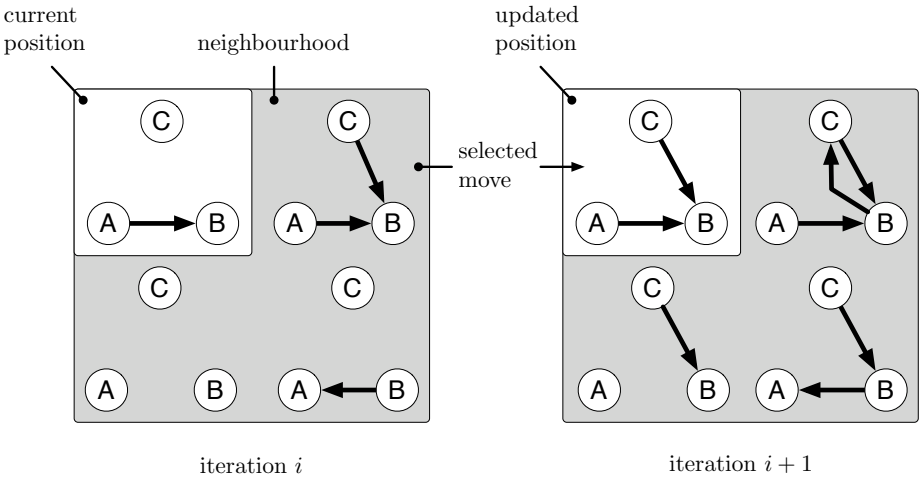


Figure D.1: Schematic illustration of a local optimisation method. A solution is slightly varied in order to yield neighbouring, i.e. similar results. One of these is selected and thereafter, the procedure may be repeated.

methods. These differ with respect to their particular strategy to improve solutions, but shared principles can also be found. Common concepts are discussed next for both, search heuristics or sampling methods, which are jointly identified as *optimisation techniques*. Accordingly, the search- and sampling-space is abstracted as *solution space*.

Many optimisation methods vary a potential solution slightly to yield another, similar solution. If the modification improves the result, the better solution might replace the original one in the next optimisation step, which repeats the variation-selection procedure (Fig. D.1). The exact rules on when to move from one solution to another depend on the optimisation method and will be presented later. Procedures to find similar solutions, i.e. networks with akin structures, can be identical for different techniques. In order to slightly vary a given network a single link could, for example, be added to the network; or an existing one could be deleted or reversed [Friedman et al., 1998, Rajapakse and Zhou, 2007, Husmeier, 2003, Friedman, 1997, Bernard and Hartemink, 2005, Friedman and Goldszmidt, 1996, Friedman et al., 1999b, Tucker et al., 2003]. Such modification is commonly called a *local step*. The set of all altered networks that can be reached from the current network by one local step are called the *neighbourhood* of this network.[1] Optimisation methods often refer to the

[1]Note that the operation applied to a network in a local step can have an impact on computation time: Adding or deleting one link changes the parent configuration of one network node only; when a decomposable score (Section E.1.1) is used, all nodes' scores stay unaltered except the one whose parents are changed. Evaluating this node's score and combining it with cached scores of all unaltered nodes efficiently yields the score of the full network. Alternatively, a local step might reverse an existing link and thereby cause higher computational

Algorithm 3 (Greedy / Hill climbing algorithm) Initially, a starting position (i.e. solution) is chosen. The Greedy algorithm evaluates the neighbourhood of the current position, which gets updated to the highest scoring neighbour. If local steps cannot increase the score any further a local maximum has been found and the search is over. Commonly, many Greedy runs are performed starting from different random positions.

choose (random) starting position y
repeat
  set $x := y$
  determine valid positions close to x,
  i.e. its neighbourhood: $\text{nbhd}(x) = \{n_1, \ldots, n_m\}$
  set potentially new position $y := \arg\max_{z \in \{x\} \cup \text{nbhd}(x)} \text{score}(z)$
until $x = y$
return x

neighbourhood of a solution rather than the actual local steps needed in order to generate it. This abstract description facilitates a very general formulation of the algorithm, which is used for presenting methods in the following.

D.1 Optimisation Methods for Network Inference

Any generic search method that operates on a discrete solution space is suitable for network learning. For example, Greedy search [Cormen et al., 2001, pp.370]; evolutionary algorithms [Bäck, 1996, Ashlock, 2004] like genetic algorithms [Whitley, 1994]; or particle swarm optimisation [Kennedy and Eberhart, 1995, Eberhart et al., 2001]. Additionally, suitable techniques can also be found in the large class of sampling methods: Metropolis-Hastings [Metropolis et al., 1953, Hastings, 1970], simulated annealing [Kirkpatrick et al., 1983, Cerny, 1985], and Gibbs sampling [Casella and George, 1992, Robert and Casella, 2004], for example. Due to the large variety of methods only few important concepts can be presented here; for further details and additional methods please consult the corresponding literature.

Search heuristics can be categorised into deterministic and probabilistic ones. A prominent deterministic method is *Greedy search*, which repeatedly performs local steps in the direction of highest score improvement (Algorithm 3). Such simple deterministic strategy, however, can easily get trapped in local extrema (Fig. D.2); this can be prevented by adding a stochastic component to the algo-

costs, because parent configurations of two nodes change, which need to be *re-scored*. Local steps associated with high costs can be advantageous if they generate diverse neighbourhoods, which might improve optimisation results. Such can happen if more different regions of the solution space can potentially be visited.

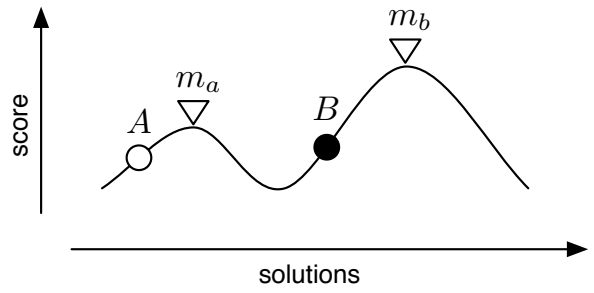


Figure D.2: Schematic illustration of solution space and associated score values. Two local maxima m_a and m_b exists, where m_b is also the global maximum. Greedy search, started at point A, would climb up to the local maximum m_a, but could not reach the better solution m_b. The search process is said to be *trapped* in a local extremum. Conversely, if a Greedy search is started at point B it will reach the optimum solution m_b. Note that, as the structure of the search space becomes rougher, many more local extrema exist, such that search runs may end quickly in one of them when using deterministic methods.

rithm. A modification of Greedy search, the *stochastic hill climber* algorithm, not only considers the steepest local step but also others, as long as they increase the score (Algorithm 4). Considering more possibilities for altering a solution can reduce the risk of getting trapped in local extrema. Additionally, a larger and more diverse neighbourhood can yield a better potential coverage of the solution space, which can improve solutions.

Not all optimisation techniques iterate over local steps; instead of considering only small changes to the current solution other methods use larger iteration-

Algorithm 4 (Stochastic hill climbing algorithm) Initially, a starting position (i.e. solution) is chosen. The algorithm evaluates the neighbourhood of the current position and randomly chooses a new position from those neighbouring solutions that exhibit a higher score value than the current solution. If no better solution exists in the neighbourhood a local maximum has been found and the search is over. Optimally, many runs of the algorithm should be performed starting from different random positions.

```
choose (random) starting position x
loop
  determine valid positions close to x,
  i.e. its neighbourhood: nbhd(x) = {n_1, ..., n_m}
  if ∃ z ∈ nbhd(x) : score(z) > score(x) then
    update position; set x := z
  else
    return x
  end if
end loop
```

Algorithm 5 (Genetic algorithm) A set of random solutions is used as the first generation population, which enters the algorithmic imitation of evolution: A population of individuals reproduces to generate new, varied individuals; only the fittest (i.e. best scoring) individuals enter the next iteration of propagation. Overall fitness of the population can increase by repeating the reproduction and selection process multiple times. The result of the process is given by the final population of solutions.

```
set p_0 = random population
for t = 0 to end-time do
  for all individuals I in p_t do
    assess fitness of I
  end for
  p_{t+1} := slight mutations and re-combinations of fittest individuals from p_t
end for
return p_{end-time}
```

steps, which can result in substantially different neighbours. These techniques are called *global optimisation methods* and include genetic algorithms (Algorithm 5), for example. Genetic algorithms are a mixture of selection and recombination, where selected solutions are recombined to new ones; these can be considerably different from the primary solutions and a large number of local steps would be needed in order to transform one of them to the recombined one. Due to the increased step size of global optimisation methods, inspected solutions can show a large variety. A diversified coverage of the solutions space can be beneficial to find solutions in situations where multiple different optima exist.

Besides search heuristics it is possible to use sampling methods for network inference as well. Generally, these techniques are used to sample from complex probability distributions in order to approximate them. They can, however, be used for optimisation purposes as well: Sampling methods generate samples, which are distributed according to a target distribution; in particular, samples corresponding to areas with relatively high probability-mass will occur more frequently than others. Setting the target distribution according to a network scoring function allows for sampling networks; networks with a higher score are thereby more likely to be sampled than low scoring ones. By collecting large numbers of samples, good scoring networks can be found. Methods that have been used to sample networks belong to the class of *Markov-Chain Monte Carle* (MCMC) methods; in particular, the Metropolis-Hastings algorithm [Rajapakse and Zhou, 2007] and Gibbs sampling [Casella and George, 1992, Robert and Casella, 2004] have been used for network inference.

Sampling methods are thought to approximate the target distribution and they therefore generate both high and low probability samples. When using such

method for network optimisation purposes this means that bad scoring networks will be among samples, although the primary interest lies on high scoring ones. Modified MCMC methods exist, which act as optimisation methods rather than sampling methods. Samples generated by these re-casted methods overrepresent high probability to low probability regions and are thus likely to have more high scoring networks among their samples than ordinary samplers. Simulated Annealing is a prominent biased sampling method, which can be used for network inference (Algorithm 6).

