Reconstructing Gene Networks from Large Scale Gene Expression Data

by

Patrik D'haeseleer

Industrieel Ingenieur, Electrical Engineering, Katholieke Industriele Hogeschool Oost-Vlaanderen, Belgium, 1988
Burgerlijk Ingenieur, Electrical Engineering, Rijksuniversiteit Gent, Belgium, 1991
M.S., Computer Science, Stanford University, 1993

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Science

The University of New Mexico

Albuquerque, New Mexico

December, 2000

Dedication

To Carla, for her love and support.

Acknowledgments

As much as a dissertation is supposed to be the work of one person, this one might never have gotten finished but for the support and encouragement of some key people. First and foremost, I would like to thank my advisor, Stephanie Forrest, who—after a year of prodding—let me drag her into the (then esoteric) field of gene expression analysis. Despite the fact that this research topic is entirely outside of her own expertise, I could not have asked for a better advisor. Steph has a remarkably good taste in research, an excellent sense of strategy, ... and a whip behind the door for when procrastination gets the better of her students. I greatly appreciate her open personality, her sense of humor, and the freedom she gave me to "do my own thing". She showed me what it takes to lead a successful, vibrant, and fun research group. I feel truly privileged to have been her student.

No doubt this dissertation would have looked very different if I had not met Roland Somogyi at the Santa Fe Institute. Not only because he and his group at NIH—including Stefanie Fuhrman and Xiling Wen—produced the first high-quality data sets on which I could apply my network models, but also because of his open-mindedness regarding computational approaches and his willingness to freely share his data. His enthusiasm and very vocal support for the sort of analyses I have been pursuing has opened many doors for me.

The Santa Fe institute itself has played a key role in shaping the ideas that went into this work, and for that matter, into my view of interdisciplinary science. SFI was one of the main reasons I came to UNM (the other main one being Stephanie), and its open atmosphere and endless stream of interesting visitors have made me a loyal fan. Some of my first thoughts about gene networks as a possible dissertation topic date from the '95 Complex Systems Summer School. The Institute also provided the catalyst for my meeting with Roland, when he came to SFI to present his first gene expression data set, seeking help in analyzing this large-scale data. Next, it brought me into contact with Andreas Wagner, who at the time was doing a post-doc at SFI. Andreas has an incredibly sharp mind when it comes to biology and statistics, and an exacting taste for "good science". His presence on my dissertation committee has definitely helped to keep me honest regarding the biology, as well as substituting for some of Stephanie's lack of expertise in this field.

Dave Peabody, the second biologist on my committee, helped spark my love for the intricate beauty of molecular biology, through the "Advanced cellular and molecular biology" course I took with him. He also helped me out by being a sounding board for my very earliest gene network ideas, when neither of us knew whether this was going somewhere. My fourth committee member, Barak Pearlmutter, is one of the smartest people I know, and just happened to be one of the experts in the specific sort of neural network models I was contemplating. His help on the neural network side—including the use of his cbp code have been essential in the later chapters.

I have had a great time being a graduate student at UNM, thanks in large part to the incredibly stimulating environment of the Adaptive Computation group. The Bad Boys—Terry Jones, Ron Hightower and Derek Smith—set the tone with the irreverent attitude, playfulness and straight-talking honesty I have come to appreciate so much. Some of my closest friends originated in this group. Jason Stewart was one of the very first people I met at UNM. Apart from being a great friend, he has helped me a lot with his own background in biology, and in dragging me through this final stretch. Anil Somayaji also carries the title of "Most Valuable Group Member". I don't know how we could have survived without his technical expertise. His constant willingness to help out, and his overall sunny disposition make him a great person to have around. During his short stay at UNM, Carlo Maley quickly found a way into our hearts. I have fond memories of the many late-night roleplaying sessions at his place. A late addition, Dennis Chao is quickly achieving a reputation as a creative genius—from Doom as a tool for process control, to power laws of "Wheeee". Never a dull moment (and rarely a productive one...) with Dennis around. I include Lance Williams here as an honorary grad student. Lance is the single most approachable professor I know, with the possible exception of Stephanie, and his quick wit and warm personality were greatly appreciated. Many thanks also to Steve Hofmeyr, Jim Inoue, Christy Warrender, Ken Ingham, Sean Elicker, Geoff and Julie Hunsicker, and all the rest for making my life that much more interesting, and for their help finalizing my dissertation. I am proud to call these people my friends. Special thanks also to Jason and Anil for proofreading almost the entire manuscript, to Jim and Lance for preserving my sanity by dragging me out of my office to play ping-pong—even at 5 a.m.—and to Dennis for comic relief.

Last—but not least—my thanks to Carla Koopal for being in my life the past six years. And especially for her patience in these past couple of months.

This research has been partially supported by grants from the National Science Foundation (IRI-9157644, IRI-9711199, CDA-9503064, and ANIR-9986555), the Office of Naval Research (N00014-95-1-0364, N00014-99-1-0417), Defense Advanced Projects Agency (AGR F30602-00-2-0584), and the Intel Corporation.

Reconstructing Gene Networks from Large Scale Gene Expression Data

by

Patrik D'haeseleer

ABSTRACT OF DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy
Computer Science

The University of New Mexico

Albuquerque, New Mexico

December, 2000

Reconstructing Gene Networks from Large Scale Gene Expression Data

by

Patrik D'haeseleer

Industrieel Ingenieur, Electrical Engineering, Katholieke
Industriele Hogeschool Oost-Vlaanderen, Belgium, 1988
Burgerlijk Ingenieur, Electrical Engineering,
Rijksuniversiteit Gent, Belgium, 1991
M.S., Computer Science, Stanford University, 1993
Ph.D., Computer Science, University of New Mexico, 2000

Abstract

New technologies have been developed to measure the expression level of thousands of genes simultaneously. These genomic-scale snapshots of gene expression—i.e. how much each gene is "turned on"—are creating a revolution in biology. However, such large-scale data also creates an urgent need for computational tools to make sense of it all.

Genes encode proteins, some of which in turn regulate other genes. Now that the human genome is within our grasp, we need to start thinking of the next step: determining the structure of this intricate network of genetic regulatory interactions. Many different modeling methodologies could be used to model such gene networks.

Analysis of various network models shows that, given a sufficiently constrained model, data requirements should scale well. Additive regulation models—where the regulatory inputs are combined using a weighted sum—can be used as a first-order approximation to the gene network. We can infer regulatory interactions directly from the data, by fitting these simple network models to large scale gene expression data. The amount of data typically is insufficient to derive a fully determined network model. Nevertheless, we can extract the most well-determined interactions in the network, using knowledge of the error levels on the measurements in a Monte-Carlo analysis of the resulting variability in the network parameters.

Using this methodology, a linear model is fit to a data set on development and injury in the central nervous system. The results compare favorably with the literature on the genes involved. Next, a more realistic, nonlinear model is presented, resulting in a set of nonlinear differential equations equivalent to a specific type of recurrent neural network. This model should allow for a closer fit with the biological reality, but requires more computational effort to fit to real data sets.

Contents

List of Figures **xvi**

List of Tables **xviii**

Glossary **xix**

1 Introduction **1**

1.1 Functional Genomics 1

1.2 An intermediate representation 4

1.3 Additive regulation models: A simple model of gene interaction . . . 5

1.4 Caution to the reader 9

1.5 Overview 11

2 Large scale measurements of internal cell state **13**

2.1 What are the important variables? 13

2.2 mRNA levels 16

2.2.1 Oligonucleotide and cDNA microarrays 16

2.2.2 Other approaches 20

2.2.3 Truly large-scale gene expression 22

2.3 Protein levels 22

3 Related work **24**

3.1 Fitting known networks 24

3.2 Gene circuits: spatial patterns in development 26

3.3 Correlation Metric Construction 27

3.4 Reverse engineering of Boolean networks 28

3.5 Co-expression clustering 29

3.5.1 Extraction of regulatory motifs 30

3.5.2 Inference of functional annotation 31

3.5.3 Classification 32

3.5.4 Which clustering method to use? 33

3.5.5 Related methods 35

3.6 Networks of clusters 36

3.7 Bayesian networks 37

4 Modeling issues **40**

4.1 Level of biochemical detail 40

4.2 Boolean or continuous 41

4.3 Deterministic or stochastic 43

4.4 Spatial or non-spatial 44

4.5 Forward and inverse modeling 45

5 Data requirements for network inference **46**

5.1 Sample complexity 47

5.1.1 General network models 47

5.1.2 Boolean, fully connected 49

5.1.3 Boolean, connectivity K 49

5.1.4 Boolean, linearly separable, connectivity K 51

5.1.5 Continuous, additive, fully connected 52

5.1.6 Continuous, additive, connectivity K 52

5.1.7 Clustering 53

5.1.8 Summary 55

5.2 The Curse of Dimensionality 58

5.3 Types of data 59

5.4 Combining different data types 60

6 A linear model of CNS development and injury **61**

6.1 A first-order approximation 61

6.2 Data sets 65

6.3 Fitting the model 68

6.4 Results and validation 73

6.4.1 Biologically plausible properties? 74

6.4.2 Robust parameters 78

6.4.3 Results: Kainate parameters 80

6.4.4 Results: Gene-to-gene parameters 86

6.4.5 A control model 90

7 Modeling gene networks with recurrent neural networks 93

7.1 Nonlinear differential equations 94

7.2 Dynamic recurrent neural networks 96

7.3 Learning algorithms 97

7.3.1 Adjusting learning rate using delta-bar-delta 100

7.3.2 ML interpretation of minimizing E 102

7.4 The Curse of Dimensionality revisited 103

7.4.1 Simplifying the model 104

7.4.2 Early stopping 108

7.5 Experiments with synthetic data sets 109

7.5.1 Generating synthetic data sets 109

7.5.2 Evaluating the networks 112

7.5.3 Results 116

7.5.4 The next step: Calculating Z-scores for the weights 117

8 Conclusions **126**

8.1 The story so far... 126

8.2 Directions for future research 128

8.2.1 Refinements of the linear model 128

8.2.2 Refinements of the neural network model 129

8.2.3 Incorporating biological knowledge 132

8.3 A look towards the future 133

Appendices **137**

A Overview of clustering methods **138**

A.1 Distance measures and preprocessing 139

A.2 Clustering algorithms 140

B Fitting the linear model **145**

C Linear models based on randomized data **148**

C.1 Simultaneously matching slopes and expression levels 148

C.2 Matching the covariance matrix 149

D Derivation of Backpropagation Through Time **153**

D.1 General derivation 153

D.2 BPTT for gene networks 156

D.3 Recurrent Backpropagation 159

References **161**

List of Figures

1.1 Example of a simple regulatory network. 5

2.1 In-situ synthesis of an oligonucleotide chip. 17

2.2 Gene expression measurement using a cDNA microarray. 19

6.1 Schematic illustration of a node in the linear network model. 63

6.2 Schematic illustration of a node in the linear model for the CNS data. 69

6.3 Original and reconstructed time series for three selected genes. . . . 73

6.4 Histogram of average parameter values $\bar{w}$. 75

6.5 Average magnitude of parameters vs. Z-score. 81

6.6 Estimates for kainate parameters K_i and their Z-score Z_{K_i}. 85

6.7 Alternative models for the interaction between BDNF, IGF II, and S100β. 88

7.1 Schematic illustration of a node in the nonlinear network model. . . 95

7.2 Unrolling a recurrent network. 99

7.3 A simple noise model for the CNS data sets. 112

7.4 Original network weights versus trained network weights. 113

7.5 Effect of early stopping on performance. 121

7.6 Effect of weight decay on performance. 122

7.7 Effect of weight elimination on performance. 123

7.8 Effect of weight decay in combination with weight elimination. . . . 124

7.9 Effect of weight elimination in combination with weight decay. . . . 125

List of Tables

5.1 Sample complexity for various network models. 56

6.1 Robust kainate parameters 82

6.2 Robust gene-to-gene parameters 87

Glossary

2D-PAGE	Two-dimensional polyacrylamide gel electrophoresis.
5HTR	Serotonin Receptors.
5-HT_{1B}	Serotonin (5-hydroxytryptamine) receptor 1B.
A_i	Maximum transcription rate for gene i.
AChE	Acetylcholinesterase.
AChR	Acetylcholine Receptors.
Autoreceptor	An autoreceptor, present at a nerve ending, is a receptor that regulates, via positive or negative feedback processes, the synthesis and/or release of its own physiological ligand.
b_i	Bias factor for gene i.
BDNF	Brain-Derived Neurotrophic Factor.
BPTT	Backpropagation Through Time.
cDNA	Complementary DNA; complementary single-stranded DNA copy of a messenger RNA, produced by reverse transcription.
CMC	Correlation Metric Construction.

CZE	Capillary Zone Electrophoresis.
D_i	Decay rate of mRNA for gene gene i.
DAG	Directed Acyclic Graph.
DBN	Dynamic Bayesian network.
Diauxic shift	Shift from anaerobic to aerobic metabolism.
DNA	Deoxy riboNucleic Acid; carrier of the genetic information in cells.
E	Neural network error term.
η	Neural network learning rate.
EM	Expectation Maximization algorithm.
EST	Expressed Sequence Tag; short DNA sequence derived from expressed mRNA.
Eukaryote	Organism composed of one or more cells with visibly evident nuclei.
G67I80/86	Glutamate decarboxylase 67 (GAD67) splice variants I80 and I86.
G67I86	Glutamate decarboxylase 67 (GAD67) splice variant I86.
GA	Genetic Algorithm.
GABA	γ-aminobutyric acid; a fast-acting neurotransmitter.
GABA-R	GABA Receptors.
GAD	Glutamic acid decarboxylase; synthesizes GABA from glutamate.
GEM	Gene Expression Matrix.
GFAP	Glial Fibrillary Acidic Protein.

Glossary

GluR	Glutamate Receptors.
GRα4	$GABA_A$ receptor subunit α4 (similar for GRβ2, γ1, ...).
Hypertrophy	A considerable increase in the size of an organ or tissue, caused by enlargement of its cellular components.
ICS	Intracellular Signaling.
IGF II	Insulin-like growth factor II.
In situ	"In place".
In vitro	"In glass", i.e. within a controlled laboratory environment, rather than in the living organism.
In vivo	"In life", i.e. within the living organism.
K	Number of regulatory inputs per gene.
Ki	Effect of kainate level on change in expression level of gene i.
$\kappa(t)$	Kainate concentration at time t.
LBCL	Large B-Cell Lymphoma.
LOOCV	Leave-One-Out Cross Validation.
MDS	Multi-Dimensional Scaling.
ML	Maximum Likelihood.
mRNA	Messenger RNA; a complementary copy of a stretch of DNA encoding a gene.
Microarray	Array consisting of up to tens of thousands of probes, typically used for measurement of gene expression.

N	Number of genes.
n	Number of expression measurements.
NFM	Neurofilament Medium.
NME	Neurotransmitter Metabolizing Enzymes.
ORF	Open Reading Frame; putative gene sequence.
P_w	P-value of parameter w, indicating the probability that the true value of w is zero or has opposite sign from its mean value.
PCA	Principle Component Analysis.
PCR	Polymerase Chain Reaction; a method for amplifying a specific DNA sequence using DNA polymerase.
Prokaryote	Cellular organism that does not have a distinct nucleus.
RBN	Random Boolean Network.
RBP	Recurrent Backpropagation.
RNA	RiboNucleic Acid.
RT-PCR	Reverse Transcriptase Polymerase Chain Reaction.
SA	Simulated Annealing.
SAGE	Serial Analysis of Gene Expression.
Sequencing	Determining the nucleotide sequence of DNA or RNA, or the amino acid sequence of a protein.
SOM	Self-Organized Map.

Glossary

SPN	Stochastic Petri Net.
SVD	Singular Value Decomposition.
SVM	Support Vector Machine.
T_i	Effect of tissue type on expression level of gene i.
τ	Indicator variable for tissue type ($\tau = 0$ for spinal cord, $\tau = 1$ for hippocampus).
TCP	T-complex protein.
Transcription	Copying DNA into RNA.
Translation	Synthesis of a protein from its corresponding mRNA.
w_{ji}	Effect of expression level of gene j on change in expression level of gene i.
$y_i(t)$	Expression level of gene i at time t.
Z_w	Z-score of parameter w, indicating how many standard deviations its mean is away from zero.

Chapter 1

Introduction

SAM:
It could all be coincidental.
JACK:
There are no coincidences, Sam.
Everything's connected, all along the line.
Cause and effect. That's the beauty of it.
Our job is to trace the connections and reveal them.
— Terry Gilliam ("Brazil")

1.1 Functional Genomics

All science is either physics or stamp collecting.
— Ernest Rutherford, physicist

Genes code for proteins, some of which in turn regulate other genes. This network of gene regulation, combined with protein interactions, can be very complex. The traditional approach to research in Molecular Biology has been an inherently local one, examining and collecting data on a single gene, a single protein or a single reaction at a time. This is, of course, the classical reductionist stance: To understand

the whole, one must first understand the parts. Over the years, this approach has led to some remarkable achievements, allowing us to make highly accurate biochemical models of such favorites as bacteriophage Lambda [183, 22].

However, with the advent of the "Age of Genomics" an entirely new class of data is emerging. As the goal of *structural* genomics—sequencing entire genomes—comes into sight, the focus is gradually shifting to *functional* genomics.

> Specifically, functional genomics refers to the development and application of global (genome-wide or system-wide) experimental approaches to assess gene function by making use of the information and reagents provided by structural genomics. It is characterized by high throughput or large scale experimental methodologies combined with statistical and computational analysis of the results. [107]

Biology used to be a *data-poor* science, out of necessity having to rely on carefully designed hypothesis and meticulously planned experiments. Over the past couple of years, however, it has been rapidly evolving into a *data-rich* field, opening up the possibility of data-driven research—for which Hood coined the term "discovery science" [1]—rather than hypothesis-driven research. Such analysis-without-hypothesis has often been compared pejoratively to a fishing expedition. But perhaps, as Geshwind [83] states, it is "fishing, but with a stick of dynamite in a stocked pond". Obviously, there is a trade-off to be made between *unbiased* analysis—allowing for the possibility of entirely innovative conclusions—and *uninformed* analysis—ignoring all the accumulated wisdom of the field.

Unfortunately, the arrival of this flood of large scale data has so far not been accompanied by an equal abundance of computational techniques to handle the data. Researchers who were used to looking at perhaps a few tens of measurements from very focused experiments are suddenly faced with literally tens of thousands or even

millions of measurements. Initially, analysis of this data was mainly of a descriptive nature, consisting of little more than lists of how many genes were previously unknown, which genes are under or overexpressed under certain circumstances, etc. More recently, simple statistical techniques such as clustering and classification are being discovered, and—occasionally—reinvented. The goal of this dissertation is to develop computational tools to analyze this data at a higher level of complexity, by attempting to determine the underlying network of regulatory interactions that causes the behavior observed in these large scale measurements.

Of course, this large-scale data is an equally valuable resource for researchers who are focusing on individual genes. But can we really expect to construct a detailed biochemical model of, say, an entire yeast cell with some 6000 genes (only about 1000 of which were defined before sequencing began, and about 50% of which are clearly related to other known genes), by analyzing each gene and determining all the binding and reaction constants one by one?

Rather than waiting until we have worked out all the biochemical details, we would like to be able to analyze such large systems in a genome-wide fashion at some intermediate level of representation, without having to go all the way down to the exact biochemical reactions. At the very least, such an intermediate-level analysis could help guide the traditional biochemical approach towards those genes most worthy of attention among these thousands of newly discovered genes. Ideally, a sufficiently predictive and explanatory model at an intermediate level might obviate the need for an exact understanding of the system at the biochemical level. For now, we will be satisfied with "cherry-picking" the most salient features of the regulatory networks, without trying to achieve an accurate model of the entire system.

1.2 An intermediate representation

Everything is deeply intertwingled.
— Theodor Holm Nelson

I intend to focus on genetic regulatory networks at the level of single cells. This ignores the extra complexity that comes with cell to cell interactions and spatial differentiation (see Reinitz and Sharp [185] in Section 3.2 for example), but is still of major importance to cellular biology. A biological system can be considered to be a state machine, where the change in internal state of the system depends on both its current internal state and any external inputs. The goal is to observe the state of a cell and how it changes under different circumstances, and from this to derive a model of how these state changes are generated. The state of a cell consists of all those variables—both internally and externally—which determine its behavior. Included are the concentrations of all the chemical species (DNA, RNA, proteins, metabolites, etc.) involved in the inner working of the cell, concentrations in the environment of the cell, receptors presented on the membrane, volume, position in the cell cycle, location of structural components within the cell, and so on. A sufficiently informative subset of these will have to be chosen, usually consisting of concentrations of certain key elements within the cell.

It is unlikely we will ever achieve a simultaneous measurement of the full set of important variables within a cell. In the immediate future, it seems likely we will primarily be focusing on mRNA data, plus perhaps protein data (see Chapter 2). Exogenous inputs or important intermediates which are missing in our set of measurements are impossible (or at least very difficult) to model. It should be emphasized that these models are, therefore, not intended to imply biochemical *mechanism*, but merely a higher-level view of regulation. This distinction is especially important for the small data sets used in Chapters 6 and 7.

The intermediate representation most familiar to molecular and cell biologists is a directed graph, with the nodes representing the key elements—often genes, proteins or metabolites—being modeled, and the arcs representing how these influence the production or destruction of others. To formalize this sort of description, we might want to add weights—positive or negative—to these arcs, and define how the inputs to a node interact. Figure 1.1 illustrates how a simple network model might be represented. Even though it consists of only six nodes, the dynamical behavior of the network is far from obvious. Nevertheless, the network representation provides a clear and concise summary of the regulatory interactions, and higher-level structures (such as the two pathways from a to e) can easily be extracted.

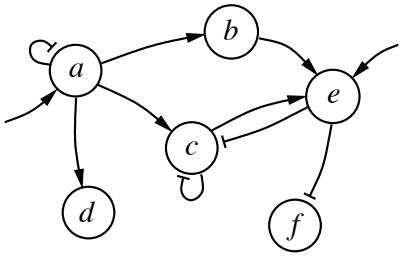

Figure 1.1: Example of a simple, 6-node regulatory network. For simplicity, no input-output mapping is specified, and interactions have been given a sign (regular arrowheads are positive, flat ones are negative) but not a specific weight. Nodes a and e receive external inputs (e.g. signaling molecules). Nodes a and c are auto-inhibitory, i.e. they will repress their own activation. Notice also the two pathways for upregulation of e by a.

1.3 Additive regulation models: A simple model of gene interaction

One of the simplest ways to model a system of interacting variables is to assume that the change in each variable over time is given by a weighted sum of all other

variables[1]:

$$\Delta y_i = \sum_j w_{ji} y_j + b_i \tag{1.1}$$

where y_i is the level of the ith variable, b_i is a bias term indicating whether i is expressed or not in the absence of regulatory inputs, and weight w_{ji} represents the influence of j on the regulation of i. We will say that A is a regulator of B if the network model predicts a causal relationship between the level of A and the change in level of B (i.e., an "arrow" in the network), regardless of the underlying mechanism of this regulation. Note that this is a more general interpretation of the terms "regulator" and "regulate" than is normally used in biology.

For a continuous-time system we get the corresponding differential equation:

$$\frac{dy_i}{dt} = \sum_j w_{ji} y_j + b_i \tag{1.2}$$

Because of the nature of interactions between regulatory factors, gene regulation is often context sensitive, e.g. A upregulates C, but only if B is present as well. The model presented here cannot implement such a nonlinear interaction between A and B in the regulation of C. However, the model should be able to extract the linear component of this regulation, i.e. that both A and B upregulate C, even if the regulation is not independent.

Obviously, an additive model like this will be a gross simplification for almost any natural system, but modeling a gene network with such a minimal model might allow us to extract at least the "Most Significant Bits" of information we're looking for: Which genes regulate which other genes (i.e. which interaction factors w_{ji} are

[1]With an additional noise component $\epsilon(t)$, such a system is generally called a first-order auto-regressive, or AR(1) time series model [95].

nonzero)? If gene j regulates gene i, is j an inducer or repressor of i (i.e. is w_{ji} positive or negative)?

In Chapter 6, we will examine a purely linear model such as this, apply it to real gene expression data, and compare the results with the literature on the genes involved.

Note that the variables in Equation 1.2 can theoretically become negative, or unboundedly large. Since these variables typically correspond to concentration levels, we may want to impose realistic upper and lower bounds. Most genes exhibit a sigmoidal dose response curve: As the concentration of the inducing regulatory signals increases, the gene activation at first increases slowly, then more rapidly, and finally saturates at a maximum level. For an added level of realism, we therefore add a sigmoidal transfer function to Equation 1.2:

$$\frac{dy_i}{dt} = S\Big(\sum_j w_{ji} y_j + b_i\Big) \tag{1.3}$$

where $S(\cdot)$ is some sigmoidal function, e.g. $S(x) = (1 + e^{-x})^{-1}$, $S(x) = \tanh(x)$, or a more biologically justified dose-response curve (although it should be noted that some studies indicate that the behavior of the entire network may not be very sensitive to the exact shape of the sigmoid [85]).

Note that the addition of a nonlinear response also allows us to model a large class of interesting nonlinear interactions between regulators. For example in the example above, where both A and B must be present to upregulate C, w_{AC} and w_{BC} individually may be too small to exceed the lower threshold of the sigmoidal $S(\cdot)$, but their combination may be large enough to cause a significant upregulation of gene C.

Because decay of gene products is often an important factor in their regulation, we can also add an extra decay term to each gene as follows:

$$\frac{dy_i}{dt} = S\Big(\sum_j w_{ji} y_j + b_i\Big) - D_i y_i \tag{1.4}$$

where D_i is the decay rate for gene i. The resulting model is very close in form to a specific type of recurrent neural networks, and can be fitted to real data in the same manner. Chapter 7 develops such a model and describes tests of the model on synthetically generated data sets.

The idea to use a neural network representation to model regulatory networks is not new, dating back at least to Bray's work on cell signaling and parallel distributed processing networks [37]. The reasons for using a neural network model, rather than a more general differential equation model, are twofold. The neural network has a straightforward graphical representation which is close to what researchers are already used to—a very important advantage considering that refinement of these sorts of models usually benefits greatly from collaboration with scientists in the field. Secondly, a large variety of efficient learning algorithms have already been developed for neural networks, whereas determining the parameters in a more general differential equation model would require more general-purpose optimization methods.

Various researchers have used variants of this representation to model genetic regulatory networks. Most notably, Mjolsness, Reinitz and Sharp [164] used a gene regulation model as in Equation 1.4, interspersed with a simple model of cell division, to model small gene networks involved in pattern formation during the blastoderm stage of development in Drosophila. Weaver *et al.* [241] used a discrete-time version of Equation 1.3, and showed it is possible to reconstruct randomly created networks of this kind, given enough time series data generated by the network.

Unfortunately, with so many researchers arriving at similar models independently, a variety of different names have been invented for them: connectionist model (Mjolsness *et al.* [164]), linear model (D'haeseleer *et al.* [66]), linear transcription model (Chen *et al.* [51]), weight matrix model (Weaver *et al.* [241]). Considering the core of all these models is the use of a weighted sum to implement gene regulation, I propose we file them under the more general classification of *additive regulation models*[2]. This distinguishes these models from other representations which may make use of weight matrices, such as Savageau's power law formalism[195]: $dy_i/dt = \alpha_i \prod y_j^{v_{ji}} - \beta_i \prod y_j^{w_{ji}}$, where the two terms account for the production and destruction of the gene product i, v_{ji} and w_{ji} are the kinetic orders, and α_i and β_i the rate constants for these elemental processes.

One implicit assumption of these models is that the concentrations of the chemical species are continuous, i.e. that stochastic fluctuations due to single molecules can be ignored. We know that this does not hold at least for some proteins which are present in concentrations of only a couple of molecules per cell. Indeed, there are indications that stochastic fluctuations may actually be exploited by some organisms [22]. However, differential equations are widely used to model biochemical systems. Hopefully, a continuous approach will prove to be appropriate for the majority of interesting mechanisms.

1.4 Caution to the reader

A number of new technologies are producing a flood of genomic-scale data about the internal state of a cell. Unfortunately, even though these data sets look large to a biologist, they are large "along the wrong dimension", i.e. a large number of variables

[2]In statistics, models consisting of a nonlinear function of a weighted sum of inputs are also called *Generalized Linear* models [159], whereas models consisting of a weighted sum of nonlinear (nonparametric) functions of inputs are called *Generalized Additive* models [231].

are measured, but the number of individual measurements of any one variable is still relatively small.

The network models employed here require a substantial number of data points. For example, a common rule of thumb in the neural network community is to use at least a couple times more measurements than weights in the network. This would imply hundreds of data points for the small set of 65 genes used in Chapter 6, or tens of thousands of data points for yeast ($\approx$ 6000 genes). Conventional wisdom would suggest that these sorts of models are underdetermined given the small number of data points currently available.

I intend to show that a shortage of data points does not invalidate the use of these models, as long as we can determine which parts of the model are well determined versus poorly determined. Indeed, much of this dissertation could be viewed as an exercise in distinguishing the few nuggets of well determined interactions from an otherwise poorly determined model. Unfortunately this does mean that it is not yet possible to infer a complete network model as in Figure 1.1. For now, we will settle for being able to infer those individual connections within the network which are best supported by the data.

Even for those relatively well determined parts of the model, we may not be able to show results with the same level of significance as with some simpler (but less powerful) methods. However, I view this approach not so much as a direct way to find "scientific truth" (however that is defined in one's favorite discipline: $P < 0.05$?), but rather as a way to derive interesting new hypotheses to guide experimentalists in further investigation.

No doubt, as the measurement technologies mature, and larger data sets become publicly available (and calibrated with each other), the usefulness and accuracy of the network models developed in this dissertation will increase. The trend towards

data sets with large numbers of measurements (see Section 2.2.3) definitely bodes well in that respect.

1.5 Overview

Chapter 2 will provide a brief overview of the technologies currently available for measuring the internal state of a cell on a large scale, as well as some of the publicly available data sets, with a focus on mRNA expression data and protein data. In Chapter 3, I list some of the related work in this area, especially with respect to techniques that have been used (or could be used) to infer genetic regulatory networks from large scale expression data. Chapter 4 will address some of the issues that arise—and decisions that have to be made—when attempting to model genetic regulatory networks, especially regarding trade-offs between realism and tractability of the model.

In Chapter 5 I derive a number of estimates for how much data is needed for various simple network models, and compare this with the amount of data needed for clustering—currently the most favorite analysis tool for gene expression data. Combined with estimates from other sources, this provides a clear overview of the features of a model that are important to reduce the data requirements: limited connectivity of the network model, and restrictions on the type of regulatory functions.

In Chapter 6, I further develop the most simple type of additive regulation models—a purely linear one. Such a linear model can be seen as a first-order approximation. I illustrate how this model can be applied to a real gene expression data set, analyze the results in detail, and show that they compare favorably with the literature on the genes involved.

Chapter 7 points out some weaknesses in the linear model and develops a more

realistic network model, introducing a nonlinear dose-response curve, as well as a separate mRNA decay term. Such a model—essentially a set of nonlinear differential equations—is equivalent to a specific type of recurrent neural network. I test this methodology on some artificially generated networks, and explore a number of standard techniques to improve the performance.

Finally, in Chapter 8, I will present my conclusions on the efficacy of these models, present a number of ways to extend these results further, and look ahead at what the future may bring, and how these models would fit in.

Large portions of this dissertation have been published previously, in collaboration with Dr. Roland Somogyi and his team, who kindly provided the data used in Chapter 6. In D'haeseleer *et al.*(1997) [65], I presented an initial analysis of the first data set provided by this group, mainly focusing on clustering methods. Little of this material is included here, except for occasional references (e.g. in Appendix A.1). In D'haeseleer *et al.*(1999) [66], I introduced the linear model, applied it to all three data sets used in Chapter 6, and hinted at the "robust parameters" approach used in Section 6.4.2 and the neural network model developed in Chapter 7. Some of the neuroscience results in Chapter 6 are to be published in a book chapter: Fuhrman *et al.*(2000, in press) [81], and the data requirement estimates (Chapter 5), modeling issues (Chapter 4) and overview of clustering methods (Section 5.1.7 and Appendix A) will appear in a review paper: D'haeseleer *et al.*(2000, in press) [64] (reprinted here by permission of Oxford University Press). The published material included here is primarily my own work, with the exception of some sanity-checking and suggestions from Stefanie Fuhrman and Roland Somogyi on the neuroscience results in Chapter 6; and the clustering overview, which was co-written with Shoudan Liang and appears here in Appendix A (copied verbatim from the review paper [64]).

Chapter 2

Large scale measurements of internal cell state

Data! Data! Data!, he cried impatiently.
I can't make bricks without clay.
— A. Conan Doyle

What sort of data on the internal state of a cell would we *like* to have, in order to reconstruct the genetic regulatory network? What sort of technologies are *available* to biologists to measure the internal state of the cell on a genomic level? This Chapter attempts to give a brief overview of the collection of large-scale expression data, and to give a sense of the ever-increasing scale of the data sets that are publicly available.

2.1 What are the important variables?

The state of a cell consists of all those variables—both internal and external—which determine its behavior. According to the Central Dogma of molecular biology, the

activity of a cell is determined by which of its genes are expressed—i.e., which genes are "turned on", resulting in the active production of the respective proteins. When a particular gene is expressed, its DNA is first *transcribed* into the complementary messenger RNA (mRNA), which is then *translated* into the specific protein this gene codes for. We can measure the level of expression of each gene—how much each gene is "turned on"—by measuring how many mRNA copies are present in the cell. To quote Eric Lander:

> The mRNA levels sensitively reflect the state of the cell, perhaps uniquely defining cell types, stages, and responses. To decipher the logic of gene regulation, we should aim to be able to monitor the expression level of all genes simultaneously ... [138]

The cartoon picture of the Central Dogma presented above is, of course, incomplete. Apart from the classical DNA $\rightarrow$ mRNA $\rightarrow$ protein pathway, the genes in the DNA are themselves regulated by the presence or absence of certain proteins. Furthermore, many of the interactions in the cell occur entirely at the protein level, which can cause significant discrepancies between protein and mRNA levels. Studies investigating the correlation between steady-state protein and mRNA levels find poor ($r = 0.36$ [94], $r = 0.48$ [18]) to moderately good ($r = 0.76$ [82]) correlations, with 10 to 20-fold variation in protein levels for genes with the same mRNA levels. (So far none have looked at the more interesting issue of whether *changes* in protein levels can be explained by changes in mRNA level. I.e., is the variation of protein levels due to a different "amplification constant" because of differences in translation and protein decay rates, or is it primarily due to regulation at the protein level?) Proteins can also undergo a number of different post-translational modifications, so each mRNA may correspond to several functionally different versions of the protein (depending on phosphorylation state, dimerization state, different protein folding,

etc.) Clearly, protein levels also form an important part of the internal state of a cell.

mRNA and protein levels do not make up the entire state of a cell. One could imagine measuring a number of other parameters, including cell volume, growth rate, methylation states of DNA, localization of proteins and mRNA within the cell, ion levels in neurons, etc. One class of data which could be very important to measure is levels of metabolites and nutrients. For example, DeRisi *et al.* [63] measured the change in mRNA levels during the transition from glucose metabolism to ethanol metabolism (the "diauxic shift") in yeast. Glucose levels were measured to represent how far into the diauxic shift the system was. It would have been interesting to measure at the same time the ethanol level, as well as a number of other metabolites involved such as acetate, pyruvate, glycogen, trehalose, etc. Specific assays exist for many of the more common metabolites. However, in order to be able to measure many of them simultaneously, we might want to use more general techniques. For example, Arkin *et al.* [23] uses capillary zone electrophoresis [171] to measure eight of the small molecular species in an in vitro glycolysis reaction.

Currently, most studies trying to infer expression mechanisms from cell state data use mRNA levels, because they are the easiest to measure (especially with the new large-scale gene expression technologies). Large-scale protein measurements tend to be very incomplete (typically only measuring the highest abundancy proteins), but can be supplemented with more exact measurements of individual proteins which are known to play an important role. When collecting time series data for example, protein levels could at least be collected at the start and end point. Similarly, when measuring gene expression data on a process involving metabolism (and which cellular process doesn't?), ideally an effort would be made to quantify the most important metabolite and nutrient levels. For now, I have decided to focus primarily on mRNA data, in the hope that this will already give us a large part of the overall

picture. If and when additional types of data become available, it should be feasible to fit them into the same modeling framework. See, for example, Section 8.2.2 for a suggested model incorporating mRNA and protein data.

2.2 mRNA levels

2.2.1 Oligonucleotide and cDNA microarrays

Most mRNA assays use probes consisting of single-stranded DNA sequences, either derived from mRNAs via reverse transcription, or synthesized based on known mRNA sequences. If the mRNA comes into contact with its complementary DNA probe, it will form a stable mRNA-DNA hybrid. If mRNA labeled with fluorescent dye is passed over the DNA probes, the amount of hybridization can be measured by the amount of fluorescence at each probe and will be proportional to the amount of specific mRNA present, and hence to the level of expression of the corresponding gene. (In practice, because of the higher stability of DNA over RNA, the mRNA's are first reverse-transcribed into cDNA. The "probes" then consist of DNA complementary to the cDNA.)

Oligonucleotide arrays: One of the main proponents of this approach is Affymetrix, whose "GeneChip" arrays [76], consist of small glass plates with thousands of oligonucleotide DNA probes (short stretches of nucleotides, typically 25-mers, in Affymetrix' case) attached to their surface. The oligonucleotides are synthesized directly onto the surface using a combination of semiconductor-based photolithography and light-directed chemical synthesis, as illustrated in Figure 2.1. Due to the combinatorial nature of the process and the high-tech approach, very large numbers of mRNAs can be probed at the same time. However, manufacturing and reading

of the chips requires expensive equipment. Current chips have over 65,000 different probes, with several probes (and controls) for each mRNA.

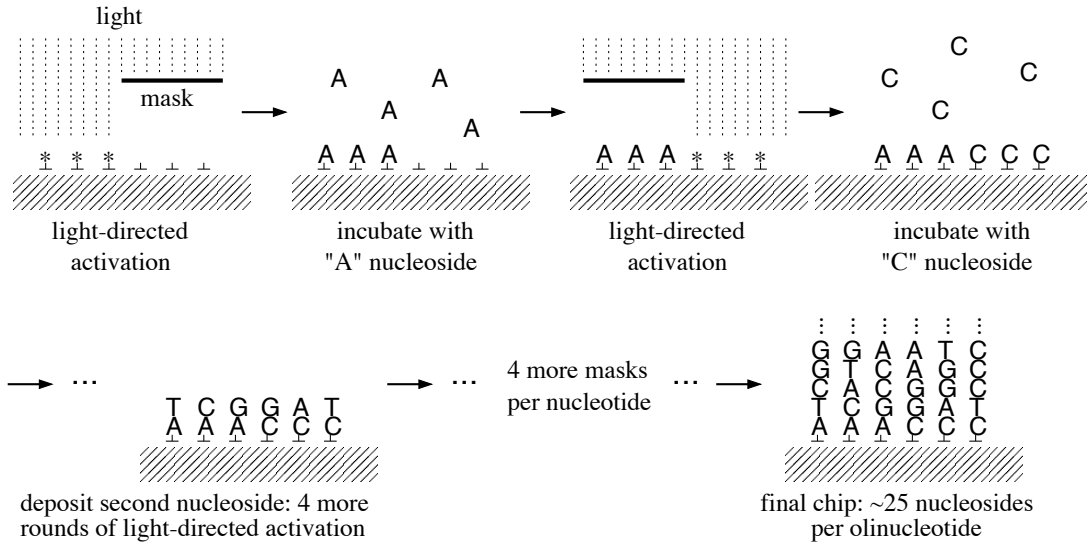

Figure 2.1: Construction of an oligonucleotide chip through in-situ light-directed synthesis. Photolithographic masks are used to protect specific areas of the chip surface from light-directed activation. Nucleosides (A,T,C or G) will attach to the activated sites. Using four complementary masks, all oligonucleotides under construction are extended by one nucleoside. Due to the combinatorial nature of the synthesis process, only $4n$ masks are needed for any number of length-n oligonucleotides.

Affymetrix currently manufactures GeneChip sets for >60,000 human genes and EST's (Expressed Sequence Tags, snippets of expressed mRNA); 30,000 mouse genes and EST's; all 6,400 yeast ORF's (Open Reading Frames, putative genes); >8300 Arabidopsis genes and EST's; >4200 known E. coli ORF's plus probe sets for stable RNA species and intergenic regions; as well as several special-purpose chip sets.

The following are some milestones achieved using this technology: In 1997, Wodicka *et al.* [251] illustrated the capability of oligonucleotide chip technology to do genome-wide analysis on yeast gene expression. Holstege *et al.* [113] in 1998 used various temperature-sensitive yeast mutants to dissect the regulatory interactions of components of the transcription initiation machinery in yeast. In 1999, Winzeler

et al. [250] published the start of an ongoing project to characterize the function of all yeast genes using thousands of individual gene deletion mutants. Expression time series were collected for 558 strains, at six time points during growth in rich and minimal medium. Golub *et al.* [89] used oligonucleotide chip data to classify 38 bone marrow samples into different cancer types.

cDNA microarrays: Developed at Stanford University [197], these are a more low-tech solution to mRNA measurement. The microarrays are glass slides on which full-length cDNA[1] has been deposited by high-speed robotic printing. They are cheaper to manufacture and easy to read, but require handling a large number of cDNAs, which makes them somewhat less scalable.

Microarray measurements are carried out as differential hybridizations to minimize errors originating from cDNA spotting variability: mRNA from two different sources (e.g control and drug-treated), labeled with two different fluorescent dyes, is passed over the array at the same time, as illustrated in Figure 2.2. The fluorescence signal from each mRNA population is evaluated independently, and then used to calculate the expression ratio.

Because of the non-proprietary nature of the technology (the Brown lab at Stanford even has an online guide [62] for building your own arrayer from scratch), this is currently the most widely used technology in academia.

Some important milestones achieved with this technology: In 1997, DeRisi, Iyer and Brown [63] published the first whole-genome gene expression measurements (approximately 6400 distinct cDNA sequences), a seven-point time series on the diauxic shift (transition from sugar metabolism to ethanol metabolism) in yeast. In 1998, Spellman *et al.* [217] published a set of yeast cell-cycle time series, consisting of three

[1] "cDNA", or complementary DNA, is a long stretch of single-stranded DNA complementary to a full-length mRNA, typically 500-5000 bases long.

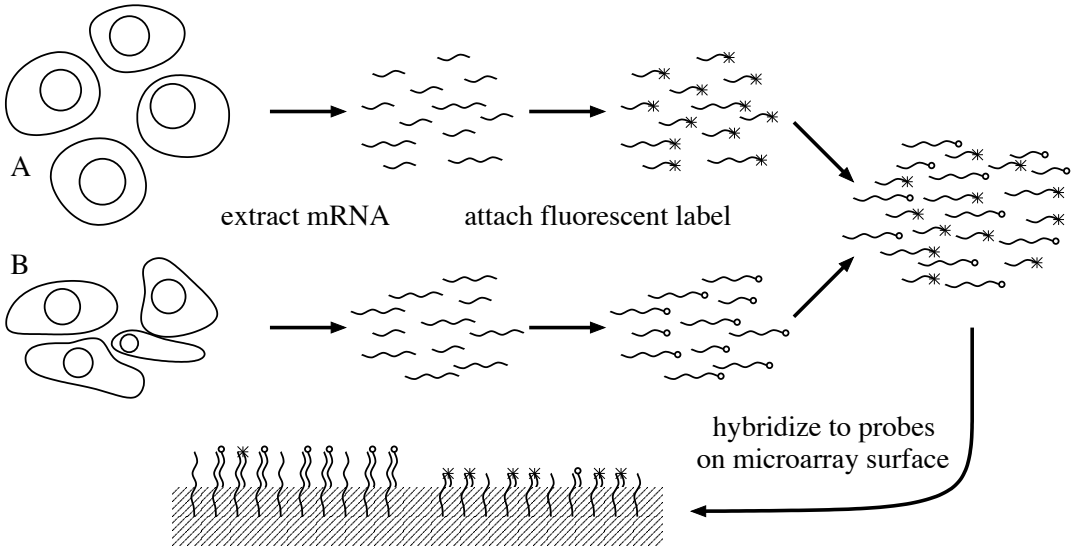

Figure 2.2:
Gene expression measurement using differential hybridization to a cDNA microarray. mRNA is extracted from two different sources (A and B) and labeled with a different fluorescent tag. The microarray has a spot corresponding to each mRNA species, containing the complementary DNA sequence. The spot for the longer mRNA species (left) shows more fluorescence at the wavelength associated with the tag used for B, the spot for the shorter mRNA species (right) shows more fluorescence from the tag used for A.

microarray data sets, and one oligonucleotide data set which had been published earlier by Cho *et al.* [53]. The four time series used different ways to synchronize the cells. The longest time series contained 24 time points, spanning almost two and a half cell cycles.[2] This is still one of the reference data sets for yeast (and for the cell cycle in general). Richmond *et al.* [186] published the first whole-genome E. coli expression measurements in 1999, using nylon membranes rather than glass slides. More recently, Alizadeh *et al.* [9] used cDNA microarray data to discover a previously unknown sub-classification within diffuse large B-cell lymphomas, associated with significantly different survival of the patients.

[2]Synchronizing the cells is usually done by halting them in some stage of the cell cycle. After release, the cells quickly lose synchrony, so in general one can only measure a few cycles.

Microarray variants: A number of variants of these approaches have been developed. Rosetta Inpharmatics developed their own manufacturing technique based on in-situ oligonucleotide synthesis using inkjet technology [34], combining some of the best properties of genechips (combinatorial synthesis based on sequence databases) and microarrays (relatively low-tech printing and read-out). They claim to be able to go from design—picking the mRNA sequences—to a finished array in a few hours. Unfortunately, their technology is strictly proprietary, and they require collaborators to do experiments on-premises at Rosetta. Recently, Hughes *et al.* [118] released a large "compendium" of their yeast expression data, gathered under standardized conditions allowing for comparisons between all measurements.

Other variants include the use of a digital micromirror array for light-directed in-situ synthesis, avoiding the use of expensive photolithography masks for each chip (Singh-Gasson *et al.* [207]); bubble jet printing technology for creation of cDNA microarrays (Okamoto *et al.* [173]); and fiber-optic arrays, where each optical fiber is tipped with a microsphere carrying oligonucleotide probes (Steemers *et al.* [218]).

2.2.2 Other approaches

SAGE: Serial Analysis of Gene Expression [236] uses a very different technique for measuring mRNA levels. First, double stranded cDNA is created from the mRNA. A single 10 base pair "sequence tag" (long enough to uniquely identify most genes) is cut from a specific location in each cDNA. Then the sequence tags are concatenated into a long double stranded DNA which can then be amplified and sequenced. This method has two advantages: the mRNA sequence does not need to be known a priori—so it will also detect previously unknown genes—and it uses sequencing technology that many labs already have. The method is rather complex though, and requires a large amount of sequencing. It has also recently been shown that the results contain

significant biases [219].

In 1997, SAGE was used to analyze the set of genes expressed during three different phases of the yeast cell cycle. A total of 4665 genes were detected [237]. In 1999, Velculescu *et al.* [235] analyzed 3.5 million tags from 19 normal and diseases human tissue types, detecting a total of 84,103 unique genes (close to some estimates of the total number of human genes). They found as many as 43,500 genes can be expressed in a single cell type, although nearly one half of the mRNA content is accounted for by only 623 genes. Recently, a public SAGE gene expression database has been announced [139].

RT-PCR: Reverse Transcriptase Polymerase Chain Reaction is a somewhat more orthodox approach, relying on amplification to measure low-abundance mRNA species. The mRNA is first reverse-transcribed into cDNA, and the cDNA is then amplified to measurable levels using PCR. PCR is known to be somewhat error-prone and may have slight differences in amplification factor, but RT-PCR uses built-in calibration techniques to achieve high accuracy coupled with an excellent sensitivity and a dynamic range covering 6-8 orders of magnitude. The method does require PCR primers for all the genes of interest, and is not inherently parallel like microarrays or SAGE, so automation is crucial to scale up. Nevertheless, it is gaining in popularity because of it's higher accuracy, partially as a way to check results from microarray analyses.

Roland Somogyi at the NIH Laboratory of Neurophysiology has used this method to measure the expression levels of 112 genes at nine different time points during the development of rat cervical spinal cord [245], and 70 genes during development and following injury of the hippocampus [212]. These are the data sets that were used to construct a linear model in Chapter 6. See Section 6.2 for a more in-depth overview of this data.

2.2.3 Truly large-scale gene expression

We are now beginning to see individual publications reporting up to hundreds of expression measurements at a time, starting with the combined cluster analysis of 79 yeast experiments (most of which were already published elsewhere) by Eisen *et al.* [70]. More recently, Alizadeh *et al.* [9] published 128 microarray analyses of 96 samples of diffuse large B-cell lymphoma, using a specially designed "Lymphochip" containing 17,856 cDNA clones. Alizadeh and Staudt [10] used the same Lymphochip to extend this analysis to 242 experiments, adding measurements on various other normal and malignant lymphocytes, as well as a total of thirteen time series consisting of activation of T-cells and peripheral blood monocytes. In the same issue, Manger and Relman [152]—also at Stanford—published a data set of 112 experiments, containing thirteen times series on the response of monocytes against various pathogens. Most recently, Hughes *et al.* [118] released the aforementioned compendium of 300 mutations and chemical treatments in yeast, probably more than doubling the current publicly available data on yeast. Several public gene expression databases are also in development [71, 167, 168].

2.3 Protein levels

Protein levels are harder to quantify than mRNA levels. The basic method, two-dimensional polyacrylamide electrophoresis (2D-PAGE), is straightforward enough, and has been around for about 25 years [172]. 2D-PAGE separates proteins on a two-dimensional sheet of gel, first in one direction based on their isoelectric point, and then in the other direction based on their molecular weight. The result is a two-dimensional image with a large number of protein "spots" The intensity of each spot is proportional to the amount of the specific protein present.

It is not a priori known which protein each spot represents, although the position of known proteins can be estimated. Also, new microsequencing and mass spectrometry techniques allow spots to be identified with proteins of which the sequence is known. Still, most spots on 2D gels tend to be unidentified as yet. Furthermore, the resolution of the spots may not be high enough to separate all proteins, only the highest expressed proteins form clearly identifiable spots (typically limited to less than 1400-1500 spots per gel)[82, 93], and 2D gel results have traditionally been hard to reproduce because of sensitivity to operating parameters and a host of possible artifacts. These problems have been somewhat alleviated by the use of highly standardized protocols and higher accuracy techniques [33].

Despite these problems, 2D gel electrophoresis has been very productive the past couple of years, with a number of *proteome* databases springing up around the world. Currently, there are several 2D gel databases available for a variety of organisms. One of the most important ones is the SWISS-2DPAGE database [20, 19], containing a total of 692 entries from human, yeast, E. coli and Dictyostelium.

Chapter 3

Related work

This Chapter consist of two broad, partially overlapping segments. Firstly, those techniques which were around when I started on this line of research and occasionally served as inspiration for my own work, especially the "gene circuits" work in Section 3.2. Other work in this category include Correlation Metric Construction (Section 3.3), some of the early work on inference of Boolean networks (Section 3.4, especially [213]), and the work on diauxic shift in Section 3.1. Secondly, there are those techniques which have been introduced since then, some of which have occasionally been influenced by my own early work, such as a few of the clustering approaches in Section 3.5 (and Appendix A), some networks of clusters (Section 3.6, in particular [239, 234]), and some of the work on Bayesian networks (Section 3.7, in particular [165]).

3.1 Fitting known networks

So far, relatively little work has been done trying to fit such large-scale information to known networks. One notable early example is in the work on diauxic shift in

yeast [63]. The authors focused mainly on genes coding for enzymes known to play a role in glucose or ethanol metabolism, using the expression data to quantify changes in the different parts of the known metabolic pathway. Expression levels of one deletion mutant and one overexpression mutant were measured as well. This study merely observed how the expression levels for the enzymes changed to compensate for the changes in demand for the different metabolic pathways. No effort was made to propose a model for the regulation of these genes, i.e. to find the *regulatory* network that maintains the *metabolic* network.

More recently, Roberts *et al.* [187] used large-scale gene expression data to elucidate the signaling and crosstalk in several yeast MAPK (mitogen-activated protein kinase) pathways: the pheromone response pathway; the PKC (protein kinase C) pathway activated by cell surface stress; the HOG (high-osmolarity glycerol) pathway activated by osmotic stress; and the filamentous growth pathway. They examined a large number of mutants, under various amounts of exposure to the α-factor pheromone, for a total of 46 experiments, and were able to verify—and clarify—the known pathways. They also found evidence of higher-order interactions between the pathways, such as the sequential activation of the pheromone and PKC pathways during formation of mating projections.

The two examples above analyzed the data mainly in a classical reductionist, one-gene-at-a-time fashion (with the exception of some cluster analysis in [187]). For a more computationally-oriented analysis, Zien *et al.* [258] have suggested a scoring function for putative pathways, based on how well they are supported by gene expression data, and found that it assigns statistically significant scores to biologically relevant pathways. von Dassow *et al.* [238] (see below in Section 3.2) fit the parameters of a small predefined network to an idealized spatial pattern of expression.

3.2 Gene circuits: spatial patterns in development

Mjolsness, Reinitz and Sharp have used a neural network approach to model small gene networks involved in pattern formation during the blastoderm stage of development in Drosophila [164, 185]. Cell divisions at this stage are under the control of a maternal clock, so they used a simplified cellular model with synchronized cell divisions along a longitudinal axis, alternated with updating the gene expression levels using a neural network. Simulated annealing was used to find a least-squares fit to real gene expression data. Because of the hybrid nature of the model, classical neural network learning algorithms do not apply. The lack of an efficient learning algorithm implies that this method is limited to very small networks (6 genes and a total of 50 parameters for the *eve* stripe example in [185]).

The model was able to successfully replicate the pattern of *eve* stripes (named after the *even-skipped* gene) in Drosophila, as well as some mutant patterns on which the model was not explicitly trained [204]. It has since also been successfully applied by Marnellos *et al.*to early neurogenesis [154] and to the emergence of ciliated cells in Xenopus [153].

Some of the methodology used in the work on the segment polarity network by von Dassow *et al.* [238] shows distinct flavors of the gene circuit approach as well. They proposed a simple network diagram (although more detailed than in the gene circuits approach) of the segment polarity network in Drosophila, and attempted to find a set of parameters that would fit the desired spatial pattern. This approach allowed them to get around the significant problem of having to determine all the parameters "from the ground up", as in the work on modeling lysis-lysogeny switch in phage Lambda [22]. They showed that their initial model could not be fit to the desired spatial patterns, and found two additional interactions (supported by the literature) that would allow a fit. The main difference from the gene circuits approach is that the

latter generally makes no prior assumptions regarding the structure of the network, while the work by von Dassow *et al.*starts with a fixed network structure, allowing for a more detailed model.

3.3 Correlation Metric Construction

Adam Arkin and John Ross at Stanford University have been working on a method called Correlation Metric Construction, to reconstruct reaction networks from measured time series of the component chemicals [21, 23]. This approach is based in part on electronic circuit theory, general systems theory [48, 57, 58] and multivariate statistics.

The system (a small chemical reaction model in [21], a reactor vessel with chemicals implementing glycolysis in [23]) is driven using random (and independent) inputs for some of the chemical species, while the concentration of all the species is monitored over time. First, the time-lagged correlation (cross-correlation) matrix is calculated, and from this a distance matrix is constructed based on the maximum correlation between any two chemical species. This distance matrix is then fed into a simple clustering algorithm to generate a tree of connections between the species. Finally, the chemical species and the tree connecting them is displayed using multidimensional scaling (MDS), mapping each species to a point in 2D space while trying to preserve the distances between each prescribed in the distance matrix.

So far, they have not used the information regarding the time lag between species at which the highest correlation was found, which could be useful to infer causal relationships. Also, they have not yet taken advantage of more sophisticated methods from general systems theory, based on mutual information, to infer dependency. They are currently working on both of these issues.

3.4 Reverse engineering of Boolean networks

Boolean networks were first introduced by Kauffman in the late 60's [129] as an abstraction of genetic regulatory networks: each gene is modeled as being either "ON" or "OFF", and the state of each gene at the next time step is determined by Boolean function of its inputs at the current time step. Despite their extreme simplicity, Boolean networks have proven useful for developing insights in the behavior of large interconnected networks [130]. Clear analogies are found for such concepts as cell types (attractors in state space), differentiation (transition between attractors), stability of expression patterns (basins of attraction), and so on. Based on the—somewhat debatable—hypothesis that real gene regulation networks should exhibit features of random Boolean networks (RBN's), he derived plausible scaling laws for the size and number of attractors (showing a similar scaling coefficient as is found when plotting number of cell types versus size of DNA for a variety of species), and illustrated the importance of the degree of connectivity of the network, and the type of Boolean functions on its dynamical behavior (showing that for similar number of inputs and Boolean functions found in the biological literature, RBN's are close to the "edge of chaos", i.e. show near maximal complexity, between completely ordered and completely disordered).

Despite the poor fit of the Boolean network models to biological reality (see also Section 4.2), they have recently been used for the analysis of real gene expression data [214, 213, 223, 142, 3, 4, 6, 5, 128, 117, 119]. A variety of algorithms has been developed for *reverse engineering* of Boolean networks, starting with Somogyi *et al.* [213]. The work of Akutsu *et al.* [3, 4, 6, 5] has been especially productive, with careful analysis of computational complexity and sample complexity of several variants, including noisy Boolean networks [6, 5]. Both Akutsu *et al.* [3] and Ideker *et al.* [119] look at the issue of learning from overexpression or underexpression mutants; and Karp *et al.* [128] and Ideker *et al.* [119] suggest ways to determine the

optimum set of additional experiments.

One avenue that has been largely unexplored—despite the use of the "reverse engineering" term—are the algorithms developed in the engineering community. Reverse engineering has a long history in electrical engineering of digital circuits [127, 158]: given a certain specification of the desired input-output mappings, what is the cheapest way to design a digital circuit to implement this? Several highly efficient algorithms exist, some of which, such as ESPRESSO [39, 38], are publicly available. Some of the assumptions on which these algorithms rely do not necessarily make biological sense, such as minimizing the number of gates to implement the Boolean functions.[1] A special purpose reverse engineering tool for biological networks would also allow the networks to be biased towards biologically plausible features (types of Boolean functions, number of inputs per gene, reuse of inputs from important regulators, etc.).

Real gene expression levels tend to be continuous rather than discrete, however, and discretization can lead to a large loss of information, as described in [65]. Furthermore, important concepts in systems theory, such as positive and negative feedback, either cannot be implemented with discrete variables, or lead to a radically different dynamical behavior (see also Section 4.2).

3.5 Co-expression clustering

Currently, by far the most popular computational technique for large-scale analysis of gene expression data is clustering of genes (or experiments) based on similarity of expression patterns. This method is only tangentially related to gene network inference (with the exception of Sections 3.5.1 and 3.6), but because of its importance

[1] Although analysis of data collected by Harris [99] shows that Boolean abstractions of real regulatory function in the literature do tend to be of minimal complexity in this sense.

in analyzing gene expression patterns, it is included here for completeness. Much of the material in this section can also be found in [64]. Briefly, a clustering method generally consists of two distinct components: a distance measure that indicates how similar two gene expression pattern are (or more generally, two clusters); and a clustering algorithm, which uses some heuristics to identify clusters of similar gene expression patterns, based on the distance measure. The section of our review paper on specific distance measures and clustering algorithms used in gene expression analysis is reproduced here in Appendix A.

For further information on coexpression clustering, Claverie [55] provides a preliminary review of gene expression analysis techniques, focusing on coexpression clustering. Niehrs and Pollet [169] provide an overview of very tightly coexpressed groups of genes (which they call "synexpression groups" that have been identified based on large-scale gene expression data. Some useful textbooks on clustering include Massart and Kaufman [155], Aldenderfer and Blashfield [8] and Kaufman and Rousseeuw [131].

There are three broad areas of application for co-expression clustering: (1) extraction of regulatory motifs (co-regulation from co-expression); (2) inference of functional annotation; (3) classification of cell types, tissue samples, etc.

3.5.1 Extraction of regulatory motifs

The transcriptional regulation of a gene occurs primarily by binding of various regulatory proteins—transcription factors—to the DNA just "upstream" from the starting position of the actual gene—the *promoter region*. Interactions between these transcription factors, other regulatory proteins and the RNA polymerase complex (responsible for transcribing the DNA of the gene into mRNA) determine whether transcription of the gene is initiated. Each transcription factor binds to a specific

pattern of DNA in the promoter, so the presence or absence of these "motifs" influences how the gene is regulated.

If the promoter regions of the genes are known—as is the case for yeast—it is possible to identify the regulatory motifs shared by a cluster of genes with very similar expression levels throughout a number of experiments. In other words, it is possible to infer co-regulation from co-expression. Hampson *et al.* [96] found that genes with extremely similar upstream regions have similar activity levels. A variety of algorithms to extract common regulatory motifs from gene clusters have been developed [40, 189, 233, 252, 257, 96, 111, 208, 44]. For example, Tavazoie *et al.* [228] identified eighteen biologically significant DNA-motifs in the promoter region of genes clustered based on yeast cell-cycle data [217]. Most motifs were highly selective for the cluster in which they were found, and seven were known regulatory motifs for the genes in their respective clusters. If we can combine this knowledge with identification of proteins binding to these regulatory motifs [170], it may be possible to build up the network of regulatory interactions regulator by regulator.

3.5.2 Inference of functional annotation

Considering the wealth of data coming out of various genome projects, it is no surprise that the function of the majority of sequenced genes is currently unknown. If an unknown gene tightly clusters with a number of genes which all have the same function, we may be able to use "Guilt By Association" [240] to infer the function of the unknown gene. For example, in developing rat spinal cord, co-expression clusters were found to segregate according to functional genes families [245]. Tavazoie *et al.* [228] found clusters to be significantly enriched for genes with similar functions. Chu *et al.* [54] identified yeast genes with previously unknown functions based

on their expression pattern during sporulation, and confirmed their functional role through deletion experiments.

3.5.3 Classification

One of the most popular uses of gene expression data is for classification tasks such as identifying and distinguishing cell types, tissue samples, malign versus benign cancers, etc. Especially classification of cancer samples has received a lot of interest [256, 179, 89, 240, 9, 29, 209, 188, 198, 46, 180]. Because of its promise as a diagnostic tool, this could very well be the first application of gene expression technology that will have a direct impact on how medicine is routinely performed.

For classification, one often clusters "along the other dimension", i.e. clustering samples based on similarity of expression levels across a number of genes, rather than clustering genes based on similarity of expression levels across a number of samples. In some instances, it has proven useful to cluster along both dimensions [243, 70, 13, 9, 52]. Such *two-way clustering* also allows for easy visualization of the data.

Clustering is typically *unsupervised*, without any prior knowledge of classes in the data. In this case it takes on an explorative role, and can be used to discover classes to explain heterogeneity within the samples. For example, Golub *et al.* [89] were able to automatically derive a classification to distinguish between acute myeloid leukemia, and acute lymphoblastic leukemia. Alizadeh *et al.* [9] classified samples of large B-cell lymphoma (DLBCL) into two distinct—and previously unknown—subtypes. The newly discovered classes corresponded to significantly different survival times, and should probably be considered separate diseases.

Clustering can also be *supervised*, with knowledge of what classes the samples belong to. In this case the goal is to discover an efficient classifier that can be used on samples with unknown classification; or to discover genes which are associated

with the classes in question—which may clarify the reason behind the existence of the classes. Although we have mentioned classification within the context of clustering, classification has a long history within the Artificial Intelligence and Machine Learning communities. Some of these methods are now being applied to classification of expression patterns as well. For example, Brown *et al.* [42] investigates the use of Support Vector Machines (SVM's) to predict functional roles for unknown yeast genes. Ben-Dor *et al.* [29] compares SVM's, Boosting [202], and their own clustering-based classifier [30].

3.5.4 Which clustering method to use?

Currently, the most popular clustering techniques are the agglomerative hierarchical clustering algorithm of Eisen *et al.* [70], the self-organizing maps of Tamayo *et al.* [225], and—because of its simplicity—the K-means algorithm. However, this may reflect a premature convergence on easily available tools [69, 224] (and appealing visualization of results), rather than a consensus on a set of best practices.

Different clustering methods can have very different results, and—at this point—it is not yet clear which clustering methods are most useful for gene expression analysis. Each combination of distance measure and clustering algorithm will tend to emphasize different types of regularities in the data. Some may be useless for what we want to do. Others may give us complementary pieces of information. Jain and Dubes [124] state:

> There is no single best criterion for obtaining a partition because no precise and workable definition of "cluster" exists. Clusters can be of any arbitrary shapes and sizes in a multidimensional pattern space. Each clustering criterion imposes a certain structure on the data, and if the data happens to conform to the requirements of a particular criterion,

the true clusters are recovered.

It is impossible to accurately evaluate how "good" a specific clustering is without referring to what the clustering will be used for. However, once an application has been identified, it may be possible to evaluate the quality of the clustering for that particular application. For example, if we want to extract regulatory motifs from clusters, we can compare clustering methods based on the P-values (i.e., significance levels) of the resulting motifs. Similarly, for functional classification, we can compare P-values associated with enrichment of clusters in certain functional categories. It is unlikely that there would be a single best clustering method for all applications. Considering the overwhelming number of combinations of distance measures and clustering algorithms—far too many to try them all each time—the field is in dire need of a comparison study of the main combinations for some of the standard applications, such as functional classification or extraction of regulatory motifs. A promising development in this respect is the upcoming Critical Assessment of Techniques for Microarray Data Analysis (CAMDA) Conference [143], which "aims at establishing the current state of the art in microarray data mining, identifying what progress has been made, and highlighting where future effort may be focused." Yeung *et al.* [254] also developed a number of *internal* and *external criteria* for cluster validation. Internal criteria look at internal properties of the clustering, such as variance within versus between clusters, while external criteria compare a clustering result to a given "gold standard".

If we want to use gene clusters to infer regulatory interactions, synthetic data generated from small but detailed models of regulatory networks could provide a useful touchstone for comparing clustering methods. Preliminary results comparing SOM, K-means, FITCH and Autoclass—all using Euclidean distance—showed very poor performance of all clustering methods in identifying a metabolic pathway with associated regulation of the enzymes by the metabolites [160].

Because of the nature of genetic regulation, there is no reason to require each gene to fall in only one single cluster—as many of the standard algorithms do. A gene may have several functional annotations, and its promoter region may contain several regulatory motifs. As more data becomes available to accurately delineate expression behavior under different conditions, we should consider using some of the clustering methods that partition genes into non-exclusive clusters. Alternatively, several clustering methods could be used simultaneously, allocating each gene to several clusters based on the different regularities emphasized by each method.

3.5.5 Related methods

The clustering methods mentioned so far are inherently *local* methods, depending on local clumping of genes in the high-dimensional space determined by the experiments. An alternative approach is to examine some of the more *global* properties of the data, using principle component analysis (PCA), or the related singular value decomposition (SVD).[2] The assumption is that the principle components of gene expression may represent independent regulatory processes. This assumes of course that the expression level of each gene can be decomposed into a linear superposition of regulatory processes.

For example, Raychaudhuri *et al.* [184] find that over 90% of the variation in the yeast sporulation data set by Chu *et al.* [54] can be explained by the first two principle components. Similarly, Holter *et al.* [114] use a more elaborate pre-processing and SVD to extract the "characteristic modes" of gene expression in the same sporulation data set [54], a yeast cell cycle time series from Spellman *et al.* [217], and a time series of serum-treated human fibroblasts from Iyer *et al.* [122]. They found the

[2]Although the popular clustering method by Eisen *et al.* [70] uses SVD for the final ranking of genes, this is not really part of the clustering algorithm itself, but mainly done for visualization purposes.

first two modes to capture 76%, 62% and 69% of the variation, respectively. Alter *et al.* [14] used a similar analysis (although with greater emphasis on normalizing and filtering the data) on two other cell cycle time series from Spellman *et al.* [217], finding approximately 40% of the variation in the top two "eigengenes". As expected, in each case the two first modes for the cell cycle time series are approximately sinusoidal and 90° out of phase. It remains to be seen whether these techniques will tell us anything more than that many genes show broad trends in gene expression across all genes. At the very least they may provide a useful tool for data filtering and dimensionality reduction, whether for visualization or for further computational manipulation.

Hastie *et al.* [102] developed a clustering technique which can be thought of as finding representative clusters for each of the principal components in the data, which performs quite well in a supervised setting for classification of diffuse large B-cell lymphoma [9] based on survival time. Note that all these approaches may be adversely affected by highly correlated experiments (such as neighboring samples in a time series, or multiple copies of the cell cycle).

3.6 Networks of clusters

One of the main problems in trying to derive networks of regulatory interactions from large-scale expression data is the dimensionality of the data sets. With this in mind, several groups have tackled the easier problem of deriving regulatory interactions between *clusters* of genes. At least two of these [239, 234] were inspired by my own work on linear models of rat CNS expression levels (Chapter 6, [66]). As with SVD (or PCA) based analysis of gene expression data (Section 3.5.5), this approach tends to analyze broad, general trends. It is as yet unclear how these results can be extended to specific regulatory interactions, or whether they will give us a higher-level insight of coordination between functional "modules" within a cell.

Wahde and Hertz [239] reduced 65 genes from the rat CNS data sets [245, 212] (see also Section 6.2) down to the four "waves" of expression identified in [245] using the FITCH hierarchical clustering algorithm [73], and then used a genetic algorithm (GA) to derive a small, 4-node recurrent neural network, similar to the ones used in Chapter 7 (although without the extra decay term).

Chen *et al.* [50] used average linkage clustering to reduce 3131 significantly fluctuating genes in the yeast cell cycle data set from Cho *et al.* [53] down to 308 clusters, then used simulated annealing (SA) to optimize a qualitative network based on timing of peaks in the data.

Mjolsness *et al.* [162] used Expectation-Maximization (EM) to reduce 2467 genes from the combined yeast data set in Eisen *et al.* [70] down to 15 clusters, and then used SA to derive a recurrent neural network (identical to the type used in Chapter 7) of part of one of the cell cycle time series from Spellman *et al.* [217].

Van Someren *et al.* [234] took the same 2467 yeast genes in Eisen *et al.* [70], drastically reduced this set by excluding those whose expression ratio never exceed $\pm$2-fold, then used complete linkage clustering to reduce them further to $T-1$ clusters (where T is the total length of the time series), then generated a linear model (as in Chapter 6) of the cdc15 and α-factor time series from Spellman *et al.* [217] ($T = 15$ and $T = 18$ respectively). [3]

3.7 Bayesian networks

Bayesian Networks model each variable as a conditional probability function with respect to a subset of the other variables. Their stochastic nature makes them

[3] $T - 1$ is the largest number of variables for which the linear model of Chapter 6 is well-determined, given a time series of T data points.

excellent candidates for modeling gene regulation systems where we may expect to see stochastic effects, or simply data sets with large amounts of noise. Learning algorithms for both the structure and parameters of Bayesian networks have been developed [43, 103]. Bayesian networks tend to generalize better than regular neural networks, and can be used for causal inference.

Most research on Bayesian networks so far has focused on acyclic networks (Directed Acyclic Graphs), and static systems with discrete variables and/or linear Gaussian models. Hofman and Tresp [110] present an approach to learning Bayesian networks of continuous variables with nonlinear interactions, using nonparametric density estimation, but it is unclear how well this technique would work with limited amounts of data.

Friedman *et al.*used Bayesian networks to generate a causal model of the yeast cell cycle data from Spellman *et al.* [217], using either a model with discretized expression levels (e.g. Boolean, or underexpressed / normal / overexpressed) [78], or a linear Gaussian model [79]. The latter treats the expression level of a gene as being normally distributed around a mean which is a linear sum of inputs. The implicit assumption of a static model (i.e. relating the expression levels at a single point in time with respect to each other) may be valid for developmental time series which we can assume to be in a quasi-equilibrium most of the time, but may be problematic for the cell cycle data, where sampling rate was typically on the same order of magnitude as initiation of transcription.[4] Therefore, rather than true causal relationships, the results may reflect co-regulation of genes. Likewise, the assumption of an acyclic network is a poor fit for the data (i.e. they construct an *acyclic* model of the cell *cycle*). However, as the authors suggest, it is also possible to generate temporal models, describing the future expression levels based on the current levels.

[4]Sampling rate was as fast as 7 minutes per time point for α-factor synchronization [217], whereas yeast mRNA half lives (which also determine the rate of *increase* of mRNA levels [98]) average around 20 minutes [112]

(Alternatively, one could relate the current *slopes* in gene expression with the current expression levels. In fact, a linear Gaussian Bayesian network of the expression slopes with respect to the expression levels would be largely equivalent to the linear model proposed in Chapter 6.)

Dynamic Bayesian networks (DBN's) can be generated by "unrolling" the network in time. They are no longer restricted to equilibrium situations, nor to acyclic networks (as long as there is at least one time delay involved in any cycle). Murphy and Mian [165] provide an excellent overview of different DBN variants and their learning algorithms, and how these relate to various gene expression models. In particular, they point out the similarity of learning DBN's with discrete variables and unknown structure, to Boolean network reverse engineering algorithms such as REVEAL [142]; mention the issue of inventing hidden nodes in the case of partial observability; show the equivalence of linear Gaussian DBN's and the linear model used in Chapter 6; and suggest the use of the Extended Kalman Filter or recurrent neural networks techniques [177] as in Chapter 7 for the nonlinear case.

Chapter 4

Modeling issues

The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct, which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work.
— John von Neumann

Various types of gene regulation network models have been proposed, and the model of choice is often determined by the question one is trying to answer. In this Chapter we will briefly address some of the decisions that need to be made when constructing a network model, the tradeoffs associated with each, and the choices made for the modeling approaches in this dissertation.

4.1 Level of biochemical detail

Gene regulation models can vary from the very abstract—such as Kauffman's random Boolean networks [130]—to the very concrete—like the full biochemical interaction models with stochastic kinetics in Arkin *et al.* [22]. The former approach is the

most mathematically tractable, and its simplicity allows examination of very large systems (thousands of genes). The latter fits the biochemical reality better and may carry more weight with the experimental biologists, but its complexity necessarily restricts it to very small systems. For example, the detailed biochemical model of the five-gene lysis-lysogeny switch in Lambda phage [22] included a total of 67 parameters—resulting from almost 50 years of research on Lambda—and required supercomputers for its stochastic simulation (in 1998).

In-depth biochemical modeling is very important for understanding the precise interactions involved in common regulatory mechanisms. However, it is doubtful we could construct such a detailed molecular model of, say, an entire yeast cell with some 6000 genes by analyzing each gene individually and determining all the binding and reaction constants for each molecular interaction one-by-one—at least not in the near future. Likewise, from the perspective of drug target identification for human disease, we cannot realistically hope to characterize all the relevant molecular interactions one-by-one as a requirement for building a predictive disease model. There is a need for methods that can handle large-scale data in a global fashion, and that can analyze these large systems at some intermediate level, without going all the way down to the exact biochemical reactions. For this reason, and because of the limited amount of data available, we will choose a more abstract model, and attempt to infer very general regulatory interactions without specifying the precise mechanism.

4.2 Boolean or continuous

The Boolean (ON/OFF) approximation implicitly assumes highly cooperative binding (very "sharp" activation response curves) and/or positive feedback loops to make the variables saturate in ON or OFF positions. However, if one examines real gene expression data, it seems clear that genes spend a lot of their time at intermediate

values: gene expression levels tend to be continuous rather than binary. Furthermore, important concepts in control theory that seem indispensable for gene regulation systems either cannot be implemented with Boolean variables, or lead to a radically different dynamical behavior: amplification, subtraction and addition of signals; smoothly varying an internal parameter to compensate for a continuously varying environmental parameter; smoothly varying the period of a periodic phenomenon like the cell cycle, etc. Feedback control (see e.g. [77]) is one of the most important tools used in control theory to regulate system variables to a desired level, and reduce sensitivity to both external disturbances and variation of system parameters. Negative feedback with a moderate feedback gain has a stabilizing effect on the output of the system. However, negative feedback in Boolean circuits will always cause oscillations, rather than increased stability, because the Boolean transfer function effectively has an infinite slope (saturating at 0 and 1). Moreover, Savageau [196] identified several rules for gene circuitry (bacterial operons) that can only be captured by continuous analysis methods. Positive and negative modes of regulation were respectively linked to high and low demand for expression, and a relationship was established between the coupling of regulator and effector genes and circuit capacity and demand.

Some of these problems can be alleviated by hybrid Boolean systems. In particular, Glass [84, 86] has proposed sets of piecewise linear differential equations, where each gene has a continuous-valued internal state, and a Boolean external state. Researchers at the Free University of Brussels [230, 229] have proposed an asynchronously updated logic with intermediate threshold values. These systems allow easy analysis of certain properties of networks, and have been used for qualitative models of small gene networks, but still do not seem appropriate for quantitative modeling of real, large-scale gene expression data.

Since we are primarily interested in modeling real gene expression data, we will

opt for a continuous-valued model.