Algorithm 6 (Simulated annealing) One aspect of statistical mechanics is to investigate the behaviour of systems consisting of large numbers of atoms in the limit of low temperature where aggregate state changes (e.g. solidification) occur [Feynman et al., 1963]. In these situations the system's randomly changing configuration tends towards a state of low energy; this stochastic convergence is mimicked by simulated annealing [Kirkpatrick et al., 1983]. Initially, a starting state is chosen randomly whose energy will be minimised. Subsequent steps to one of its neighbouring states are chosen randomly; a neighbouring state is either accepted or rejected depending on four factors: the energy of the current and the potential state, a control variable called *temperature*, and a random number. The probability distribution combining these factors is specific to the application; however, it should depend on the temperature to facilitate the state to change nearly randomly while the temperature is high. Regions of low energy can initially be found this way. As the temperature gradually drops the probability of accepting moves to higher energy states should decrease. Thereby, the state is likely to get locked in an region of low energy.

set S = random state
initialise sample set $M = \emptyset$
for $T = T_{\max}$ down to 1 **do**
  determine valid positions close to S,
  i.e. its neighbourhood: nbhd$(S) = \{N_1, \ldots, N_m\}$
  randomly select a neighbouring state $B \in$ nbhd(S)
  if $P\left(energy\left(S\right), energy\left(B\right), T/T_{\max}\right) \leq$ random[0,1] **then**
    accept new state: $S := B$
    and add to sample set: $M := M \cup B$
  end if
  if energy$(S) <$ sufficiently low energy **then**
    break
  end if
end for
return M

Appendix E

Efficient Network Learning

Network inference from high dimensional data sets generally is computationally very intensive. In dimensions where a full pairwise correlation analysis is still feasible, the significantly larger number of multivariate relations may prevent their exhaustive evaluation (Section B.2). They can thus only be selectively assessed, using techniques discussed in appendix D, for example. The success of these methods largely depends on how many structures can be considered in the given time. It is therefore important that scores used to evaluate networks (Chapters 2 and 3) are implemented efficiently, in order to maximise the number of inspected structures. Practical approaches to this important aspect are discussed in this appendix. Universal implementation concepts are given first, before score-specific comments explain how BD scores and the SSS can be efficiently computed.

E.1 General Optimisation Potential

Any approach to network inference will most likely benefit from reduction in dimensionality. Similarly, an increase in computing power is desirable regardless of the method, since improved execution speed allows to inspect more networks in the same time period. This section will discuss how so called decomposable network scores facilitate both: reducing the complexity of the inference problem and harnessing more compute resources in order to accelerate evaluation. The latter, parallelised or distributed network learning, can also be applied to non-decomposable scores; however, the BD scores as well as the SSS are decomposable as the next section explains.

E.1.1 Score Decomposability

A mathematical function is said to be *decomposable*, if it can be written as a combination of independent terms. How these terms are actually combined and what determines their independence is specific to the context of the function. Network scoring functions are said to be decomposable, if the score of the full network can be expressed as a combination of partial scores associated with its nodes [Pe'er, 2005]. More precisely: Each node together with its parents constitutes a separate *family*. Composing all families yields the full network. If the score of the full networks is indeed a combination of *family-scores*, the score is considered to be decomposable. Any additional, node-independent expressions are irrelevant in this context and do not affect decomposability.

One example of a decomposable score is the SSS (Chapter 3): On page 43, the score of a full network is explicitely defined as the product of all node scores, i.e. a product of family-scores. But as indicated earlier, BD scores also decompose, as can be seen from the score's equation:

$$BD(G|D) = \underbrace{P(G)}_{=\text{prior on network}} \underbrace{\prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})}}_{=\text{family-score of node } i} . \quad \text{(E.1)}$$

The outer most product, taken over all variables, combines the family-scores of individual nodes. Note that the prior probability $P(G)$ is not node specific but affects all nodes likewise. It thus does not affect the decomposability of the score. Scores other than the ones discussed in this thesis are decomposable as well; for example the minimal description length (MDL) [Lam and Bacchus, 1994] and the Bayesian information criterion (BIC) [Schwarz, 1978].

Exploiting Decomposability Through Caching and Parallelisation

Network inference is often done with optimisation techniques that involve local steps (Appendix D): Such methods repeatedly change a network slightly, for example by adding a link, removing a link, or reversing a link. These local operations change the parent configuration of few nodes only: one family is changed by the addition or removal of a link, and two nodes' parents change through a link-reversion. At the same time, configurations of all remaining nodes are left unchanged. When applying a decomposable score with such methods, the fact that generally only a minority of family-scores changes has been widely recognised and exploited for optimisation [Friedman et al., 1998, Friedman, 1997, Friedman and Goldszmidt, 1996, Boyen et al., 1999, Friedman et al., 2000, Dojer et al., 2006]: After each local step the score of the resulting network has

to be calculated. Most family-scores are identical to their previous value before the network was changed; caching them thus eliminates the need for a re-calculation. Only families that have been affected by the change must be scored. This inconspicuous practice can lead to significant efficiency improvements, because caching involves only negligible organisational costs compared to the computationally more demanding (re-)scoring of a family. This positive effect must not be disregarded since it applies to every single of the numerous local step during the inference procedure.

The fact that family-scores are independent of each other also allows to compute them separately — not necessarily in a serial fashion. Processors in modern computers often have multiple cores, which can perform computations truly in parallel. An efficient implementation of a score might use such resource by calculating family-scores simultaneously on different cores. Depending on the platform used, separate threads or independent child-processes may be suitable programming techniques; however, details are not discussed here, as they will be familiar to the experienced programmer.

Exploiting Decomposability in Conjunction Structural Independence

The optimisation proposals made so far can be applied to learn both static BNs and stochastic processes. This is not the case for the next suggestion, which cannot be applied if resulting networks are required to be acyclic or have to fulfil any other global property. However, for processes like DBNs or the ones corresponding to the SSS it is in general possible to use the fact that families in the network are *structurally independent*. To illustrate this feature consider BNs as a counter example: These models are associated with directed acyclic graphs (DAGs) (Section B.1), i.e. they must not contain any cycle. In order to guarantee that a graph is cycle-free it is not sufficient to analyse separate families only, but the whole graph must be considered. This is because the different families are *structurally dependent* on each other: Changing one family can lead to an invalid cycle, which renders the whole network invalid. This situation can only be cured by changing other families in return. Families in DBNs, on the other hand, do not depend on each other, but yield a valid graph as long as all the parent configurations are proper. Such structural independence is important because, together with a decomposable score, it implies that the highest scoring network is composed of all highest scoring family-scores. This means that each node's optimal family can be determined independently [Murphy and Mian, 1999]. An efficient inference algorithm for structurally independent networks will utilise this fact. Optimising each node's parent configuration separately to find the best scoring network is of significant advantage compared to an opti-

misation operating on the space of full networks. This is because learning a network family-wise drastically reduces the dimension of the learning problem (Section B.2), such that optimisation methods are more likely to succeed: For heuristics the search space is significantly smaller, such that a higher percentage of structures can be inspected for good solutions. MCMC methods are benefited likewise, because the dimension of the target distribution is reduced. This can speed up convergence in the burn-in phase and result in better samples thereafter.

Utilising decomposability in conjunction with structural independence is an often overlooked optimisation target. The reason for this is likely to be found in the domain of non-dynamic BNs — the original domain of network scores — whose acyclicity constraint hinders the application of the outlined procedure. It should, however, be noted that node-wise optimisation has been used for BNs as a pre-processing step by the *sparse candidate algorithm* [Friedman et al., 1999b]: Only links that show good performance in the node-wise search are considered as candidate-links in DAGs during the succeeding search over full networks.

The next section is concerned with how network inference can be performed in a distributed fashion across different computers. As will be explained, the efficiency of such task devision largely depends on how efficiently duties and results are communicated. A suitable concept for this, based on unique identification numbers for parent configurations and graphs, will be presented.

E.1.2 Distributed Network Inference

The analysis of large, high dimensional data-sets can cause enormous computational demands, such that corresponding calculations can take unacceptably long when utilising a single computer. Dividing the workload up onto multiple machines, however, might shorten the effective waiting time for results considerably. Such *distributed computing* is mainly a challenge to computer skills; but these rather technical problems about managing multiple machines are not considered here. Instead, theoretical aids will be provided in order to support practical approaches to distributed network inference.

In order to explain the context of the rest of this section consider the following situation (Fig. E.1a): We are interested in the scores of particular graphs, which have therefore been compiled in a list. (The list might for example contain all graphs with less than 5 links.) All listed graphs are to be assessed in order to yield another list, which contains their score values. How this result list can be used is discussed in section B.3; here the focus is on how this simple work-flow can be split up and distributed among different machines (Fig. E.1b).

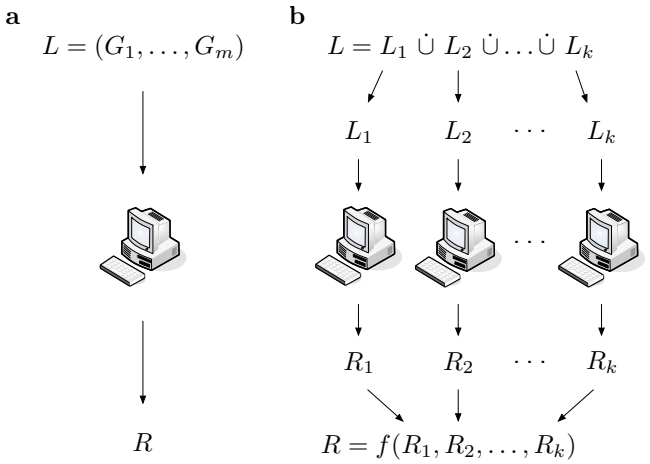
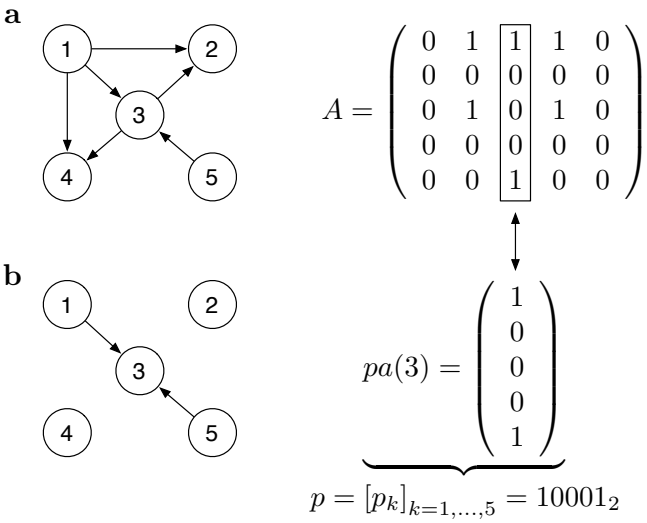


Figure E.1: Work-flow-scheme for network inference on single/multiple computer(s). **a** List of interest L is composed of all graphs G_i that are to be analysed (e.g. scored). The list is processed by a single machine, which generates an appropriate list of results R. **b** Analysis of list of interest L split up amongst k machines. Therefore L is partitioned into sub-lists L_i, which get transferred to the different compute nodes. These process the list analogous to the work-flow shown in (a). In the final step, the results R_i of each sub-list are collected and suitably joined via an operation f in order to yield the list of all results R.