4.3 Deterministic or stochastic

One implicit assumption in continuous-valued models is that fluctuations in the range of single molecules can be ignored. Differential equations are already widely used to model biochemical systems, and a continuous approach may be sufficient for a large variety of interesting mechanisms. However, molecules present at only a few copies per cell do play an important role in some biological phenomena, such as the lysis-lysogeny switch in Lambda phage [183]. In that case, it may be impossible to model the behavior of the system exactly with a purely deterministic model.

These stochastic effects—which have mainly been observed in prokaryotes—may not play as much of a role in the larger eukaryotic cells. In yeast, most mRNA species seem to occur at close to one mRNA copy per cell [237, 113], down to 0.1 mRNA/cell or less (i.e. the mRNA is only present 10% of the time or less in any one cell). Low copy numbers like these could be due to leaky transcription and not have any regulatory role. Also, if the half-life of the corresponding protein (typically measured in hours or days) is much larger than the half-life of the mRNA (averaging around 20 min in yeast [112]), the protein level may not be affected by stochastic fluctuations in mRNA. Analysis of mRNA and protein decay rates and abundances may allow us to identify those few genes for which stochastic modeling may prove necessary.

Particle-based models can keep track of individual molecule counts, and often include much biochemical detail and/or spatial structure. Of course, keeping track of all this detail is computationally expensive, so they are typically only used for small systems. A related modeling technique is Stochastic Petri Nets (SPN's), which can be considered a subset of Markov processes, and can be used to model molecular

interactions [90]. Whereas fitting the parameters of a general particle model to real data can be quite difficult, optimization algorithms exist for SPN's. Hybrid Petri Nets [11, 156] include both discrete and continuous variables, allowing them to model both small-copy number and mass action interactions.

Additional sources of unpredictability can include external noise, or errors on measured data. The Bayesian approach to unpredictability is to construct models that can manipulate probability distributions rather than just single values. Stochastic differential equations could be used for example. Of course, this does add a whole new level of complexity to the models. Alternatively, a deterministic model can sometimes be extended by a simplified analysis of the variance on the expected behavior.

Since the role of stochasticity is unclear for the systems we're interested in (typically eukaryotes), we will choose for the simpler of the two approaches: a deterministic model.

4.4 Spatial or non-spatial

Spatiality can play an important role, both at the level of intercellular interactions, and at the level of cell compartments (e.g. nucleus vs. cytoplasm vs. membrane). Most processes in multicellular organisms, especially during development, involve interactions between different cells types, or even between cells of the same type. Some useful information can probably be extracted using a nonspatial model, but eventually a spatial model may be needed.

Spatiality adds yet another dimension of complexity to the models: spatial development, cell type interactions, reservoirs, diffusion constants, etc. In some cases, the abundance of data—spatial patterns—can more than make up for the extra

complexity of the model. For example, Mjolsness *et al.* [164] used a time series of one-dimensional spatial patterns to fit a simple model of eve stripe formation in Drosophila. Models like the ones proposed by Marnellos and Mjolsness [154] for the role of lateral interactions in early Drosophila neurogenesis provide experimentally testable predictions about potentially important interactions.

Current large-scale gene expression data typically does not include any spatial aspects, so we will use a non-spatial model.

4.5 Forward and inverse modeling

Some of the more detailed modeling methodologies listed above have been used to construct computer models of small, well-described regulatory networks. Of course, this requires an extensive knowledge of the system in question, often resulting from decades of research. In this dissertation, we will not focus on this forward modeling approach, but rather on the inverse modeling, or reverse engineering problem: given a specific set of measurements, what can we deduce about the unknown underlying regulatory network? Reverse engineering typically requires the use of a parametric model, the parameters of which are then fit to the real-world data. If the connection structure of the regulatory network (i.e. which genes have a regulatory effect on each other) is unknown, the parametric model will necessarily have to be very general and simplistic, providing little insight into the actual molecular mechanisms involved. Once the network structure is well known, a more detailed model might be used to estimate individual mechanism-related parameters, such as binding and decay constants.

Chapter 5

Data requirements for network inference

Number is the ruler of forms and ideas,
and is the cause of gods and demons.
— Pythagoras

In this Chapter, we will start by examining the amount of data needed to be able to reconstruct various different network models—a question with great practical importance, but unfortunately no exact answers. As we increase the number of variables to model, the size of the parameter space increases exponentially. This "Curse of Dimensionality" is examined in Section 5.2. Lastly, the data requirements for network inference imply we may need to combine data sets from different sources and of different types. These issues are explored in the final two Sections.

5.1 Sample complexity

The ambitious goal of network reverse engineering comes at the price of requiring more data points. How many data points are needed to infer a gene network of N genes depends on the complexity of the model used to do the inference. As we will see, constraining the connectivity of the network (number of regulatory inputs per gene) and the nature of the regulatory interactions can dramatically reduce the amount of data needed.

5.1.1 General network models

We can derive an absolute lower bound on the amount of information—in bits—needed to construct general network models, using Information Theory[1]. Suppose we want to derive a *sparse* network model of N genes, where each gene is only affected by K other genes on average (the "connectivity" of the network). This corresponds to constructing a sparsely connected, directed graph with N nodes and NK edges. There are N^2 possible edges between all N genes, and only NK actual edges, so there are $\binom{N^2}{NK}$ possible models of N genes with K interactions on average. To specify the correct model, we then need

$$\log \binom{N^2}{NK} = \log \frac{N^2!}{(NK)!(N^2 - NK)!} \tag{5.1}$$

bits of information. We can use Stirling's approximation to the factorial ($n! \approx \sqrt{2\pi n}(n/e)^n$) to derive an approximation for $\log \binom{a}{b}$ (see, e.g., [59]):

$$\log \binom{a}{b} \approx a \log(a) - b \log(b) - (a - b) \log(a - b) \tag{5.2}$$

[1] first developed by Shannon [201], see Cover and Thomas [59] for a good introduction.

for $a, b \gg 1$. Equation 5.1 then becomes:

$$\begin{aligned}
&\log \binom{N^2}{NK} \\
&\quad \approx N^2 \log\left(N^2\right) - NK \log(NK) - \left(N^2 - NK\right) \log\left(N^2 - NK\right) && (5.3) \\
&\quad = N\Big(N \log(N) - K \log(K) - (N-K) \log(N-K)\Big) && (5.4) \\
&\quad \approx NK \log(N/K) && (5.5)
\end{aligned}$$

bits of information. The last approximation holds for $K \ll N$, such that $\log(N - K) \approx \log(N)$. Since each data point consists of N measurements, we will need at least $\Omega(K \log(N/K))$ data points to fully specify a model of this kind. Note that, by Equations 5.4 and 5.2, we would get the same growth rate for a model with *exactly* K inputs per gene: $N \log \binom{N}{K} \approx \log \binom{N^2}{NK}$. A similar derivation for undirected graphs (i.e. inferring regulatory interactions, without specifying the *causal* relationship) leads to a lower bound of $\Omega(K \log(N/2K))$ data points.

If we further want to specify whether the interaction is positive or negative, this only requires one extra bit of information per connection in the network. In general, if we want to specify p_n parameters per gene, with λ_n bits of precision each; and p_k parameters per interaction, with λ_k bits of precision, we get $NK \log(N/K) + \lambda_n p_n N + \lambda_k p_k NK$ bits, or at least $\Omega(K \log(N/K) + \lambda_n p_n + \lambda_k p_k K)$ data points. Note that we have not specified how these regulatory inputs are combined, whether the regulatory function is linear or nonlinear, etc. Each link and each node in the network could correspond to some arbitrary, parametrized nonlinear function.

5.1.2 Boolean, fully connected

In a fully connected Boolean network, the output of each gene is modeled as a general Boolean function of the outputs of all N genes. This means we need to specify the output of each single gene, for each of the 2^N possible different states of the network. In other words, we need to measure all possible 2^N input-output pairs. This is clearly inconceivable for even fairly small numbers of genes.

5.1.3 Boolean, connectivity K

If we reduce the connectivity of the Boolean network to an average of K regulatory inputs per gene, the data requirements decrease significantly. To fully specify a Boolean network with limited connectivity, we need to specify the connection pattern between the N nodes (genes) and the rule table for a function of K inputs at each. An absolute lower bound of $\Omega\left(2^K + K\log(N/K)\right)$ can be derived using information theory.[2] A tighter lower bound can be found by looking at a slightly simpler model, where we assume the pattern of connectivity is given, by calculating how the number of independently chosen data points should scale with K and N. Since this is a simpler model, its data requirements should be a lower bound to the requirements for the model with unknown connections.

Every data point (i.e. every input-output pair, specifying the state of the entire Boolean network at time t and $t+1$), specifies exactly one of 2^K entries in each rule table: Given this particular combination of the K inputs to each gene at time t, the output of the gene is given by its state at time $t+1$. We will estimate the probability P that all N rule tables are fully specified by n data points, and calculate how the number of data points n needs to scale with P, the number of genes N, and

[2] As shown in Section 5.1.1, we need $K\log(N/K)$ bits per gene to specify the connection pattern, and 2^K bits per gene to specify the Boolean function.

connectivity K.

The probability that one of 2^K entries in a specific rule table is *not* specified by a single data point is equal to $1 - 2^{-K}$. For n (independent) data points this becomes $(1 - 2^{-K})^n$. Since every data point has to specify exactly one entry in each rule table, the probabilities for each individual entry in a rule table to be unspecified are not entirely independent (e.g. if $2^K - 1$ entries are unspecified, the remaining entry has to be specified). However, for $P \approx 1$ (i.e. we have enough data to have a good chance at a fully specified model), these probabilities will be extremely close to zero, and we can approximate them as being independent. The probability that all 2^K entries in a single rule table are specified by n data points is then approximately:

$$1 - 2^K \left(1 - 2^{-K}\right)^n \tag{5.6}$$

The probability that all N rule tables are fully specified by n data points then becomes:

$$P \approx \left(1 - 2^K \left(1 - 2^{-K}\right)^n\right)^N \tag{5.7}$$

Taking base-2 logarithms, we find:

$$C_1 = -\log(P) \tag{5.8}$$

$$\approx -N \log\left(1 - 2^K \left(1 - 2^{-K}\right)^n\right) \tag{5.9}$$

Further simplifying using $\log_2(1 - z) \approx -z \log_2(e)$ for $z \ll 1$ (keeping in mind that the quantity in Equation 5.6 is very close to 1), and taking logs again:

$$C_1 \approx N2^K \left(1 - 2^{-K}\right)^n \log(e) \tag{5.10}$$

$$\begin{aligned} C_2 &= -\log\left(C_1 / \log(e)\right) && (5.11)\\ &\approx -\log(N) - K - n\log\left(1 - 2^{-K}\right) && (5.12)\\ &\approx -\log(N) - K + n2^{-K}\log(e) && (5.13) \end{aligned}$$

If $P \approx 1$, C_1 will be a small, and C_2 a moderate positive constant (e.g. for $P = 0.9$ and $P = 0.999$, C_2 is 3.25 and 9.97 respectively). We can now express n, the number of data points needed, in terms of N, K and C_2:

$$n \approx 2^K \left(K + \log(N) + C_2\right) / log(e) \tag{5.14}$$

which is $\Theta\left(2^K \left(K + log(N)\right)\right)$. This estimate agrees well with preliminary experimental results by Liang *et al.* [142] and Akutsu *et al.* [4].

5.1.4 Boolean, linearly separable, connectivity K

In addition to constraining the number of inputs per gene, we could also constrain the type of Boolean functions used in the network. A natural choice is the set of linearly separable Boolean functions, i.e., those that can be implemented using a weighted sum of the inputs, followed by a threshold function. Linearly separable functions are well-behaved, in the sense that inputs always have either an upregulating or downregulating effect. Non-linearly separable functions can have inputs that are upregulating or downregulating, depending on the state of the other inputs (the classical example of this is the Boolean XOR). Interestingly, the vast majority

of genes whose regulation is described in the literature seem to have regulation functions which are linearly separable, when abstracted down to the Boolean level [99].[3] Combining a reduced connectivity with linearly separable Boolean functions reduces the data requirements to $\Omega(K \log(N/K))$ [104].

5.1.5 Continuous, additive, fully connected

When we look at network models with continuous-valued expression levels, we need to choose a parametrized model of regulation functions. (As opposed to Boolean functions, functions over the reals are not enumerable, so we would need infinite amounts of data to fit a "general" continuous-valued function). As mentioned in Section 1.3, and in analogy with Section 5.1.4, we will focus on *additive regulation models*. For models with continuous expression levels, the data requirements are less clear than for the Boolean models. In the case of linear (D'haeseleer et al., 1999) or quasi-linear[4] additive models [241], fitting the model is equivalent to performing a multiple regression, so at least $N + 1$ data points are needed for a fully connected model of N genes[5].

5.1.6 Continuous, additive, connectivity K

Data requirements for sparse additive regulation models are as yet unknown, but based on the similarity with the equivalent Boolean model, we speculate it to be of the form $\Omega(K \log(N/K))$. A promising avenue of further research in this area may

[3] Although notable exceptions to this certainly exist: in Drosophila, *hunchback*, one of the key regulatory genes in embryonic development, has a concentration-dependent regulatory effect on *Krüppel* [199].

[4] Also known as *generalized linear*.

[5] Note that this result is not directly comparable to the Boolean case: the fully connected Boolean network uses arbitrary Boolean functions, and the estimate for linearly separable Boolean functions (equivalent to the additive functions used here) assumes $K \ll N$

be the results on sample complexity for recurrent neural networks, which have a very similar structure to the models presented here. An analysis based on PAC-learning shows that the number of training instances needed to accurately learn the *dynamical behavior* (as opposed to the network weights) for a fully connected network is lower-bounded by $\Omega(N)$ and upper-bounded by $\mathcal{O}(N^4)$ [136]. However, this is based on a worst-case analysis, and might be reduced to $\mathcal{O}(N)$ for the general case [216]. There are a few neural network techniques (such as Winnow and Weighted Majority [144, 145, 146]) that are known to scale as $\mathcal{O}(K \log(N)$ and perform quite well in the presence of many irrelevant inputs. However, these techniques are specific for classification tasks with feedforward networks, using a multiplicative weight update (one could think of them as doing a binary search on the decision surface). It is unclear how these algorithms could be extended to a recurrent network with continuous outputs.

5.1.7 Clustering

Finally, to allow for comparison with gene clustering methods, we examined data requirements for clustering based on pairwise correlation comparisons. In that case, as the number of genes being compared increases, the number of data points will have to increase proportional to $\log(N)$, in order to maintain a constant, low level of false positives. Claverie [55] arrived at a similar logarithmic scaling for binary data (absent/detected).

For this simple abstraction of clustering, we will say that two genes cluster together if their correlation is significantly greater (with a significance level α) than a certain cutoff value ρ_c. We test whether we can exclude the null hypothesis $\rho < \rho_c$ based on the measured correlation coefficient r over the available data points. Because of the large number of comparisons being made, we need to reduce the signif-

icance level for the correlation test with the number of tests each gene is involved in. We can use the *Bonferroni correction*, $\alpha = \alpha'/N$, in order to keep the expected number of false positives for each gene constant.[6] In order to be able to use the same cutoff-value for the measured correlation r_α to decide whether two genes cluster together, the number of data points will have to increase as the significance level for each test grows smaller.

If the real correlation coefficient ρ is close to 1.0, the distribution of the measured correlation coefficient r is very asymmetrical. The following z-transformation, developed by Fisher [75], is approximately normally distributed with mean $z(\rho)$ and variance $1/(n-3)$ (with n the number of data points):

$$z(r) = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) \tag{5.15}$$

We can now devise a single-sided test on $z(r)$ to answer the question: If $z(r) > z(r_\alpha)$, what is the significance level with which we can reject the hypothesis $z(\rho) < z(\rho_c)$ (and thus $\rho < \rho_c$)? At the tail of the normal distribution, the area under the normal curve to the right of $z(r_\alpha)$ can be approximated by:

$$\alpha = \int_{z=z(r_\alpha)}^{\infty} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-z(\rho_c))^2}{2\sigma^2}}\, dz \approx \frac{\sigma}{\sqrt{2\pi}\,(z(r_\alpha) - z(\rho_c))} e^{-\frac{(z(r_\alpha)-z(\rho_c))^2}{2\sigma^2}} \tag{5.16}$$

Taking natural logs, replacing α with the Bonferroni correction $\alpha = \alpha'/N$, and with $\sigma = 1/\sqrt{n-3}$, we arrive at:

[6] In fact, it is sufficient that the false positives do not grow faster than the *true* correlations. If we assume the number of true correlations per gene increases at least as N^γ with the number of genes (with $0 < \gamma < 1$, i.e. both the number of clusters and number of genes per cluster increases), then $\alpha = \alpha'/N^{1-\gamma}$ suffices. (For example, when the number of true correlations grows linearly with N, i.e. $\gamma = 1$, we can allow the number of false positives to grow linearly as well, so no correction is needed.) When we plug $\alpha = \alpha'/N^{1-\gamma}$ into Equation 5.16, the resulting growth rate for n is similar to the one for $\alpha = \alpha'/N$.

$$\begin{aligned}\ln(\alpha') - \ln(N) &\approx -\frac{1}{2}\ln(n-3) - \ln\left(\sqrt{2\pi}\left(z(r_\alpha) - z(\rho_c)\right)\right) \\ &\quad -(n-3)\left(z(r_\alpha) - z(\rho_c)\right)^2/2 \end{aligned} \tag{5.17}$$

$$\begin{aligned} n &\approx 3 + \frac{2}{\left(z(r_\alpha) - z(\rho_c)\right)^2} \cdot \\ &\quad \left(\ln(N) + \ln(1/\alpha') - \ln(n-3)/2 - \ln\left(\sqrt{2\pi}\left(z(r_\alpha) - z(\rho_c)\right)\right)\right) \end{aligned} \tag{5.18}$$

Although this defines n recursively, as a function of $\ln(n-3)$, the dominant term will be $\ln(N)$. In other words, if we want to use the same cutoff value r_α to decide whether $\rho > \rho_c$, we need to scale the number of data points logarithmically with the number of genes. Strictly speaking, this analysis only holds for correlation tests, but we can expect similar effects to play a role in other clustering algorithms.

5.1.8 Summary

Table 5.1 provides an overview of some of the models considered, and estimates of the amount of data needed for each. These estimates hold for independently chosen data points, and only indicate asymptotic growth rates, ignoring any constant factors. Note also that these estimates reflect the amount of data needed to be able to reconstruct the *entire* network correctly.[7] As mentioned before, we are content with being able to extract the most significant interactions.

For reasonably constrained models, the number of data points needed tends to scale with $\log(N)$ rather than N, and that the data requirements for network inference are at least a factor K larger than for clustering.

[7]For example, the sample complexity for reconstructing a *single* gene in a Boolean, sparsely connected network scales with $2^K K$, rather than $2^K(K + \log(N))$ (using Equation 5.6 rather than 5.7 in Section 5.1.3)

Model	Data needed	
General:	$K\log(N/K) + \lambda_n p_n + \lambda_k p_k K$	
Boolean:		
fully connected	2^N	
connectivity K	$2^K(K + \log(N))$	[64, 142, 4]
linearly separable, connectivity K	$K\log(N/K)$	[104]
Continuous:		
additive, fully connected	$N + 1$	
additive, connectivity K	$K\log(N/K)$ (*)	[64]
Clustering:		
pairwise correlation	$\log(N)$	[64]

Table 5.1: Sample complexity for various network models. Fully connected: each gene can receive regulatory inputs from all other genes. Connectivity K: at most K regulatory inputs per gene. Additive, linearly separable: regulation can be modeled using a weighted sum. Pairwise correlation: significance level for pairwise comparisons based on correlation must decrease inversely proportional to number of comparisons. (*): conjecture.

In practice, the amount of data may need to be orders of magnitude higher because of non-independence and large measurement errors (see also [222]). Higher accuracy methods such as RT-PCR yield more bits of information per data point than cDNA microarrays or oligonucleotide chips, so fewer data points may be required to achieve the same accuracy in the model. (Conversely, if measurement accuracy is low, more data points may be required.) So far none of the sample complexity estimates on this sort of network models includes accuracy of the data. However, we may be able to use a rough guideline provided by information theory, by looking at the information capacity of a Gaussian channel. It can be shown that the maximum amount of information (in bits) encoded in a Gaussian distributed variable with variance $P = \sigma_P^2$, when measured together with an additional Gaussian noise with variance $N = \sigma_N^2$, is given by:

$$I = \frac{1}{2}\log_2(1 + \frac{P}{N}) \approx \log_2(\sigma_P) - \log_2(\sigma_N) \tag{5.19}$$

where the approximation is within 5% for $\sigma_N < \sigma_P/3$. In other words, every halving of the measurement error increases the amount of information per measurement by one bit. This may not sound much, but consider that with a 10% measurement noise, the information capacity is only $I \approx 3.3$ bits per measurement. The logarithmic scaling does indicate a decreasing usefulness of improving accuracy much further, especially in view of significant amounts of inherent variability in the systems being measured.

Note that modeling real data with Boolean networks discards a lot of information in the data sets, because the expression levels need to be discretized to one bit per measurement. In the example above, with $I \approx 3.3$ bits per measurement, discretizing to one bit would throw away almost 70% of the information contained in the signal. Continuous models will tend to take better advantage of the available information in the data.

Another important issue in design of gene expression experiments is whether to allocate the—so far—often limited and expensive supply of microarrays or oligonucleotide chips to collecting more replicates, or more individual data points. Again, from an information theoretic point of view, n replicates reduce the noise variance by a factor of n, increasing the information content at most with $\log(n)$. Independent measurements on the other hand increase the information content proportional to n, and are therefore—*theoretically*—preferred. However, if the noise on the measurements is significant, it will generally be much harder to extract this additional information without a very good model of the noise involved. Replicates have often been required for publication for other types of biological experiments (usually at least triplicates, so a standard error can be estimated), and it seems like the consensus may be moving in that direction for expression data as well [87, 141]. As the cost per experiment decreases, this issue will likely resolve itself in favor of doing more measurements altogether, i.e., more experiments and more replicates per experiment.

5.2 The Curse of Dimensionality

Measuring more variables allows for a *more exact* model, but makes the *correct* model exponentially harder to find.

When faced with the task of modeling an unknown process, our intuition tells us to observe as many parameters of the system as possible. This is clearly reflected in the current tendency to measure the expression levels of more and more genes simultaneously, rather than to measure these expression levels as often as possible.

However, in Machine Learning it is well known that the more variables one models, the *harder* the modeling task becomes, because the space of models to be searched increases exponentially with the number of parameters of the model, and therefore with the number of variables. This is often referred to as the Curse of Dimensionality [28].

Does this mean that our intuition about modeling is wrong? Not necessarily. Although we humans do want to be able to look at as many variables of the problem as possible, we rather quickly select those we think are really important to the system, and simply ignore the others. Our reason for wanting to know all the variables is so we wouldn't miss any of the important ones, not so we could include all the non-important ones in our model. In order to achieve an accurate model, we must at least measure those variables which are important to the process being studied. If some intermediate variables are not measured, it may be possible to infer them during the modeling process, but this can be very hard. We should be as inclusive as possible in which variables we measure, and try to eliminate redundant variables after the data is collected. Careful selection of the input variables is crucial to get around the Curse of Dimensionality. Use of a priori information can also help narrow down the range of plausible models. As we saw in Section 5.1, narrowing down the range of plausible models by putting on additional—realistic—constraints can simplify the

search for the best model considerably. For example, constraining the genes to be regulated by no more than 5-7 other genes will simplify the number of regulatory interactions we need to consider. Similarly, for Boolean networks, constraining the types of Boolean functions to those that are biologically plausible can significantly reduce the number of Boolean rules that match the data.

Constraining the model by using a priori information about what is biologically known or plausible is probably the most important weapon we have to fight the Curse of Dimensionality. How precisely to include this information into the inference process is the true art of modeling.

5.3 Types of data

To infer the regulation of a single gene, we need to observe the expression of that gene under many different combinations of expression levels of its regulatory inputs. This implies sampling a wide variety of different environmental conditions and perturbations. Therefore, the gene network inference techniques we will cover all have one thing in common: they tend to be data-intensive.

Gene expression time series yield a lot of data, but all the data points tend to be about a single dynamical process in the cell, and will be related to the surrounding time points. Therefore, a 10-point time series can generally be expected to contain less information than a data set of ten independent expression measurements under different environmental conditions, or with mutations in different pathways. The advantage of a time series is that it can provide crucial insights into the dynamics of the process. On the other hand, data sets consisting of individual measurements provide an efficient way to map the attractors of the network. Both types of data, and multiple data sets of each, will be needed to unravel the regulatory interactions of the genes.

5.4 Combining different data types

The need for large amounts of data means that successful network modeling efforts will probably have to use data from different sources, and deal with different data types such as time series and steady-state data, different error levels, incomplete data, etc. Whereas clustering methods can use data from different strains, in different growth media etc., combining data sets for reverse engineering of regulatory networks requires that differences between the experimental conditions be quantified much more precisely. Likewise, data will have to be calibrated properly to allow comparison between data sets. Relative expression ratios have limited usefulness unless they can be calibrated with respect to other data sets post facto (e.g. using expression levels relative to a given standard). In this respect, there is a growing need for a reliable reference in relative expression measurements. An obvious approach could be to agree on a standard strain or tissue pool and carefully controlled growth conditions to use in all data collection efforts on the same organism. Alternatively, a reference mRNA population with fixed relative concentrations of mRNA's could be generated artificially, or perhaps even derived directly from the genomic DNA.

As individual data sets become larger, the amount of analysis that can be done within a single data set increases as well. But unless we can have confidence in comparing results from different experimenters, we potentially miss out on an enormous resource: the combined data of all researchers examining the same organism.

Chapter 6

A linear model of CNS development and injury

We will start by examining the most simple form of additive regulation models: a purely linear one, where changes in expression levels are linearly correlated with expression levels of other genes. This first-order approximation model is then applied to a set of real-world gene expression time series on development and injury of rat central nervous system. We first examine some of the higher-level properties of the resulting linear model (such as limited connectivity of the network), and find that they are biologically plausible. Next, we develop a methodology to identify those specific weights in the network which are well-defined by the data. The results of this analysis compare favorably with what can be found in the literature regarding these genes.

6.1 A first-order approximation

Have no fear of perfection – you'll never reach it.

— Salvador Dali

The whole idea of correctness is totally overrated.
— Stephanie Forrest, 10/29/99

As we will show, even the simplest form of the additive regulation model (Equation 1.1) can give interesting and suggestive results. Of course, a linear model such as this is unlikely to be much more than a caricature of the real system, and should be thought of as a first-order approximation. This is because its purely linear form cannot correctly model nonlinear interactions. However, we do expect it to be able to capture many important linear components of gene regulation. In that sense, it has similar strengths and weaknesses as using linear (Pearson) correlation to analyze any real-world variables. Although it is not an optimally fitting model, the majority of applied statistics is, similarly, based on linear correlations. The value of a coarse model like this is mainly exploratory. It serves to direct further detailed investigation by suggesting novel hypotheses about the system.

Let us first rewrite Equation 1.1 as a difference equation, explicitly introducing the time step Δt:

$$\frac{\Delta y_i(t)}{\Delta t} = \sum_j w_{ji}\, y_j(t) + b_i \tag{6.1}$$

where $y_i(t)$ is the expression level of gene i at time t, $\Delta y_i(t) = y_i(t + \Delta t) - y_i(t)$, w_{ji} indicates how much the level of gene j influences gene i, and b_i is a constant bias factor to model the activation level of the gene in the absence of any other regulatory inputs. Each "node" in the regulatory network model performs a simple summation of its inputs, as illustrated in Figure 6.1.

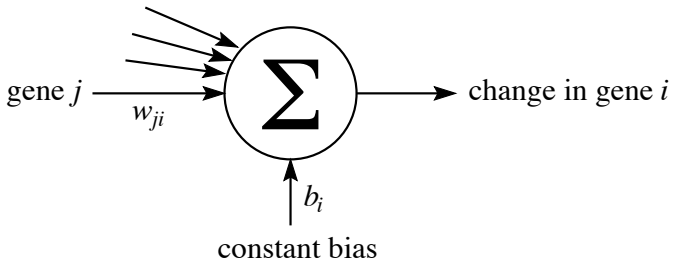

Figure 6.1: Schematic illustration of a node in the linear network model. The input from all regulatory genes is summed up, together with a constant bias term. The result determines the change (i.e., slope) in expression level of the corresponding gene.

Note that we could equivalently rewrite this equation as an *update rule*, by multiplying both sides by Δt and adding $y_i(t)$:

$$y_i(t + \Delta t) = \sum_j w'_{ji}\, y_j(t) + b'_i \tag{6.2}$$

where $w'_{ji} = \Delta t\, w_{ji}$ ($+1$ if $i = j$), and $b'_i = \Delta t\, b_i$. In this more general form of an update rule, there is no implicit assumption that $\mathbf{y}(t)$ should be a smooth—or even continuous—function in time. It is included here mainly to illustrate the similarity with the Boolean network formulation, and some earlier work on continuous models using an update rule formulation.[1] Note also that the parameters w'_{ji} and b'_i are dependent on the time step Δt in this formulation.

It is important to keep in mind that it is not the current expression *level* which is regulated in the first place, but rather the transcriptional state of the gene. Whereas

[1]For example, Weaver *et al.* [241] generated random sparse weight matrices $\mathbf{w}'$ for an update rule similar to Equation 6.2, and showed that the corresponding network models can be reconstructed given enough data generated by the network. In their experiments, the generated "expression levels" $\mathbf{y}(t)$ often jumped around erratically from time point to time point. Comparing Equations 6.1 and 6.2 shows that $\mathbf{w}' = \Delta t\, \mathbf{w} + \mathbf{I}$, so in order to get a smooth time series for small Δt, $\mathbf{w}'$ should be close to the identity matrix. It is the weight matrix $\mathbf{w}$ which corresponds to our intuitive notion of a connection matrix, not $\mathbf{w}'$.

the transcriptional state may show an on-off behavior at small time scales, the actual expression level is due to the accumulation of mRNA, essentially related to the integral of the transcriptional state of the gene over time[2]. Since we want to model real gene expression, with expression levels $\mathbf{y}$ that are smooth in time, i.e. $\mathbf{y}(t+\Delta t) \approx \mathbf{y}(t)$, we will instead use the difference equation formulation of Equation 6.1 If we choose Δt small enough, the parameters w_{ji} and b_i of Equation 6.1 will approach the parameters of the corresponding differential equation (and therefore be independent of the time step Δt):

$$\frac{dy_i(t)}{dt} = \sum_j w_{ji}\, y_j(t) + b_i \tag{6.3}$$

In addition to regulation by other genes within the data set, the genes may also be affected by changes in a number of exogenous inputs which we will have to include in the model (e.g. externally added chemicals in a toxicological experiment, depletion of nutrients in the growth medium, changing temperature, etc.):

$$\frac{\Delta y_i(t)}{\Delta t} = \sum_j w_{ji}\, y_j(t) + \sum_k v_{ki}\, x_k(t) + b_i \tag{6.4}$$

where $x_k(t)$ is the level of exogenous input k at time t, v_{ki} accounts for the effect of this input on the expression level of gene i.

Because of the need for fairly large amounts of data, measured under different conditions, we may need to combine several data sets. In fact, the data I will be using (see Section 6.2 contains measurements on two different tissue types. Differences in gene expression between tissues are caused by regulatory inputs to the genes. Some of these regulatory inputs will be included as variables in our model, others might not.

[2]This observation also was the inspiration for Glass's work on modeling gene regulation using a hybrid Boolean model with piecewise linear dynamics [84, 86], where the expression level increases or decreases linearly, depending on whether the gene is ON or OFF.

We could account for those extra regulatory variables which are purely tissue-specific (i.e. they vary depending on tissue, but do not vary within a given tissue) by adding an additional "endogenous" input for each. However, under the linear assumption, the total effect of all these tissue-specific inputs can be summarized with a single tissue-specific term T_{li} for each additional tissue l:

$$\frac{\Delta y_i(t)}{\Delta t} = \sum_j w_{ji}\, y_j(t) + \sum_k v_{ki}\, x_k(t) + \sum_l T_{li}\, \tau_l + b_i \tag{6.5}$$

where τ_l is an indicator variable which is 1 *iff* the particular data we are modeling comes from tissue l (otherwise 0), and T_{li} sums up the tissue-specific differences in regulation by other variables that are not included in the data set. Equivalently, we can think of genes having a different default expression state within each tissue. For a single tissue, this default expression state was modeled using the bias term b_i. Likewise, we can think of $T_{li}\,\tau_l + b_i$ (which is constant, but different for each tissue type) as modeling the default expression state in tissue l.

Given the time series $y_i(t)$, finding these parameters requires solving a least squares system of linear equations, or, equivalently, performing a multiple regression of each gene on all other genes. In Section 6.3 we will show how we can apply a model such as this on real data.

6.2 Data sets

It is a capital mistake to theorize before one has data.
Insensibly one begins to twist the facts to suit theories,
instead of theories to suit facts.
— A. Conan Doyle

Wen *et al.* [245] have published a Gene Expression Matrix of 112 mRNA species measured at nine different stages during the development of rat cervical spinal cord:

embryonic days 11-21 (E11, E13, E15, E18, E21), postnatal days 0-14 (P0=E22, P7, P14), and adult (A=P90). More recently, the same team developed a similar data set [212] of 70 mRNA species measured at nine time points during development of rat hippocampus (E15, E18, P0, P3, P7, P10, P13, P25, A=P60), and at ten more time points (0h=P25, 0.5h, 1.5h, 3h, 6h, 24h, 48h, 10d, 21d, 32d, 49d) following injury of the central nervous system by injection with kainate (kainic acid), a glutamatergic agonist which causes seizures, localized cell death, and severely disrupts the normal gene expression patterns.

The unequal spacing of time points was carefully chosen to coincide with the varying rate of development and response to injury of the rat central nervous system. The genes measured are only a tiny fraction of the total number of genes expressed in these tissues. However, they were selected to be representative of some of the major gene families assumed to play an important role in CNS development, intracellular signaling or transcriptional regulation in general: neurotransmitter synthesizing and metabolizing enzymes, neurotransmitter receptors, various signaling peptides (neurotrophins, heparin binding growth factors, insulin-like growth factors) and their receptors, cell cycle proteins, transcription factors, as well as developmental marker proteins and some expressed sequence tags (EST's).

Each data point in these time series is the result of measurements on three separate animals. This ensures high accuracy, eliminates some of the variability between individuals, and gives us an idea of the variability at each point ("triplicate standard deviation", see Section 6.4.2 for an example of how this additional information can be exploited). When I started working on these data sets, these were the largest publicly available gene expression time series in terms of number of time points, using a high fidelity gene expression assay. As of this writing, they still stand out for their relatively high quality, although they have since been surpassed in terms of number of genes and number of data points.

Considering the large amount of overlap between the mRNA species for the data sets (65 species in common) and the related tissue types (rat cervical spinal cord and hippocampus), it is possible to join them into one larger data set of 65 genes by 28 time points, consisting of 1) cervical spinal cord development, 2) hippocampus development, and 3) hippocampus injury. The regulatory "hardware" of the genes is the same, though different parts of it might be active in different contexts. Combining data from different tissues allows us to get a more complete picture of the regulatory interactions, provided we account for tissue-specific differences in regulation.

As mentioned before, The choice of these data sets should be viewed in a historic perspective (even though they are only a couple of years old!): they were the best that was available at the time. However, it should be pointed out that they are far from optimal for the sort of models we are interested in. In particular, they consist essentially of whole-tissue samples, measuring the average expression levels in the entire cervical spinal cord or entire hippocampus of an individual. These tissue can be further subdivided into different anatomical regions, each of these regions typically consists of several functionally different layers of cells, and each of these layers consist of different cell types. This obviously violates our earlier statement that we want to focus on genetic regulatory networks at the level of single cells, ignoring cell to cell interactions and spatial differentiation. Yet, as we shall show, even from these coarse-grained, whole-tissue measurements, we are able to derive genetic regulatory interactions which compare well with the existing literature.

Initial analysis of the Gene Expression Matrix presented in Wen *et al.* [245] was based mainly on similarities between temporal gene expression patterns measured using a Euclidean distance metric [213]. Genes were clustered hierarchically, and waves of activation were identified, representing sets of genes that were turned on in a sequential manner during the course of development.

I have previously presented a preliminary statistical analysis of this data set (D'haeseleer *et al.* [65]), in which relationships between individual genes were inferred based on linear correlation, rank correlation and mutual information. Several gene pairs with high linear correlation were identified, as well as a number of genes with high rank correlation but non-significant linear correlation. Although the number of data points per gene was insufficient to derive real results, the use of mutual information (see, e.g. [201, 59]) to derive causal inferences was illustrated. Since then, a few other groups have analyzed this data as well. For example, Wahde and Hertz [239] used the clusters derived in Wen *et al.* [245] to construct a little cluster network (see also Section 3.6).

6.3 Fitting the model

Truth . . . and if mine eyes
Can bear its blaze, and trace its symmetries,
Measure its distance, and its advent wait,
I am no prophet - I but calculate.
— Charles MacKay ("The Poetical Works of Charles MacKay")

The data sets used here cover two tissue types, and include one single exogenous output to the system (kainate). Equation 6.5 becomes:

$$\frac{\Delta y_i(t)}{\Delta t} = \sum_j w_{ji}\, y_j(t) + K_i\, \kappa(t) + T_i\, \tau + b_i \tag{6.6}$$

where $\kappa(t)$ is the kainate level at time t, K_i is the influence of kainate on gene i, τ is an indicator variable for tissue type ($\tau = 0$ for spinal cord, $\tau = 1$ for hippocampus), and T_i accounts for all the differences in regulation between tissue types. Figure 6.2 shows schematically what a "node" in the corresponding linear network model looks like.

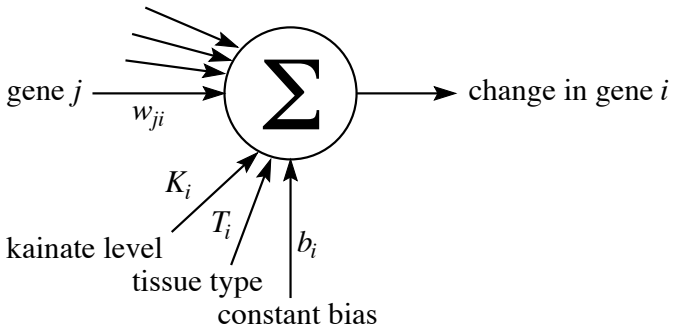

Figure 6.2: Schematic illustration of a node in the linear network model for the CNS development and injury data. The input from all regulatory genes is summed up, together with an input from the kainate level, a constant bias term, and an additional term to cover tissue-specific differences in regulation. The result determines the change (i.e., slope) in expression level of the corresponding gene.

Because the original data sets consist of raw ratiometric RT-PCR measurements, we first normalize the expression level of each gene with respect to its maximum level over all three data sets. This gives us a basis to compare the interaction strengths of the genes. Normalization is more commonly done with respect to the average signal, or with respect to the standard deviation of the signal. However, since this data is a coarse and non-uniform sampling of a time-series, these concepts are ambiguous (Should we average over the data set? Over the interpolated time series? Should we weight the time series based on developmental speed?). In addition, the maximum expression level of a gene is a useful biochemical concept, related to its production and decay rates. This choice of normalization also allow easier comparison with Chapter 7, where we use a sigmoidal transfer function to cap the production rate of the genes to a maximum level.

The linear model in Equation 6.6 can be fit to a time series finely sampled at equidistant time points Δt. Considering the extremely non-uniform spacing of the measurements (half hour interval after kainate injection, more than two months interval before the final adult cervical spinal cord measurement), we next constructed

a finely interpolated time series from the data. Because the modeled variables correspond to concentration levels, we need to avoid negative values in the interpolation. This is achieved by first taking the logarithm of the expression values, applying the interpolation on these log expression levels, and then taking the exponential of the resulting interpolation. We use a piecewise cubic interpolation method, more specifically a multivariate variant of Akima interpolation [2]. This is a local, C^1 (continuous in the first derivative) method, where the interpolation only depends on the nearest data points, and which does not tend to show the spurious excursions between data points common to, for example, cubic spline interpolation (which also imposes C^2 continuity). An interpolation rate of 10 time points per hour gives us 5 interpolated points between the two closest measurements: fine enough to yield a reasonable approximation to the differential equation, while still allowing us to calculate the least squares fit over the entire 7-month data set. We get 24241 interpolated time points for the spinal cord data (101 days), 16081 for the hippocampus development data set (67 days), and 11761 for the hippocampus kainate injury data set (49 days), for a total of 52083 interpolated time points.

The kainate concentration $\kappa(t)$ is zero during the spinal cord and hippocampus time series, jumps from zero to one at 0h for the kainate time series, and then exponentially decays back to zero: $\kappa(t) = e^{-(t-0h)/D_{KA}}$. Kainate tends to disappear from the brain after several hours [244]. We chose an estimated decay constant of $D_{KA} = 100$ min, corresponding to a half-life of 69.3 min.[3]

Note that the (nonlinear) interpolation has a crucial side-effect: it introduces an implicit additional smoothness constraint on the time series between the measured data points. This smoothness constraint is justified by the effort that went into determining at what time points measurements should be taken. If the measurement

[3]As we will see in Section 6.4.2, knowing the exact *in vivo* decay rate D_{KA} for kainate is not crucial, as randomly varying D_{KA} within a fairly large range has little effect on the results.

rate is fast enough to keep up with the fastest developmental or perturbational changes in the system, we can assume the trajectory of the system between data points to be smooth.

Normally, trying to fit a model with 68×65 parameters (including the additional terms in Equation 6.6) using only 28×65 data points would lead to a highly underdetermined system. In other words, we would be able to find infinitely many models—with different sets of parameters—that all fit the data perfectly. However, the additional smoothness constraint on the data, allows us to exclude all those models that behave very erratic in between the measured data points. In addition, it also assures that the system has a single optimum, so the fitting becomes (barely) feasible. We do expect there to be many dimensions in which the optimum is poorly determined, corresponding to parts of the model for which not enough data is available. Section 6.4.2 will illustrate how one can identify which parts of the model are well or underdetermined.

The actual fitting of the model to the data requires a small amount of linear algebra, which is summarized in Appendix B. The end result is a matrix $\mathbf{W}^+$, containing the least squares fit of the parameters w_{ji}, K_i, T_i and b_i in Equation 6.6. The computational complexity of finding a least-squares solution for a linear model is $\mathcal{O}(TN^2)$, where T is the total number of time points in the interpolation, and N is the number of genes. Not surprisingly, the shortage of original data points relative to the number of dimensions of the problem results in a poorly conditioned system, with condition number $6.1 \cdot 10^4$. This condition number gives an upper bound for how much the relative error in $\mathbf{Y}$ (the interpolated gene expression time series) could be magnified in the least squares solution, $\mathbf{W}^+$.[4] In other words, if we are given $\mathbf{Y}$ plus some small error term $\delta\mathbf{Y}$, the resulting weight matrix will be $\mathbf{W}^+$ plus some error $\delta\mathbf{W}^+$. For a poorly conditioned system, the *relative* error $||\delta\mathbf{W}^+||/||\mathbf{W}^+||$ may be

[4]This is just yet another way of saying the model is poorly determined: a large range of parameter sets $\mathbf{W}$ all show a good fit with the input data $\mathbf{Y}$

much larger than the relative error in the input, $||\delta\mathbf{Y}||/||\mathbf{Y}||$, and the magnification of this error is upper-bounded by the condition number of the system (in this case, the condition number of the augmented input matrix, $\widetilde{\mathbf{Y}}$, see Appendix B):

$$\frac{||\delta\mathbf{W}^{+}||}{||\mathbf{W}^{+}||} \leq cond(\widetilde{\mathbf{Y}})\frac{||\delta\mathbf{Y}||}{||\mathbf{Y}||} \tag{6.7}$$

However, this is a worst-case scenario and assumes, among other things, that the error in $\mathbf{Y}$ can vary independently for each interpolated time point. In reality, the nonlinear interpolation spreads out any errors in the original data sets over a range of interpolated time points, improving the conditioning of the system with respect to the original data sets. In Section 6.4.2, we show that the noise in the input data gets multiplied by a factor of "only" 29.7 in $\mathbf{W}^{+}$: not as bad as $6.1 \cdot 10^4$, but still poorly conditioned. In fact, the main goal of Section 6.4.2 is to determine which parts of $\mathbf{W}$ are the least affected by the poor conditioning of the system (and the noise in the input data).

Likewise, the condition number of $\mathbf{W}$ is $6.3 \cdot 10^4$, indicating that small amounts of noise in $\mathbf{Y}(t)$ could result in large changes in the slope $d\mathbf{Y}(t)/dt$. However, it turns out that the dynamical behavior of the system is surprisingly robust. If we initialize the system with the gene expression levels measured at the very first time point and apply the model iteratively, we can reconstruct the trajectory through state space almost perfectly for all three data sets. Figure 6.3 shows the original and reconstructed time series for three representative genes. The interpolated time series (not shown) are nearly indistinguishable from the reconstruction. The very close fit is likely due to overfitting, but it does show that errors do not accumulate, despite the poor conditioning of $\mathbf{W}$. Analysis of the eigenvectors of the linear system also reveals that the final expression levels are close to fixed points of the system (within 3% for the spinal cord and hippocampus "adult" expression levels, within 9% for the final hippocampus injury expression levels): the linear model settles into an attractor

in state space corresponding to the adult expression levels of the real organism.

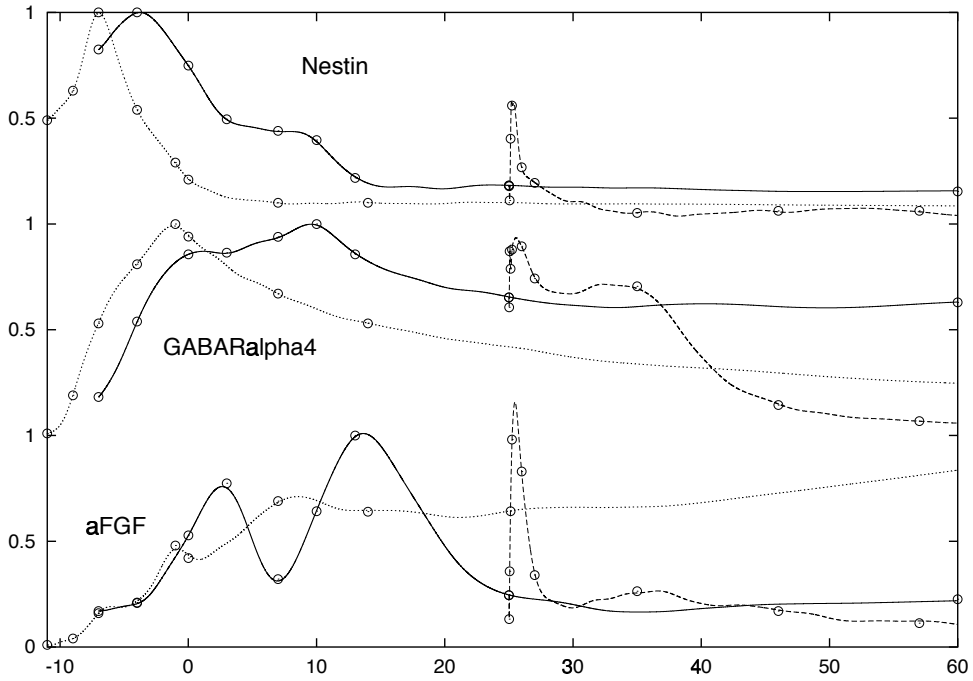

Figure 6.3: Original (dots) and reconstructed time series (lines) for nestin(top), GRa4 (middle) and aFGF (bottom). Time is in days from birth (day 0, corresponding to postnatal day P0 or embryonic day E22). Dotted line: spinal cord, starting day -11 (E11). Solid line: hippocampus development, starting day -7 (E15). Dashed line: hippocampus kainate injury, starting day 25 (P25)

6.4 Results and validation

A theory has only the alternatives of being right or wrong.
A model has the third possibility: it may be right, but irrelevant.
— M. Eigen

Before we address the issue of which individual parameters are well determined versus poorly determined, Section 6.4.1 will look at some of the overall properties of $\mathbf{W}$. Just as the average of a large number of poor estimators can yield a good

estimator, the hope is that these global properties may be better determined than the individual parameters. Next, Section 6.4.2 shows how we can "separate the wheat from the chaff": identify the few well determined interactions in the network model. In Section 6.4.3 we put the class of most robust parameters (those due to the effect of kainate on the genes) to the test by comparing them with what is known in the literature. Section 6.4.4 does the same for the most robust gene-to-gene parameters. Finally, Section Section 6.4.5 compares the results of the linear model with a linear model based on randomized data.

6.4.1 Biologically plausible properties?

The linear model assumes that every gene is regulated by every other gene. However, when we look at the least squares fit of the model to the real data, we find that many of the parameters of the model are close to zero. Figure 6.4 shows a distribution of interaction weights that is very sharply peaked around zero (with 25th and 75th percentiles at $\pm 0.258\,\sigma$, compared to $\pm 0.674\,\sigma$ for a normal distribution). This means the connection matrix is a good approximation to a sparse matrix, i.e., each gene is only influenced by a limited number of others, as we would expect for the real connection matrix. For a rough estimate of the number of "nonzero" parameters, we can fit the distribution with a mixture of two zero-centered Gaussians: more than 80% of the parameters get assigned to the narrowest Gaussian ($\sigma_1 = 0.068$), the rest to the much broader second Gaussian ($\sigma_2 = 0.375$).

The sum of input weights to each gene is close to zero, i.e. there seem to be no genes that are primarily upregulated or downregulated. In fact, the distribution of the sums of input parameters is significantly much closer to zero than would be expected based on the distribution of parameters ($\sigma = 0.542$ versus expected $\sigma = 1.648$).[5] This may partially be an artifact of the model, because there are no

[5]The expected standard deviation of the sum (assuming the parameters are picked ran-

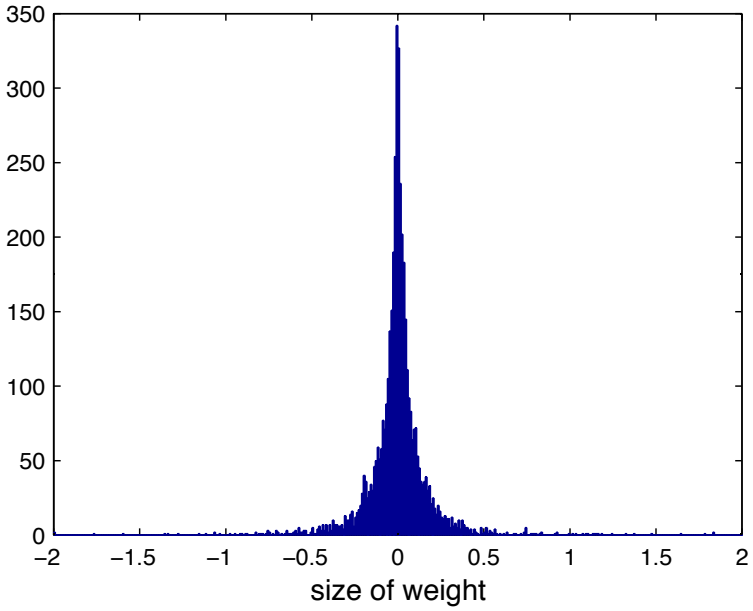

Figure 6.4: Histogram of average parameter values $\bar{w}$. Note the sharp peak at zero.

fixed upper and lower expression thresholds for each gene. Predominantly positive (or negative) inputs to a gene would cause a increasing (decreasing) expression level, so positive and negative inputs must be balanced. More surprisingly, the distribution of input sum is close to zero even if we exclude the bias term b_i for each gene. In other words, we see few instances of genes which are "OFF" in the absence of regulatory inputs and which are upregulated by those inputs, or "ON" in the absence of regulatory inputs and downregulated.

Looking at the sum of output parameters from each gene (or other input such as kainate etc.), we see that this sum varies significantly more than expected based on the distribution of parameters ($\sigma = 2.513$ versus expected $\sigma = 1.611$).[5]In other words, there seem to be genes that have a predominantly positive (e.g. GRγ1) or negative (e.g. IGF II) regulatory effect on the other genes, which is in agreement

domly from the distribution of parameters) is $\sqrt{68}\,\sigma_{params}$ for the input vector, $\sqrt{65}\,\sigma_{params}$ for the output vector, with $\sigma_{params} = 0.200$

with our biological knowledge.

Whereas the sum of input or output vectors tells us about the sign of regulation, the magnitude of the vectors informs us about the strength of regulation. We see a few significantly larger (e.g. GRγ1) and especially more small magnitude output vectors than expected, given the distribution of parameters.[6] It seems likely that the model has discovered that some genes are important regulators, while many others are not. This explanation is reinforced by a significant negative correlation ($r = -0.46$) between output magnitude and average triplicate variability for the gene, i.e. genes with less variation among the three replicates per time point had higher output magnitude. Important regulators are presumably more tightly regulated themselves, and thus would be expected to show less variability.

Surprisingly, we see a similar pattern for the input vectors: a few genes have large regulatory input parameters,[7] many others have all small regulatory inputs. Here, this variation is explained by a significant correlation ($r = 0.79$) between the magnitude of input vectors, and the standard deviation of slopes between time points. Since the linear model correlates expression levels with changes in expression levels, genes with rapid changes between time points will tend to have larger regulatory input parameters.[8] Each gene has a characteristic scale for its input parameters, corresponding to how fast the expression level of the gene changes throughout the time series. Instead of a mixture of "zero" and "nonzero" parameters, distribution

[6]Based on 100 random permutations of the parameter matrix.

[7]In our earlier work [66], this was assumed to be a sign of a poorly determined gene: poorly determined variables are often fitted using a number of very large inputs, which mostly cancel each other out. However, as we will see in Sections 6.4.4 and 6.4.3, one of the genes with highest input magnitude (BDNF) also has very well determined input parameters.

[8]No such correlation was found between output vector magnitude and average expression level, and no other significant correlations were found between input and output magnitudes, average and standard deviation of expression levels, slopes, or average triplicate standard deviation.

of parameter values in Figure 6.4 should probably be considered as a mixture of distributions at these different scales.

When we divide the genes into functional categories, other interesting patterns emerge. The categories used were: *5HTR* (Serotonin Receptors), *AChR* (Acetylcholine Receptors), *GABA-R* (GABA Receptors), *GluR* (Glutamate Receptors), *ICS* (Intracellular Signaling), *NME* (Neurotransmitter Metabolizing Enzymes, including GAD), *cell cycle*, *glial*, *growth factor*, *insulin and IGF*, *neuronal*, *neurotrophin*, *progenitor*, *synaptic*, *trans-regulation*, and *other*.

NME and *GluR* are the main input classes, with weights coming from these genes on average more than twice as large as from other genes. Both categories are known to play an important role in development and injury of the central nervous system. Also important are *ICS* (46% larger weights), *5HTR* (45% larger) and *trans regulation* (35% larger). A notable exception to primary regulation by *NME* and *GluR* is *growth factor*, which gets most input from *ICS*. We also observed that there is a tendency for genes in one functional class to receive more inputs from genes in the same class.

In summary, the least squares solution for the linear model results in a sparsely connected network, in which all genes have both positive and negative inputs, some genes are predominantly positive or negative regulators, there are a small number of important regulators with stable expression patterns, regulatory inputs are scaled by the speed of change in expression level of each gene, some of the main regulatory gene categories are known to play an important role, and there is more regulation within a functional category than between categories. All these high-level properties can be considered plausible from a biological point of view.

6.4.2 Robust parameters

All theorems are true.
All models are wrong.
And all data are inaccurate.
What are we to do?
We must be sure to remain uncertain.
— Leonard A. Smith

Because we expect large parts of the model to be underconstrained, we performed a Monte Carlo analysis to assess the effect of noise in the input data on the resulting parameters, and used this to determine the most robust parameters. As mentioned earlier, every value in the original data sets is really an average of triplicate experiments. This gives us high accuracy, and a rough estimate of the standard deviation at each measurement. We used this information to construct 40 new input data sets, adding a small amount of Gaussian noise (with the same standard deviation) to each. We then generated the linear model for each of these perturbed data sets, and analyzed the variability of the parameters over those 40 perturbed models.

To reflect our uncertainty about the kainate decay constant D_{KA} used to generate the kainate time series, we also lognormally perturbed D_{KA} around its estimated value of 100 min. This did not qualitatively change the results[9]. Similarly, continuity or discontinuity in the slope of the interpolated kainate time series[10] at the time of kainate injection had little effect on the results, indicating that the time resolution in the kainate time series is sufficient to capture the initial dynamics of the response.

[9]Over the 40 perturbed models, D_{KA} varied from a minimum of 45.55 min (half-life of 31.57 min) to a maximum of 205.13 min (half-life of 142.18 min).

[10]When interpolating the kainate time series, the default slope at 0h is determined by the 0h and 0.5h data points. In contrast, the slope for unperturbed animals (at postnatal day 25, but without kainate injection) can be estimated from the hippocampus development time series. Using the slope calculated based solely on the kainate time series would therefore be equivalent to introducing a discontinuous jump in slope. Alternatively, we can force the kainate time series interpolation to start with the slope found at P25 in the developmental time series.

The results listed below are for D_{KA} lognormally perturbed around 100 min, and a discontinuous slope of the interpolated kainate time series.

For each parameter in the model, we calculated the average magnitude $\bar{w}$ of the parameter, and compared it to its standard deviation σ_w over all 40 perturbed models. Note that although the original triplicate standard deviations are only a very rough estimate of the real variability for each gene and each time point, σ_w will be the result of some weighted average of a large number of these. In fact, for the specific weight w_{ji}, $\sigma_{w_{ji}}$ will be some weighted average of *all* the triplicate standard deviations for both y_i and y_j, at all time points, over all data sets. Just as the average of two poor estimators is itself a more accurate estimator (in fact, with half the variance of the original estimators), $\sigma_{w_{ji}}$ will have much greater accuracy than any of the triplicate standard deviations it is based on.

The *Z-score* of a parameter w is defined as $Z_w = |\bar{w}|/\sigma_w$, and indicates how many standard deviations the mean of the parameter is away from zero. From this Z-score, we then compute a *P value*, indicating the probability that the "real" value of the parameter for the best-fit linear model is zero, or even has opposite sign from $\bar{w}$ (i.e., the probability that this weight w is a *false positive*). We could simply count what fraction of the perturbed models have zero or even the opposite sign for the parameter in question. However, this would require many more than 40 runs to get sufficient accuracy in the P values. If we assume each parameter has a similar distribution, we can look at the distribution of all parameters, each normalized with respect to its mean $\bar{w}$ and standard deviation σ_w.[11] To estimate the P value for a specific value of Z, we count the number of instances where $(w - \bar{w})/\sigma_w > Z$, and divide by the total number of parameters ($40 \times 68 \times 65$). The Z-scores (or their

[11]The perhaps more standard—but less accurate—approach would be to assume all parameters have a Gaussian distribution over the 40 perturbed models. The distribution estimated above turns out to have a slightly sharper peak and longer tails than the Gaussian, resulting in larger P values for high Z, and smaller P values for low Z.

derived P values) are then used to identify robust parameters.

Note that the P values used here do not necessarily indicate the probability that the parameters found correspond to real biological regulatory interactions. They simply reflect the probability, given the noise on the input data, that the best-fit linear model for the true expression time series includes a parameter with this sign. In some instances, fitting a nonlinear interaction using a linear model may require a number of spurious linear terms. These parameters may be necessary for a good fit, and thus receive a high Z-score. Our hope is that gene regulation has sufficiently strong linear component that this first-order approximation with a linear model will mainly yield biologically relevant results.

6.4.3 Results: Kainate parameters

Figure 6.5 shows the Z-scores and average magnitude of all parameters. Surprisingly, the most robust parameters in the model are the parameters K_i, indicating the effect of kainate on each gene (black dots in Figure 6.5). This is probably because of the very fast and drastic effect of kainate-induced seizures on the system, as compared to the slow and subtle changes during development. Table 6.1 lists a number of the kainate $\rightarrow$ gene parameters with the highest Z-scores. Note that a few parameters (e.g. Kainate $\rightarrow$ 5-HT_{1B}) have a high Z-score but a low average magnitude. Such highly consistent but small parameter values may reflect a real but minor regulatory influence, or simply an absence of regulation (e.g. compensation for nonlinear effects).

Since it is unlikely that kainate actually regulates all these genes directly, we must assume there are some intermediate steps missing. Including an even earlier time point may shed some light on the precise sequence of regulation, especially for the BDNF/IGF II/S100β trio which also show gene-to-gene interactions (see Section 6.4.4). It should be noted, however, that most of the existing literature on

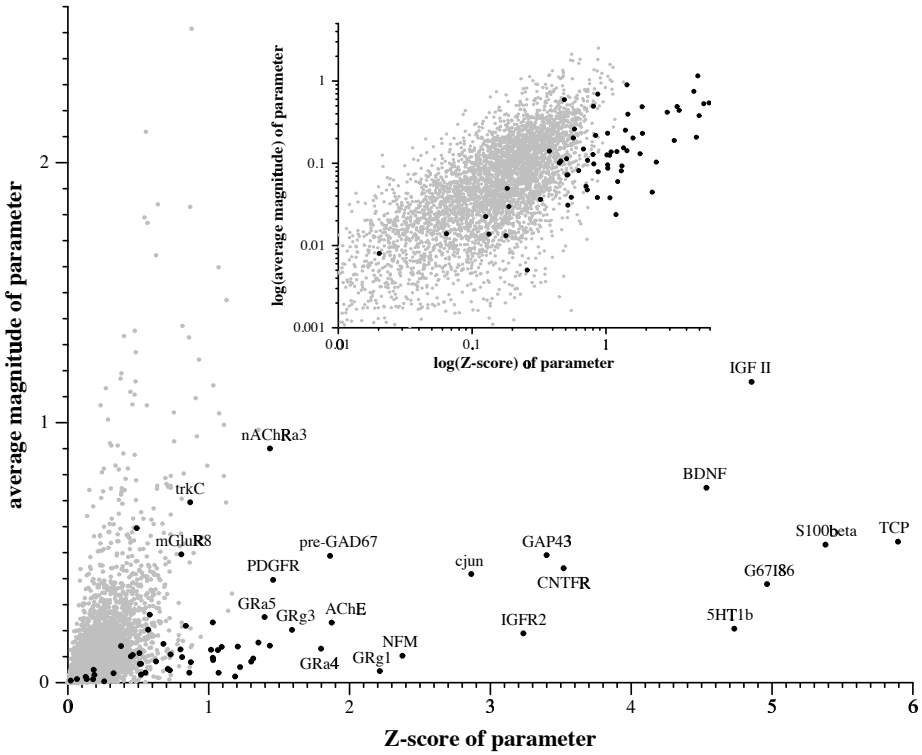

Figure 6.5: Left: Average magnitude of parameters vs. Z-score. Black points are kainate $\rightarrow$ gene parameters. Inset: log-log plot shows clearly that the kainate parameters on average have significantly higher Z_w and higher $|\bar{w}|$.

kainate response looks at much coarser time scales (typically hours), and that the 0.5 hr time interval in this data set is the shortest reported in the literature for kainate response.

Kainate $\dashv$ IGF II: Kar *et al.* [126] found that IGF I, IGF II and insulin receptor sites show a marked decrease after kainate administration, suggesting "possible involvement of these growth factors in the cascade of neurotrophic events that is associated with the reorganization of the hippocampal formation observed following kainate-induced seizures." We found a four-fold decrease in IGF II mRNA levels one half hour after onset of seizures, followed by a two-fold increase in IGF I after 6 hours, and a large decrease of all IGF's and IGF receptors around 10-21 days. Our model

Parameter	$\bar{w}$	σ_w	Z_w	P_w
Kainate $\dashv$ IGF II	-1.157	0.238	4.854	$4.355 \cdot 10^{-4}$
Kainate $\rightarrow$ BDNF	$+0.750$	0.165	4.534	$7.834 \cdot 10^{-4}$
Kainate $\rightarrow$ TCP	$+0.542$	0.092	5.894	$2.828 \cdot 10^{-5}$
Kainate $\rightarrow$ S100β	$+0.531$	0.099	5.379	$1.923 \cdot 10^{-4}$
Kainate $\rightarrow$ G67I86	$+0.379$	0.076	4.964	$3.620 \cdot 10^{-4}$
Kainate $\rightarrow$ 5-HT$_{1B}$	$+0.208$	0.044	4.732	$5.430 \cdot 10^{-4}$

Table 6.1: Robust kainate parameters. IGF II: insulin-like growth factor II; BDNF: brain-derived neurotrophic factor; TCP: T-complex protein; G67I86: glutamate decarboxylase 67 (GAD67) splice variant I86; 5-HT$_{1B}$: serotonin (5-hydroxytryptamine) receptor 1B

suggests that it is IGF II which initially sets off the widespread changes in expression levels of insulin, the insulin-like growth factors, and their receptors following kainate administration.

Kainate $\rightarrow$ BDNF: BDNF is upregulated by kainate via two different promoters in hippocampal neurons [166], and the BDNF mRNA increase due to kainate is not blocked by protein synthesis inhibitors, indicating BDNF is regulated as an immediate early gene [47]. In the kainate injury time series, BDNF expression levels increase five-fold one half hour after onset of seizures. In the adult brain, BDNF is thought to play a major role in the development of kainate-induced hypertrophy in granular neurons of the dentate gyrus region of the hippocampus: administration of antisense deoxynucleotides for BDNF (sequestering the complementary BDNF mRNA) after kainate administration totally prevented neuronal hypertrophy [92]. Hippocampal BDNF levels are also correlated with severity of seizures and the extent of neuronal loss in the CA1 and CA3 regions of the hippocampus, and administration of exogenous BDNF exacerbates the damage to CA3 neurons [190]. Interestingly, in immature (20-day-old) rats, which normally do not show neuronal loss following kainic-acid induced seizures, BDNF apparently has a neuroprotective effect: anti-

sense deoxynucleotide administration results in longer seizure duration and loss of CA1 and CA3 pyramidal cells and hilar interneurons inside the dentate gyrus [226].

Kainate → TCP: The case for kainate regulation of TCP (T-complex protein) is rather speculative, even though it is the single most robust parameter in our linear model. One intriguing link is the mapping of the epilepsy susceptibility locus EJM1 on chromosome 6 [192], near a human homologue of the mouse T-complex [68]. If TCP is indeed a major gene involved in kainate neurotoxicity, TCP gene defects might cause increased susceptibility to epilepsy.

Kainate → S100β: S100β is known to be upregulated five fold in human temporal lobe epilepsy [91]. S100β protects hippocampal neurons from damage induced by glucose deprivation [25], "suggesting that its elevation in neurological disorders may be a compensatory response." Perhaps its overexpression is a consequence of glucose deprivation due to neuronal hyperexcitation by kainate. Lastly, S100β induces apoptotic cell death in astrocytes [116], which protect against kainate neurotoxicity [157]. Hence, overexpression of S100β might cause aggravation of kainate toxicity by astrocyte apoptosis.

Kainate → G67I86: G67I86 is an embryonic splice variant[12] of GAD67 [35], expressed in mice from E10.5 to E15.5 (corresponding to rat E12 to E17), and not detectable in adult brain [221]. GAD synthesizes the fast-acting neurotransmitter GABA from glutamate. The short leader peptide translated from G67I86 is not enzymatically active, but is thought to exert some unknown regulatory function [221]. Mature GAD67 mRNA was known to be upregulated in hippocampal dentate granule cells four hours after kainic acid injection [200, 67]. However, the more fine-grained time series used here shows that G67I86 mRNA levels increase first (0.5h-1.5h),

[12]G67I86 contains a 80 bp insert not found in the adult GAD67 mRNA. This insert includes a stop codon which truncates the translation of the mRNA, resulting in a short leader peptide rather than the full-length GAD67 protein.

followed by a second embryonic splice variant G67I80[13] (1.5h), and finally the adult GAD67 mRNA (1.5h-24h). This is the same sequence in which these splice variants occur during development [221], indicating that GAD67 expression after kainate injury may be recapitulating its developmental program. Such recapitulation of developmental processes plays an important role in regeneration of the peripheral nervous system following injury, and has also been implicated in the central nervous system [56, 255, 253].

Kainate $\rightarrow$ 5-HT$_{1B}$: Kainate administration causes a release of serotonin in the hippocampus [227], which would be expected to provoke a compensatory down-regulation of the 5-HT$_{1B}$ serotonin receptor instead [125]. However, the interaction between serotonin and 5-HT$_{1B}$ is more complicated than that: 5-HT$_{1B}$ is an autoreceptor [125], i.e., activation of the receptor causes an inhibition of serotonin release. In addition, receptor activation will also cause a desensitization of the 5-HT$_{1B}$ receptor [182]. It is conceivable that this complex set of feedback loops might cause a transient upregulation of 5-HT$_{1B}$ by kainate. Indeed, the 5-HT$_{1B}$ expression time series shows a transient upregulation, peaking at 1.5-3h, followed by a decrease below the original expression level.

The direct effect of kainate is a transient phenomenon, lasting at most a couple of hours. It could be argued that we might be able to derive these kainate parameters directly from the first few time points in the kainate injury time series, in which all the effort involved in integrating results from three separate time series is entirely superfluous.

One can indeed find reasonable, intuitive estimators for K_i and the corresponding Z-score Z_{K_i} using the change in expression level between the first two time points, $\Delta y_i(0, 0.5)$ and the triplicate error $\sigma_i(0)$ and $\sigma_i(0.5)$ at those time points (see also

[13] G67I80 contains a 6bp shorter insert than G67I86, and is translated into the leader peptide, plus a truncated but enzymatically active GAD67 protein.

Figure 6.6):

$$k_i = \frac{\Delta y_i(0, 0.5)}{0.5} \approx K_i \qquad \text{(slope of the time series)} \tag{6.8}$$

$$z_i = \frac{\Delta y_i(0, 0.5)}{\sigma_i(0) + \sigma_i(0.5)} \approx Z_{K_i} \tag{6.9}$$

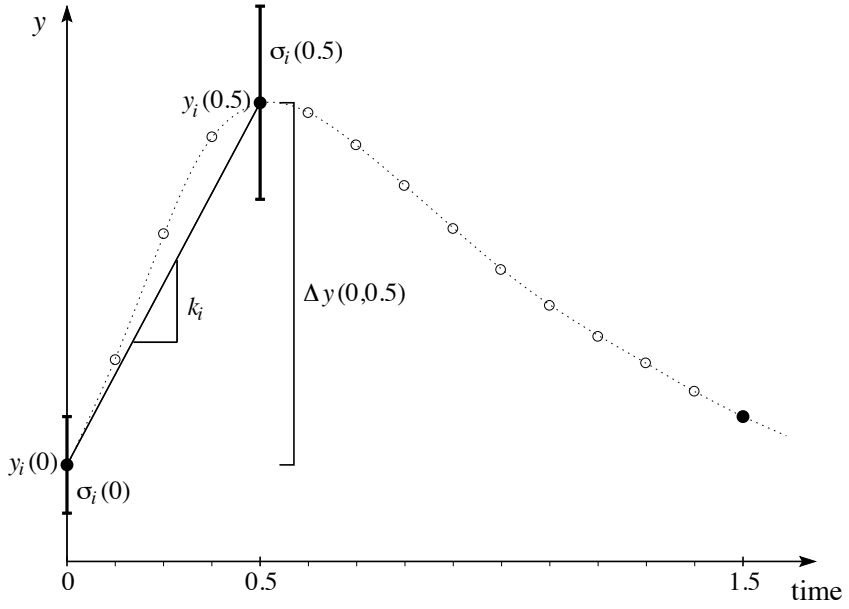

Figure 6.6: Estimates for kainate parameters K_i and their Z-score Z_{K_i}, based on the first two time points in the kainate time series. The original data points are solid, interpolated time points hollow. k_i is the slope between the two first time points in the data set, z_i is a measure of the significance of the change in expression level.

These simple estimators k_i and z_i show a reasonable correlation to the results obtained by fitting the linear model to all three complete time series: $r = 0.84$ and $r = 0.83$ respectively. Unfortunately, this nice correlation starts to break down in the most interesting region: for those genes with high Z-score. For the ten genes with highest Z-score, the correlation between k_i and K_i is only barely significant:

$r = 0.67$, and the correlation between z_i and Z_{K_i} is no longer significant: $r = 0.47$. The estimates are even worse for the six genes listed in Table 6.1:

- k_i strongly underestimates K_i for IGF II ($k_i = 0.696$; $K_i = 1.157$)
- k_i strongly overestimates K_i for BDNF ($k_i = 1.215$; $K_i = 0.750$).
- z_i strongly underestimates Z_{K_i} for S100β ($z_i = 3.333$; $Z_{K_i} = 5.379$)
- z_i strongly underestimates Z_{K_i} for 5-HT$_{1B}$ ($z_i = 3.667$; $Z_{K_i} = 4.732$),
- z_i strongly overestimates Z_{K_i} for G67I86 ($z_i = 12.333$; $Z_{K_i} = 4.964$)
- z_i strongly overestimates Z_{K_i} for BDNF ($z_i = 7.156$; $Z_{K_i} = 4.534$).

Of the six genes with largest Z-scores, TCP is the only one with accurate estimations. Both estimators are particularly unreliable for high values. Although their accuracy is probably sufficient to pick up most of the important interactions, it is clear that adding in the rest of the kainate time series, as well as the two developmental time series, significantly improves the results.

6.4.4 Results: Gene-to-gene parameters

Kainate is an exogenous input to the system, so the immediate effects of kainate administration are easy to isolate. In addition, kainate injury as a model of temporal lobe epilepsy is very well studied. The gene-to-gene interactions on the other hand are much harder to unravel, both *in vivo* as *in vitro*, and consequently less information about them is available in the literature.

In the linear model, the parameters accounting for the gene-to-gene interactions have much smaller Z-scores than for the kainate-to-gene interactions. The gene-to-gene parameters with $Z_w > 1.0$ are listed in Table 6.2. Interestingly, GRγ1 and

IGF II—accounting for seven out of ten entries in the table—also have the highest magnitude output vectors, which we interpreted as a sign of important regulatory genes in Section 6.4.1. None of the Z-scores are significant at the $P = 0.05$ level, although in total we only expect about one false positive in this table of ten parameters. Remember that the goal of this model is primarily to generate interesting new hypothesis to guide further research. From that point of view, nine out of ten is quite acceptable.

Parameter	$\bar{w}$	σ_w	Z_w	P_w
GFAP $\dashv$ GFAP	-0.277	0.243	1.138	0.097
BDNF $\dashv$ BDNF	-0.973	0.719	1.353	0.072
IGF II $\dashv$ BDNF	-1.598	1.494	1.070	0.106
BDNF $\dashv$ S100β	-0.343	0.294	1.165	0.093
IGF II $\dashv$ S100β	-0.693	0.617	1.123	0.099
GRγ1 $\rightarrow$ GRα4	$+1.144$	1.108	1.032	0.112
GRγ1 $\rightarrow$ GRβ2	$+1.036$	0.965	1.074	0.106
GRγ1 $\rightarrow$ G67I80/86	$+1.471$	1.307	1.126	0.098
GRγ1 $\rightarrow$ AChE	$+0.992$	0.895	1.108	0.101
GRγ1 $\rightarrow$ NFM	$+0.795$	0.718	1.108	0.101

Table 6.2: All gene-to-gene parameters with Z-score greater than 1.0. GFAP: glial fibrillary acidic protein; BDNF: brain-derived neurotrophic factor; IGF II: insulin-like growth factor II; G67I80/86: glutamate decarboxylase 67 (GAD67) splice variants I80 and I86; AChE: acetylcholinesterase; NFM: neurofilament medium; GRα4, GRβ2, and GRγ1: $GABA_A$ receptor subunits α4, β2, and γ1.

GFAP $\dashv$ GFAP; BDNF $\dashv$ BDNF: It is interesting to note that two out of the ten gene-to-gene parameters in Table 6.2 are autoregulatory, i.e., a gene down-regulating itself. Although these specific genes are not known to regulate themselves, in general such negative feedback loops are an important homeostatic mechanism.

BDNF, IGF II $\dashv$ BDNF, S100β: BDNF and S100β seem to be regulated by BDNF itself, and IGF II. Moreover, the regulation by IGF II is in both cases roughly twice as strong as the regulation by BDNF (2.02 times as strong for S100β, 1.64 times

as strong for BDNF, well within the error bounds on these parameters). This might lead us to infer the presence of a hidden node[14] regulating BDNF and S100β, as in Figure 6.7. All three of these genes are growth factors, playing a role in differentiation and development of neurons, as well as in neurite outgrowth. Both IGF II and BDNF induce differentiation of CNS stem cells-derived neuronal precursors, and IGF I and BDNF may act together or sequentially to promote differentiation [24] (the combination of IGF II and BDNF was not examined, but IGF II was found to have a similar effect as IGF I on differentiation). Furthermore, IGF II and S100β have almost opposite effects on the growth of developing serotonin and dopamine neurons in vitro [147]. The interactions between BDNF, IGF II and S100β may play an important role in differentiation of developing neurons into different cell types.