The obvious approach is to divide the list of interest into sub-lists; these are then distributed among available resources. On each machine the sub-list is processed analogous to case discussed first (Fig. E.1a). Finally, the separate results are collected and suitably joined, such that the same result-list is achieved as if the analysis had been performed on a single computer, but faster. The purpose of this explanation is to highlight the need for additional organisational effort, which is associated with such task division: duties and results have to be suitably distributed and joined, respectively. Improper task-division would miss distributing all graphs of interest, such that these remain unevaluated. It would also be non-optimal if graphs are evaluated redundantly on multiple compute nodes due to overlapping sub-lists. Corresponding demands apply to the output side of the work-flow — the result list: Solutions should neither be missed nor represented superfluously. Two conditions must be met in order to fulfil these requirements: (1) unambiguous communication of objects of interest; and (2) definite assignment of corresponding results. Different approaches meeting these demands exist but differ widely in their efficiency. However, effective communication concepts are vital in order to ensure that time gained by distributed computing is not consumed by transmitting tasks and results. Therefore, two different approaches are discussed in the following.

In order to compose, split, and communicate a list of graphs of interest, the structures themselves must be suitably represented. Such could be done by each graphs' adjacency matrix (Section B.1), which fully specifies the size and connectivity of a graph. Defining the list of interest as a list of matrices corresponding the graphs is straightforward and can be used for communication in a distributed computing scheme (Fig. E.1b). However, representing graphs this way renders the list L relatively voluminous, which makes its communication costly. It would thus be favourable if the adjacency matrix could be bijectively mapped to a more compact representation, e.g. a unique decimal number by which graphs were unambiguously identified. Indeed, such one-to-one relationship can be easily established by interpreting the adjacency matrix as a binary number. This is done by composing the elements of the matrix in a arbitrary but fixed order, which yields a binary string. Treating the string as a number with base 2 facilitates its conversion to a number of any other base [Bronstein et al., 1999, pp.931]. For convenience, base 10 might be chosen in order to yield a familiar decimal number for each graph. To reconstruct a graph from a given number this needs to be re-converted to base 2 before interpreting the digits of this number as entries of the adjacency matrix. Similar to a full graph, this base-conversion concept can also be applied to parts of a graphs, such as a particular parent configuration (Fig. E.2). Whether applying this simple mapping to full networks or parent configurations, it results in a compact representation,



c

$$
\begin{aligned}
decimal(p) &= \textstyle\sum_{k=1}^{5} p_k \cdot 2^{k-1} \\
&= 1 \cdot 2^0 + 0 \cdot 2^1 + 0 \cdot 2^2 + 0 \cdot 2^3 + 1 \cdot 2^4 = 17_{10} \\
binary(17_{10}) &= \left[\left(17_{10} \bmod 2^k\right) \operatorname{div} 2^{k-1} \right]_{k=5,\ldots,1} \\
&= [1,0,0,0,1] \qquad\qquad = 10001_2
\end{aligned}
$$

Figure E.2: Identifying graph structures with decimal numbers. **a** Graph with five nodes and its adjacency matrix. **b** Family of node 3 and its corresponding parent vector, which can be interpreted as a binary number p. **c** Conversion of node 3's parent vector to a decimal number and back.

which can be used to efficiently communicate individual structures of interest.

(Technical Note) For practical application of the base-conversion concept two aspects should be noted: First, for an unambiguous re-conversion of an identification number to base 2, the number of vertices in the graph must be given. This is to ensure that the binary number will be of appropriate length, when one or more 0's prefix the first digit 1. Second, this representation of adjacency matrices can only be more efficient that the matrices themselves, when the latter are not stored as bit-pattern. In detail: Most computers store information in binary formats, which allow to represent an adjacency matrix on the level of individual bits, each of which can code for the existence or abundance of a link. In total n^2 bits would be needed to store the full matrix; however, commonly the smallest memory-block that can be allocated is (at least) a byte, which consists of 8 bits. Unless the number of required bits n^2 happens to be a multiple of 8, a minimal overhead of unused bits is involved; but despite this, the bit-pattern representation of matrices would be the most efficient coding for storage. For practical application, however, bit-patterns are not very convenient to use, because processing them generally requires a decomposition and up-conversion to larger data-types. Because of this, less efficient representations are often preferred; for example, saving the adjacency matrix as a text file where each of its entries is represented by a character 0 or 1. In such cases the presented number-conversion concept can save significant storage-space: Data-types used to represent characters generally have enough bits to represent at least 256 different characters (8 bits) out of which only two are used for binary adjacency matrices. Converting the matrix to a decimal number and storing it as text makes use of 10 different digits (from 0 to 9) by which the total number of required characters is reduced. It would be most efficient if the whole scope of characters would be used for each digit; i.e. if the matrix would be converted to a base 256 number. In practice this number is likely to be even larger, since modern character data-types commonly use more than 8 bits per digit. However, these considerations are specific to computing platforms and need therefore be left to the programmer who is knowledgeable about internals of the used system.

General concepts in order to improve the efficiency of network inference have been considered so far. Different scores offer additional, but specific enhancement possibilities, which are discussed in the remainder of this section for both the BD scores and the SSS.

E.2 Score Specific Optimisation Potential

Scoring functions play a central role in network inference with respect to two aspects: (1) They rate networks, and therefore determine the outcome, i.e. the quality of the analysis. Whether a score's assessment is appropriate or not is discussed throughout the main part of the thesis. The following sections thus focus on point (2): Because large numbers of networks are assessed, scoring functions are a major factor affecting the required computing time. This makes scores central with respect to the efficiency of the analysis, which can be enhanced by exploiting particular properties of each score. The following sections discuss BD scores and the SSS separately in order to account for their different points of action for efficiency improvements.

E.2.1 Suggestions for BD Scores

All of the BD scores involve multiple nested products [equations (2.9) and (2.14)]: The most outer one is taken over all nodes; within that, a product is taken over each node's joint parent states with another product over the child's states nested within. On a computer multiplications are generally computationally expensive and consume significantly more time than additions. This common characteristic of processors can be taken into account by calculating logarithmic scores: A logarithm is applied to the scoring function, by which, according to the associated rules, products become sums.[1] The BD score then reads as

$$\log BD(G|D) = \log P(G) + \sum_{i=1}^{n}\sum_{j=1}^{q_i}\log\left(\frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij}+N_{ij})}\right) + \sum_{k=1}^{r_i}\log\left(\frac{\Gamma(N'_{ijk}+N_{ijk})}{\Gamma(N'_{ijk})}\right) . \quad \text{(E.3)}$$

In this equation the computationally most costly operation is presumably the Γ-function, which generally cannot be further resolved. However, in cases where its argument is an integer value, it can be substituted with a factorial [equation (2.13)]. For one of the BD score's variants all pseudo counts N'_{ijk} are integer's by definition: the K2 score. Its formula [equation (2.14)] can therefore

[1] The exponential function (with corresponding base a) a^x makes this operation fully reversible in case the absolute score value is required. In order to compare and rank networks log-scores can be directly compared to each other. This is because the logarithm is a monotonically increasing function, since [Bronstein et al., 1999, p.374]

$$\frac{d\log_a(x)}{dx} = \frac{1}{x\ln(a)} \overset{!}{>} 0 \quad (0 < a \neq 1, 0 < x) , \quad \text{(E.2)}$$

by which the numerical ordering of arguments is preserved.

be resolved to

$$\log K2(G|D) = \log P(G) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \frac{\sum_{l=1}^{N'_{ij}-1} \log(l)}{\sum_{l=1}^{N'_{ij}+N_{ij}-1} \log(l)} + \sum_{k=1}^{r_i} \frac{\sum_{l=1}^{N'_{ijk}+N_{ijk}-1} \log(l)}{\sum_{l=1}^{N'_{ijk}-1} \log(l)} \; . \quad \text{(E.4)}$$

Calculating the score according to this equation can be considerably faster. However, a minor trick might be considered additionally: caching calculated logarithms. More precisely, the sums of log-values $\sum_{l=1}^{N} \log(l)$ calculated for different upper bounds N are worthwhile to be stored in order to reduce the number of costly log-function calls. Indeed, the maximal upper bound N that can appear for any network can be determined from the data; all sums up to that bound can then be calculated off-line, i.e. prior to the network assessments, in order to produce a look-up table that is used for the numerous calculations thereafter.

The considerations above can help to compute the score value efficiently from the sufficient statistics N_{ijk}. Determining the latter, however, is generally a costly procedure, which can easily take multiple times longer than the score calculation itself. An efficient implementation of the BD score thus cannot ignore time-saving strategies to determine the required counts from the data. Such are discussed in the following.

Extracting Sufficient Statistics for the BD Scores From Data

Determining the statistics N_{ijk} means inspection of the data in order to count the occurrence of state combination: *How often were the parents of node i in joined state j while the child was in state k?* The simplest approach to this tedious counting task is to define a multi-dimensional array such that the state of each variable can be used to index one of the dimensions. Thereby, each joint parent-child-state corresponds to a distinct element in the array. Initialising the array with zeros and counting up an element each time its associated pattern is found in the data yields the required statistics. This approach is straightforward to implement and, in principle, computationally efficient; however, it can be very memory intensive. In situations where many parents exist and/or variables can take many different values the array demands significant amounts of memory.[2]

[2]The amount of memory that must be allocated for the array can be calculated as follows: Assume the number of possible variable states r_i is equal to r for all nodes i; let p_{max} denote the maximum number of parents per node; and let b denote the number of bytes required for each counter-element in the array. A total of $b \cdot r^{p_{max}+1}$ bytes is then required for the array.

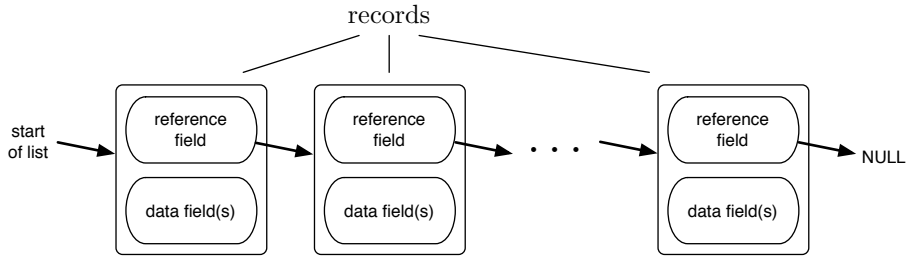


Figure E.3: Schematic illustration of a linked-list data structure. The basic entity of a list are records, each of which contains reference- and data-fields. The beginning of a list is given by a pointer to its first record. Records themselves are connected chain-like through a reference to the successive record in each of them. The void reference `NULL` marks the end of the list.

This can negatively impact on computational performance, as caching mechanisms in the processor are likely to fail if huge objects are manipulated in a non-serial fashion [Fog, 2008].

One possibility that can reduce memory usage are so called *sparse arrays*. (Details can be found in the literature, for example [Goldwasser et al., 2002, pp.33], [Smith, 2004, p.71], or [Das, 2006, pp.12].) To the user these data-structures are identical to a normal array; internally, only values that are different from zero are actually stored, i.e. consume memory. Unsurprisingly, sparse arrays come with a management overhead both computationally and concerning memory. They are thus more costly than normal arrays, when many entries are non-zero, but in other cases they are very efficient. For the BD score this advantage will be especially noticeable when high dimensional data with many variable states are analysed: Unless data-length is exceptionally long, the number of data-points will be far lower than the number of all possible joint parent-child-states. Thus, even if every data-point exhibits a different parent-child-state, their number will not be sufficient in order to render a significant number array entries non-zero. Sparse arrays should therefore be considered even if they are not natively supported by the programming language used. In this case, as outlined next, linked lists can be used to mimic these arrays.

(Determining Sufficient Statistics Efficiently by Ranking Joint Parent-Child-States and Usage of Linked Lists) *Lists* are data structures that consist of separate *records*, which are unidirectionally linked in a serial fashion (Fig. E.3) [Cormen et al., 2001, pp.204]. Each record has a *reference field*, in order to point to another record, and also *data fields*, which contain the actual information to be stored. In contrast to an array, which generally has a fixed size, lists can be dynamically expanded by appending or inserting records

(thought changes to the corresponding reference fields). Like sparse arrays, they consume little memory if few records are stored only.

In order to use lists for the determination of counts N_{ijk}, the data-fields of each record need to facilitate two things: (1) to identify a particular joint parent-child-state; and (2) to count its occurrence. With appropriate records defined, the data can be traversed while updating the list: For every new vector of data, i.e. a joint parent-child-state that did not occur so far, a record is added to the list. For any vector that has been observed already, the counter in the corresponding record is increased. The efficiency of such list implementation heavily depends on how quickly data-vectors can be compared: In order to test whether a parent-child-state is already in the list, every data-vector needs to be checked against the corresponding field in a record until it matches up or the end of the list is reached. Numerous comparisons between the data-vector and record fields are thus required and happen to be the main time consumers. This is mainly because the comparison of two vectors requires multiple element-wise comparisons. Thus, if the number of values to compare could be reduced, the counting-process would speed up. In the following, a computationally cheap mapping of joint parent-child-state vectors to unique one-dimensional ID numbers is proposed. To test whether two IDs match only one comparison is needed; costs associated with the generation of the IDs can thus pay off by comparison operations saved.

Similarly to the conversion of graphs to decimal numbers presented earlier (Section E.1.2, Fig. E.2), joint parent-child-state combinations can be mapped. The only difference is that data-vectors to convert are not binary numbers, but they can have a different base for each digit, such that the formalism is more intricate. In section 2.2.1 the vector of joint parent states has been introduced, which is a sub-vector of the data for all nodes that only contains entries belonging to the parents. This notation is expanded here in order to not only accommodate the states of the parents but also the child's state as the first element in the *joint parent-child-state vector* $d_t^{(i,pa_i)}$. If node i has $p_i := \#pa_i$ parents $pa_i = \left\{pa_1^{(i)}, \ldots, pa_{p_i}^{(i)}\right\}$ the vector reads as

$$d_t^{(i,pa_i)} = \left(\underbrace{d_t^{(i)}}_{\text{child-state}}, \underbrace{d_t^{\left(p_1^{(i)}\right)}, \ldots, d_t^{\left(p_{p_i}^{(i)}\right)}}_{parents' states} \right) . \tag{E.5}$$

In order to simplify formulae below the data are assumed to be transformed, such that the values taken by each component i are numbered from zero onwards, i.e. $d_t^{(i)} \in \{0, \ldots, r_i - 1\}$. The number of different joint parent-child-state vectors can be calculated from the number r_i of different states variable i can take: The

vector $d_t^{(i,pa_i)}$ can take $r_i \cdot \prod_{l \in pa_i} r_l$ different combinations of values. All these different combinations can be mapped to a unique integer ID via

$$\text{ID}\left(d_t^{(i,pa_i)}\right) = d_t^{(i)} + \sum_{l=1}^{p_i} \left[d_t^{\left(pa_{p_l}^{(i)}\right)} \cdot r_i \cdot \prod_{z=1}^{l-1} r_{pa_z^{(i)}} \right] , \quad \text{where } \prod_{z=1}^{0} = 1 \; . \tag{E.6}$$

Each joint vector is thereby systematically assigned a number between 0 and $r_i \cdot \prod_{l \in pa_i} r_l - 1$. This can be best seen by ordering vectors by their ID, which yields the following scheme:

ID	$d_t^{(i)}$	$d_t^{\left(pa_1^{(i)}\right)}$	$d_t^{\left(pa_2^{(i)}\right)}$	$\ldots$	$d_t^{\left(pa_{p_i}^{(i)}\right)}$
0	0	0	0	$\ldots$	0
1	1	0	0	$\ldots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r_i - 1$	$r_i - 1$	0	0	$\ldots$	0
r_i	0	1	0	$\ldots$	0
$r_i + 1$	1	1	0	$\ldots$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r_i \cdot \prod_{l \in pa_i} r_l - 1$	$r_i - 1$	$r_{pa_1^{(i)}} - 1$	$r_{pa_2^{(i)}} - 1$	$\ldots$	$r_{pa_{p_i}^{(i)}} - 1$

The expressions simplify, if all variables i can potentially take the same number of different values, i.e. $\forall\, i : r_i = c$. In this situation equation (E.6) reduces to

$$\text{ID}\left(d_t^{(i,pa_i)}\right) = d_t^{(i)} + \sum_{l=1}^{p_i} d_t^{\left(pa_{p_l}^{(i)}\right)} \cdot c^l \; . \tag{E.7}$$

This formula interprets the joint vector $d_t^{(i,pa_i)}$ as a backwards written number (in the number system with base c), which is converted to the decimal number system [Bronstein et al., 1999, pp.931]. Vice versa, in order to reconstruct the joint parent-child-state vector from a given ID use the following inverse mapping:

$$\begin{aligned} d_t^{(i)} &= \text{ID} \;\; \text{mod} \;\; r_i \\ d_t^{\left(pa_1^{(i)}\right)} &= (\text{ID} \;\; \text{div} \;\; r_i) \,\text{mod} \;\; r_{pa_1^{(i)}} \\ &\vdots \\ d_t^{\left(pa_{p_i}^{(i)}\right)} &= \left(\text{ID} \;\; \text{div} \;\; r_i \cdot \prod_{z=1}^{p_i - 1} r_{pa_z^{(i)}} \right) \text{mod} \;\; r_{pa_{p_i}^{(i)}} \; . \end{aligned} \tag{E.8}$$

The inverse mapping is needed after the list has been constructed according to

the data, in order to extract the sufficient statistics. In more detail, the way linked-lists and the ID can be used together involves two steps: Given a node i and its parents for which the sufficient statistics are to be determined...

1. Create a linked-list (using IDs) to count the occurrence of different parent-child-states in the data.

2. After all data-points have been processed, the counts N_{ijk} for a specific parent-state j and child-state k can be retrieved from the list: The combination of states j and k a particular ID represents is given by

$$j = \text{ID } \text{ div } r_i \quad \text{and} \quad k = \text{ID } \text{ mod } r_i \ , \tag{E.9}$$

   such that the counter in the corresponding record can be associated.

Describing this procedure involves an extensive formalism, but which can be efficiently implemented and thereby accelerate network inference with BD scores.

In the final section the focus is on the SSS and its optimisation potential. As outlined earlier (Section E.1.1), an efficient implementation of network inference will exploit the decomposability of this score, but further improvements are possible, as outlined next.

E.2.2 Tuning Up the Snap Shot Score

Like the BD scores, the SSS offers ways to save computation time. The most obvious point of action of this score is to calculate the activity level series of each node off-line, i.e. prior to a network search, such that they can be quickly retrieved afterwards. This is possible since spike trains and decay constant are already known before starting the inference procedure and because these only two determinants of the activity level remain fixed; activity level series therefore remain the same throughout the search, too. For each individual network to score the activity level series are required, which makes it beneficial to avoid their redundant calculation.