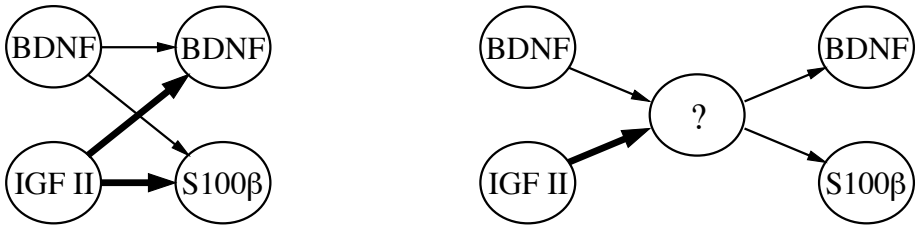

Figure 6.7: Alternative models for the interaction between BDNF, IGF II, and S100β. Note that BDNF was drawn twice for clarity.

GRγ1 $\rightarrow$ GRα4, GRβ2: GRγ1 is a $GABA_A$ receptor subunit. Each pentameric $GABA_A$ receptor consists of five subunits, and so far, 19 mammalian subunit types (plus several splice variants) have been identified, grouped into seven classes: 6 α subunit types, 4 β, 3 γ, 1 δ, 1 ϵ, 1 π, and 3 ρ [26]. In the CNS, $GABA_A$ receptors generally consist of combinations of α and β subunits, plus one or more of the γ, δ, or ϵ types, allowing for possibly hundreds of different $GABA_A$ receptors. The upregulation of α4 and β2 by γ1 seems to imply the coordinate regulation of an

[14]Interestingly, the combination (BDNF + 1.8 IGF II) shows an even higher Z-score for regulating BDNF and S100β (1.434 and 1.332), and a higher average Z-score overall (0.503 versus 0.476 for BDNF alone and 0.436 for IGF II alone)

$\alpha 4\beta 2\gamma 1$ $GABA_A$ receptor by its $\gamma 1$ subunit. This specific receptor combination has not previously been described, perhaps because $\alpha 4$ and $\gamma 1$ are less common subunits, $\alpha 4$ antibodies have yielded inconsistent results, and a frequently used $\beta 2/3$ antibody does not distinguish between $\beta 2$ and $\beta 3$ subunits. However, $\alpha 4\beta\gamma$ [15] has been detected in the cortex, striatum and hippocampal pyramidal cells [174], $\alpha 4\beta 2\delta$ has been detected in thalamus and hippocampal dentate granule cells [174] (δ is known to substitute for γ in some receptors), $\alpha 4$, $\beta 2$, and δ-mRNA levels are tightly correlated in individual dentate granule cells [41], and the hippocampus does contain some of the highest concentrations of both $\alpha 4$ [133] and $\gamma 1$ [134].

GRγ1 $\rightarrow$ G67I80/86: GABA is implicated in neuronal development, and it is thought that GAD (the enzymes(s) which synthesize GABA from glutamate) regulates the expression of GABA receptors via GABA, and that GABA receptor activation in turn regulates GAD expression [215]. GAD67, $\alpha 4$, $\beta 1$ and $\gamma 1$ expression is associated with proliferation and development in the rat embryonic and early postnatal CNS [148]. Considering the timing, this GAD67 expression presumably consists mainly of the embryonic splice variants G67I80 and G67I86. Total $GABA_A$ receptor mRNA was found to be highly correlated (R=0.99) with total GAD mRNA in cervical spinal cord [215], and it seems likely that the $GABA_A$ receptor subunits which appear transiently during spinal cord development ($\alpha 4$, $\alpha 5$, $\beta 1$, $\beta 2$, $\gamma 1$, and $\gamma 3$) would be highly correlated with the transiently expressed GAD67 variants.

GRγ1 $\rightarrow$ AChE: GABA has been conjectured to control the development of cholinergic neurons, and indeed, AChE expression is downregulated by activation of $GABA_A$ receptors [132]. Exposure to GABA has also been shown to downregulate $GABA_A$ receptor subunit $\gamma 1$, as well as $\alpha 1$, $\beta 2$, $\beta 4$, and $\gamma 2$ [27], so perhaps the effect of GABA on AChE (and $\beta 2$) is due to downregulation of $\gamma 1$.

GRγ1 $\rightarrow$ NFM: NFM (neurofilament medium) is a neuronal marker, so it is

[15] The precise subtype of β and γ was not identified

not surprising that NFM would be upregulated in conjunction with a number of neurotransmitter receptors ($\alpha 4$, $\beta 2$, $\gamma 1$) and neurotransmitter metabolizing enzymes (G67I80/86, AChE). It has also been noted that GAD family mRNA expression parallels neurofilament expression [215].

Interestingly, GRγ1 and IGF II—accounting for seven out of ten entries in the table—also have the highest magnitude output vectors, which we interpreted as a sign of important regulatory genes in Section 6.4.1. Whereas IGF II is known to be an important regulator, no such role for GRγ1 has been postulated before. The GRγ1 gene product is part of a receptor complex, and would not be expected to play any direct regulatory role. Nevertheless, it is not unheard off for a protein with a primarily structural role in the cell to also have a regulatory effect (for example, CASK, a cytoskeleton protein acting as a structural girder for cell junctions, is known to enter the nucleus and directly regulate gene expression [115]). Alternatively, some other factor not included in our data set may be driving the coordinate regulation of these genes, and be most highly correlated with GRγ1 (e.g., GRγ1 may have few other regulatory inputs, and may show a faster response to this regulator than the other genes), in which case the best causal explanation within the scope of the data set would be regulation by GRγ1. Either way, the model shows a significant coordinate regulation of these gene and, lacking any other explanations, further investigation of the role of GRγ1 may be warranted.

6.4.5 A control model

One way to illustrate the power of a modeling methodology is to apply it to a set of randomized data and compare the results with a model based on real data. Intuitively, we would like to see the model to return much more significant results on real data than on randomized data, in which case we could assume that the model

managed to extract the salient features that are present in the real data but absent in the randomized data.

Of course, there are various ways to contruct a "randomized" data set: purely random values taken from some distribution, permutations of the original data sets, etc. Ideally, one would like to use a randomized data set that has all the same properties of the real data, with the exception of the property the model is trying to capture—in this case, causal relationships between the genes. This way, the difference in performance on the two data sets can only be due to the presence or absence of the property of interest.

For the linear model and the data sets used here, it has proven quite difficult to generate a randomized data set that did not *outperform* the real data, in terms of Z-scores of gene-to-gene interactions in the resulting linear model. What makes the real data sets so difficult to deal with for the model, is the very slow, smooth changes in expression levels, combined with a high amount of correlation between the expression levels of various genes. Not surprisingly, if we choose a randomization which perturbs either of these factors, the randomized data will be easier to model with the linear model.

A number of different randomizations were tried, progressively matching more of the properties of the real data, while still avoiding to add in any causal relationships between the simulated genes. The linear model essentially correlates expression levels with changes in expression levels, so at a minimum the randomized data should have the same distributions for expression levels and slopes of expression levels. The former could be achieved for example by randomly permuting the (normalized) expression levels within each time point. However, this would result in more extreme changes in expression levels between time points. Larger slopes can be expected to result in larger parameter values. Moreover, the changes in expression level for each time interval would become more significant, with respect to the triplicate standard

deviation of the measurement on either side of the interval.

One of the most important factors that reduces the level of significance of the real model is the correlation between the genes in the real data set. Appendix C shows how we can generate randomized data sets that match the real data in (1) the distribution of expression levels at each time point, (2) the distribution of changes in expression levels between time points, and (3) the variance and covariance between simulated genes. The results obtained from these randomized data sets are quite similar to those obtained from the real data sets, with the following exceptions:

1. The control model has significantly smaller parameter weights ($\sigma_c = 0.149$ versus $\sigma_r = 0.200$ for the real data).

2. The control model has slightly fewer gene-to-gene parameters with high Z-score (an average of 8.4 with $Z_w > 1.0$, over 5 randomized data sets; compared to 10 using the real data).

Although these results are somewhat disappointing, it should be kept in mind that a parameter with high Z-score in the control model does not mean that the model has "discovered" an imaginary genetic regulatory interaction, but simply that there is a high probability that the best fit linear model to this specific randomized data set contains a nonzero value for this parameter. In other words, if our assumption is that this data was generated by a linear network model, then this model must have some specific nonzero parameters. The fact that we have violated that assumption by generating the data in a randomized fashion does not invalidate the possibility that the best-fit parameters to *real* data may reflect biological reality. The real validation test for a model such as this invariably has to come from comparison with the literature (as in the previous Sections) or, ideally, verification or refutation of the predictions in the laboratory.

Chapter 7

Modeling gene networks with recurrent neural networks

As a net is made up of a series of ties,
so everything in this world is connected by a series of ties. If anyone thinks
that the mesh of a net is an independent, isolated thing, he is mistaken.
It is called a net because it is made up of a series of interconnected meshes,
and each mesh has its place and responsibility in relation to other meshes.
— *Buddha*

The performance of the linear model is quite promising, but the model has some quite unrealistic properties. In order to compensate for these, we add a nonlinear transfer function (similar to a dose-response curve), and an explicit mRNA decay term, resulting in a set of nonlinear differential equations similar to a recurrent neural network. I will briefly introduce some background on training neural networks, including such issues as the delta-bar-delta training rule, early stopping, weight decay, and weight elimination. Next, I demonstrate how we can randomly generate small "synthetic" networks with similar properties and dynamical behavior as found in the real data, and how to reconstruct the network by training a recurrent neural network on coarsely sampled, noisy time series data. Various measures of perfor-

mance are suggested, and various settings of the learning algorithm are explored to optmize the performance of the modeling methodology. Results seem promising, although possibly not quite as good as for the linear model. Finally, I briefly discuss how to extend the "robust parameters" analysis used in Chapter 6 to the nonlinear model, which should allow for a better distinction between the "true" and "false" weights.

7.1 Nonlinear differential equations

Although the linear model performs better than expected, it has some serious drawbacks. One issue is that a linear, additive system can only have a single attractor in state space. If we interpret attractors of genetic regulatory networks as cell types (as suggested by Kauffman [130]), this would mean only cells with a single stable cell type could be modeled. In fact, the virtual τ_l inputs in Equation 6.5 do allow for separate cell types. For example, eigenvector analysis of the linear model in Chapter 6 does show a separate attractor for the spinal cord data set (here of course corresponding to an tissue-wide average expression level, not a single cell type) and the hippocampus data set. Nevertheless, there can only be a single stable attractor for each "tissue type". The adult expression levels in the normal developmental hippocampus time series and the kainate injury time series actually lie on the same, one-dimensional attractor which intersects both adult expression patterns. In order to achieve different *point* attractors, it is essential to introduce nonlinearity in the model.

A second important drawback of the linear model is that its assumptions about gene regulation are very unrealistic. For example, nothing prevents expression levels in Equation 6.5 from becoming negative, or from increasing without bounds. Even though the best-fit model to the real expression data does not show this behavior,

it would be preferable to constrain the outputs of the model to conform to physical reality by imposing a lower and upper bound on concentration levels. The obvious solution is to switch to a model as in Equation 1.4:

$$\frac{dy_i}{dt} = A_i\, S\Big(\sum_j w_{ji}\, y_j + b_i\Big) - D_i\, y_i \tag{7.1}$$

Figure 7.1 shows a schematic representation of what a "node" in the nonlinear network would look like, for the specific case of the data sets used in Chapter 6—with an additional kainate input and tissue type indicator variable.

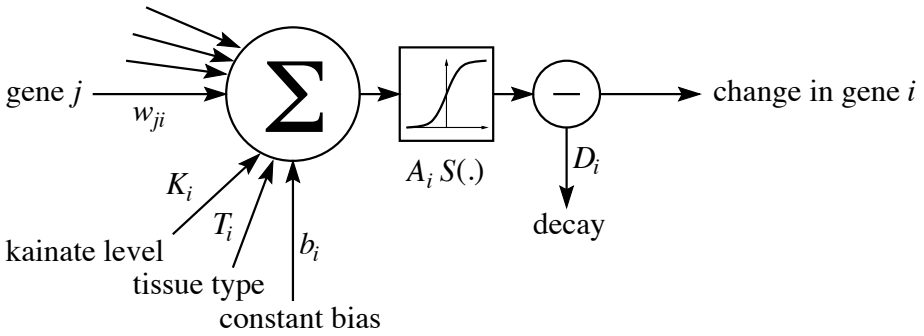

Figure 7.1: Schematic illustration of a node in the nonlinear (neural) network model. The input from all regulatory genes is summed up, together with an input from the kainate level, a constant bias term, and an additional term to cover tissue-specific differences in regulation. This weighted sum of inputs is passed through a sigmoidal transfer function $S(\cdot)$ and a proportional decay term is subtracted. The result determines the change (i.e., slope) in expression level of the corresponding gene.

The sigmoidal transfer function $S(\cdot)$ (bounded between 0 and 1) introduces the needed nonlinearity and imposes an upper bound on the production rate of the mRNA. Although the total input $\sum w_{ji}\, y_j + b_i$ to the gene may become negative (strong repression of the gene), since $S(\cdot)$ is lower-bounded to zero and the decay term $D_i\, y_i$ is proportional to the expression level, y_i can never decrease below zero. Note that this model does not include *regulated* decay (or protection from decay)

of gene products induced by other regulatory factors, only passive first-order decay. The decay term $D_i\, y_i$ imposes a maximum concentration level, at the point where production and decay balance each other out:

$$\max(y_i) = A_i/D_i \tag{7.2}$$

The set of nonlinear ordinary differential equations given in Equation 7.1 could be fitted to a data set using general optimization methods, essentially by doing some form of gradient descent on the parameters. For example, Mjolsness *et al.* [164] use Simulated Annealing to simultaneously find a set of network parameters and diffusion parameters to fit a spatial expression pattern. However, as we shall see in the next section, very efficient optimization techniques for precisely this sort of differential equation have already been developed within the field of neural networks.

7.2 Dynamic recurrent neural networks

Instead of using general optimization methods, we can view Equation 7.1 as defining a continuous-time, or *dynamic*, recurrent neural network. The advantage of this approach is that the neural network community has developed efficient algorithms to fit parameters of this specific type of network (see Pearlmutter [177] for a review). In addition, we can apply some of the techniques that have been developed to reduce the connectivity of the network, and easily incorporate additional biological knowledge by imposing Bayesian priors on properties of the network or individual weights.

Neural networks have a reputation for being a "black box" modeling technique: only useful for prediction, without allowing one to learn about the system being modeled by "opening the box". The main reason for this reputation is that neural networks are often applied to systems which are not expected to show any similarity

to the network structure used in the neural network. Here we have a one-to-one mapping between genes and their regulatory interactions, and network nodes and weights, so we expect the resulting network structure to show some similarity to the real regulatory network.

The black box approach here would be to connect the "gene" node outputs to a number of hidden layers, which then would feed back into the gene nodes. No doubt this would allow a closer fit to the data (at the expense of needing more data to avoid overfitting), but we would have less hope of being able to match the network of hidden nodes to biological reality. On the other hand, if we have measurements regarding intermediary nodes (e.g. protein concentrations), we may be able to add these nodes and preserve the one-to-one mapping. Also, if there is sufficient evidence for shared patterns of regulation among genes, we may add a hidden node to the network, in the hope the new hidden node corresponds to a real regulatory intermediate.

Albertini and Sontag [7] showed that for recurrent network models such as Equation 7.1, the input/output behavior uniquely determines all the weights. In other words, given enough data on the dynamical behavior of the system under different conditions, there is only a single such recurrent neural network that best matches the data. It has also been shown [205, 206] that feedforward networks of this type are as powerful as Turing Machines or analog computers.

7.3 Learning algorithms

As is illustrated in Chapter 2, real world data comes in a variety of types. Some data, such as the Gene Expression Matrix or the diauxic shift data, may consist of a time series, where each point reflect the state of the cells at a particular time in a dynamic process. These sets provide us with a lot of training data at a time, but each time point will be related to the surrounding time points, reducing the amount

of information present in the data. For these data sets, we need to train the neural network to follow the same trajectory when starting from the same initial state.

Other data, more conventionally, describes the steady state of the cells when presented with a certain environment, or with a given mutation, usually either a deletion or overexpression of a certain gene. For this type of data, we will need to train the network to exhibit a given stable attractor when presented with a set of external inputs (mutations can usually be represented using a virtual external input to the affected genes). Learning rules to learn attractors tend to be somewhat simpler, because they do only need to learn a fixed point, and not the exact dynamics.

In order to take advantage of all the data present on an organism, we would want to use both types of data to train the same network. However, these data sets do require different learning algorithms. It should be possible to combine both types of data, and different data sets of each type, into a single error term which can be used to train the network. Or we can alternate training using the different learning rules, just as we alternate training on different training data within the same set. However, it may be necessary to weight the different data sets by how much information they really contain, otherwise the single point steady state data may be practically ignored compared to the much larger time series data sets. Integrating different sets of data and different learning rules will likely be an important factor in making this sort of model feasible. Learning algorithms for dynamic recurrent neural networks are reviewed in Pearlmutter [177], and the two most relevant ones (Backpropagation through Time and Recurrent Backpropagation) are derived for the specific form of Equation 7.1 in Appendix D.

Backpropagation Through Time (BPTT), developed by Werbos [247, 248], allows a recurrent neural network to be trained to follow a specified trajectory through state space. Any recurrent network, simulated over n discrete time steps can be represented by an n-layered, purely feedforward network, by "unrolling" the recurrent net (See

Figure 7.2): one simply replicates all the nodes n times, and updates their activation levels one layer per time step. This has been the basis for most learning algorithms for recurrent neural networks, because now any learning algorithm for feedforward nets can be applied. BPTT uses the standard backpropagation algorithm [246, 191] to train recurrent neural networks this way.

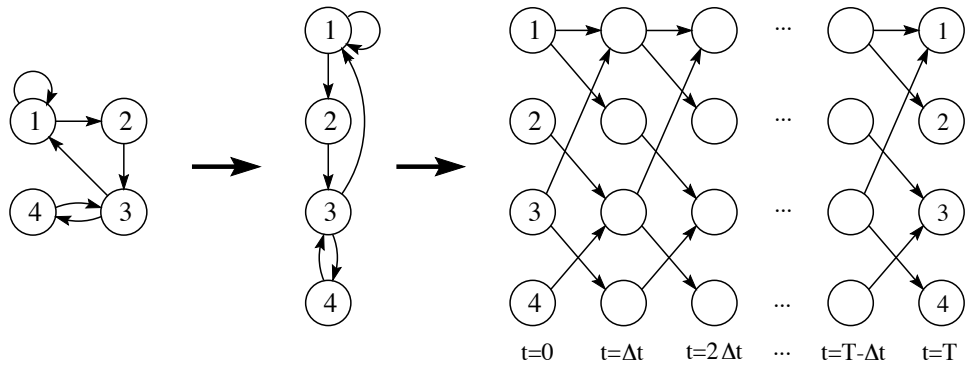

Figure 7.2: Example of a simple, 4-node recurrent network, unrolled in time from $t = 0$ to $t = T$ with a time step Δt.

Pearlmutter [176] has derived a continuous-time version of BPTT, applicable to Equation 7.1. Simulating the network using the differential equation form, rather than with discrete time steps, has a number of advantages. First of all, it is unclear what time step we should use so the dynamics of the differential equation system do not get lost in the simulation. Secondly, if the training data is sampled at non-uniform intervals, the number of time steps to unroll the recurrent network may be different for each training sample. Lastly, by keeping in mind that we're really dealing with a set of differential equations, we may be able to use some special simulation techniques that are faster than simply running a set of difference equations.

Appendix D.2 derives the continuous-time BPTT learning rule for Equation 7.1. The computational complexity of the algorithm is $\mathcal{O}(QTP)$, where Q is the number of iterations needed to train the network, T is the number of discretized time points in the entire time series, and P is the number of weights (in our case, $P = \mathcal{O}(N^2)$;

if we started out with a sparse network, we would have $P = \mathcal{O}(NK)$). All the runs below were done with a modified version of Pearlmutter's CBP software [175] implementation of BPTT.

The idea behind training a neural network is to find the derivative of an error term E with respect to the individual weights w_k (i.e. the w_{ij}, A_i and D_i in Equation 7.1) of the network. These derivatives can then be used to do gradient descent on the weights, updating them in the direction that minimizes E:

$$\Delta w_k = -\eta \frac{\partial E}{\partial w_k} \tag{7.3}$$

where η is a learning rate indicating what size steps should be taken in parameter space (see also Section 7.3.1). The error term E in general can take the form of an weighted integral over time. In our case, since we only have a sparsely sampled time series to fit, the error term will simplify to something like:

$$E = \sum_{t=t_1}^{t_n} \sum_i (y_i(t) - \hat{y}_i(t))^2 \tag{7.4}$$

where $\hat{y}_i(t)$ is the measured expression level of gene i at time point t. More elaborate error terms are easily added. For example, we could add an integral of the magnitude of the second derivative to impose a smoothness constraint (although, as we will see later, there is a more elegant way to do this). Or we could add a term indicating how poorly the model fits certain biological expectations. The equations derived in Appendix D.2 only require that we can easily differentiate the total error E with respect to the gene expression levels y_i.

7.3.1 Adjusting learning rate using delta-bar-delta

The choice of learning rate η in Equation 7.3 is crucial in the performance of the learning algorithm. If the learning rate is too large, the weight update may overshoot

it's optimum value, causing the weight vector to jump back and forth in parameter space. If the learning rate is too small, a large number of iterations will be needed to reach the optimum. This problem is exacerbated by the fact that different regions of parameters space often have different degrees of "ruggedness", resulting in a different optimal learning rate at different times during training. Similarly, different weights may have their own optimal learning rate.

The *delta-bar-delta* algorithm [123] uses a simple heuristic to get around these problems. Essentially, it maintains a separate learning rate η_k for each weight w_k, and automatically adjusts these during training. When the current gradient ("delta") for the weight points in the same direction as the time-averaged gradient ("delta-bar") during previous iterations, η_k is increased (up to a pre-defined maximum). Conversely, when the current gradient points in the opposite direction (i.e. we just overshot the optimum), η_k is decreased. More formally:

$$\delta_k(t) = \frac{\partial E}{\partial w_k}(t) \tag{7.5}$$

$$\bar{\delta}_k(t) = \theta\bar{\delta}_k(t-1) + (1-\theta)\delta_k(t) \tag{7.6}$$

where the time index t indicates iteration number, and $\bar{\delta}_k(t)$ is an exponentially weighted average (with $0 < \theta < 1$) of the gradient δ_k at previous iterations. Note that the sign of $\bar{\delta}_k\delta_k$ will indicate whether the gradient δ_k points in the same direction as before (hence, "delta-bar-delta"). The learning rate η_k is then updated using:

$$\Delta\eta_k(t) = \begin{cases} \epsilon_1 & \text{if } \bar{\delta}_k(t-1)\delta_k(t) > 0 \\ -\epsilon_2\eta_k(t) & \text{if } \bar{\delta}_k(t-1)\delta_k(t) < 0 \end{cases} \tag{7.7}$$

where ϵ_1 and ϵ_1 are some small constants, determining the speed at which the learning rate can vary.

7.3.2 ML interpretation of minimizing E

Note that the standard quadratic error term in Equation 7.4 is equivalent to finding a Maximum Likelihood fit of the model to the data, under the assumption of Gaussian noise on the measurements. The likelihood of the model given the data is defined as:

$$P(\hat{\mathbf{y}}|\mathbf{w}) = \prod_{i,t} P\left(\hat{y}_i(t)|\mathbf{w}\right) \tag{7.8}$$

Maximizing the likelihood thus finds the model which would be most likely to produce the observed data $\hat{\mathbf{y}}$. Note that, via Bayes' rule (see Equation 7.18), this is equivalent to maximizing $P(\mathbf{w}|\hat{\mathbf{y}})$, if we have no prior assumptions on the distribution of the observed data $P(\hat{\mathbf{y}})$ and the weights of the model $P(\mathbf{w})$. If we assume the measured outputs are given by the values $y_i(t)$ produced by the model, plus a zero-centered Gaussian noise term:

$$\begin{aligned} P\left(\hat{y}_i(t)|\mathbf{w}\right) &= \mathcal{N}\left(\hat{y}_i(t), \mu = y_i(t), \sigma = s\right) && (7.9) \\ &= \frac{1}{\sqrt{2\pi}s} e^{-\frac{(y_i(t)-\hat{y}_i(t))^2}{2\sigma^2}} && (7.10) \end{aligned}$$

then the negative log-likelihood becomes:

$$\begin{aligned} -\log(P\left(\hat{\mathbf{y}}|\mathbf{w}\right)) &= -\sum_{i,t} \log \mathcal{N}\left(\hat{y}_i(t), \mu = y_i(t), \sigma = s\right) && (7.11) \\ &\propto \sum_{i,t} (y_i(t) - \hat{y}_i(t))^2 && (7.12) \end{aligned}$$

and thus minimizing a quadratic error terms as in Equation 7.4 is equivalent to minimizing the negative log-likelihood, which is equivalent to maximizing the likelihood.

7.4 The Curse of Dimensionality revisited

As mentioned earlier, high-dimensional systems are hard to model, especially if comparatively few training samples are available. The problem is generally stated as one of *generalization*, i.e. being able to produce correct outputs on new inputs. Note that our goal here is slightly different: we want to predict the structure of the network, not its behavior under new conditions. Of course, the goals are identical in the limit, as sufficient data on the dynamical behavior will uniquely determine the network parameters. [193] mentions three conditions that are typically necessary for good generalization:

1. The inputs to the network contain sufficient information pertaining to the target. This validates our desire to measure as many variables of the system as possible. If we overlook an important input to the system, obviously the model will not be able to learn the influences that input exerts.

2. The function we are trying to learn should be, in some sense, smooth. We can expect this to be true for most biological systems, because they are generally stable with respect to small perturbations.

3. The training set should be a sufficiently large and representative subset ("sample" in statistical terminology) of the set of all cases that we want to generalize to (the "population" in statistical terminology). This is our main problem!

The data sets used in Chapter 6 are on the order of 65 genes $\times$ 28 time steps [245, 212]. For yeast, publicly available data is on the order of 6400 genes $\times$ 400 experiments [217, 118]. Rules of thumb for training neural networks require that a few times as many training cases as parameters are available to avoid any loss of generalization due to overfitting. The use of good regularizers and early stopping can

prevent overfitting with fewer training samples, but clearly we are nowhere near the required amount of data at the moment.

Considering we do not (yet?) have sufficient large-scale data to achieve a good model solely based on this data, it is important to include as much extra knowledge as possible (see Section 8.2.3), but also that we try to minimize the size of the model (number of parameters). As conjectured in Table 5.1 (and as proven in the Boolean case), if we can reduce the number of inputs per gene, we may be able to derive a reasonable network from a number of data points that only grows with the *logarithm* of the number of genes.

7.4.1 Simplifying the model

Weight Decay

Weight decay is a simple technique to reduce unneeded connections in the network by decaying their weights to zero. Intuitively, the hope is that, by putting a constant downwards pressure on the weights, those that are unimportant for fitting the data will decay down to zero, whereas those that are important will be maintained at a higher level. Weight decay adds a penalty term equal to the sum of the squared weights to the error function:

$$E' = E + \alpha \sum_k w_k^2 \tag{7.13}$$

where α is the weight decay rate. The effect on the weight update rule is an extra negative term proportional to the size of the weight:

$$\Delta w_k = -\eta \frac{\partial E'}{\partial w_k}$$

$$= -\eta\frac{\partial E}{\partial w_k} - \eta\alpha w_k \tag{7.14}$$

Interestingly, weight decay is equivalent to biasing the network with respect to a Bayesian prior on the weights, where the prior distribution of weights is a zero-mean Gaussian:

$$P(w_k) = \mathcal{N}(w_k, \mu = 0, \sigma = s) \tag{7.15}$$

$$= \frac{1}{\sqrt{2\pi}s}e^{-\frac{w_k^2}{2\sigma^2}} \tag{7.16}$$

then

$$P(\mathbf{w}) = \prod_k \mathcal{N}(w_k, \mu = 0, \sigma = s) \tag{7.17}$$

Whereas Maximum Likelihood finds the model that maximizes $P(\hat{\mathbf{y}}|\mathbf{w})$, Bayesian inference maximizes the more correct term $P(\mathbf{w}|\hat{\mathbf{y}})$, i.e rather than finding the model for which the observed data is most likely, it finds the model which is most likely given the data (the maximum *a posteriori* or MAP model). The two are related by Bayes' rule:

$$P(\mathbf{w}|\hat{\mathbf{y}}) = \frac{P(\hat{\mathbf{y}}|\mathbf{w})P(\mathbf{w})}{P(\hat{\mathbf{y}})} \tag{7.18}$$

If we assume all observed data sets $\hat{\mathbf{y}}$ are equiprobable, we can ignore the term in the denominator. To find the maximum, we again take the negative log of these probabilities, and with Equations 7.10 and 7.17 we get:

$$-\log\left(P(\mathbf{w}|\hat{\mathbf{y}})\right) \propto \beta\sum_{i,t}(y_i(t)-\hat{y}_i(t))^2+\alpha\sum_k w_k^2 \tag{7.19}$$

$$\propto \beta E_D+\alpha E_W \tag{7.20}$$

where E_D is the normal quadratic error term indicating the fit between the model outputs and the measured data, E_W is a quadratic (weight decay) error term indicating the fit between the model parameters and their expected distribution, and α and β are rate parameters indicating the balance between these two error terms. Minimizing a quadratic error term plus weight decay is thus equivalent to finding the MAP model under the assumption of Gaussian measurement noise and Gaussian distributed weights.

Weight decay (also known as *ridge regression* in statistics) can also be interpreted as a classic example of a *regularizer*, an additional term added to the optimization to make the system "better behaved" (e.g., smoother[1] or better conditioned). In other words, adding the additional constraint of minimizing the sum of weights can reduce an underdetermined model—with an infinite number of possible solutions—to a well-determined one—corresponding to the single solution that optimizes both E_D and E_W. It has been shown that a weight decay regularizer can result in significant improvements in generalization [108, 137].

Weight Elimination

Other penalty terms besides the sum of squared weights can be used. Whereas weight decay tends to shrink the large weights more than the small ones (often preferring many small weights over a single large one), weight elimination [242] tends to work

[1] In fact, the additional smoothness constraint imposed by the interpolation in Chapter 6 can likewise be interpreted as a regularizer.

mainly on the small weights. This makes it more suitable to eliminate unneeded connections.

$$E' = E + \gamma \sum_k \frac{w_k^2}{w_0^2 + w_k^2} \tag{7.21}$$

where γ is the weight elimination rate, and w_0 is a weight threshold. For small weights $w_k \ll w_0$, the weight elimination error term is quadratic, as in weight decay. For large weights $w_k \gg w_0$, the weight elimination error term is approximately 1.[2] As for weight decay, we can think of weight elimination as imposing a Bayesian prior on the distribution of weights. As shown in [242], weight elimination corresponds to a prior probability distribution over the weights which resembles a narrow Gaussian around zero (indicating unneeded connections), superimposed on a much wider, uniform distribution for the relevant weights. This is exactly the sort of distribution we would expect for a noisy reconstruction of a sparsely connected network. Therefore, we would expect weight elimination to perform better than weight decay for this particular application.

[242] suggests choosing w_0 of order one for activation levels of order one, and adaptively changing γ based on the error E. Alternatively, if we want to train towards a network with a desired number of parameters (e.g. on the order of the amount of data available) we may want to update w_0 such that the specified fraction of weights falls below w_0.

Pruning algorithms

Weight decay and weight elimination do not actually reduce the total number of connections. They simply allow certain weights to evolve towards zero, hopefully

[2] One could therefore think of the weight elimination error term as *counting* the number of large weights.

indicating that the associated connections are superfluous. *Pruning* algorithms, on the other hand, directly remove connections. The two best known ones are Optimal Brain Damage [60], and Optimal Brain Surgeon [101]. Both compute a *saliency* measure for weights in the network, as an estimate of how much the training error will increase if the weight were to be removed. Connections with the smallest saliency can then be deleted, after which the remaining weights are updated through retraining. A variant of Optimal Brain Surgeon has been used to prune recurrent networks [178].

7.4.2 Early stopping

During training of a neural network on limited data, the network typically progresses from being underfit, to adequately fit, to overfit. The overfit network may fit the training data significantly better, but only at the expense of generalization, i.e. it will perform significantly worse on new data. If we could stop training before overfitting occurs, we would get a network which generalizes better to conditions it has not been trained on, and therefore—hopefully—has a network structure which is a better approximation of the regulatory network.

Early stopping is usually performed by splitting the available data in a training and validation set, training only on the former, and stopping when the validation error starts to increase (or training to convergence on the training data, and picking the weights for which the validation error was minimal). The obvious drawback is that splitting the data reduces the amount of data available for training even further. In practice, this problem can be overcome—at the cost of additional computation—using, e.g., cross-validation: repeatedly leaving out a small subset of the data and training on the rest, averaging the generalization error over the left-out subsets. Here, since we will be dealing primarily with synthetic data, we can simply assess the performance based on comparison with the known goal.

7.5 Experiments with synthetic data sets

In the linear model, we could derive the best fit model in a single step using a little linear algebra. For the nonlinear model, we need to iteratively update the network parameters, where each iteration of the neural network training algorithm involves integrating differential equations forwards and back across the entire time span. We can expect this to take several magnitudes more computation time (hours, versus seconds for the linear model). In order to speed up experimentation, we start out examining some small synthetic data sets, generated from a known, randomly generated network. This also allows us to compare the performance using different variants (e.g. weight decay versus weight elimination), because the "target" network is known.

7.5.1 Generating synthetic data sets

For now, I have chosen to use networks with $N = 20$ nodes ("genes") and $K \approx 4$ inputs per node. Since the computational complexity grows linear with the number of weights (and thus quadratic with N), these smaller networks should train about ten times as fast per time step and per iteration, compared to a network with $N = 65$. Also, in order to focus on learning the connection weights of the network, I have assumed unit production and decay rates ($A_i = D_i = 1$).

The goal is to generate some small artificial networks with similar properties and dynamical behavior as found in the real data sets, sample this dynamical behavior at a similar time resolution as found in the real data, add a similar level of noise to the measurements, and then try to reconstruct the original network from this limited amount of coarsely sampled, noisy data. The following is the sequence of steps to be taken to generate the synthetic networks and data:

Generate a sparse, 21×20 connection matrix (the extra input dimension is for a constant bias term). Each weight has a 80% chance of being set to zero, 20% chance $K \approx 4$ of being drawn from a standard normal distribution. Such random networks do not tend to show very interesting dynamics: most nodes very quickly saturate at 0 or 1. In what follows we will call those weights that are nonzero in the synthetic network "true weights", and those that are zero "false weights". Note that, because of the Gaussian distribution of the nonzero weights, many of them will actually be close to zero as well. Realistically, we can only hope to reconstruct those nonzero weights which are within the tails of the distribution.

Reduce the sum of inputs to each node. As we saw in Chapter 6 (Section 6.4.1), the results for the linear model show that sums of input weights (i.e. $\sum_j w_{ji}$) are about three times smaller than expected, based on the distribution of weights. I.e., positive and negative regulation seems carefully balanced to keep the genes within their active range. We will likewise attempt to reduce the standard deviation of input sums to one third the expected level, by iteratively re-randomizing the inputs to nodes with large input sums. (Because re-randomizing only the largest input sum at each iteration would result in a truncated distribution, we actually add a small amount of Gaussian noise[3] to the input sums before ranking them. This way, each node has some weighted probability of being re-randomized, and the resulting distribution of input sums is approximately Gaussian.) Re-randomizing approximately preserves the distribution of weights, although the networks tend to have slightly fewer non-zero weights (typically around 70, rather than the expected 84). Networks generated using this approach tend to have fixed points in which many nodes settle to intermediate values, but the fixed point is still reached fairly quickly, with little interesting dynamics.

[3]With standard deviation equal to the standard deviation of input sums we are aiming for

Scale the network weights to show interesting dynamics. The above explanation assumed a standard normal distribution for the non-zero weights. If we scale the weights up by a constant factor, more interesting dynamics emerge. When the weights become too large, the nodes again start saturating at 0/1, or oscillations tend to set in. Some experimentation showed that multiplying the weights by 5 (i.e. the weights are drawn from a $\mathcal{N}(\mu = 0, \sigma = 5)$ Gaussian distribution) yields a dynamical behavior which subjectively resembles the real data sets.

Generate and sample time series. The networks were started at a random initial state, and Equation 7.1 was simulated for 500 time steps at a time resolution of $\Delta t = 0.1$ (typically sufficient to reach the fixed point). In the real CNS data sets, the average change per gene between time points is approximately 16% of its maximal expression level (i.e. an average change of 0.16 if all genes are normalized to have a maximum of 1.0). The synthetic trajectories were then sub-sampled at a similar rate, resulting in an average of 5-6 time points per time series before reaching the fixed point.

Add noise similar to the real data. In order to get a fair comparison, we should also assess the effect of noise in the input data on our ability to reconstruct the network. We can use the triplicate standard deviations in the real CNS data sets to derive a simple noise model, and then add similar amounts of noise to our sampled synthetic data. As Figure 7.3(a) shows, the amount of noise is somewhat correlated with the absolute expression level, but little further structure can be identified because of the large spread in the raw expression levels. When we look at normalized data (Figure 7.3(b)), the picture becomes much clearer: the noise level is clearly correlated with the normalized expression level. The regression line drawn in the figure is $s = 0.0112 + 0.0832y$, with y the normalized expression level and s the corresponding (normalized) triplicate standard deviation. This is the model used

to add noise to the synthetic data. The increasing spread in the observed triplicate standard deviation can largely be explained by the fact that these standard deviations are very rough estimates, based on a small sample. The spread of triplicate standard deviations in the real data is only slightly larger than expected based on this simple error model (i.e., the real data has slightly more measurements with higher and lower than expected triplicate standard deviation).

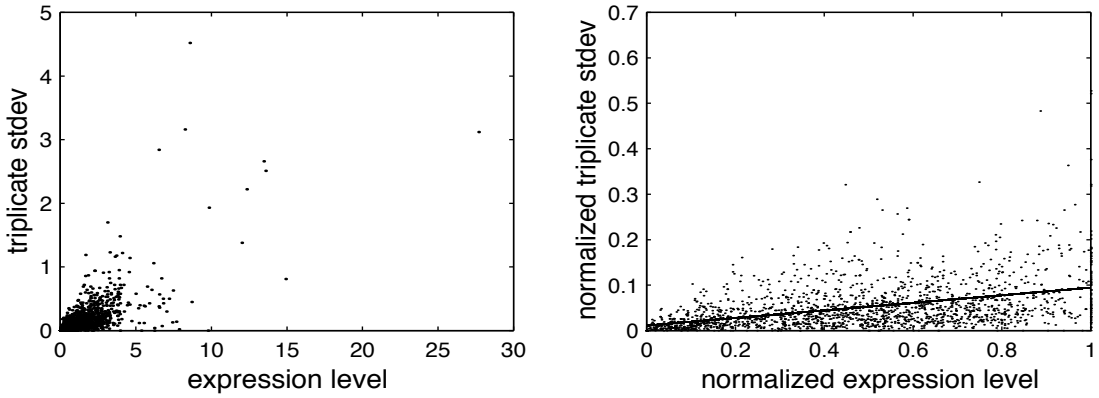

Figure 7.3: A simple noise model for the CNS data sets. (a) Raw expression levels and standard deviation over the triplicate experiments vary over a wide range. Some correlation can be observed, but it obscured by the uneven distribution of the data. (b) After normalizing expression levels to a maximum of 1.0, the correlation is much clearer, although the distribution is still heteroscedastic (i.e., the spread increases with expression level).

7.5.2 Evaluating the networks

To compare different network models we need to have some measure of their performance. Classically, one would look at the error term used to train the network to rank them with respect to how well they match the data (ideally using a separate test set to evaluate the generalization error. However, here the primary goal is not to match the synthetic data, but to match the structure of the network that generated

this data in the first place. In other words, given the weight matrix resulting from training the neural network on the synthetic data, how "close" is this weight matrix to the one from the original network. As we shall see, a number of performance measures can be used, none of which stand out as being significantly better than the rest.

Figure 7.4 shows two examples of trained network weights plotted against original network weights. Clearly visible is the approximately 80% zero weights in the original network, which are assigned nonzero values in the trained network, and thus are smeared out on the vertical axis. Also visible in Figure 7.4(b) is the effect of weight elimination, which likewise clusters a number of weights near the horizontal axis. The true weights show significant correlation with their values in the trained network, and several true weights with large values in the trained network can easily be identified.

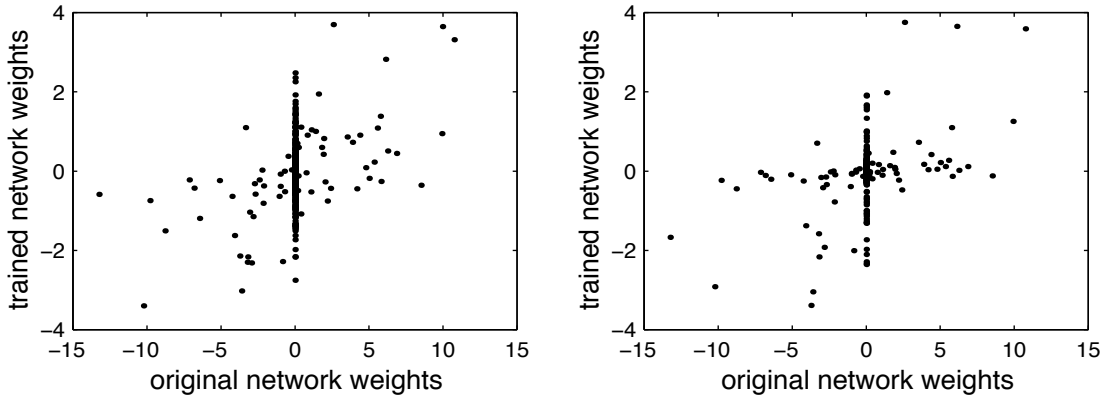

Figure 7.4: Original network weights versus trained network weights after 2000 iterations. (a) Without using weight decay or weight elimination. (b) With weight decay rate $\alpha = 0.00005$ and weight elimination rate $\gamma = 0.0005$.

Performance based on zero-nonzero classification

The first "bit" of information we are interested in, is which genes regulate which other genes. In other words, can we distinguish the *true* weights (nonzero in original network) from the *false* weights (zero in original network). We will try to divide the weights of the trained network based on their size into "nonzero" and "zero" (or "significant" and "insignificant") classes, and then assess how many errors we have made doing so.

As mentioned earlier for the linear model (Section 6.4.1), one simple way to classify the weights is by fitting the weight distribution by a mixture of two Gaussians, and identify the ones in the wider Gaussian as nonzero. F_{mix} is the fraction of true weights among the weights identified as nonzero in this way. Note that the use of weight elimination should make this mixture model more plausible, as it tries to enforce such a mixture of Gaussians on the weight distribution.

Alternatively, if we have an estimate of the number NK of true nonzero weights, we could simply take the top NK largest weights in the trained network and classify these as nonzero. F_{top} is the fraction classified correctly using this approach.

Rather than taking the top n weights and calculating what fraction F are true weights, we can reverse the question: What number n of the largest weights should we look at to have no more than a fraction $P(= 1 - F)$ false positives (i.e. false weights identified as nonzero in the trained network). In particular, we will look for the value of n at which we see no more than $P = 0.25$ and $P = 0.5$ false weights (N_{25} and N_{50}).

In practice, as we will see in Section 7.5.3, nonzero classification based on a mixture of Gaussians typically results in the largest nonzero class, and thus poorest performance. Picking the top NK largest weights typically gives better performance, and N_{50} and N_{25} are progressively more restrictive in how many weights they accept

as being significant.

Correlation-based performance measures

If we also want to assess the accuracy of the weights values given by the trained network, we could examine, for example, the sum of squared errors between the original network weights and the trained network weights. This is related to the Pearson (linear) correlation between the two distributions. We will call this measure R_{pa} ("*p*" for Pearson, "*a*" for *all* weights).

Pearson correlation assumes Gaussian distributions, and we know that at least the weights in the original network are strongly non-Gaussian (in fact, the distribution consists of about 80% zeros, plus 20% Gaussian distributed weights). Similarly, when using weight elimination, the resulting weight distribution of the trained network will be strongly non-Gaussian. When dealing with non-Gaussian distributions, rank correlation measures are preferred over linear correlation. R_{sa} measures the Spearman rank correlation over all weights.

Finally, we can identify two more-or-less independent components that determine the performance of a given network. Firstly, do the true weights show significantly greater magnitudes than the false weights in the trained network? Secondly, are the weights derived for those true weights correlated with their real values? M_{tf} is a measure for this first component, consisting of the ratio of the standard deviation of the true weights over the standard deviation of the false weights (i.e. the standard deviations of their values in the trained network). R_{st} is a measure for the second component, consisting of the Spearman rank correlation over the true weights only.

7.5.3 Results

Five random networks were generated as described above, and three time series were collected with a total of 18 time points (networks generating fewer or more than 18 time points in the first three time series sampled were rejected, to keep the results comparable). Four of the networks had three time series with six points each, one network had one series each of 5, 6 and 7 time points. To more clearly show the trends in performance with various parameter settings, for each network, all the runs were performed with the same random number. Also, the performance measures given below are the average over all five network, unless otherwise noted.

Early stopping: We will first examine the effect of early stopping on the performance measures defined in the previous section, using a mixture of weight elimination and weight decay. Figure 7.5 clearly shows that measures reach a peak (or plateau) at 1000-2000 iterations. For the remaining experiments, we will therefore only show results at 2000 iterations.

Weight decay: Next, we can examine the effect of weight decay on the performance. The weight decay parameter α was sampled logarithmically from $\alpha = 0.00001$ to $\alpha = 0.001$. The results are shown in Figure 7.6. The effect of weight decay is rather minimal, and is only evident in the increase in number of weights we can classify at P=0.25 and P=0.5 (N_{25}, N_{50}). Although not obvious from these graphs, weight decay does have a marked effect on the magnitude of all weights, reducing them overall by about 50% for $\alpha = 0.001$ compared to $\alpha = 0$.

Weight elimination: The weight elimination parameters γ was sampled logarithmically from $\gamma = 0.0001$ to $\gamma = 0.002$. The results are shown in Figure 7.7. Weight elimination performs markedly better than weight decay, improving most

of our performance measures, presumably because it provides a better prior model for the weight distribution of the original network. The threshold weight w_0 (see Equation 7.21 was scaled automatically to match a mixture of Gaussians with 20% of weights in the larger Gaussian. Note that because of this, F_{mix}—the performance using a mixture of Gaussians to identify the "nonzero" weights—approaches F_{top}—the performance when classifying only the NK largest weights as nonzero.

Combination of weight decay and weight elimination: Clearly, weight elimination seems to outperform weight decay. However, in practice we may actually want to use a mixture of both: weight elimination puts almost no constraints on the size of weights much larger than w_0, so we may want to constrain these with a small amount of weight decay. Preliminary experiments indicated that the optimal performance is found roughly in the neighborhood of $\alpha = 0.00005, \gamma = 0.0005$. We therefore scanned α between 0.000005 and 0.001 for $\gamma = 0.0005$, and scanned γ between 0.0001 and 0.002 for $\alpha = 0.00005$. The results are shown in Figures 7.8 and 7.9. Again, the use of weight elimination leads to clear improvements in almost all performance measures, whereas weight decay now seems almost entirely superfluous.

7.5.4 The next step: Calculating Z-scores for the weights

So far, the analysis of the results derived using the neural network approach consisted only of identifying "true" weights based on the size of the weights in the trained network. The results are promising, but still far from exciting, typically allowing us to identify the ten or so highest weights as true weights with a significance of only P=0.25 (i.e. $N_{25} \approx 10$). The assumption that the magnitude of a weight reflects its importance is known to perform rather badly in practice in neural networks (see, e.g., Bishop [32], p.360). In fact, if we had followed the same strategy for the linear model, we would have achieved rather poor results as well. If we look back at Figure 6.5,

we observe that most of the largest weights in the linear model turn out to have large σ_w, i.e. they are poorly determined by the data. There, the solution was to explicitly take σ_w into account when trying to decide which weights were real. In order to improve on our results of the previous section, we should be able to do the same for the nonlinear model.

We can derive the standard deviations (and, in fact, the covariances) of the weights in the network by examining the *Hessian*, i.e. the matrix of second derivatives of the error E with respect to the weights:

$$H_{k,l} = \frac{\partial^2 E}{\partial w_k \partial w_l} \tag{7.22}$$

(Note that the Hessian is of the size number of weights $\times$ number of weights, and thus scales with the fourth power of the number of nodes in the network.)

Earlier (Section 7.4.1) we showed that training the network to convergence is equivalent to finding the weight matrix $\mathbf{w_{MP}}$ that maximizes the posterior probability $P(\mathbf{w}|\hat{\mathbf{y}})$ of the weights given the measurement data $\hat{\mathbf{y}}$ (see Equation 7.18). By Taylor-expanding the log posterior probability around this optimum $\mathbf{w_{MP}}$ with $\Delta\mathbf{w} = \mathbf{w} - \mathbf{w_{MP}}$, it can be shown (see MacKay [149]) that:

$$P(\mathbf{w}|\hat{\mathbf{y}}) \approx P(\mathbf{w_{MP}}|\hat{\mathbf{y}}) \exp\left(-\frac{1}{2}\Delta\mathbf{w}^{\mathrm{T}}\mathbf{H}\Delta\mathbf{w}\right) \tag{7.23}$$

In other words, the posterior probability of the weights can be approximated near its optimum as a Gaussian with covariance matrix $\mathbf{H}^{-1}$, the inverse of the Hessian evaluated at $\mathbf{w_{MP}}$. The desired standard deviations of the weights are then given by the square roots of the diagonal elements of the inverse of the Hessian. The Z-score of a weights would then become:

$$Z_{w_k} = \frac{|w_k|}{\sqrt{[\mathbf{H}^{-1}]_{kk}}} \tag{7.24}$$

Note that this is directly related to the *saliency* of a weight, as derived by Hassibi and Stork within the framework of the "Optimal Brain Surgeon" (OBS) pruning technique [101]:

$$L_{w_k} = \frac{1}{2}\frac{w_k^2}{[\mathbf{H}^{-1}]_{kk}} = \frac{1}{2}Z_{w_k}^2 \tag{7.25}$$

They arrived at this formula from an entirely different direction: the saliency of a weight is a measure for how much the error E would change if we were to prune away the corresponding weight w_k, and optimally adjust the other weights under the assumption of a locally quadratic error surface. Whereas OBS is primarily interested in the weights with smallest saliency, and iteratively removes these to prune the network, we would like to find the weights with *highest* saliency, because these are the most robust ones.

Various approaches have been developed to calculate the Hessian (see Bishop [32] for an overview). A simple approximation can be derived by observing that the backpropagation algorithm already calculates the first derivatives of the error term. We perturb each weight w_l in turn by a small amount ϵ and observe how the first derivative $\partial E/\partial w_k$ changes. To reduce residual errors to $\mathcal{O}(\epsilon^2)$, we use a symmetrical central difference formulation:

$$\frac{\partial^2 E}{\partial w_k \partial w_l} = \frac{1}{2\epsilon}\left(\frac{\partial E}{\partial w_k}(w_l + \epsilon) - \frac{\partial E}{\partial w_k}(w_l - \epsilon)\right) + \mathcal{O}(\epsilon^2) \tag{7.26}$$

Unfortunately, the computed Hessian turns out to be very poorly conditioned, with condition number of the order of 10^{-9}. Although it is not numerically impossible to derive the inverse of the Hessian, and thus the variances of the weights, the results are expected to be highly inaccurate. Remember that in the linear model we also had to deal with a poorly conditioned system (Section 6.3), but there the condition number was not quite as bad (around 10^{-4}) and we were primarily interested in the

larger entries (i.e. larger network parameters) of the result. Here, we are interested in the *smallest* entries (on the diagonal), which are likely to contain a larger relative error, and even worse: the *inverse* $1/[\mathbf{H}^{-1}]_{kk}$ of these smallest entries. Not surprisingly, the Z-scores calculated in this fashion showed very little relationship with the true weights of the network.

Further experimentation along these lines is continuing. One of the factors exacerbate the conditioning of the Hessian is the fact that the input variables are not zero-mean, introducing a large constant component in all the weights leading to a given node, giving the Hessian a block-diagonal appearance and possibly drowning out some of the smaller true second derivatives. We can eliminate this block diagonal by subtracting the mean from all the inputs and training the network using this data. In principle, this should only affect the bias weights b_i, which now have to compensate for the lack of a constant component in the other inputs. It may also be possible to derive a better approximation for the inverse of the Hessian directly (rather than estimating the Hessian first and inverting) using the technique used by Hassibi and Stork for OBS [101]. Finally, as a last resort we could take the same approach as we did for the linear model: generate new input data sets with small amounts of noise added, and directly observe the variance in the weights.

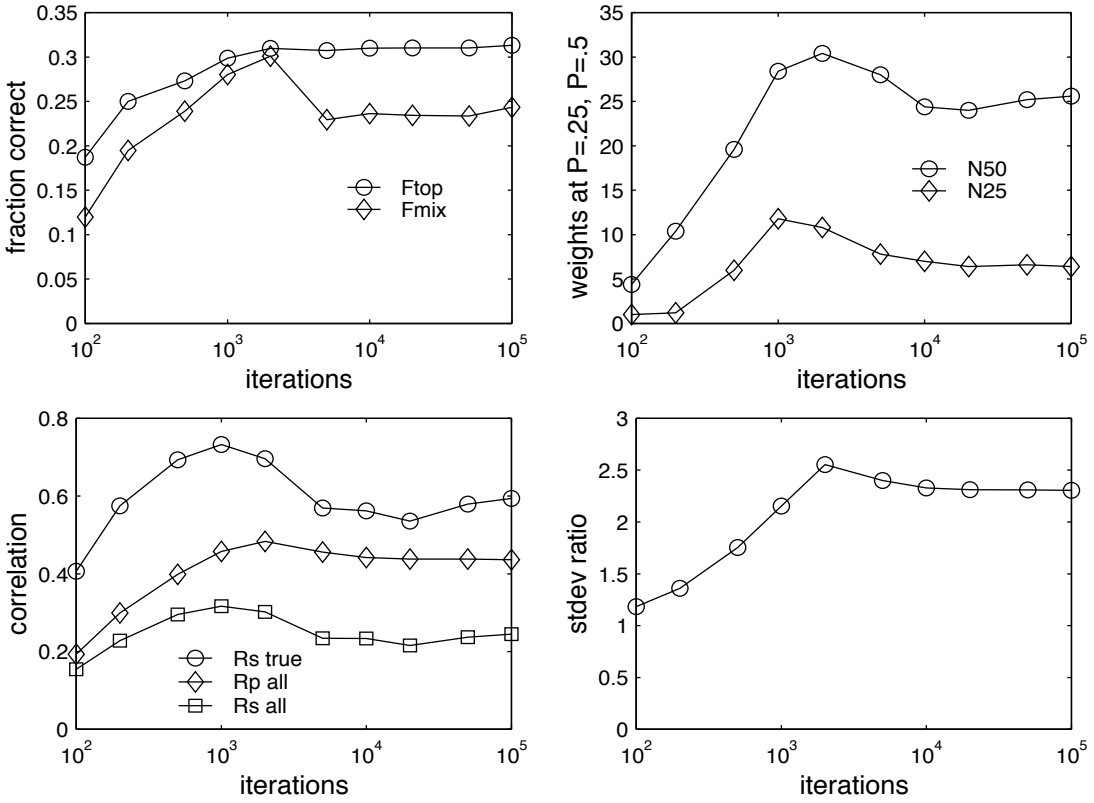

Figure 7.5: Effect of early stopping on performance, for number of iterations from 100 to 100,000 (note the logarithmic scale). Top left: fraction of true positives when fitting weight distribution using a mixture of Gaussians (F_{mix}) and fraction of true positives within the top KN largest weights (F_{top}). Top right: number of weights at P=0.25 and P=0.50 (N_{25}, N_{50}). Bottom left: Spearman rank correlation of true weights (R_{st}), of all weights (R_{sa}); and linear Pearson correlation of all weights (R_{pa}). Bottom right: ratio of standard deviation of true weights versus all weights.

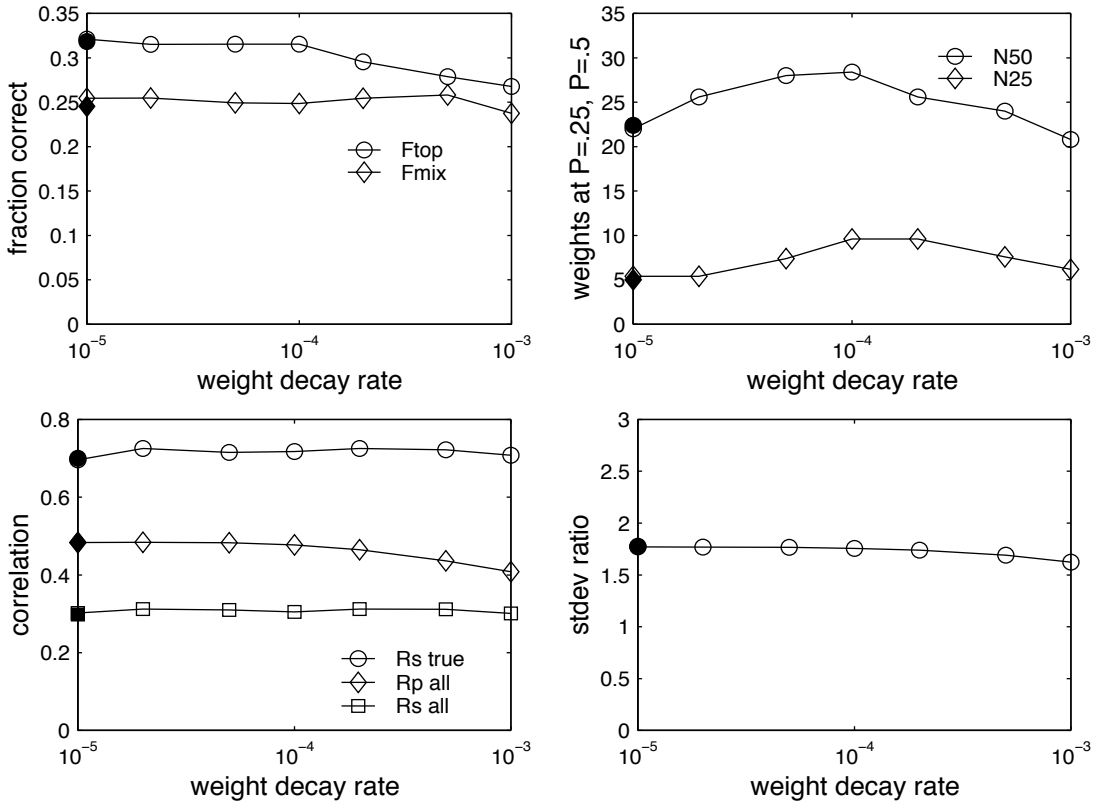

Figure 7.6: Effect of scaling weight decay rate from $\alpha = 0.00001$ to 0.001. The solid markers on the vertical axis are the performance for $\alpha = 0$ (note the logarithmic scale for α).

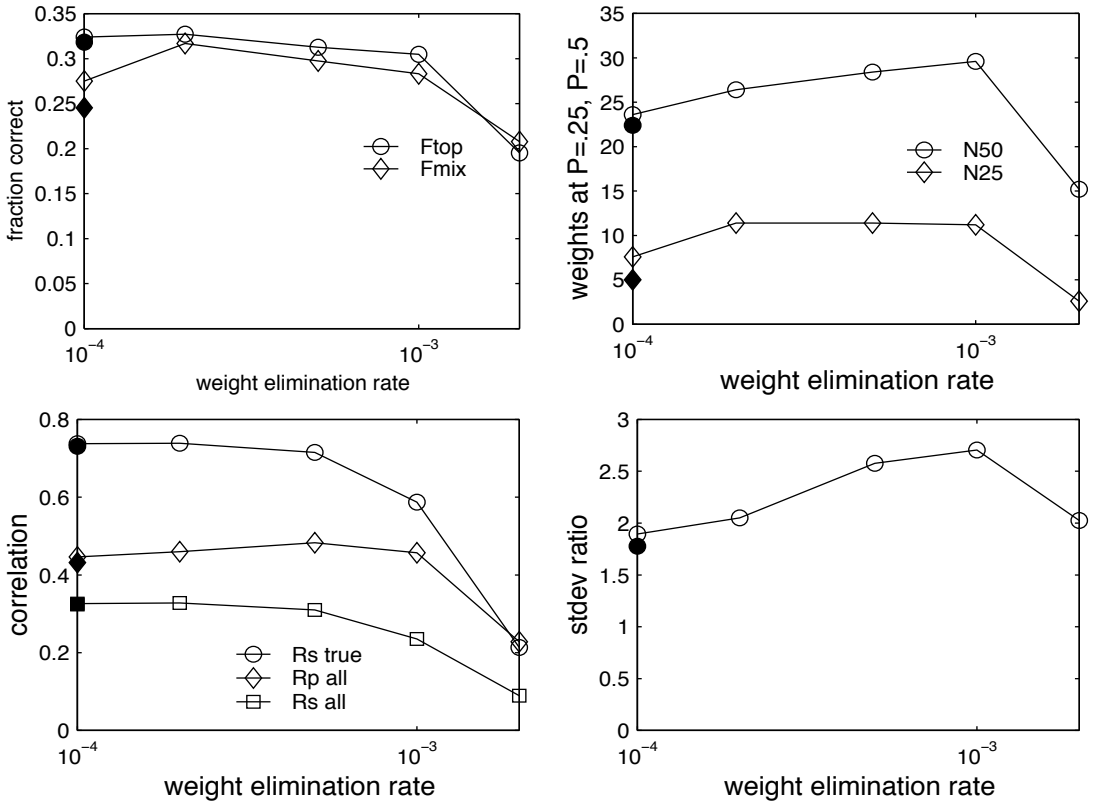

Figure 7.7: Effect of scaling weight elimination rate from $\gamma = 0.0001$ to 0.002. The solid markers on the vertical axis are the performance for $\gamma = 0$ (note the logarithmic scale for γ).

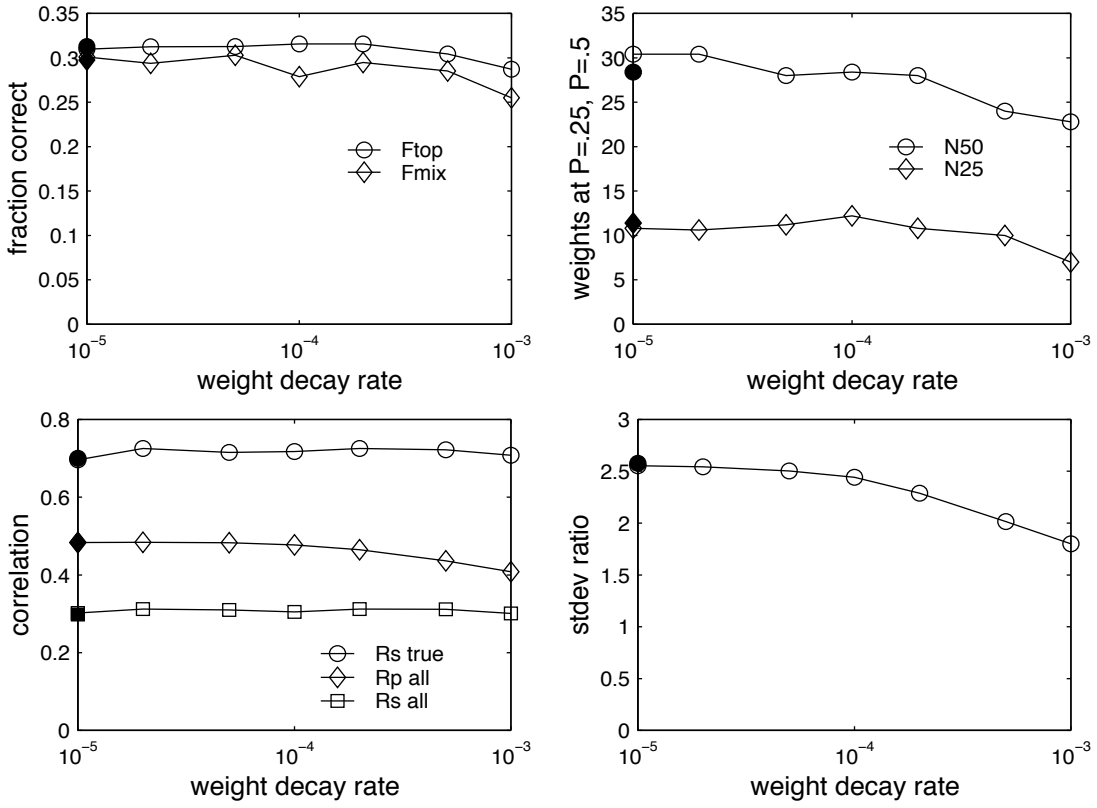

Figure 7.8: Effect of scaling weight decay rate from $\alpha = 0.00001$ to 0.001, for weight elimination rate $\gamma = 0.0005$. The solid markers on the vertical axis are the performance for $\alpha = 0$ (note the logarithmic scale for α).

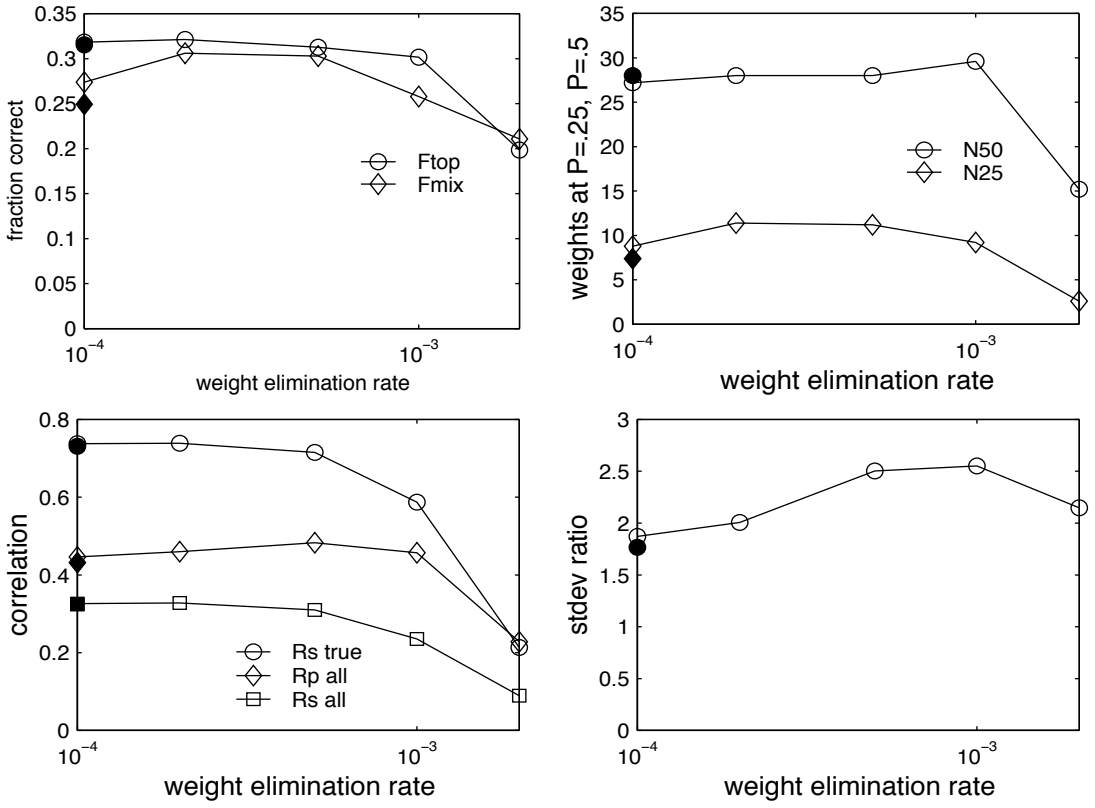

Figure 7.9: Effect of scaling weight elimination rate from $\gamma = 0.0001$ to 0.002, for weight decay rate $\alpha = 0.00005$. The solid markers on the vertical axis are the performance for $\gamma = 0$ (note the logarithmic scale for γ).

Chapter 8

Conclusions

This is not the end.
It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.
— Winston Churchill, 10 November 1942

8.1 The story so far...

Rather than giving up on these network models because they are officially "underdetermined", I have shown that they can indeed be applied to infer at least *part* of the regulatory interactions between genes from large-scale gene expression data. The first important result is a rather theoretical one: the estimates of data requirements in Chapter 5 show that, as long as we impose sufficient constraints on the network models, their data requirements might only scale logarithmically with the number of variables (number of genes). This compares favorably with the data requirements for clustering, although it is still perhaps an order of magnitude or more larger.

In practice, the lack of data compared with the number of parameters of the data turned out to be much more of a stumbling block than I had originally anticipated.

In retrospect, the underdetermined nature of the model should not have come as a surprise, simply based on the dimensionality of the data, and the significant correlations between the measurements. Nevertheless, I showed that it is indeed possible to identify some portion of the significant weights in the model, using the knowledge we have regarding the variability of the individual measurements. This points out yet again how crucial it is to know the error behavior of the data one is working with. A common trend towards the usage of replicate experiments may allow for more widespread use of this technique.

The linear model used in Chapter 6 is an extreme simplification, and should be regarded only as a first-order approximation. The tissues studied consist of multiple functional regions, multiple layers within each region, and multiple cell types within each layer, all of which can be expected to exhibit different expression patterns during development and injury. Also, the number of variables used is only a small fraction of the important variables that play a role in these tissues. Protein, neurotransmitter, and neuropeptide levels are missing entirely. Nevertheless, we find we can isolate several important known regulatory interactions. Other predictions generated by the model seem quite plausible when compared with current knowledge, and form useful new hypotheses that can guide further experimentation.

Finally, I showed that it is possible to move to a more realistic, nonlinear model, even with limited amounts of data. Although this nonlinear model should in principle be able to match the underlying regulatory network more closely, the actual fitting of the network becomes more complex and expensive, requiring training a neural network over thousands of iterations, where each iteration involves a forwards and backwards numerical integration through time. Nevertheless, even looking only at the magnitude of the weights we can identify approximately the top ten largest weights (out of a total of 420 weights) as "true" weights, with a P-value of 0.25. This does not sound like a great achievement, but as we saw in the linear model,

significance levels of a weight are only poorly correlated with their magnitude. As in the linear model, we should be able to achieve much better identification of the true weights by also taking the variability of each weight into account. Experiments along these lines are currently ongoing.

8.2 Directions for future research

8.2.1 Refinements of the linear model

Some further refinements could still be made to the linear model. For example, to capture the change in developmental speed around birth, we could explicitly add an additional input to the system for the "birth" event.

Rather than using the ordinary least squares solution, we could use a weighted least squares. This would allow us to (1) weigh expression levels according to the corresponding triplicate standard deviations on the measurements, (2) weigh interpolated time points based on the location within the interpolation interval (higher weight close to the real data points), (3) give equal weight to all the intervals between real data points (at the moment, their "weight" in the least squares solution is essentially proportional to the length of the interval, giving much higher weight to the final data points which are months apart). Note that the use of the Monte Carlo analysis in Chapter 6 essentially already covers the first two points: measurement with a larger triplicate standard deviation will get perturbed more, resulting in a smaller contribution to the Z-score of the associated weights. Similarly, perturbations in the real data points will probably cause larger perturbations in the interpolated time points, particularly in those interpolated points farthest away from the real data points.

Lastly, since the linear model essentially performs a multiple regression of all

genes on all genes, perhaps we could exploit some of the techniques developed to determine the significant inputs in multiple regression. Some of these are based on adding additional penalty terms to the optimization to account for the number of inputs, size of input weights, etc. At the extreme, this can reduce to a linear case of the sort of network optimization techniques discussed in Chapter 7.

8.2.2 Refinements of the neural network model

We have really only scratched the surface of modeling genetic regulatory networks using neural network models. Clearly, the work on identifying the "true" weights of the network based on their Z-score will have to be further developed (this is work in progress).

The introduction of dynamic Bayesian network methodology for gene expression analysis is an especially promising development. The nonlinear neural network developed in Chapter 7 is essentially similar to a nonlinear dynamic Bayesian network, as pointed out by Murphy and Mian [165]. Bayesian networks do not so much provide a different model, but rather a new perspective from an area which has a very thorough theoretical foundation, just as the neural network perspective provided us with useful insights and efficient tools to tackle a set of nonlinear differential equations.

Beyond these issues, there a wide variety of possible elaborations on the basic model that might be useful. Here are some of the more promising ones:

Fitting steady-state data using RBP

As already briefly mentioned in Section 7.3, we may also want to take advantage of the large number of data sets that do not consist of time-series data, but rather of individual measurements under different conditions, i.e. we would like to train

the model to have certain fixed points. Recurrent Backpropagation (RBP), independently developed by Pineda [181] and Almeida [12], provides us with an update rule for the weights to minimize the difference between the measured expression level $\hat{y}_i^0(t)$ and the fixed point of the model, $y_i^0(t)$. As in BPTT, it is trivial to add in extra error terms, provided we can easily differentiate the total error E with respect to the gene expression levels y_i. Appendix D.3 gives the RBP learning rule for Equation 7.1.

If we want to integrate BPTT and RBP into a single modeling methodology, we can simply treat fixed points as a special case of time series. One could either train on a two-point time series, where $\hat{y}_i(0) = \hat{y}_i(\Delta t) = \hat{y}_i^0$. This approach is similar to *teacher forcing* [249], and has the disadvantage that the achieved fixed point may not necessarily be a *stable* one. Alternatively—and closer to how RBP works—we could train on a two-point time series, where $\hat{y}_i(0) = \hat{y}_i(T) = \hat{y}_i^0$, with T sufficiently large that the system reaches its own (stable) fixed point y_i^0 (assuming one exists).

Time delays

Another possibility is to add time delays between the output of one unit and its effect on another. This could correspond to time delays incurred due to the transcription and translation steps, possible transport of the generated protein through the cell, or simply the effect of a series of intermediate steps that are not explicitly being modeled as nodes in the network. The total input to each node then becomes

$$x_i(t) = \sum_j w_{ji} y_j(t - \tau_{ji}), \tag{8.1}$$

(See Pearlmutter [176] for a derivation of the corresponding learning rules for the delays τ_{ji}) For data such as the Gene Expression Matrix [245], where expression patterns are measured days apart, the time resolution is too coarse to notice time delays between individual genes. However, for data sets such as the yeast cell cycle [217],

where the sampling rate is on the order of the response time of many genes, time delays may be crucial to correctly model the observed behavior. Likewise, circadian rhythms tend to involve long time delays, likely corresponding to regulatory proteins slowly being transported into the nucleus.

Hidden nodes

Several genes may be regulated by the same intermediate regulator not included in the data set (e.g. the end product of a long signalling cascade that combines several different signals), resulting in a large number of similar connections to these genes. It may be possible to simplify the model by adding an extra hidden node for this intermediate regulator, trading off a larger number of nodes for a smaller number of connections. (Hidden nodes can be trained using exactly the same learning rules as mentioned above.) Section 6.4.4 provides an example of a situation where it might be possible to infer a hidden node (although in this case, the addition of a hidden node does not significantly decrease the complexity of the model).

Two-layer networks

As mentioned earlier, protein interactions play an important role within a cell. It may be possible to include the effect of these interactions at the level of their influence on gene regulation, but modeling them explicitly would provide a model that is closer to the biological reality. The obvious extension would be a two-layer network, corresponding to mRNA levels and protein levels.

This model is more complex than the one proposed before. However, separating protein interactions from gene regulations may significantly cut down on the number of inputs to each gene. If these protein interactions are essential to the correct working of the system, they would have to be included as connections between units

in the simple model anyway. Furthermore, the number of extra connections from the mRNA layer to the protein layer is only on the order of the number of genes, and the weights on these connections may be preset based on the known properties of the mRNAs. Lastly, this kind of model would allow us to integrate both mRNA expression data and protein level data very naturally.

8.2.3 Incorporating biological knowledge

A LITTLE KNOWLEDGE CAN GO A LONG WAY
— Jenny Holzer

Obviously, if we have any a priori knowledge about the system we are trying to model, we should try to build that knowledge into the model. This may take the form of hardwiring known parts of the model, or simply biasing the training algorithm to generate a network that reflects the a priori knowledge.

Known or inferred protein interactions

Protein interactions can be inferred in a number of ways. There are some specialized interaction assays for specific classes of proteins such as kinases of DNA binding proteins. It is also possible to infer interaction based on similarity to other proteins that are already known to interact, or even based on three dimensional configuration. However, there is also a general protein interaction assay, the yeast 2-hybrid method [74]. For example, Uetz *et al.* [232] recently used large-scale yeast 2-hybrid screens to identify 957 putative interactions, involving 1004 yeast proteins.

Given reliable information on which proteins interact with which others, we would be able to fix a large part of the structure of our recurrent network, instead of having to starts out with a fully interconnected net. However, none of these techniques can

capture all of the possible protein interactions. Once sufficient information of this nature becomes available, we might be able to construct an a priori probability distribution of the weights of the connections in the network, and then use the deviation from this distribution as a penalty term in the total error (Equation 7.4) to bias the training of the network.