Computations can further be accelerated in the situation where none of the channels shows any spiking over a certain period. Without any spikes, activity levels decay to zero and do not affect score values: The score is the same whether zero activity levels are summed or not, such that silent periods can be omitted.[3] Saved summations speed up the score calculation accordingly. Further, two more advanced options to optimise the score exist, which are discussed

[3] When omitting sections of the data that show no activity, care has to be taken in order to ensure that concatenation of the remaining parts does not cause artefacts, i.e. spikes triggering a snapshot of a positive activity level, which should be zero. Joined partitions must therefore be separated by a zero activity section, according to the shift constant.

now. One of them utilises a characterisation result from chapter 4 in order to conclude whether a network search can be aborted or if a continuation to score more complex parent configurations can be beneficial. But first, a strategical improvement of the order at which parent configurations are scored is presented.

Re-using Joined Activity-Level Series

It has already been mentioned that activity level series of individual nodes can be calculated and cached prior to the network search. This not possible for joined activity level series because these depend on the parent configuration to score, which changes throughout the search. For configurations with multiple parents, channels thus have to be joined on demand, such that the child-node can be scored. The process of joining activity level series and calculating the score has been investigated in profiling implementations. These revealed that the two steps involved in assessing a configuration differ widely concerning their computational costs: Joining activity level series is much more expensive than taking and summing snapshots; the time spent scoring a configuration (100% time) splits up very unequally between joining (≈90% time) and actual score calculation (≈10% time).[4] In order to ensure good performance of the network inference algorithm, special care should therefore be taken when implementing the function to join channels.

Additionally to improvements of the join operation itself, it is also possible to reduce the total number of joins to be calculated. As outlined earlier (Section E.1.1), each node's parent configuration can be optimised independently: Separately for each node i, different parent configurations are scored in order to identify its best scoring parents (Fig. E.4a). This decomposed learning strategy easily facilitates to harness multiple compute nodes, each of which optimises configurations for another node. However, the scoring of configurations can be made more efficient by avoiding the redundant evaluation of joins: A join does not depend on the child, but on the parents only — for any set of parent nodes it is thus the same for any child node. This can be used in an alternative scoring strategy in which a join is re-used several times once it is computed (Fig. E.4b): A set of parents is chosen and their join is calculated; after that, different child nodes are scored using that join. With this concept, the computationally expensive join only needs to be evaluated only once and time for otherwise redundant calculations is saved; computation time by which the search can either be shortened or improved by evaluating more configurations in the same time.

[4]The SSS was separately implemented in C [Kernighan and Ritchie, 1988] and Python [van Rossum and et al.]. Both implementations were optimised before making measurements. The programming language did not have any significant effect on temporal proportions between joining channels and calculating the score.

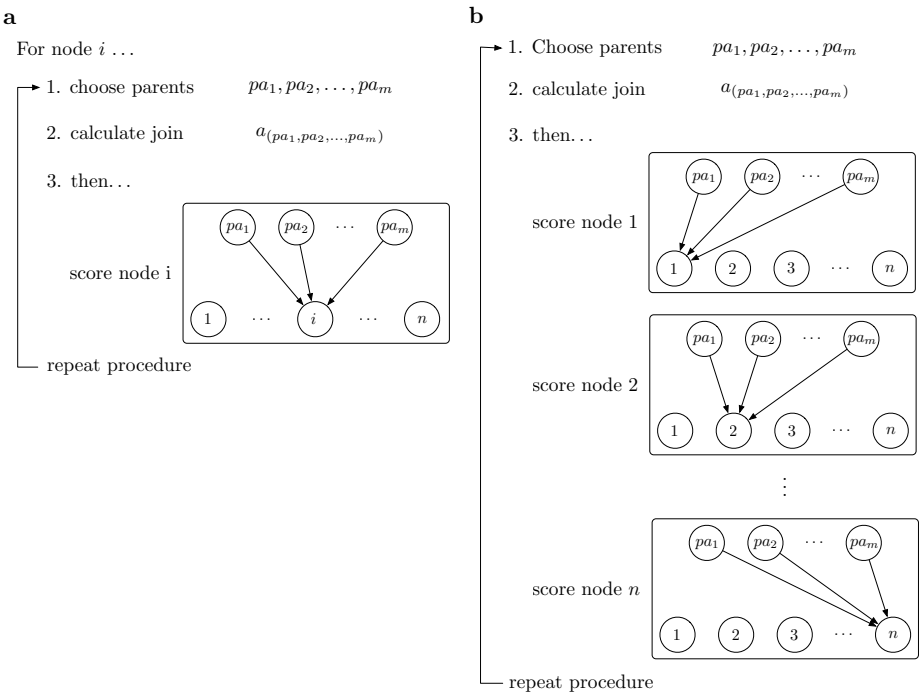


Figure E.4: Comparison of two different learning strategies: node-wise and join-wise. **a** Node-wise learning of parent configurations: The parent configurations are selected and scored for each node i separately. If the same parent configuration is scored for different nodes, the joined activity level of the parents is evaluated redundantly. **b** Join-wise learning of parent configuration: A parent configuration is chosen, which is then scored for every node i. This re-use of joins is proper, because these do not depend on the child node, but on the parents only. Compared to the node-wise procedure shown in (a), computation time can thus be saved, because the join has to be calculated only once for all nodes.

Strategies proposed so far covered how distributed compute resources can be used for network inference, how simple features of the SSS can improve the efficiency of the score, and how joins can be re-used in order to save computation time. All these efforts aim at evaluating networks as quickly as possible, which is important because the best scoring structure cannot be derived from the data directly, but all potential networks must be probed instead. Fortunately, a theoretical result gained in section 4.1 facilitates conclusions about scores of large groups of networks: Under particular circumstances, simpler configurations than those in the groups are known to be superior, such that the latter need not to be considered for scoring at all. The details on this concept and its implementation are subject to the next section.

Reduced Search Depth

The SSS has been investigated and characterised in chapter 4. One of the results was that expanding a join by a non-overlapping channel cannot increase the score value beyond that of the better scoring of the two (Corollary 2). In the best case, the expanded, more complex join scores just as well as one of the simpler configurations; then, according to Occam's razor, the simpler configuration should be preferred (Section 1.4.2). As a consequence, if it is known that a configuration cannot be improved further by making it more complex, there is no need to score configurations that are worse. The exclusion of subordinate configurations can reduce the number of configurations to score dramatically, which benefits the performance of network inference. However, the conditions of corollary 2 must be met in order to use it: It must be ensured that two activity level series are non-overlapping before excluding their join from being scored. In the following it is therefore discussed how overlapping activity can be efficiently determined.

Key to fast activity overlap-checking are two facts: (1) Activity overlap between any two channels can be detected prior to the search; and (2) A join and a single channel have overlapping activity, if at least one of the joined channels has overlapping activity with the single one. In order to show how the conditions of corollary 2 can be checked quickly, a (symmetric) activity overlap matrix $O = (o_{ij})_{i,j\in\{1,\dots,n\}}$ is defined as follows:

$$o_{ij} = \begin{cases} 0, & \text{channels } i \text{ and } j \text{ have non-overlapping activity,} \\ 1, & \text{otherwise.} \end{cases} \tag{E.10}$$

This matrix needs to be calculated and stored prior to the network search. During the search it is used in order to determine any overlap between the

activity of a single channel a_k and a join of multiple channels $a_{(j_1,\ldots,j_r)}$. This can be done with the *overlap function* $o(\,\cdot\,,\cdot\,)$, which is defined as

$$o\left(a_k, a_{(j_1,\ldots,j_r)}\right) = \max_{i=1,\ldots,r} o_{k,j_i} \ . \tag{E.11}$$

If any of the channels j_i in the join has overlapping activity with channel k, the join itself has overlapping activity with channel k. In this case the value $o\left(a_k, a_{(j_1,\ldots,j_r)}\right)$ will be equal to one; and zero otherwise.[5] According to corollary 2, only if activity overlaps it is worthwhile to score the complex join $a_{(k,j_1,\ldots,j_r)}$ of all channels. Otherwise this configuration cannot improve the score value but becomes more complex only and should thus not be considered.

Determining the overlap of activity according to the approach described above is a very efficient and computationally considerably cheaper than comparing activity level series directly with each other. This is especially the case when the activity level series do not overlap, since both activity level series must be compared at all times (at which any of them is not zero). Since data-sets analysed with the SSS will generally contain many data-points, a direct comparison requires numerous time-consuming comparisons in order to guarantee that activity between two series does not overlap. In contrast, using the overlap matrix requires at most n comparisons, irrespective of the length of the data and will thus be significantly cheaper.

[5]It is assumed that channel k is not among the already joined channels $j_1, \ldots, j_r$. Joining a channel twice does not have any effect (see examples 5 and 6 in section 3.2); the learning procedure should prevent such unnecessary computational effort.

Bibliography

L. F. Abbott. Lapicque's introduction of the integrate-and-fire model neuron (1907). *Brain research bulletin*, 50(5-6):303–4, 1999.

M. Abeles. *Local cortical circuits: an electrophysiological study*. Springer-Verlag, Berlin, Heidelberg, New York, 1982.

C. Adams, J. Simonotto, S. J. Eglen, and E. Sernagor. Multielectrode array recordings of neural activity patterns in the developing retina of the cone rod homeobox knockout (Crx-/-) mouse. In *6th International Meeting on Substrate-Integrated Microelectrodes*, pages 195–196, Reutlingen, Germany, 2008.

A. M. H. J. Aertsen, G. L. Gerstein, M. K. Habib, and G. Palm. Dynamics of neuronal firing correlation - modulation of effective connectivity. *Journal of Neurophysiology*, 61(5):900–917, 1989.

E. M. Airoldi. Getting started in probabilistic graphical models. *PLoS Computational Biology*, 3(12):e252, 2007.

R. Albert and A. L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, 2002.

C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine Learning*, 50(1-2):5–43, 2003.

D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer, 2004.

L. Astolfi, F. Cincotti, D. Mattia, M. G. Marciani, L. A. Baccalá, F. D. Fallani, S. Salinari, M. Ursino, M. Zavaglia, and F. Babiloni. Assessing cortical functional connectivity by partial directed coherence: Simulations and application to real data. *IEEE Transactions on Biomedical Engineering*, 53(9): 1802–1812, 2006.

L. A. Baccalá and K. Sameshima. Partial directed coherence: a new concept in neural structure determination. *Biological Cybernetics*, 84(6):463–74, 2001.

T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, USA, 1996.

W. Bair, C. Koch, W. Newsome, and K. Britten. Power spectrum analysis of bursting cells in area MT in the behaving monkey. *Journal of Neuroscience*, 14(5 Pt 1):2870–92, 1994.

R. Barbieri, M. A. Wilson, L. M. Frank, and E. N. Brown. An analysis of hippocampal spatio-temporal representations using a Bayesian algorithm for neural spike train decoding. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(2):131–6, 2005.

C. Barry, R. Hayman, N. Burgess, and K. J. Jeffery. Experience-dependent rescaling of entorhinal grids. *Nature Neuroscience*, 10(6):682–684, 2007.

T. Bayes. An essay toward solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763.

E. A. Bender and R. W. Robinson. The asymptotic number of acyclic digraphs II. *Journal of Combinatorial Theory*, (Series B 44):363–369, 1988.

E. A. Bender, L. B. Richmond, R. W. Robinson, and N. C. Wormald. The asymptotic number of acyclic digraphs. *Combinatorica*, 6(1):15–22, 1986.

A. Bernard and A. J. Hartemink. Informative structure priors: joint learning of dynamic regulatory networks from multiple types of data. *Pacific Symposium on Biocomputing*, pages 459–70, 2005.

U. S. Bhalla. How to record a million synaptic weights in a hippocampal slice. *PLoS Computational Biology*, 4(6):e1000098, 2008.

B. Bollobs. *Modern Graph Theory*. Graduate Texts in Mathematics. Springer, 1998.

S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, 2009.

S. Bornholdt and H. G. Schuster. *Handbook of graphs and networks: from the Genome to the Internet*. John Wiley and Sons, 2003.

A. Borst and F. E. Theunissen. Information theory and neural coding. *Nature Neuroscience*, 2(11):947–57, 1999.

X. Boyen, N. Friedman, and D. Koller. Discovering the hidden structure of complex dynamic systems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 91–100, Stockholm, Sweden, 1999. Morgan Kaufmann.

V. Braitenberg and A. Schüz. *Cortex: Statistics and Geometry of Neuronal Connectivity*. Springer, 2nd edition, 1998.

I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Harri Deutsch, 4th edition, 1999.

E. N. Brown, R. E. Kass, and P. P. Mitra. Multiple neural spike train data analysis: state-of-the-art and future challenges. *Nature Neuroscience*, 7(5): 456–461, 2004.

V. H. Brun, M. K. Otnass, S. Molden, H. A. Steffenach, M. P. Witter, M. B. Moser, and E. I. Moser. Place cells and place recognition maintained by direct entorhinal-hippocampal circuitry. *Science*, 296(5576):2243–6, 2002.

N. Brunel and M. C. W. van Rossum. Lapicque's 1907 paper: from frogs to integrate-and-fire. *Biological Cybernetics*, 97(5-6):337–339, 2007.

E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional systems. *Nature Reviews Neuroscience*, 10(3): 186–198, 2009.

W. Buntine. Theory refinement on bayesian networks. *Proceedings of Uncertainty in Artificial Intelligence*, pages 52–60, 1991.

W. L. Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering*, 8(2):195–210, 1996.

D. V. Buonomano and W. Maass. State-dependent computations: spatiotemporal processing in cortical networks. *Nature Reviews Neuroscience*, 10(2): 113–25, 2009.

J. Burge, T. Lane, H. Link, S. Qiu, and V. P. Clark. Discrete dynamic Bayesian network analysis of fMRI data. *Human Brain Mapping*, 30(1):122–37, 2009.

G. Buzsaki. Theta oscillations in the hippocampus. *Neuron*, 33(3):325–40, 2002.

A. J. Cadotte, T. B. DeMarse, P. He, and M. Ding. Causal measures of structure and plasticity in simulated and living neural networks. *PLoS ONE*, 3(10): e3355, 2008.

G. Casella and E. I. George. Explaining the Gibbs sampler. *The American Statistician*, 46(3):167–174, 1992.

V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

D. M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H.-J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer Verlag, 1996.

D. M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks: Search methods and experimental results. *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pages 112–128, 1995.

D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5: 1287–1330, 2004.

E. S. Chornoboy, L. P. Schramm, and A. F. Karr. Maximum-likelihood identification of neural point process systems. *Biological Cybernetics*, 59(4-5): 265–275, 1988.

K. J. Cios, W. Pedrycz, R. W. Swiniarski, and L. A. Kurgan. *Data Mining: A Knowledge Discovery Approach*. Springer, 2007.

D. L. Cohn. *Measure theory*. Birkhäuser, 1980.

E. C. Cooper and D. H. Lowenstein. Hippocampus, 2002.

G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, 1992.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

L. D. Costa, F. A. Rodrigues, G. Travieso, and P. R. V. Boas. Characterization of complex networks: A survey of measurements. *Advances in Physics*, 56(1): 167–242, 2007.

R. T. Cox. Probability, frequency and reasonable expectation. *American Journal of Physics*, 14(1):1–13, 1946.

A. Czurko, H. Hirase, J. Csicsvari, and G. Buzsaki. Sustained activation of hippocampal pyramidal cells by 'space clamping' in a running wheel. *The European Journal of Neuroscience*, 11(1):344–52, 1999.

V. V. Das. *Principles of Data Structures Using C and C++*. New Age International, 1st edition, 2006.

P. Dayan and L. F. Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. The MIT Press, 1st paperback edition, 2005.

L. de Almeida, M. Idiart, and J. E. Lisman. The input-output transformation of the hippocampal granule cells: from grid cells to place fields. *Journal of Neuroscience*, 29(23):7504–12, 2009.

J. Demas, S. J. Eglen, and R. O. Wong. Developmental loss of synchronous spontaneous activity in the mouse retina is independent of visual experience. *Journal of Neuroscience*, 23(7):2851–60, 2003.

R. Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer, 3rd edition, 2006.

M. T. Do, S. H. Kang, T. Xue, H. Zhong, H. W. Liao, D. E. Bergles, and K. W. Yau. Photon capture and signalling by melanopsin retinal ganglion cells. *Nature*, 457(7227):281–7, 2009.

N. Dojer, A. Gambin, A. Mizera, B. Wilczynski, and J. Tiuryn. Applying dynamic Bayesian networks to perturbed gene expression data. *BMC Bioinformatics*, 7:249, 2006.

D. L. Dowe, S. Gardner, and G. Oppy. Bayes not bust! Why simplicity is no problem for Bayesians. *The British Journal for the Philosophy of Science*, 58: 709–754, 2007.

J. E. Dowling. *The retina: an approachable part of the brain*. Harvard University Press, 1987.

R. Eberhart, Y. Shi, and J. Kennedy. *Swarm Intelligence*. Artificial Intelligence. Morgan Kaufmann, 2001.

C. Echtermeyer, T. V. Smulders, and A. V. Smith. Causal pattern recovery from neural spike train data using the Snap Shot Score. *Journal of Computational Neuroscience*, to appear, DOI: 10.1007/s10827-009-0174-2, 2009.

S. R. Eddy. What is dynamic programming? *Nature Biotechnology*, 22(7): 909–10, 2004.

M. Eichler. On the evaluation of information flow in multivariate systems by the directed transfer function. *Biological Cybernetics*, 94(6):469–82, 2006.

A. D. Ekstrom, M. J. Kahana, J. B. Caplan, T. A. Fields, E. A. Isham, E. L. Newman, and I. Fried. Cellular networks underlying human spatial navigation. *Nature*, 425(6954):184–8, 2003.

S. Eldawlatly, Y. Zhou, R. Jin, and K. Oweiss. Reconstructing functional neuronal circuits using dynamic Bayesian networks. In *30th Annual International IEEE Engineering in Medicine and Biology Society (EMBS) Conference*, volume 2008, pages 5531–4, Vancouver, British Columbia, Canada, 2008.

G. B. Ermentrout and N. Kopell. Parabolic bursting in an excitable system coupled with a slow oscillation. *SIAM Journal on Applied Mathematics*, 46 (2):233–253, 1986.

T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27 (8):861–874, 2006.

W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. John Wiley and Sons, 3rd edition, 1950.

R. P. Feynman, R. B. Leighton, and M. Sands. The principles of statistical mechanics. In *Lectures on Physics: Mainly Mechanics, Radiation and Heat*, volume 1. Addison Wesley, 1963.

M. E. Fisher. Critical phenomena. In F. J. W. Hahne, editor, *Critical phenomena*, volume Lecture notes in Physics, pages 1–139. Springer, Berlin, 1983.

R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6): 345, 1962.

A. Fog. Optimizing software in C++: An optimization guide for Windows, Linux and Mac platforms, 14/01/2008, 2008.

T. C. Foster, C. A. Castro, and B. L. McNaughton. Spatial selectivity of rat hippocampal neurons: dependence on preparedness for movement. *Science*, 244(4912):1580–2, 1989.

N. Fourcaud-Trocme, D. Hansel, C. van Vreeswijk, and N. Brunel. How spike generation mechanisms determine the neuronal response to fluctuating inputs. *Journal of Neuroscience*, 23(37):11628–11640, 2003.

M. Franzius, R. Vollgraf, and L. Wiskott. From grids to places. *Journal of Computational Neuroscience*, 22(3):297–9, 2007.

N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *14th International Conference on Machine Learning (ICML 1997)*, pages 125–133, Nashville, Tennessee, USA, 1997. Morgan Kaufmann.

N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 252–262, 1996.

N. Friedman and D. Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50 (1-2):95–125, 2003.

N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 139–147, 1998.

N. Friedman, M. Goldszmidt, and A. Wyner. Data analysis with Bayesian networks: A bootstrap approach. In *Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 206–215, San Francisco, 1999a. Morgan Kaufman.

N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. *Proceedings of Uncertainty in Artificial Intelligence*, pages 206–215, 1999b.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

K. J. Friston. Functional and effective connectivity in neuroimaging: A synthesis. *Human Brain Mapping*, 2:56–78, 1994.

T. Furukawa, E. M. Morrow, and C. L. Cepko. Crx, a novel otx-like homeobox gene, shows photoreceptor-specific expression and regulates photoreceptor differentiation. *Cell*, 91(4):531–541, 1997.