Hints from theoretical biology

One of the goals of theoretical biology is to find general principles that are at work in biological systems. We may be able to use some of these principles to guide us in our search for a good model.

In particular, Michael Savageau has been examining principles of gene regulation. On one hand, Savageau's *demand theory* [194] relates to the mode of regulation of single genes, stating that genes whose products are in high demand should be positively regulated, and vice versa. On the other hand, his analysis of small coupled gene circuits based on six "criteria for functional effectiveness" (stability, robustness, decisiveness, efficiency, responsiveness and selectivity) gives an indication of what circuit structures and parameter settings can be expected [109]. If correct, these principles could be used to assess the biological plausibility of the final model. It is as yet unclear how they could be used to guide the training of the recurrent network.

8.3 A look towards the future

As of this writing, we are still in the exponential phase of deployment of large-scale gene expression measurement technologies. Frost & Sullivan [80] estimate approximately a doubling in the number of arrays used for each of the next two years, with a prediction of well over 1.5 million arrays used in 2003. Considering the

nearly constant stream of new technologies, this may very well be an underestimate. Miniaturization, automation and mass-production will likely reduces the cost per gene expression experiment to a few dollars per chip. Once these technologies start influencing our daily lives—probably primarily as diagnostic tools in a hospital setting—there is no predicting how pervasive they will become.

As these genome-scale technologies mature, we can expect to see:

- More whole-genome measurements, rather than selected subsets of genes, increasing the need for analysis tools that can deal with large amounts of superfluous variables.

- Higher accuracy, allowing better distinction between genes with similar expression patterns. At the moment, some people still view array data essentially as qualitative data: useful as a first approach, but in need of validation by other means if one actually wants to publish a result. With increasing accuracy, automation, and understanding of the errors, large scale gene expression technology will likely become generally accepted as a quantitative measurement tool.

- Possibly higher time resolution, as we get more experience with response times of the very fastest genes. To observe the very fastest changing genes, we may very well have to resort to lab-on-a-chip approaches to do the measurement *in situ*, before the mRNA decays. For now, time resolutions on the order of a few minutes are definitely feasible, and sufficient for the vast majority of mRNA species.

- More data points, making the sorts of approaches presented here more effective. As mentioned before, in order to infer the regulation of any gene, one has to thoroughly exercise the different inputs to the gene. The recent trickle of very

large data sets (such as Hughes *et al.* [118]: 300 separate measurements on yeast, all calibrated) are likely only the beginning.

- More replicates, resulting in better error models and a better appreciation of why and when genes show increased variability. Currently, replicates are mainly used for averaging (thus reducing the error variance), and for assigning significance levels to the amount of up- or down-regulation of a gene. However, as I have illustrated in Chapter 6, they also provide a crucial tool to identify well-determined regulatory interactions in the data.

Especially the advent of larger data sets and more data sets with replicates Should make the modeling methodologies developed here more widely applicable. We can also expect to see production of more large-scale non-mRNA data, bringing with it an increased need for integration between disparate data types within the same computational analysis, as well as integration with other information sources, such as literature data bases, etc. The Bayesian approach to learning neural networks—adding additional knowledge as priors on the resulting network—provides for a very flexible tool to integrate these disparate types of data.

After the explosion of genomic-scale data, we are finally starting to see a smattering of computational tools that can deal with this data. I hope the techniques I have developed here will be a useful addition to this growing genomic biologist's tool chest. Much work yet remains to be done, and as the technologies and analysis tools develop, we will likely identify other challenges.

As the saying goes: "in the land of the blind, the one-eyed man is King" large-scale gene expression technology has given us an "eye" into the internal workings of cells. It's still only one eye, so we're only seeing half the picture. And it's still somewhat blurry, but we're furiously developing lenses. But what a difference one eye makes...

My pen is at the bottom of a page,
Which, being finished, here the story ends;
'Tis to be wished it had been sooner done,
But stories somehow lengthen when begun.
— Byron

Appendices

A Overview of clustering methods **138**

A.1 Distance measures and preprocessing 139

A.2 Clustering algorithms 140

B Fitting the linear model **145**

C Linear models based on randomized data **148**

C.1 Simultaneously matching slopes and expression levels 148

C.1 Matching the covariance matrix 149

D Derivation of Backpropagation Through Time **153**

D.1 General derivation 153

D.2 BPTT for gene networks 156

D.3 Recurrent Backpropagation 159

Appendix A

Overview of clustering methods

I believe the day will come when the biologist will—without being a mathematician—not hesitate to use mathematical analysis when he requires it.
— Karl Pearson, in Nature, 1901

This appendix appeared earlier as part of a review paper, written in collaboration with Shoudan Liang and Roland Somogyi [64] (reprinted here by permission of Oxford University Press). This particular section was co-written with Shoudan Liang. It is up-to-date up to about the end of 1999. In the months since then, a number of new clustering algorithms have been developed [100, 105, 106, 52, 111, 203, 140]. One has to wonder whether this plethora of clustering algorithms represents genuine advances in the field, or rather whether it is illustrative of the ease with which one can invent "yet another clustering algorithm". One promising note is the increased use of robust statistics, such as "jackknife correlation" [106], leave-one-out cross validation (LOOCV) [29], etc.

A.1 Distance measures and preprocessing

Most clustering algorithms take a matrix of pairwise distances between genes as input. The choice of distance measure—used to quantify the difference in expression profiles between two genes—may be as important as the choice of clustering algorithm. Distance measures can be divided into at least three classes, emphasizing different regularities present within the data: a) similarity according to positive correlations, which may identify similar or identical regulation; b) similarity according to positive and negative correlations, which may also help identify control processes that antagonistically regulate downstream pathways; c) similarity according to mutual information, which may detect even more complex relationships.

So far, most clustering studies in the gene expression literature use either Euclidean distance or Pearson correlation between expression profiles as a distance measure. Other measures used include Euclidean distance between expression profiles and slopes (for time series [245]), squared Pearson correlation [65], Euclidean distance between pairwise correlations to all other genes [72], Spearman rank correlation [65], and mutual information [65, 161, 45].

Conspicuously absent so far are distance measures that can deal with the large numbers of highly related measurements in the data sets. For example, clustering yeast genes based on all publicly available data will be highly biased towards the large cell cycle data sets: 73 data points in 4 time series, containing almost 8 complete cell cycles [217], whereas only a single data point may be present for various stress conditions, mutations, etc. Correlation between the experiments will also lead to highly elliptical clusters, which form a problem for clustering methods that are biased towards compact, round clusters (such as K-means). A distance measure that can deal with the covariance between experiments in a principled way (e.g. Mahalanobis distance [151] may be more appropriate here. For even longer time series, distance

measures based on Fourier or wavelet transforms may be considered.

A related issue is normalization and other preprocessing of the data. Distance measures that are sensitive to scaling and/or offsets (such as Euclidean distance) may require normalization of the data. Normalization can be done with respect to the maximum expression level for each gene, with respect to both minimum and maximum expression level or with respect to the mean and standard deviation of each expression profile. From a statistical point of view, we recommend using the latter, unless there is a good reason to preserve the mean expression values. When using relative expression levels (for example, microarray data), the data will tend to be log-normally distributed, so the logarithm of the relative expression values should be used. Califano *et al.* [46] suggest using a nonlinear transformation into a uniform distribution for each gene instead, which will tend to spread out the clusters more effectively.

A.2 Clustering algorithms

All clustering algorithms assume the pre-existence of groupings of the objects to be clustered. Random noise and other uncertainties have obscured these groupings. The objectives of the clustering algorithm are to recover the original grouping among the data.

Clustering algorithms can be divided into hierarchical and non-hierarchical methods. Non-hierarchical methods typically cluster N objects into K groups in an iterative process until certain goodness criteria are optimized. Examples of non-hierarchical methods include K-means, EM (expectation-maximization) and Autoclass. Hierarchical methods return a hierarchy of nested clusters, where each cluster typically consists of the union of two or more smaller clusters. The hierarchical methods can be further distinguished into agglomerative and divisive methods, depending

on whether they start with single-object clusters and recursively merge them into larger clusters, or start with the cluster containing all objects and recursively divide it into smaller clusters. In this section, we review several clustering methods for gene expression.

The K-means algorithm [150] can be used to partition N genes into K clusters, where K is predetermined by the user (see e.g. Tavazoie *et al.* [228] for an application to yeast gene expression). K initial cluster "centroids" are chosen—either by the user, to reflect representative expression patterns, or at random—and each gene is assigned to the cluster with the nearest centroid. Next, the centroid for each cluster is recalculated as the average expression pattern of all genes belonging to the cluster, and genes are reassigned to the closest centroid. Membership in the clusters and cluster centroids are updated iteratively until no more changes occur, or the amount of change falls below a pre-defined threshold. K-means clustering minimizes the sum of the squared distance to the centroids, which tends to result in round clusters. Different random initial seeds can be tried to assess the robustness of the clustering results.

The Self-Organized Map (SOM) method [135] is closely related to K-means and has been applied to mRNA expression data of yeast cell cycles as well as hematopoietic differentiation of four well-studied model cell lines [225]. The method is more structured than K-means in that the cluster centers are located on a grid. At each iteration, a randomly selected gene expression pattern attracts the nearest cluster center, plus some of its neighbors in the grid. Over time, fewer cluster centers are updated at each iteration, until finally only the nearest cluster is drawn towards each gene, placing the cluster centers in the center of gravity of the surrounding expression patterns. Drawbacks of this method are that the user has to specify a priori the number of clusters (as for K-means), as well as the grid topology, including the dimensions of the grid (typically one, two or three-dimensional) and the number of

clusters in each dimension (e.g. 8 clusters could be mapped to a 2x4 2D grid or a 2x2x2 3D cube). The artificial grid structure makes it very easy to visualize the results, but may have residual effects on the final clustering. Optimization techniques for selecting the number of clusters developed for K-means can presumably be used here too.

The Expectation-Maximization (EM) algorithm [61] for fitting a mixture of Gaussians (also known as Fuzzy K-Means [31] is very similar to K-means, and has been used by Mjolsness *et al.* [162] to cluster yeast data. Rather than classifying each gene into one specific cluster, we assign membership functions (typically Gaussians, or any other parametric probability distribution) to each cluster, allowing each gene to be part of several clusters. As in K-means, we alternately update the membership for each expression pattern, and then the parameters associated with each cluster: centroid, covariance and mixture weight. Cluster boundaries are sharp and linear in K-means, smooth and rounded in EM.

Autoclass [49] is also related to EM, in that it finds a mixture of probability distributions. In addition, it uses Bayesian methods to derive the maximum posterior probability classification, and the optimum number of clusters.

Wen *et al.* [245] used the FITCH hierarchical clustering algorithm [73] to group the expression patterns of 112 genes in spinal cord development, producing a graph similar to the phylogenetic trees familiar to most biologists [211]. The expression clusters captured the main waves of gene expression in development. While the algorithm used in this study minimizes the overall distance in the tree, the computational requirement grows with the fourth power of the number of elements, making it impractical for much larger data sets.

Eisen *et al.* [70] applied a standard agglomerative hierarchical clustering algorithm, average-linkage analysis, to large-scale gene expression data. Starting with N

clusters containing a single gene each, at each step in the iteration the two closest clusters are merged into a larger cluster. Distance between clusters is defined as the distance between their average expression pattern. After N-1 steps, all the genes are merged together into a hierarchical tree. Other hierarchical methods may calculate distance between clusters differently. In UPGMA (unweighted pair-group method using arithmetic averages [210] for example, the distance between two clusters is defined as the average distance between genes in the two clusters.

Ben-Dor and Yakhini [30] have developed a clustering algorithm based on random graph theory. Their method shares features with both agglomerative hierarchical clustering and K-means. Clusters are constructed one at a time. The gene with the largest "affinity" (smallest average distance to all other genes in the cluster) is added to the cluster, if the affinity is larger than a cutoff. A gene can also be removed from the cluster if its affinity drops below the cutoff. A finite number of clusters are constructed depending on the cutoff. The ability to remove ill-fitting genes from the cluster is an attractive feature of this algorithm. Zhu and Zhang [257] used a similar algorithm to cluster yeast sporulation data.

Alon *et al.* [13] used a divisive hierarchical algorithm to cluster gene expression data of colon cancer. The method relies on the maximum entropy principle and attempts to find the most likely partition of data into clusters at a given "cost" (sum of squared within-cluster distances). Starting from a single cluster with large cost, as the allowed cost is lowered, the cluster breaks up spontaneously into multiple clusters in order to maximize the entropy for the configuration, within the constraint of fixed total cost.

Califano *et al.* [46] have developed a clustering algorithm to identify groups of genes which can be used for phenotype classification of cell types, by searching for clusters of microarray samples that are highly correlated over a subset of genes. Only the most significant clusters are returned. The same technique could be used

to find clusters of genes that are highly coexpressed over a subset of their expression profiles. Han *et al.* [97] used a similar, partial matching approach to group objects into a hypergraph based on correlations over subsets of the data. In a hypergraph, each hyperedge (corresponding to a single cluster) connects several nodes (genes), so each node (gene) can be part of several hyperedges (clusters). Mjolsness *et al.* [163] developed a hierarchical algorithm that places objects into a directed, acyclic graph, where each cluster can be part of several parent clusters. The algorithm optimizes the number of clusters, cluster positions and partial cluster memberships of objects, such as to provide the most compact graph structure. All three clustering methods allow genes to be part of several clusters, possibly coinciding with multiple regulatory motifs or multiple functional classifications for each gene. This makes them especially appropriate for eukaryotic gene expression where genes are controlled by complex inputs from multiple transcription factors and enhancers.

Appendix B

Fitting the linear model

First, we rewrite Equation 6.6 in matrix notation:

$$\frac{\Delta\vec{\mathbf{y}}(t)}{\Delta t} = \mathbf{W}\vec{\mathbf{y}}(t) + \vec{\mathbf{K}}\,\kappa(t) + \vec{\mathbf{T}}\,\tau + \vec{\mathbf{B}} \tag{B.1}$$

where $\Delta\vec{\mathbf{y}}(t)$, $\vec{\mathbf{y}}(t)$, $\vec{\mathbf{K}}$, $\vec{\mathbf{T}}$ and $\vec{\mathbf{B}}$ are now column vectors containing the corresponding values of $\Delta y_i(t) = y_i(t+\Delta t) - y_i(t)$, $y_i(t)$, K_i, T_i and b_i for all 65 genes, and $\mathbf{W}$ is a 65×65 matrix containing the parameters w_{ji}. To simplify, we can include the parameters $\vec{\mathbf{K}}$, $\vec{\mathbf{T}}$ and $\vec{\mathbf{B}}$ as extra columns in the matrix $\mathbf{W}$ (which now becomes a rectangular 65×68 matrix), provided we add $\kappa(t)$, τ, and a unit constant as additional "inputs" to $\vec{\mathbf{y}}(t)$ in the right hand side.

$$\frac{\Delta\vec{\mathbf{y}}(t)}{\Delta t} = \left[\begin{array}{cccc} \mathbf{W} & \vec{\mathbf{K}} & \vec{\mathbf{T}} & \vec{\mathbf{B}} \end{array}\right] \cdot \left[\begin{array}{c} \vec{\mathbf{y}}(t) \\ \kappa(t) \\ \tau \\ \mathbf{1} \end{array}\right] \tag{B.2}$$

For convenience, we will call the augmented weight matrix $\widetilde{\mathbf{W}}$, and the augmented input vector $\tilde{\mathbf{y}}(t)$. Note that we have one such equation for each time interval in each

of the interpolated time series. We can combine these into a single matrix equation:

$$\frac{\Delta \mathbf{Y}}{\Delta t} = \widetilde{\mathbf{W}}\,\widetilde{\mathbf{Y}} \tag{B.3}$$

where $\Delta\mathbf{Y}$ is a 65×52080 matrix, containing $\Delta\vec{\mathbf{y}}(t) = \vec{\mathbf{y}}(t+\Delta t) - \vec{\mathbf{y}}(t)$ for all 52080 interpolated time intervals of each time series; and $\widetilde{\mathbf{Y}}$ is a 68×52080 matrix, containing $\vec{\mathbf{y}}(t)$, $\kappa(t)$, τ, and a unit constant for all but the *last* time points of each time series:

$$\Delta\mathbf{Y} = \left[\begin{array}{ccc|ccc|ccc}
y_1^s(2)-y_1^s(1) & \cdots & y_1^s(n_s)-y_1^s(n_s-1) & y_1^h(2)-y_1^h(1) & \cdots & y_1^h(n_h)-y_1^h(n_h-1) & y_1^k(2)-y_1^k(1) & \cdots & y_1^k(n_k)-y_1^k(n_k-1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
y_N^s(2)-y_N^s(1) & \cdots & y_N^s(n_s)-y_N^s(n_s-1) & y_N^h(2)-y_N^h(1) & \cdots & y_N^h(n_h)-y_N^h(n_h-1) & y_N^k(2)-y_N^k(1) & \cdots & y_N^k(n_k)-y_N^k(n_k-1)
\end{array}\right]$$

spinal cord development | hippocampus development | hippocampus kainate injury

$$\widetilde{\mathbf{Y}} = \left[\begin{array}{ccc|ccc|ccc}
y_1^s(1) & \cdots & y_1^s(n_s-1) & y_1^h(1) & \cdots & y_1^h(n_h-1) & y_1^k(1) & \cdots & y_1^k(n_k-1) \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
y_N^s(1) & \cdots & y_N^s(n_s-1) & y_N^h(1) & \cdots & y_N^h(n_h-1) & y_N^k(1) & \cdots & y_N^k(n_k-1) \\
0 & \cdots & 0 & 0 & \cdots & 0 & \kappa(1) & \cdots & \kappa(n_k-1) \\
0 & \cdots & 0 & 1 & \cdots & 1 & 1 & \cdots & 1 \\
1 & \cdots & 1 & 1 & \cdots & 1 & 1 & \cdots & 1
\end{array}\right] \tag{B.4}$$

$$\widetilde{\mathbf{W}} = \left[\begin{array}{cccccc}
w_{1,1} & \cdots & w_{N,1} & K_1 & T_1 & b_1 \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
w_{1,N} & \cdots & w_{N,N} & K_N & T_N & b_N
\end{array}\right] \tag{B.5}$$

where N is the number of genes ($N = 65$), n_s, n_h and n_k are the number of interpolated time points in the spinal cord development, hippocampus development and

hippocampus kainate injury time series, respectively (for convenience, the interpolated time points are ordered from 1 to n in each time series), and $y_i^s(t)$, $y_i^h(t)$ and $y_i^k(t)$ are the interpolated expression levels in those three time series. Note that the kainate level $\kappa(t)$ (third row from the bottom in Equation B.4) is zero except for the kainate injury time series, and that the tissue indicator variable τ (second row from the bottom in Equation B.4) is 0 for spinal cord and 1 for the two hippocampus time series.

If $\widetilde{\mathbf{Y}}$ were an invertible square matrix, we could solve for $\widetilde{\mathbf{W}}$ exactly using $\widetilde{\mathbf{W}} = \Delta\mathbf{Y}/\Delta t \cdot \widetilde{\mathbf{Y}}^{-1}$. Since $\widetilde{\mathbf{Y}}$ is rectangular, and has more rows than columns, the system is overdetermined and no exact solution for Equation B.3 is possible. However, we can find the least squares solution $\mathbf{W}^+$ using the following formula (see, e.g., [220, 88]):

$$\mathbf{W}^+ = \frac{\Delta\mathbf{Y}}{\Delta t}\widetilde{\mathbf{Y}}^{\mathrm{T}}\left(\widetilde{\mathbf{Y}}^{\mathrm{T}}\widetilde{\mathbf{Y}}\right)^{-1} \tag{B.6}$$

(Or, if $\widetilde{\mathbf{Y}}$ is rank-deficient, we could use the pseudoinverse [220, 88] to find a unique least squares solution). The resulting 65-by-68 matrix $\mathbf{W}^+$ gives us the least squares fit for the parameters w_{ji}, K_i, T_i and b_i in Equation 6.6.

Appendix C

Linear models based on randomized data

C.1 Simultaneously matching slopes and expression levels

As mentioned in Section 6.4.5, we could match the distribution of expression level at each time point in the real data sets by randomly permuting the expression values within each time point. Unfortunately, this tends to drastically change the distribution of *changes* in expression levels, which we also want to match. We can approximately match both distributions in the following way:

1. Initialize the randomized time series using a random permutation of the real, normalized expression levels for each time point.

2. Starting at the first time interval, rank the slopes between the first two time points. Rank the slopes for the first interval in the real data set. Assign the

ranked "real" slopes to the ranked randomized slopes, i.e., the randomized time series with highest slope will get assigned the highest slope in the real data set. This will shift the expression levels for the second time point.

3. Rank the new expression levels for the second time point, and assign the ranked expression levels to the correspondingly ranked randomized time series.

4. Alternatively match the ranked slopes for the next interval and match the ranked expression levels for the following time point until the slopes and expression levels have been adjusted for the entire time series. Do the same to get randomized versions of all three real data sets.

5. The randomized time series now have the exact same distribution of expression levels for each time point as the real data, but the slopes are only a crude approximation (because after we matched the slopes, we changed the expression levels of the second time point of each time interval). However, if we iterate steps 1-5 a number of times, both distributions will converge to be very close to the real data.

C.2 Matching the covariance matrix

We have now achieved a randomized data set that matches both expression levels and slopes of the real data. However, the new time series will typically be more independent that what we find for the real genes. In the real model, it is often hard to decide whether gene A should get a regulatory input from gene B or C, because B and C are highly correlated. If we were to use a randomized data set with less correlation among the variables, the model would be able to assign interaction parameters with much greater certainty. In order to get a truly comparable randomized data set, we also want to match the amount of correlation between the individual time series.

Otherwise, any differences in performance between the real and randomized data could be purely due to differences in amount of correlation, not differences in the presence or absence of causal interactions.

If we think of each data point—i.e. set of measurements of all 65 genes—as a point in a 65-dimensional space, then these 28 data points determine a 65-dimensional (hyper-)ellipsoid. The standard deviation of each variable (gene) is related to the size of the projection of this ellipsoid on the corresponding dimension. The *covariance* between genes is then be a measure of how much this high-dimensional ellipsoid lies along the diagonal of the corresponding dimensions. We want to achieve a randomized data set for which this ellipsoid (i.e. the covariance matrix of the variables) has a similar size and similar amounts of "diagonality".

The *Singular Value Decomposition* (SVD) [220, 88] of a set of zero-centered variables essentially provides us with a change of basis to the principal axes of the ellipsoid mentioned above:

$$\mathbf{Y} = \mathbf{U} \cdot \mathbf{S} \cdot \mathbf{V}^{\mathrm{T}} \tag{C.1}$$

If $\mathbf{Y}$ is a $m \times n$ matrix (in our case 28 data points by 65 genes), then $\mathbf{U}$ is an $m \times n$ matrix containing the new set of (now orthogonal) variables, $\mathbf{S}$ is a diagonal $n \times n$ matrix containing the singular values, essentially the sizes of the principal axes of the ellipsoid, and $\mathbf{V}$ is an $n \times n$ matrix with the principle axes of the original ellipsoid. The columns of $\mathbf{U} \cdot \mathbf{S}$ are also called the *Principle Components* of the data. The singular values in $\mathbf{S}$ are the square roots of the eigenvalues of the covariance matrix of the data, and $\mathbf{V}$ contains the corresponding eigenvectors.

Using subscript 'r' for the real data set, and 'c' for the control (randomized) data set, if $\hat{\mathbf{Y}}_c$ stands for the zero-centered randomized time series (i.e. we subtract the mean value of the expression level from each gene), then the following formula

stretches the principle components of the randomized time series to the same size as those of the real time series, and rotates them to lie along the principle axis of the real data:

$$\hat{\mathbf{Y}}_c \leftarrow \mathbf{U}_c \cdot \mathbf{S}_r \cdot \mathbf{V}_r^{\mathrm{T}} \tag{C.2}$$

The result is a time series which resembles the original randomized data, but which has a covariance matrix which is similar to that of the real data. We can now integrate matching slopes, expression levels and covariances within a single procedure:

1. Initialize the randomized time series using a random permutation of the real, normalized expression levels for each time point.
2. Subtract the mean expression level from each time series.
3. Calculate the SVD of the zero-centered time series, and replace the singular values with those of the real time series (Equation C.2).
4. Add back in the mean values we subtracted in step 2.
5. Starting at the first time interval, alternatively match slopes and expression levels with the real data set.
6. Repeat steps 2-6 until convergence.

Within a reasonably small number of iterations, we arrive at a randomized data set which almost perfectly matches the real data set in terms of expression levels, slopes between time points, and covariances between genes. Of course, after this elaborate matching of the randomized data to the real data, one might wonder whether we haven't just recreated the original data set. If so, we would expect to see high

correlations between the randomized time series and some of the real genes. Some correlation is to be expected, because the new time series follow the same overall trends, by following the same distribution of expression levels at each time point as the real data. Fortunately, it turns out the correlations between randomized and real time series are significantly smaller[1] than the correlations within the real time series or within the randomized time series, i.e. the new time series show more similarity to each other than to the real genes, and vice versa.

[1] $P < 0.05$, based on 200 randomized time series

Appendix D

Derivation of Backpropagation Through Time

This appendix is an extension of the derivation for continuous-time Backpropagation Through Time (BPTT) by Pearlmutter [176], expanded based on clarifications from [247], and with a few corrections. The notation is mostly identical to [176], with the exception of the use of $S(\cdot)$ for the sigmoidal transfer function, $\mathcal{E}(\cdot)$ in the integral of the error term, and separate rate terms A_i and D_i for the production and decay terms.

D.1 General derivation

Let us start with the general case. Given a set of differential equations of the form:

$$\frac{dy_i}{dt} = f_i(\mathbf{y}(t), \mathbf{w}, \mathbf{I}(t)) \tag{D.1}$$

where y_i is the activation level of node i, $\mathbf{I}$ are external inputs, and $\mathbf{w}$ is a set of weights that determines how each particular node i combines the external inputs and

activation levels to generate the slope $f_i(t)$.

Given also an error term E, which we need to minimize:

$$E = \int_{t_1}^{t_n} \mathcal{E}(\mathbf{y}(t), t)dt \tag{D.2}$$

Typically, $\mathcal{E}$ will be a measure of the distance between $\mathbf{y}(t)$ and its desired value $\hat{\mathbf{y}}(t)$, possible weighted by a function over t to indicate when a good match is most crucial. Our goal then will be to optimize the weights w of the system to minimize the error term E. Typically, this involves calculating the derivative of E with respect to the weights w, and then doing a gradient descent on w to minimize E.

If Equation D.1 is, for example, of a form:

$$\frac{dy_i}{dt} = S\Big(\sum_j w_{ji}\, y_j\Big) - y_i + I_i \tag{D.3}$$

then it defines a continuous-time recurrent neural network (See Section D.2).[1] In order to find the training rule to update the weights, we can "unroll" such a network into time using a discrete time step Δt, generating a regular, feed-forward network with as many layers as time steps in the discretized time span, and then take the limit as Δt goes to zero. Variables discretized in time will will be indicated with a tilde: $\tilde{y}_i$, etc. Using Equation D.1, we then derive the following first order difference equation:

$$\tilde{y}_i(t + \Delta t) = \tilde{y}_i(t) + \Delta t\, f_i(\mathbf{y}(t), \mathbf{w}, \mathbf{I}(t)) \tag{D.4}$$

We now introduce the *ordered derivative* [247, 248] of the error term E with respect to a variable Q_k, where the Q_k are ordered such that their values can be

[1] Although, as [247] points out, this approach is applicable to *any* dynamical system of the form D.1.

calculated in the order $Q_1, Q_2, \ldots Q_n, E$. Whereas the regular partial derivative of E with respect to Q_k refers to the *instantaneous* change in E due to a small change in Q_k, the ordered derivative is the total propagated change due to all the variables $Q_{k+1} \ldots Q_n$ that directly depend on Q_k. The chain rule for ordered derivatives (indicated by ∂^+) is as follows:

$$\frac{\partial^+ E}{\partial Q_k} = \frac{\partial E}{\partial Q_k} + \sum_{j>k} \frac{\partial^+ E}{\partial Q_j} \cdot \frac{\partial Q_j}{\partial Q_k} \tag{D.5}$$

For example, the ordered derivative of E with respect to $\tilde{y}_i(t)$ measures how much a small change to $\tilde{y}_i$ at time t affects E when that error is propagated through time. Since $\tilde{y}_i(t)$ directly affects $\tilde{y}_j(t + \Delta t)$, the chain rule results in:

$$\tilde{z}_i(t) = \frac{\partial^+ E}{\partial \tilde{y}_i(t)} = \frac{\partial E}{\partial \tilde{y}_i(t)} + \sum_j \frac{\partial^+ E}{\partial \tilde{y}_j(t + \Delta t)} \cdot \frac{\partial \tilde{y}_j(t + \Delta t)}{\partial \tilde{y}_i(t)} \tag{D.6}$$

$$= \Delta t \, e_i(t) + \sum_j \tilde{z}_j(t + \Delta t) \cdot \frac{\partial \tilde{y}_j(t + \Delta t)}{\partial \tilde{y}_i(t)} \tag{D.7}$$

where $e_i(t)$ is $\partial E / \partial \tilde{y}_i(t)$, i.e. the immediate effect of a change in $\tilde{y}_i(t)$ on E. Using Equation D.4, we get:

$$\tilde{z}_i(t) = \Delta t \, e_i(t) + \tilde{z}_i(t + \Delta t) + \Delta t \sum_j \tilde{z}_j(t + \Delta t) \cdot \frac{\partial f_j(\mathbf{y}(t), \mathbf{w}, \mathbf{I}(t))}{\partial \tilde{y}_i(t)} \tag{D.8}$$

Note that the calculation of $\tilde{z}_i(t)$ depends on the *later* values of $\tilde{z}_j(t + \Delta t)$. This is where backpropagation derives its name from: the errors due to the differences between the outputs of the nodes are propagated back through the network and used to calculate how the weights should be updated to minimize the error. We can rewrite this as a difference equation:

$$\frac{\tilde{z}_i(t + \Delta t) - \tilde{z}_i(t)}{\Delta t} = -e_i(t) - \sum_j \tilde{z}_j(t + \Delta t) \cdot \frac{\partial f_j(\mathbf{y}(t), \mathbf{w}, \mathbf{I}(t))}{\partial \tilde{y}_i(t)} \tag{D.9}$$

and taking the limit as $\Delta t \to 0$:

$$\frac{dz_i}{dt} = -e_i - \sum_j \frac{\partial f_j(\mathbf{y}, \mathbf{w}, \mathbf{I})}{\partial y_i} z_j \tag{D.10}$$

Let us now look at the effect of a small change in a particular weight w_k on the total error E. w_k does not directly affect E, but it does directly affect $\tilde{y}_j(t+\Delta t)$ for all discretized time points t. The ordered derivative of the error E with respect to the individual weights w_k becomes:

$$\frac{\partial^+ E}{\partial w_k} = \sum_{t_1}^{t_n} \frac{\partial^+ E}{\partial \tilde{y}_j(t+\Delta t)} \cdot \frac{\partial \tilde{y}_j(t+\Delta t)}{\partial w_k} \tag{D.11}$$

$$= \sum_{t_1}^{t_n} \tilde{z}_j(t+\Delta t)\Delta t \, \frac{\partial f_j(\mathbf{y}(t), \mathbf{w}, \mathbf{I}(t))}{\partial w_k} \tag{D.12}$$

and taking the limit as $\Delta t \to 0$:

$$\frac{\partial E}{\partial w_k} = \int_{t_1}^{t_n} z_j \frac{\partial f_j(\mathbf{y}, \mathbf{w}, \mathbf{I})}{\partial w_k} \, dt \tag{D.13}$$

D.2 BPTT for gene networks

Rather than the standard form of a recurrent neural network used in [176], and listed in Equation D.3, we will use the following form, which has separate rate constants for the production and decay terms, and where an external inputs are treated similar to outputs from other nodes[2]:

$$\frac{dy_i}{dt} = A_i S(x_i) - D_i y_i \tag{D.14}$$

[2]Note that in Equation D.3, a similar effect can be obtained by having separate input nodes, with all other input weight set to zero. For a fixed input I_i, such nodes will converge to an activation level of $y_i = I_i$.

where—in our case—$S(\cdot)$ is a differentiable sigmoidal bounded between $S(-\infty) = 0$ and $S(+\infty) = 1$, and x_i, the total input to node i, given by:

$$x_i = \sum_j w_{ji}\, y_j \tag{D.15}$$

With this formulation, the difference equation for $\tilde{z}_i$ becomes:

$$\frac{\tilde{z}_i(t+\Delta t) - \tilde{z}_i(t)}{\Delta t} = -e_i(t) + D_i \tilde{z}_i(t+\Delta t) - \sum_j w_{ij}\, A_j\, S'(\tilde{x}_j(t))\, \tilde{z}_j(t+\Delta t) \tag{D.16}$$

and taking the limit as $\Delta t \to 0$:

$$\frac{dz_i}{dt} = D_i\, z_i - e_i - \sum_j A_j\, w_{ij}\, S'(x_j)\, z_j \tag{D.17}$$

Equation D.12 will give slightly different formulas for the different classes of weights w_{ij}, A_i and D_i:

$$\frac{\partial^+ E}{\partial w_{ij}} = \sum_{t_1}^{t_n} \tilde{z}_j(t+\Delta t)\Delta t\, A_j\, S'(\tilde{x}_j(t))\, \tilde{y}_i(t) \tag{D.18}$$

$$\frac{\partial^+ E}{\partial A_i} = \sum_{t_1}^{t_n} \tilde{z}_j(t+\Delta t)\Delta t\, S(\tilde{x}_j(t)) \tag{D.19}$$

$$\frac{\partial^+ E}{\partial D_i} = -\sum_{t_1}^{t_n} \tilde{z}_j(t+\Delta t)\Delta t\, \tilde{y}_j(t) \tag{D.20}$$

and their continuous equivalent, taking the limit as $\Delta t \to 0$:

$$\frac{\partial E}{\partial w_{ij}} = \int_{t_1}^{t_n} A_j\, z_j\, S'(x_j)\, y_i\, dt \tag{D.21}$$

$$\frac{\partial E}{\partial A_i} = \int_{t_1}^{t_n} z_j\, S(x_j)\, dt \tag{D.22}$$

$$\frac{\partial E}{\partial D_i} = -\int_{t_1}^{t_n} z_j\, y_j\, dt \tag{D.23}$$

To train the network, one integrates the system $\mathbf{y}$ forward in time from $t = t_1$ to $t = t_n$, sets the boundary condition $z_i(t_n) = 0$,[3] and then integrates the system $\mathbf{z}$ backwards in time using Equation D.17, at the same time integrating Equations D.21–D.23 to compute $\partial E/\partial w_k$. Using these last quantities, one can then perform the standard gradient descent on the weights of the network:

$$\Delta w_k = -\eta \frac{\partial E}{\partial w_k} \tag{D.24}$$

Note that this procedure treats the very first expression measurement of each time series data set different from the others: $\hat{\mathbf{y}}(t_1)$ is used to initialize the network at time $t = t_1$, and these initial values get fed back through the network and drive its behavior from $t = t_1$ to $t = t_n$. The other measurements in the time series, at the sampling times $t_2, t_3 \ldots t_n$, are used to determine the instantaneous errors $\mathbf{e}(t)$ in the backwards propagation of the error (Equation D.17). In other words, we assume the first measurement $\hat{\mathbf{y}}(t_1)$ is correct, and use it to initialize the network, but allow an imperfect match with the other measurements (since the total error E usually cannot be minimized to zero).

We can avoid this discrepancy by treating the initial values $\mathbf{y}(t_1)$ as additional parameters of the network (one set for each time series), and optimize these to reduce the total error E as well. We get:

$$\frac{\partial E}{\partial y_i(t_1)} = z_i(t_1) \tag{D.25}$$

(using the definition of $z_i(t)$). Note that $z_i(t_1)$ will include two terms: the backwards integration of Equation D.17, corresponding to the propagated errors at subsequent time points due to the initialization $\mathbf{y}(t_1)$; and $e_i(t_1)$, the instantaneous contribution

[3]Or rather, $z_i(t_n + \Delta t) = 0$, because the outputs of the network just *after* the last measurement do not contribute to the total error E.

to the error due to a difference between the initialization $y_i(t_1)$ and the measured value $\hat{y}_i(t_1)$. Just as for all other measurements $\hat{\mathbf{y}}(t_k)$, the network will have to trade off these two terms and find a set of parameters that, for each sampling time t_k, allows a good fit for the real data $\hat{\mathbf{y}}(t_k)$, without ruining the fit at subsequent time points $t_{k+1} \ldots t_n$.

D.3 Recurrent Backpropagation

Recurrent Backpropagation (RBP) can be seen as a special case of Backpropagation Through Time (BPTT), where the dynamic behavior to be learned is a fixed point. One can imagine letting Equation D.1 relax to a fixed point, and then having the error E consist of a single measurement of the distance between the achieved fixed point $\mathbf{y}^0$ and the desired fixed point $\hat{\mathbf{y}}^0$. For Equation D.14, the fixed points are solutions of:

$$y_i^0 = \frac{A_i}{D_i} S(x_i^0) \tag{D.26}$$

Pineda [181] and Almeida [12] independently showed that the error gradient with respect to the weights w_{ij} is given by:

$$\frac{\partial E}{\partial w_{ij}} = -A_j \, z_j \, S'(x_j) \, y_i \tag{D.27}$$

where z_i is a solution to:

$$D_i \, z_i = e_i + \sum_j A_j \, w_{ij} \, S'(x_j) \, z_j \tag{D.28}$$

which can be found by relaxing the following equation to a fixed point:

$$\frac{dz_i}{dt} = -D_i \, z_i + e_i + \sum_j A_j \, w_{ij} \, S'(x_j) \, z_j \tag{D.29}$$

Note that the z_i defined here is subtly different from the z_i in Section D.2, as is evidenced by the difference in sign between Equations D.29 and D.17, and Equations D.27 and D.21. In fact, the z_i defined here is related to $z_i(t_n - \Delta t)$ in Section D.2, if we measure the distance from the desired fixed point $\hat{\mathbf{y}}^0$ at time t_n, and integrate Equation D.17 *backwards* in time for a single time step Δt. Here, we integrate Equation D.27 *forwards* in time to find its fixed point.

References

[1] Aebersold, R., Hood, L. E., and Watts, J. D. Equipping scientists for the new biology. *Nature Biotechnology 18*, 4 (April 2000), 359.