M. Fyhn, T. Hafting, A. Treves, M. B. Moser, and E. I. Moser. Hippocampal remapping and grid realignment in entorhinal cortex. *Nature*, 446(7132):190–194, 2007.

F. Gabbianti and C. Koch. Principles of spike train analysis. In C. Koch and I. Segev, editors, *Methods in Neuronal Modeling: From Ions to Networks*, page 671. MIT Press, 1998.

R. F. Galan. On how network architecture determines the dominant patterns of spontaneous neural activity. *PLoS ONE*, 3(5):e2148, 2008.

G. Ganis and S. Kosslyn. Multiple mechanisms of top-down processing in vision. In S. Funahashi, editor, *Representation and Brain*, pages 21–46. Springer, 2007.

G. L. Gerstein and A. M. Aertsen. Representation of cooperative firing activity among simultaneously recorded neurons. *Journal of Neurophysiology*, 54(6): 1513–28, 1985.

G. L. Gerstein and D. H. Perkel. Simultaneously recorded trains of action potentials: analysis and functional interpretation. *Science*, 164(881):828–30, 1969.

G. L. Gerstein, D. H. Perkel, and J. E. Dayhoff. Cooperative firing activity in simultaneously recorded populations of neurons: detection and measurement. *Journal of Neuroscience*, 5(4):881–9, 1985.

W. Gerstner and W. M. Kistler. *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, Cambridge, 1st edition, 2002.

L. M. Giocomo, E. A. Zilli, E. Fransen, and M. E. Hasselmo. Temporal frequency of subthreshold oscillations scales with entorhinal grid cell field spacing. *Science*, 315(5819):1719–1722, 2007.

M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch. *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*, volume 59 of *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 2002.

C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

L. Groarke. Following in the footsteps of Aristotle: the Chicago school, the glue-stick and the razor. *Journal of Speculative Philosophy*, 6(3):190–205, 1992.

T. Hafting, M. Fyhn, S. Molden, M. B. Moser, and E. I. Moser. Microstructure of a spatial map in the entorhinal cortex. *Nature*, 436(7052):801–806, 2005.

T. Hafting, M. Fyhn, T. Bonnevie, M. B. Moser, and E. I. Moser. Hippocampus-independent phase precession in entorhinal grid cells. *Nature*, 2008.

M. W. Hankins, S. N. Peirson, and R. G. Foster. Melanopsin: an exciting photopigment. *Trends in Neurosciences*, 31(1):27–36, 2008.

H. Hanser. *Lexikon der Neurowissenschaft*, volume 1-4. Spektrum Akademischer Verlag, 2005.

W. K. Hastings. Monte-Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79–119, 1997.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks - the combination of knowledge and statistical-data. *Machine Learning*, 20(3): 197–243, 1995.

W. Heisenberg. Über den anschulichen Inhalt der quantentheoretischen Kinematik und Mechanik. *Zeitschrift für Physik*, 43:172–198, 1927.

W. Heisenberg. The physical content of quantum kinematics and mechanics. In J. A. Wheeler and W. H. Zurek, editors, *Quantum Theory and Measurement*, pages 62–68. Princeton University Press, Princeton, 1983.

A. V. M. Herz, T. Gollisch, C. K. Machens, and D. Jaeger. Modeling single-neuron dynamics and computations: A balance of detail and abstraction. *Science*, 314(5796):80–85, 2006.

H. Heuser. *Lehrbuch der Analysis*, volume 1 of *Mathematische Leitfäden*. B.G.Teubner, Stuttgart/Leipzig/Wiesbaden, 13th edition, 2000.

K. Heyman. The map in the brain: grid cells may help us navigate. *Science*, 312(5774):680–1, 2006.

C. A. Hidalgo, N. Blumm, A. L. Barabasi, and N. A. Christakis. A dynamic network approach for the study of human phenotypes. *PLoS Computational Biology*, 5(4):e1000353, 2009.

A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117(4):500–44, 1952.

J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999.

D. H. Hubel. *Eye, brain, and vision*. Scientific American Library, 1995.

D. Husmeier. Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. *Bioinformatics*, 19(17):2271–82, 2003.

E. Jaynes and G. Bretthorst. *Probability Theory: The Logic of Science*. Cambridge University Press, 1st edition, 2003.

A. H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer-Verlag, 2001.

P. Jezzard, P. M. Matthews, and S. M. Smith. *Functional MRI: An Introduction to Methods*. Oxford University Press, Oxford, 1st edition, 2001.

G. L. Jones and J. P. Hobert. Honest exploration of intractable probability distributions via Markov chain Monte Carlo. *Statistical Science*, 16(4):312–334, 2001.

Z. W. Junning Li and M. McKeown. Dynamic Bayesian networks (DBNs) demonstrate impaired brain connectivity during performance of simultaneous movements in Parkinson's disease. In *3rd IEEE International Symposium on Biomedical Imaging: Nano to Macro*, pages 964–967, 2006.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 84(Series D):35–45, 1960.

M. J. Kaminski and K. J. Blinowska. A new method of the description of the information flow in the brain structures. *Biological Cybernetics*, 65(3):203–10, 1991.

E. R. Kandel, J. H. Schwartz, and T. M. Jessel. *Principles of Neural Science*. McGraw-Hill, 4th edition, 2000.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, WA, Australia, 1995.

B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice Hall, 2nd edition, 1988.

S. Kim, S. Imoto, and S. Miyano. Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data. *Biosystems*, 75(1-3):57–65, 2004.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

U. B. Kjaerulff and A. L. Madsen. *Bayesian networks and influence diagrams*. Information Science and Statistics. Springer, 2008.

C. Koch, C.-H. Mo, and W. Softky. Single-cell models. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, page 1344. Bradford book, 2nd edition, 2003.

K. Kording. Decision theory: What "should" the nervous system do? *Science*, 318(5850):606–610, 2007.

W. Lam and F. Bacchus. Learning Bayesian belief networks: an approach based on the MDL principle. *Computational Intelligence*, 10(3):269–293, 1994.

L. Lapicque. Recherches quantitatives sur l`excitation electrique des nerfs traitee comme une polarization. *Journal de Physiologie et Pathologie Général*, (9):620–635, 1907.

S. L. Lauritzen. *Graphical Models*. Oxford University Press, Oxford, 1996.

D. Lazer, A. Pentland, L. Adamic, S. Aral, A. L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne. Computational social science. *Science*, 323 (5915):721–723, 2009.

C. R. Legendy and M. Salcman. Bursts and recurrences of bursts in the spike trains of spontaneously active striate cortex neurons. *Journal of Neurophysiology*, 53(4):926–39, 1985.

S. Leutgeb, J. K. Leutgeb, A. Treves, M. B. Moser, and E. I. Moser. Distinct ensemble codes in hippocampal areas CA3 and CA1. *Science*, 305(5688): 1295–8, 2004.

J. Li, Z. J. Wang, and M. J. McKeown. A framework for group analysis of fMRI data using dynamic Bayesian networks. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 5992–5, 2007.

B. G. Lindsey and G. L. Gerstein. Two enhancements of the gravity algorithm for multiple spike train analysis. *Journal of Neuroscience Methods*, 150(1): 116–127, 2006.

T. J. Loredo. From Laplace to supernova Sn 1987a - Bayesian-inference in Astrophysics. *Maximum Entropy and Bayesian Methods*, 39:81–142, 1990.

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 1st edition, 2003.

D. Madigan and A. E. Raftery. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association*, 89(428):1535–1546, 1994.

L. Maffei and L. Galli-Resta. Correlation in the discharges of neighboring rat retinal ganglion cells during prenatal life. *Proceedings of the National Academy of Sciences of the United States of America*, 87(7):2861–4, 1990.

J. C. Magee. Dendritic integration of excitatory synaptic input. *Nature Reviews Neuroscience*, 1(3):181–90, 2000.

V. A. Makarov, F. Panetsos, and O. de Feo. A method for determining neural connectivity and inferring the underlying network dynamics using extracellular spike recordings. *Journal of Neuroscience Methods*, 144(2):265–79, 2005.

H. Markram. The Blue Brain Project. *Nature Reviews Neuroscience*, 7(2): 153–60, 2006.

P. M. Matthews and P. Jezzard. Functional magnetic resonance imaging. *Journal of Neurology Neurosurgery and Psychiatry*, 75(1):6–12, 2004.

R. McHenry. The new encyclopaedia britannica, 1993.

J. McNames. Optimal rate filters for biomedical point processes. In *IEEE Engineering in Medicine and Biology Society*, volume 1, pages 145–8, 2005.

M. Meister, R. O. Wong, D. A. Baylor, and C. J. Shatz. Synchronous bursts of action potentials in ganglion cells of the developing mammalian retina. *Science*, 252(5008):939–43, 1991.

N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

M. A. Moita, S. Rosis, Y. Zhou, J. E. LeDoux, and H. T. Blair. Hippocampal place cells acquire location-specific responses to the conditioned stimulus during auditory fear conditioning. *Neuron*, 37(3):485–97, 2003.

C. Molter and Y. Yamaguchi. Impact of temporal coding of presynaptic entorhinal cortex grid cells on the formation of hippocampal place fields. *Neural Netw*, 21(2-3):303–10, 2008.

D. W. Mount. *Bioinformatics: Sequence and Genome Analysis*. CSHL Press, Cold Spring Harbor, New York, 2nd edition, 2004.

R. U. Muller, E. Bostock, J. S. Taube, and J. L. Kubie. On the directional firing properties of hippocampal place cells. *Journal of Neuroscience*, 14(12): 7235–51, 1994.

K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, MIT Artificial Intelligence Laboratory, 1999.

K. P. Murphy. An introduction to graphical models. Technical report, Intel Research, 2001.

R. Narasimhan and Y. Nievergelt. *Functions of complex variables*. Birkhäuser, 2001.

M. Nawrot, A. Aertsen, and S. Rotter. Single-trial estimation of neuronal firing rates: From single-neuron spike trains to population activity. *Journal of Neuroscience Methods*, 94(1):81–92, 1999.

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical report, University of Toronto, 25. September 1993, 1993.

J. Neyman and E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London*, Series A, 231:289–337, 1933.

P. L. Nunez and R. Srinivasan. Electroencephalogram. *Scholarpedia*, 2(2):1348, 2007.

D. Q. Nykamp. Revealing pairwise coupling in linear-nonlinear networks. *SIAM Journal on Applied Mathematics*, 65(6):2005–2032, 2005.

H. Nyquist. Certain topics in telegraph transmission theory. *Proceedings of the IEEE*, 90(2):280–305, 1928.

M. Okatan, M. A. Wilson, and E. N. Brown. Analyzing functional connectivity using a network likelihood model of ensemble neural spiking activity. *Neural Computation*, 17(9):1927–1961, 2005.

J. O'Keefe and J. Dostrovsky. The hippocampus as a spatial map. Preliminary evidence from unit activity in the freely-moving rat. *Brain research*, 34(1): 171–5, 1971.

J. O'Keefe and L. Nadel. *The Hippocampus as a Cognitive Map*. Oxford University Press, 1979.

D. S. Olton and R. J. Samuelson. Remembrance of places passed - spatial memory in rats. *Journal of Experimental Psychology-Animal Behavior Processes*, 2(2):97–116, 1976.

C. W. Oyster. *The Human Eye: Structure and Function*. Sinauer Associates, Sunderland, Massachusetts, 1999.

G. Paxinos. *The rat nervous system*. Academic Press, 1995.

J. Pearl. Bayesian networks: a model of self-activated memory for evidential reasoning. In *7th Annual Conference of the Cognitive Science Society*, pages 329–334, Irvine, California, 1985.

J. Pearl. *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 1988.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, UK, 2000.

D. Pe'er. *From Gene Expression to Molecular Pathways*. PhD thesis, 2003.

D. Pe'er. Bayesian network analysis of signaling networks: a primer. *Science's STKE*, 2005(281):pl4, 2005.

D. H. Perkel, G. L. Gerstein, and G. P. Moore. Neuronal spike trains and stochastic point processes. II. Simultaneous spike trains. *Biophysical Journal*, 7(4):419–40, 1967.

B. E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d'Alche Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19 Suppl 2:ii138–48, 2003.

J. W. Pillow, J. Shlens, L. Paninski, A. Sher, A. M. Litke, E. J. Chichilnisky, and E. P. Simoncelli. Spatio-temporal correlations and visual signalling in a complete neuronal population. *Nature*, 454(7207):995–U37, 2008.

G. J. Quirk, R. U. Muller, and J. L. Kubie. The firing of hippocampal place cells in the dark depends on the rat's recent experience. *Journal of Neuroscience*, 10(6):2008–17, 1990.

R. Q. Quiroga and S. Panzeri. Extracting information from neuronal populations: information theory and decoding approaches. *Nature Reviews Neuroscience*, 10(3):173–185, 2009.

J. C. Rajapakse and J. Zhou. Learning effective brain connectivity with dynamic Bayesian networks. *Neuroimage*, 37(3):749–60, 2007.

J. C. Rajapakse, Y. Wang, X. Zheng, and J. Zhou. Probabilistic framework for brain connectivity from functional MR images. *IEEE Transactions on Medical Imaging*, 27(6):825–33, 2008.

M. J. E. Richardson. Firing-rate response of linear and nonlinear integrate-and-fire neurons to modulated current-based and conductance-based synaptic drive. *Physical Review E*, 76(2), 2007.

F. Rieke, D. Warland, R. d. R. van Steveninck, and W. Bialek. *Spikes: exploring the neural code*. MIT Press, 1st paperback edition, 1999.

C. P. Robert and G. Casella. The multi-stage Gibbs sampler. In *Monte Carlo statistical methods*, pages 337–370. Springer, 2nd edition, 2004.

R. W. Robinson. Counting labeled acyclic digraphs. In F. Harary, editor, *New directions in the theory of graphs*, pages 239–273. Academic Press, New York, 1973.

S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Computation*, 11(2):305–45, 1999.

K. Sameshima and L. A. Baccalá. Using partial directed coherence to describe neuronal ensemble interactions. *Journal of Neuroscience Methods*, 94:93–103, 1999.

G. Schwarz. Estimating dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793, 1974.

E. Sernagor, S. Eglen, B. Harris, and R. Wong. *Retinal Development*. Cambridge University Press, 1st edition, 2006.

C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

A. Siegel and H. N. Sapru. *Essential Neuroscience*. Lippincott Williams & Wilkins, 2007.

P. Smith. *Applied data structures with C++*. Jones & Bartlett Publishers, 1st edition, 2004.

V. A. Smith, J. Yu, T. V. Smulders, A. J. Hartemink, and E. D. Jarvis. Computational inference of neural information flow networks. *PLoS Computational Biology*, 2(11):e161, 2006.

O. Sporns. Graph theory methods for the analysis of neural connectivity patterns. In R. Kötter, editor, *Neuroscience Databases: A Practical Guide*, pages 169–183. Springer, 2003.

O. Sporns and G. Tononi. Classes of network connectivity and dynamics. *Complexity*, 7(1):28–38, 2002.

O. Sporns, G. Tononi, and G. M. Edelman. Connectivity and complexity: the relationship between neuroanatomy and brain dynamics. *Neural Networks*, 13(8-9):909–22, 2000.

O. Sporns, D. R. Chialvo, M. Kaiser, and C. C. Hilgetag. Organization, development and function of complex brain networks. *Trends in Cognitive Sciences*, 8(9):418–25, 2004.

R. P. Stanley. Acyclic orientations of graphs (reprinted from discrete mathematics, vol 5, pg 171-178, 1973). *Discrete Mathematics*, 306(10-11):905–909, 2006.

R. B. Stein. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5:173–94, 1965.

C. Stosiek, O. Garaschuk, K. Holthoff, and A. Konnerth. In vivo two-photon calcium imaging of neuronal networks. *Proceedings of the National Academy of Sciences of the United States of America*, 100(12):7319–7324, 2003.

S. H. Strogatz. Exploring complex networks. *Nature*, 410(6825):268–76, 2001.

D. Y. Takahashi, L. A. Baccalá, and K. Sameshima. Connectivity inference between neural structures via partial directed coherence. *Journal of Applied Statistics*, 34(10):1259–1273, 2007.

H. Tanizaki. *Nonlinear Filters: Estimation and Applications*. Springer, 1996.

R. D. Traub, D. Contreras, M. O. Cunningham, H. Murray, F. E. LeBeau, A. Roopun, A. Bibbig, W. B. Wilent, M. J. Higley, and M. A. Whittington. Single-column thalamocortical network model exhibiting gamma oscillations, sleep spindles, and epileptogenic bursts. *Journal of Neurophysiology*, 93(4): 2194–232, 2005.

W. Truccolo, U. T. Eden, M. R. Fellows, J. P. Donoghue, and E. N. Brown. A point process framework for relating neural spiking activity to spiking history, neural ensemble, and extrinsic covariate effects. *Journal of Neurophysiology*, 93(2):1074–1089, 2005.

M. Tsodyks, T. Kenet, A. Grinvald, and A. Arieli. Linking spontaneous activity of single cortical neurons and the underlying functional architecture. *Science*, 286(5446):1943–6, 1999.

A. Tucker, Y. H. Liu, and D. Garway-Heath. Spatial operators for evolving dynamic Bayesian networks from spatio-temporal data. In *Genetic and Evolutionary Computation (GECCO), Part II*, volume 2724, pages 2360–2371, Chicago, IL, 2003.

G. van Rossum and et al. Python language website, http://www.python.org/.

T. Verma and J. Pearl. Equivalence and synthesis of causal models. In P. Bonissone, M. Henrion, L. Kanal, and J. Lemmer, editors, *Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, Boston, MA, 1990. Morgan Kaufmann.

R. P. Vertes. Hippocampal theta rhythm: a tag for short-term memory. *Hippocampus*, 15(7):923–35, 2005.

J. D. Victor. Spike train metrics. *Current Opinion in Neurobiology*, 15(5): 585–92, 2005.

J. D. Victor and K. P. Purpura. Nature and precision of temporal coding in visual cortex: a metric-space analysis. *Journal of Neurophysiology*, 76(2): 1310–26, 1996.

J. D. Victor and K. P. Purpura. Metric-space analysis of spike trains: Theory, algorithms, and application. *Network*, 8:127–164, 1997.

T. P. Vogels, K. Rajan, and L. F. Abbott. Neural network dynamics. *Annual Review of Neuroscience*, 28:357–76, 2005.

C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.

S. Warshall. A theorem on boolean matrices. *Journal of the ACM*, 9(1):11–12, 1962.

D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

M. A. Wilson and B. L. McNaughton. Dynamics of the hippocampal ensemble code for space. *Science*, 261(5124):1055–8, 1993.

R. O. Wong. Retinal waves and visual system development. *Annual Review of Neuroscience*, 22:29–47, 1999.

R. O. Wong, M. Meister, and C. J. Shatz. Transient period of correlated bursting activity during development of the mammalian retina. *Neuron*, 11(5):923–38, 1993.

R. O. Wong, A. Chernjavsky, S. J. Smith, and C. J. Shatz. Early functional neural networks in the developing retina. *Nature*, 374(6524):716–8, 1995.

E. R. Wood, P. A. Dudchenko, R. J. Robitsek, and H. Eichenbaum. Hippocampal neurons encode information about different types of memory episodes occurring in the same location. *Neuron*, 27(3):623–33, 2000.

S. Wright. Correlation and causation. *The Journal of Agricultural Research*, 20 (7):557–585, 1921.

S. Yamaoka and N. Hagino. Spontaneous septal neuron activity in the rat. *Brain research*, 67(1):147–52, 1974.

J. Yu. *Developing Bayesian Network Inference Algorithms to Predict Causal Functional Pathways in Biological Systems*. PhD thesis, 2005.

R. Yuste and D. W. Tank. Dendritic integration in mammalian neurons, a century after Cajal. *Neuron*, 16(4):701–16, 1996.

K. Zhang, I. Ginzburg, B. L. McNaughton, and T. J. Sejnowski. Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells. *Journal of Neurophysiology*, 79(2):1017–44, 1998.

M. Zou and S. D. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–9, 2005.