[2] Akima, H. A new method of interpolation and smooth curve fitting based on local procedures. *J. ACM 17*, 4 (1970), 589–602.

[3] Akutsu, T., Kuhara, S., Maruyama, O., and Miyano, S. Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. In *Proceedings of the ninth annual ACM-SIAM symposium on discrete algorithms* (New York, NY, 1998), H. Karloff, Ed., ACM Press, pp. 695–702.

[4] Akutsu, T., Miyano, S., and Kuhara, S. Identification of genetic networks from a small number of gene expression pattern under the Boolean network model. In Altman et al. [16], pp. 17–28.

[5] Akutsu, T., Miyano, S., and Kuhara, S. Algorithms for identifying boolean networks and related biological networks based on matrix multiplication and fingerprint function. In Istrail et al. [121], pp. 8–14.

[6] Akutsu, T., Miyano, S., and Kuhara, S. Algorithms for inferring qualitative models of biological networks. In Altman et al. [17], pp. 290–301.

[7] Albertini, F., and Sontag, E. D. Uniqueness of weights for recurrent nets. In *Proceedings MTNS '93* (Regensburg, 1993), vol. 2 of *Systems and Networks: Mathematical Theory and Applications*, Akademie Verlag, pp. 599–602.

[8] Aldenderfer, M. S., and Blashfield, R. K. *Cluster analysis*. Quantitative applications in the social sciences. Sage Publications, Newbury Park, CA, 1984.

[9] Alizadeh, A. A., Eisen, M. B., Davis, R. E., Ma, C., Lossos, I. S., Rosenwald, A., Boldrick, J. C., Sabet, H., Tran, T., Yu, X., Powell, J. I., Yang, L., Marti, G. E., Moore, T., Lu, L., Lewis, D. B.,

TIBSHIRANI, R., SHERLOCK, G., CHAN, W. C., GREINER, T. C., WEISENBURGER, D. D., ARMITAGE, J. O., WARNKE, R., AND STAUDT, L. M. Distinct types of diffuse large b-cell lymphoma identified by gene expression profiling. *Nature 403*, 6769 (2000), 503–11.

[10] ALIZADEH, A. A., AND STAUDT, L. M. Genomic-scale gene expression profiling of normal and malignant immune cells. *Curr. Opin. Immunol. 12*, 2 (April 2000), 219–225.

[11] ALLA, H., AND DAVID, R. Continuous and hybrid Petri nets. *Journal of Circuits, Systems, and Computers 8*, 1 (1998), 159–188.

[12] ALMEIDA, L. B. A learning rule for asynchronous perceptrons with feedback in a combinatorial environment. In *Proceedings of the IEEE First International Conference on Neural Networks* (San Diego, CA, 1987), M. Caudill and C. Butler, Eds., MIT Press, pp. 609–618.

[13] ALON, U., BARKAI, N., NOTTERMAN, D. A., GISH, K., YBARRA, S., MACK, D., AND LEVINE, A. J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA 96*, 12 (1999), 6745–50.

[14] ALTER, O., BROWN, P. O., AND BOTSTEIN, D. Singular value decomposition for genome-wide expression data processing and modeling. *Proc. Natl. Acad. Sci. USA 97*, 18 (2000), 10101–10106.

[15] ALTMAN, R. B., DUNKER, A. K., HUNTER, L., AND KLEIN, T. E., Eds. *Pacific Symposium on Biocomputing '98* (Singapore, 1998), World Scientific Publishing Co.

[16] ALTMAN, R. B., DUNKER, A. K., HUNTER, L., KLEIN, T. E., AND LAUDERDALE, K., Eds. *Pacific Symposium on Biocomputing '99* (Singapore, 1999), World Scientific Publishing Co.

[17] ALTMAN, R. B., LAUDERDALE, K., DUNKER, A. K., HUNTER, L., AND KLEIN, T. E., Eds. *Pacific Symposium on Biocomputing '00* (Singapore, 2000), World Scientific Publishing Co.

[18] ANDERSON, L., AND SEILHAMER, J. A comparison of selected mRNA and protein abundances in human liver. *Electrophoresis 18*, 4 (April 1997), 533–537.

[19] APPEL, R. D., SANCHEZ, J. C., BAIROCH, A., GOLAZ, O., MIU, M., VARGAS, J. R., AND HOCHSTRASSER, D. F. SWISS-2DPAGE: Two-dimensional polyacrylamide gel electrophoresis database. http://www.expasy.ch/ch2d/.

[20] Appel, R. D., Sanchez, J. C., Bairoch, A., Golaz, O., Miu, M., Vargas, J. R., and Hochstrasser, D. F. SWISS-2DPAGE: a database of two-dimensional gel electrophoresis images. *Electrophoresis 14*, 11 (1993), 1232–1238.

[21] Arkin, A., and Ross, J. Statistical construction of chemical reaction mechanism from measured time-series. *J. Physical Chemistry 99* (1995), 970–979.

[22] Arkin, A., Ross, J., and McAdams, H. H. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells. *Genetics 149*, 4 (August 1998), 1633–1648.

[23] Arkin, A., Shen, P., and Ross, J. A test case of correlation metric construction of a reaction pathway from measurements. *Science 277* (1997), 1275–1279.

[24] Arsenijevic, Y., and Weiss, S. Insulin-like growth factor-I is a differentiation factor for postmitotic cns stem cell-derived neuronal precursors: Distinct actions from those of brain-derived neurotrophic factor. *J. Neurosci. 18*, 6 (1998), 2118–2128.

[25] Barger, S. W., Van Eldik, L. J., and Mattson, M. P. S100 beta protects hippocampal neurons from damage induced by glucose deprivation. *Brain Res. 677*, 1 (1995), 167–170.

[26] Barnard, E. A., Skolnick, P., Olsen, R. W., Mohler, H., Sieghart, W., Biggio, G., Braestrup, C., Bateson, A. N., and Langer, S. Z. International union of pharmacology. XV. subtypes of gamma-aminobutyric acid A receptors: Classification on the basis of subunit structure and receptor function. *Pharmacol. Rev. 50*, 2 (June 1998), 291–313.

[27] Baumgartner, B. J., Harvey, R. J., Darlison, M. G., and Barnes Jr, E. M. Developmental up-regulation and agonist-dependent down-regulation of GABAA receptor subunit mRNAs in chick cortical neurons. *Brain. Res. Mol. Brain. Res. 26*, 1–2 (October 1994), 9–17.

[28] Bellman, R. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[29] Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., and Yakhini, Z. Tissue classification with gene expression profiles. In Istrail et al. [121], pp. 54–64.

[30] Ben-Dor, A., Shamir, R., and Yakhini, Z. Clustering gene expression patterns. *J. Comput. Biol. 6*, 3–4 (1999), 281–97.

[31] Bezdek, J. C. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[32] Bishop, C. M. *Neural networks for pattern recognition.* Oxford University Press, Oxford, UK, 1995.

[33] Bjellqvist, B., Ek, K., Righetti, P. G., Gianazza, E., Görg, A., Westermeier, R., and Postel, W. Isoelectric focusing in immobilized pH gradients: principle, methodology and some applications. *J. Biochem. Biophys. Methods 6* (1982), 317–339.

[34] Blanchard, A. Synthetic DNA arrays. *Genetic Engineering 20* (1998), 111–123.

[35] Bond, R. W., Wyborski, R. J., and Gottlieb, D. I. Developmentally regulated expression of an exon containing a stop codon in the gene for glutamic acid decarboxylase. *Proc. Natl. Acad. Sci. USA 87*, 22 (1990), 8771–8775.

[36] Bourne, P., Gribskov, M., Altman, R., Jensen, N., Hope, D., Lengauer, T., Mitchell, J., Scheeff, E., Smith, C., Strande, S., and Weissig, H., Eds. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology* (Menlo Park, CA, 2000), AAAI Press.

[37] Bray, D. Intracellular signalling as a parallel distributed process. *J. Theor. Biol. 143* (1990), 215–231.

[38] Brayton, R. K., Hachtel, G. D., McMullen, C. T., and Sangiovanni-Vincentelli, A. L. ESPRESSO – Boolean Minimization. Available via ftp from ic.eecs.berkeley.edu in /pub/Espresso/.

[39] Brayton, R. K., Hachtel, G. D., McMullen, C. T., and Sangiovanni-Vincentelli, A. L. *Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[40] Brāzma, A., Jonassen, I., Vilo, J., and Ukkonen, E. Predicting gene regulatory elements in silico on a genomic scale. *Genome Research 8*, 11 (November 1998), 1202–1215.

[41] Brooks-Kayal, A. R., Shumate, M. D., Jin, H., Lin, D. D., Rikhter, T. Y., Holloway, K. L., and Coulter, D. A. Human neuronal gamma-aminobutyric acid-A receptors: Coordinated subunit mRNA expression and

functional correlates in individual dentate granule cells. *J. Neurosci. 19*, 19 (1999), 8312–8318.

[42] Brown, M. P. S., Grundy, W. N., Lin, D., Sugnet, C., Ares Jr, M., and Haussler, D. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA 97*, 1 (2000), 262–267.

[43] Buntine, W. Operations for learning with graphical models. *Journal of Artificial Intelligence Research 2* (1994), 159–225.

[44] Bussemaker, H. J., Li, H., and Siggia, E. D. Building a dictionary for genomes: identification of presumptive regulatory sites by statistical analysis. *Proc. Natl. Acad. Sci. USA 97*, 18 (2000), 10096–10100.

[45] Butte, A. J., and Kohane, I. S. Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. In Altman et al. [17], pp. 415–426.

[46] Califano, A., Stolovitzky, G., and Tu, Y. Analysis of gene expression microarrays for phenotype classification. In Bourne et al. [36], pp. 75–85.

[47] Castren, E., Berninger, B., Leingartner, A., and Lindholm, D. Regulation of brain-derived neurotrophic factor mRNA levels in hippocampus by neuronal activity. *Prog. Brain. Res. 117* (1998), 57–64.

[48] Cavallo, R. E., and Klir, G. J. Reconstructability analysis: overview and bibliography. *Int. J. Gen. Sys. 7* (1981), 1–6.

[49] Cheeseman, P., and Stutz, J. Bayesian classification (AutoClass): Theory and results. In *Advances in Knowledge Discovery and Data Mining*, U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. AAAI Press/MIT Press, 1996, ch. 6, pp. 153–180.

[50] Chen, T., Filkov, V., and Skiena, S. S. Identifying gene regulatory networks from experimental data. In Istrail et al. [120], pp. 94–103.

[51] Chen, T., He, H. L., and Church, G. M. Modeling gene expression with differential equations. In Altman et al. [16], pp. 29–40.

[52] Cheng, Y., and Church, G. M. Biclustering of expression data. In Bourne et al. [36], pp. 93–103.

[53] Cho, R. J., Campbell, M. J., Winzeler, E. A., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T. G., Gabrielian, A. E., Landsman, D., Lockhart, D. J., and Davis, R. W. A genome-wide transcriptional analysis of the mitotic cell cycle. *Mol. Cell 2*, 1 (July 1998), 65–73.

[54] Chu, S., DeRisi, J., Eisen, M., Mulholland, J., Botstein, D., Brown, P. O., and Herskowitz, I. The transcriptional program of sporulation in budding yeast. *Science 282*, 5389 (1998), 699–705.

[55] Claverie, J.-M. Computational methods for the identification of differential and coordinated gene expression. *Human Molecular Genetics 8*, 10 (1999), 1821–1832.

[56] Cohen, M. J., and Hall, G. F. Control of neuron shape during development and regeneration. *Neurochem. Pathol. 5*, 3 (December 1986), 331–343.

[57] Conant, R. C. Extended dependency analysis of large systems Part I: dynamic analysis. *Int. J. Gen. Sys. 14* (1988), 97–123.

[58] Conant, R. C. Extended dependency analysis of large systems Part II: static analysis. *Int. J. Gen. Sys. 14* (1988), 125–141.

[59] Cover, T. M., and Thomas, J. A. *Elements of information theory.* John Wiley & Sons, Inc., New York, 1991.

[60] Cun, Y. L., Denker, J. S., and Solla, S. A. Optimal brain damage. In *Advances in Neural Information Processing Systems 2* (San Mateo, CA, 1990), D. S. Touretzky, Ed., Morgan Kaufmann, pp. 598–605.

[61] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society 39(B)* (1977), 1–38.

[62] DeRisi, J., Iyer, V., and Brown, P. O. The MGuide: A complete guide to building your own microarrayer. Available online at http://cmgm.stanford.edu/pbrown/mguide/.

[63] DeRisi, J., Iyer, V., and Brown, P. O. Exploring the metabolic and genetic control of gene expression on a genomic scale. *Science 278*, 5338 (1997), 680–686.

[64] D'haeseleer, P., Liang, S., and Somogyi, R. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics 16*, 8 (2000), 707–726.

[65] D'HAESELEER, P., WEN, X., FUHRMAN, S., AND SOMOGYI, R. Mining the gene expression matrix: inferring gene relationships from large scale gene expression data. In *Information processing in cells and tissues* (1997), R. C. Paton and M. Holcombe, Eds., Plenum Press, pp. 203–212.

[66] D'HAESELEER, P., WEN, X., FUHRMAN, S., AND SOMOGYI, R. Linear modeling of mRNA expression levels during CNS development and injury. In Altman et al. [16], pp. 41–52.

[67] DING, R., ASADA, H., AND OBATA, K. Changes in extracellular glutamate and GABA levels in the hippocampal CA3 and CA1 areas and the induction of glutamic acid decarboxylase-67 in dentate granule cells of rats treated with kainic acid. *Brain Res. 800*, 1 (July 1998), 105–113.

[68] DURNER, M., GREENBERG, D. A., AND DELGADO-ESCUETA, A. V. Is there a genetic relationship between epilepsy and birth defects? *Neurology 42*, 4 Suppl. 2 (1992), 63–67.

[69] EISEN, M. B., SPELLMAN, P. T., BROWN, P. O., AND BOTSTEIN, D. CLUSTER. Available at http://rana.Stanford.EDU/software/.

[70] EISEN, M. B., SPELLMAN, P. T., BROWN, P. O., AND BOTSTEIN, D. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA 95*, 25 (1998), 14863–14868.

[71] EUROPEAN BIOINFORMATICS INSTITUTE. The ArrayExpress database. http://www.ebi.ac.uk/arrayexpress/index.html.

[72] EWING, R. M., KAHLA, A. B., POIROT, O., LOPEZ, F., AUDIC, S., AND CLAVERIE, J. M. Large-scale statistical analyses of rice ESTs reveal correlated patterns of gene expression. *Genome Research 9*, 10 (October 1999), 950–959.

[73] FELSENSTEIN, J. PHYLIP version 3.5c. Distributed by the author, Department of Genetics, University of Washington, Seattle., 1993.

[74] FIELDS, S., AND SONG, O. A novel genetic system to detect protein-protein interactions. *Nature 340* (1989), 245–246.

[75] FISHER, R. A. In *The advanced theory of statistics*, M. G. Kendall and A. Stuart, Eds., 3rd ed., vol. 1. Hafner Press, 1969, p. 391.

[76] FODOR, S. P. A., RAVA, R. P., HUANG, X. C., PEASE, A. C., HOLMES, C. P., AND ADAMS, C. L. Multiplexed biochemical assays with biological chips. *Nature 364* (1993), 555–556.

[77] FRANKLIN, G. F., POWELL, J. D., AND EMAMI-NAEINI, A. *Feedback control of dynamic systems*, 3rd ed. Addison-Wesley, Reading, MA, 1994.

[78] FRIEDMAN, N., LINIAL, M., NACHMAN, I., AND PE'ER, D. Using Bayesian networks to analyze expression data. In Istrail et al. [121], pp. 127–135.

[79] FRIEDMAN, N., LINIAL, M., NACHMAN, I., AND PE'ER, D. Using Bayesian networks to analyze expression data. *Submitted to Journal of Computational Biology* (2000).

[80] FROST & SULLIVAN. Opportunities for DNA microchip and array technologies. Available from http://www.frost.com/, December 1999.

[81] FUHRMAN, S., D'HAESELEER, P., LIANG, S., AND SOMOGYI, R. *Tracing genetic information flow from gene expression to pathways and regulatory networks*. MIT Press, Cambridge, MA, 2000. (in press).

[82] FUTCHER, B., LATTER, G. I., MONARDO, P., MCLAUGHLIN, C. S., AND GARRELS, J. I. A sampling of the yeast proteome. *Molecular and Cellular Biology 19*, 11 (November 1999), 7357–7368.

[83] GESCHWIND, D. Opening talk for Short Course on DNA Microarrays, at Soc. for Neuroscience Annual Meeting, October 1999.

[84] GLASS, L. Combinatorial and topological methods in nonlinear chemical kinetics. *J. Chem. Phys. 63*, 4 (1975), 1325–1335.

[85] GLASS, L., AND KAUFFMAN, S. A. Co-operative components, spatial localization and oscillatory cellular dynamics. *J. Theor. Biol. 34* (1972), 219–237.

[86] GLASS, L., AND PASTERNACK, J. S. Stable oscillations in mathematical models of biological control systems. *J. Math. Biol. 6* (1978), 207–223.

[87] GOLDSTEIN, K. M., TABOADA, E., CONWAY, A., HESS, K., GARDEN, M., NADON, R., JACKSON, R., LASKY, S. R., FAIRFIELD, E., MINNING, T., EYNON, B., AND OTHERS. Various comments on the GENE-ARRAYS mailing list, available at listserv@listserv.ucsf.edu. See also http://www.egroups.com/message/microarray/1974, September 2000.

[88] GOLUB, G. H., AND LOAN, C. F. V. *Matrix Computations*, 3rd ed. Johns Hopkins Series in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 1996.

[89] Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A., Bloomfield, C. D., and Lander, E. S. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science 286*, 5439 (1999), 531–537.

[90] Goss, P. J., and Peccoud, J. Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proc. Natl. Acad. Sci. USA 95*, 12 (1998), 6750–6755.

[91] Griffin, W. S., Yeralan, O., Sheng, J. G., Boop, F. A., Mrak, R. E., Rovnaghi, C. R., Burnett, B. A., Feoktistova, A., and Van Eldik, L. J. Overexpression of the neurotrophic cytokine S100 beta in human temporal lobe epilepsy. *J. Neurochem. 65*, 1 (1995), 228–233.

[92] Guilhem, D., Dreyfus, P. A., Makiura, Y., Suzuki, F., and Onteniente, B. Short increase of BDNF messenger RNA triggers kainic acid-induced neuronal hypertrophy in adult mice. *Neuroscience 72*, 4 (1996), 923–931.

[93] Gygi, S. P., Corthals, G. L., Zhang, Y., Rochon, Y., and Aebersold, R. Evaluation of two-dimensional gel electrophoresis-based proteome analysis technology. *Proc. Natl. Acad. Sci. USA 97*, 17 (2000), 9390–9395.

[94] Gygi, S. P., Rochon, Y., Franza, B. R., and Aebersold, R. Correlation between protein and mRNA abundance in yeast. *Molecular and Cellular Biology 19*, 3 (March 1999), 1720–1730.

[95] Hamilton, J. D. *Time series analysis*. Princeton U. Press, Princeton, NJ, 1994.

[96] Hampson, S., Baldi, P., Kibler, D., and Sandmeyer, S. B. Analysis of yeast's ORF upstream regions by parallel processing, microarrays, and computational methods. In Bourne et al. [36], pp. 190–201.

[97] Han, E.-H., Karypis, G., Kumar, V., and Mobasher, B. Clustering in a high-dimensional space using hypergraph models. Tech. Rep. 97-019, Department of Computer Science, University of Minnesota, 1997.

[98] Hargrove, J. L., and Schmidt, F. H. The role of mRNA and protein stability in gene expression. *FASEB J. 3*, 12 (October 1989), 2360–2370.

[99] Harris, S. Unpublished data, March 1997.

[100] Hartuv, E., Schmitt, A., Lange, J., Meier-Ewert, S., Lehrach, H., and Shamir, R. An algorithm for clustering cDNAs for gene expression analysis. In Istrail et al. [120], pp. 188–197.

[101] Hassibi, B., and Stork, D. G. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems 5* (San Mateo, CA, 1993), S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds., Morgan Kaufmann, pp. 164–171.

[102] Hastie, T., Tibshirani, R., Eisen, M. B., Alizadeh, A., Levy, R., Staudt, L., Chan, W. C., Botstein, D., and Brown, P. 'gene shaving' as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biology 1*, 2 (August 2000), research0003.1–0003.21.

[103] Heckerman, D. A tutorial on learning with bayesian networks. Tech. Rep. MSR-TR-95-06, Microsoft Research, Redmond, WA, 1995. Available via ftp from ftp.research.microsoft.com in /pub/Tech-Reports/Winter94-95/TR-95-06.PS.

[104] Hertz, J. Statistical issues in reverse engineering of genetic networks. http://www.nordita.dk/∼hertz/papers/dgshort.ps.gz, 1998. Poster for Pacific Symposium on Biocomputing.

[105] Herwig, R., Poustka, A. J., Muller, C., Bull, C., Lehrach, H., and O'Brien, J. Large-scale clustering of cDNA-fingerprinting data. *Genome Research 9*, 11 (November 1999), 1093–1105.

[106] Heyer, L. J., Kruglyak, S., and Yooseph, S. Exploring expression data: identification and analysis of coexpressed genes. *Genome Research 9*, 11 (November 1999), 1106–1115.

[107] Hieter, P., and Boguski, M. Functional genomics: it's all how you read it. *Science 278* (1997), 601–602.

[108] Hinton, G. E. Learning translation invariant recognition in massively parallel networks. In *PARLE: Parallel Architectures and Languages Europe* (Berlin, 1987), G. Goos and J. Hartmanis, Eds., Lecture Notes in Computer Science, Springer-Verlag, pp. 1–13.

[109] Hlavacek, W. S., and Savageau, M. A. Subunit structure of regulator proteins influences the design of gene circuitry: analysis of perfectly coupled and completely uncoupled circuits. *J. Mol. Biol. 248*, 4 (1995), 739–755.

[110] Hofmann, R., and Tresp, V. Discovering structure in continuous variables using bayesian networks. In *Advances in Neural Information Processing Systems 8* (Cambridge, MA, 1996), D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo, Eds., MIT Press, pp. 500–506.

[111] Holmes, I., and Bruno, W. J. Finding regulatory elements using joint likelihoods for sequence and expression profile data. In Bourne et al. [36], pp. 202–210.

[112] Holstege, F. C., Jennings, E. G., Wyrick, J. J., Lee, T. I., Hengartner, C. J., Green, M. R., Golub, T. R., Lander, E. S., and Young, R. A. Genome-wide expression page. Avaliable online at http://web.wi.mit.edu/young/expression/.

[113] Holstege, F. C., Jennings, E. G., Wyrick, J. J., Lee, T. I., Hengartner, C. J., Green, M. R., Golub, T. R., Lander, E. S., and Young, R. A. Dissecting the regulatory circuitry of a eukaryotic genome. *Cell 95*, 5 (November 1998), 717–728.

[114] Holter, N. S., Mitra, M., Maritan, A., Cieplak, M., Banavar, J. R., and Fedoroff, N. V. Fundamental patterns underlying gene expression profiles: simplicity from complexity. *Proc. Natl. Acad. Sci. USA 97*, 15 (2000), 8409–8414.

[115] Hsueh, Y. P., Wang, T. F., Yang, F. C., and Sheng, M. Nuclear translocation and transcription regulation by the membrane-associated guanylate kinase CASK/LIN-2. *Nature 404*, 6775 (2000), 298–302.

[116] Hu, J., and Van Eldik, L. J. S100 beta induces apoptotic cell death in cultured astrocytes via a nitric oxide-dependent pathway. *Biochim. Biophys. Acta 1313*, 3 (1996), 239–245.

[117] Huang, S. Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery. *J. Mol. Med. 77*, 6 (June 1999), 469–80.

[118] Hughes, T. R., Marton, M. J., Jones, A. R., Roberts, C. J., Stoughton, R., Armour, C. D., Bennett, H. A., Coffey, E., Dai, H., He, Y. D., Kidd, M. J., King, A. M., Meyer, M. R., Slade, D., Lum, P. Y., Stepaniants, S. B., Shoemaker, D. D., Gachotte, D., Chakraburtty, K., Simon, J., Bard, M., and Friend, S. H. Functional discovery via a compendium of expression profiles. *Cell 102*, 1 (2000), 109–126.

[119] IDEKER, T. E., THORSSON, V., AND KARP, R. M. Discovery of regulatory interactions through perturbation: inference and experimental design. In Altman et al. [17], pp. 302–313.

[120] ISTRAIL, S., PEVZNER, P., AND WATERMAN, M., Eds. *Proceedings of the third annual international conference on computational molecular biology* (New York, NY, 1999), ACM Press.

[121] ISTRAIL, S., PEVZNER, P., AND WATERMAN, M., Eds. *Proceedings of the fourth annual international conference on computational molecular biology* (New York, NY, 2000), ACM Press.

[122] IYER, V. R., EISEN, M. B., ROSS, D. T., SCHULER, G., MOORE, T., LEE, J. C. F., TRENT, J. M., STAUDT, L. M., HUDSON JR., J., BOGUSKI, M. S., LASHKARI, D., SHALON, D., BOTSTEIN, D., AND BROWN, P. O. The transcriptional program in the response of human fibroblasts to serum. *Science 283*, 5398 (1999), 83–97.

[123] JACOBS, R. A. Increased rates of convergence through learning rate adaptation. *Neural Networks 1*, 4 (1988), 295–307.

[124] JAIN, A. K., AND DUBES, R. C. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[125] JOHANNING, H., PLENGE, P., AND MELLERUP, E. Serotonin receptors in the brain of rats treated chronically with imipramine or RU24969: support for the 5-HT1B receptor being a 5-HT autoreceptor. *Pharmacol. Toxicol. 70*, 2 (February 1992), 131–134.

[126] KAR, S., SETO, D., DORE, S., CHABOT, J. G., AND QUIRION, R. Systemic administration of kainic acid induces selective time dependent decrease in [125I]insulin-like growth factor I, [125I]insulin-like growth factor II and [125I]insulin receptor binding sites in adult rat hippocampal formation. *Neuroscience 80*, 4 (1997), 1041–55.

[127] KARNAUGH, M. The map method for synthesis of combinational logic circuits. *AIEE Transactions, Part I Communication and Electronics 72* (November 1953), 593–598.

[128] KARP, R. M., STOUGHTON, R., AND YEUNG, K. Y. Algorithms for choosing differential gene expression experiments. In Istrail et al. [120], pp. 208–217.

[129] KAUFFMAN, S. A. Metabolic stability and epigenesis in randomly connected nets. *J. Theor. Biol. 22* (1969), 437–467.

[130] KAUFFMAN, S. A. *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, New York, 1993.

[131] KAUFMAN, L., AND ROUSSEEUW, P. J. *Finding groups in data: An introduction to cluster analysis*. John Wiley & Sons, New York, 1990.

[132] KENIGSBERG, R. L., HONG, Y., AND THÉORÊT, Y. Cholinergic cell expression in the developing rat medial septal nucleus in vitro is differentially controlled by GABAA and GABAB receptors. *Brain Res. 805*, 1–2 (1998), 123–130.

[133] KHAN, Z. U., GUTIERREZ, A., MEHTA, A. K., MIRALLES, C. P., AND DE BLAS, A. L. The alpha 4 subunit of the GABAA receptors from rat brain and retina. *Neuropharmacology 35*, 9–10 (1996), 1315–1322.

[134] KHAN, Z. U., GUTIERREZ, A., MIRALLES, C. P., AND DE BLAS, A. L. The gamma subunits of the native GABAA/benzodiazepine receptors. *Neurochem. Res. 21*, 2 (February 1996), 147–159.

[135] KOHONEN, T. *Self-Organizing Maps*, 2nd ed., vol. 30 of *Springer Series in Information Sciences*. Springer, New York, 1997.

[136] KOIRAN, P., AND SONTAG, E. D. Vapnik-Chervonenkis dimension of recurrent neural networks. *Discrete Applied Math 86* (1998), 63–79.

[137] KROGH, A., AND HERTZ, J. A. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems 4* (San Mateo, CA, 1992), J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., pp. 950–957.

[138] LANDER, E. S. The new genomics: global views of biology. *Science 274* (1996), 536–539.

[139] LASH, A. E., TOLSTOSHEV, C. M., WAGNER, L., SCHULER, G. D., STRAUSBERG, R. L., RIGGINS, G. J., AND ALTSCHUL, S. F. SAGEmap: A public gene expression resource. *Genome Research 10*, 7 (July 2000), 1051–1060.

[140] LAZZERONI, L., AND OWEN, A. Plaid models for gene expression data. Tech. rep., Stanford University, 2000.

[141] LEE, M.-L. T., KUO, F. C., WHITMORE, G. A., AND SKLAR, J. Importance of replication in microarray gene expression studies: Statistical methods and evidence from repetitive cDNA hybridizations. *Proc. Natl. Acad. Sci. USA 97*, 18 (2000), 9834–9839.

[142] LIANG, S., FUHRMAN, S., AND SOMOGYI, R. REVEAL, a general reverse engineering algorithm for inference of genetic network architectures. In Altman et al. [15], pp. 18–29.

[143] LIN, S. Critical assessment of techniques for microarray data analysis (CAMDA). http://bioinformatics.duke.edu/camda/, 2000.

[144] LITTLESTONE, N. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning 2* (1988), 285–318.

[145] LITTLESTONE, N. Redundant noisy attributes, attribute errors, and linear-threshold learning using Winnow. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory* (University of California, Santa Cruz, 1991), ACM Press, pp. 147–156.

[146] LITTLESTONE, N., AND WARMUTH, M. K. The weighted majority algorithm. *Information and Computation 108*, 2 (February 1994), 212–261.

[147] LIU, J. P., AND STERNBERG, P. S-100 beta and insulin-like growth factor-II differentially regulate growth of developing serotonin and dopamine neurons in vitro. *J. Neurosci. Res. 33*, 2 (October 1992), 248–256.

[148] MA, W., AND BARKER, J. L. GABA, GAD, and GABA(A) receptor alpha4, beta1, and gamma1 subunits are expressed in the late embryonic and early postnatal neocortical germinal matrix and coincide with gliogenesis. *Microsc. Res. Tech. 40*, 5 (1998), 398–407.

[149] MACKAY, D. J. C. Probable networks and plausible predictions - a review of practical bayesian methods for supervised neural networks. Available at ftp://wol.ra.phy.cam.ac.uk/pub/www/mackay/network.ps.gz, 1995.

[150] MACQUEEN, J. Some methods for classification and analysis of multivariate observation. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), L. M. L. Cam and J. Nyeman, Eds., vol. 1, University of California Press, pp. 281–297.

[151] MAHALANOBIS, P. C. On the generalized distance in statistics. In *Proceedings of the National Institute of Sciences of India* (1936), vol. 12, pp. 49–55.

[152] MANGER, I. D., AND RELMAN, D. A. How the host 'sees' pathogens: global gene expression responses to infection. *Curr. Opin. Immunol. 12*, 2 (April 2000), 215–218.

[153] MARNELLOS, G., DEBLANDERE, G. A., MJOLSNESS, E., AND KINTNER, C. Delta-Notch lateral inhibitory patterning in the emergence of ciliated cells in Xenopus: experimental observations and a gene network model. In Altman et al. [17], pp. 326–337.

[154] MARNELLOS, G., AND MJOLSNESS, E. A gene network approach to modeling early neurogenesis in Drosophila. In Altman et al. [15], pp. 30–41.

[155] MASSART, D. L., AND KAUFMANN, L. *The interpretation of analytical chemical data by the use of cluster analysis*. John Wiley & Sons, New York, 1983.

[156] MATSUNO, H., DOI, A., NAGASAKI, M., AND MIYANO, S. Hybrid Petri net representation of genetic regulatory network. In Altman et al. [17], pp. 338–349.

[157] MATTSON, M. P., AND RYCHLIK, B. Glia protect hippocampal neurons against excitatory amino acid-induced degeneration: involvement of fibroblast growth factor. *Int. J. Dev. Neurosci. 8*, 4 (1990), 399–415.

[158] MCCLUSKY, E. J. Minimization of boolean functions. *Bell Sys. Tech. J. 35* (1956), 1417–1444.

[159] MCCULLAGH, P., AND NELDER, J. A. *Generalized Linear Models*, 2nd ed. Chapman & Hall, London, UK, 1989.

[160] MENDES, P. Metabolic simulation as an aid in understanding gene expression data. In *Proceedings of Workshop on computation of Biochemical Pathways and Genetic Networks* (Berlin, 1999), E. Bornberg-Bauer, A. D. Beuckelaer, U. Kummer, and U. Rost, Eds., Logos Verlag, pp. 27–33.

[161] MICHAELS, G. S., CARR, D. B., ASKENAZI, M., FUHRMAN, S., WEN, X., AND SOMOGYI, R. Cluster analysis and data visualization of large-scale gene expression data. In Altman et al. [15], pp. 42–53.

[162] MJOLSNESS, E., MANN, T., NO, R. C., AND WOLD, B. From co-expression to co-regulation: An approach to inferring transcriptional regulation among gene classes from large-scale expression data. In *Advances in Neural Information Processing Systems 12* (Cambridge, MA, 2000), S. A. Solla, T. K. Leen, and K.-R. Müller, Eds., MIT Press.

[163] MJOLSNESS, E., NO, R. C., AND WOLD, B. Multi-parent clustering algorithms for large-scale gene expression analysis. Tech. Rep. JPL-ICTR-99-5, Jet Propulsion Laboratory Section 367, 1999.

[164] MJOLSNESS, E., SHARP, D. H., AND REINITZ, J. A connectionist model of development. *J. Theor. Biol. 152*, 4 (1991), 429–454.

[165] Murphy, K., and Mian, S. Modeling gene expression data using Dynamic Bayesian Networks. Tech. rep., University of California, Berkeley, 1999. http://www.cs.berkeley.edu/~murphyk/Papers/ismb99.ps.gz.

[166] Nakayama, M., Gahara, Y., Kitamura, T., and Ohara, O. Distinctive four promoters collectively direct expression of brain-derived neurotrophic factor gene. *Brain. Res. Mol. Brain. Res. 21*, 3–4 (1994), 206–218.

[167] National Center for Biotechnology Information. Gene Expression Omnibus (GEO). Available online at http://www.ncbi.nlm.nih.gov/geo/.

[168] National Center for Genome Resources. GeneX: a collaborative internet database and toolset for gene expression data. Available online at http://www.ncgr.org/research/genex/.

[169] Niehrs, C., and Pollet, N. Synexpression groups in eukaryotes. *Nature 402*, 6761 (1999), 483–487.

[170] Nordhoff, E., Krogsdam, A.-M., Jørgensen, H. F., Kallipolitis, B. H., Clark, B. F. C., Roepstorff, P., and Kristiansen, K. Rapid identification of DNA-binding proteins by mass spectrometry. *Nature Biotechnology 17*, 9 (September 1999), 884–888.

[171] Oefner, P. J. Surface-charge reversed capillary zone electrophoresis of inorganic and organic anions. *Electrophoresis 16*, 1 (1995), 46–56.

[172] O'Farrell, P. H. High resolution two-dimensional electrophoresis of proteins. *J. Biol. Chem. 250* (1975), 4007–4021.

[173] Okamoto, T., Suzuki, T., and Yamamoto, N. Microarray fabrication with covalent attachment of DNA using Bubble Jet technology. *Nature Biotechnology 18*, 4 (April 2000), 438–441.

[174] Olsen, R. W., and DeLorey, T. M. GABA and glycine. In *Basic neurochemistry: molecular, cellular and medical aspects*, G. J. Siegel, B. W. Agranoff, R. W. Albers, S. K. Fisher, and M. D. Uhler, Eds., 6th ed. Lippincott-Raven Publishers, Philadelphia, 1999, pp. 335–346.

[175] Pearlmutter, B. A. The CBP neural network simulator. Available via ftp from ftp.cs.cmu.edu in /usr/anon/user/bap/cbp.tar.gz.

[176] Pearlmutter, B. A. Learning state space trajectories in recurrent neural networks. *Neural Computation 1*, 2 (1989), 263–269.

[177] Pearlmutter, B. A. Gradient calculations for dynamic recurrent neural networks: a survey. *IEEE Transactions on Neural Networks 6*, 5 (1995), 1212–1228.

[178] Pedersen, M. W., and Hansen, L. K. Recurrent networks: Second order properties and pruning. In *Advances in Neural Information Processing Systems 7* (Cambridge, MA, 1995), G. Tesauro, D. Touretzky, and T. Leen, Eds., MIT Press.

[179] Perou, C. M., Jeffrey, S. S., van de Rijn, M., Rees, C. A., Eisen, M. B., Ross, D. T., Pergamenschikov, A., Williams, C. F., Zhu, S. X., Lee, J. C., Lashkari, D., Shalon, D., Brown, P. O., and Botstein, D. Distinctive gene expression patterns in human mammary epithelial cells and breast cancers. *Proc. Natl. Acad. Sci. USA 96*, 16 (1999), 9212–9217.

[180] Perou, C. M., Sørlie, T., Eisen, M. B., van de Rijn, M., Jeffrey, S. S., Rees, C. A., Pollack, J. R., Ross, D. T., Johnsen, H., Akslen, L. A., Fluge, Ø., Pergamenschikov, A., Williams, C., Zhu, S. X., Lønning, P. E., Børresen-Dale, A.-L., Brown, P. O., and Botstein, D. Molecular portraits of human breast tumours. *Nature 406*, 6797 (2000), 747–752.

[181] Pineda, F. J. Generalization of back-propagation to recurrent neural networks. *Physical Review Letters 59*, 19 (1987), 2229–2232.

[182] Pleus, R. C., and Bylund, D. B. Desensitization and down-regulation of the 5-hydroxytryptamine1B receptor in the opossum kidney cell line. *J. Pharmacol. Exp. Ther. 261*, 1 (April 1992), 271–277.

[183] Ptashne, M. *A genetic switch*, 2nd ed. Cell Press & Blackwell scientific publications, Cambridge, MA, 1992.

[184] Raychaudhuri, S., Stuart, J. M., and Altman, R. B. Principal components analysis to summarize microarray experiments: Application to sporulation time series. In Altman et al. [17], pp. 452–463.

[185] Reinitz, J., and Sharp, D. H. Mechanism of *eve* stripe formation. *Mechanisms of Development 49* (1995), 133–158.

[186] Richmond, C. S., Glasner, J. D., Mau, R., Jin, H., and Blattner, F. R. Genome-wide expression profiling in Escherichia coli K-12. *Nucleic Acids Res. 27*, 19 (1999), 3821–3835.

[187] Roberts, C. J., Nelson, B., Marton, M. J., Stoughton, R., Meyer, M. R., Bennett, H. A., He, Y., Dai, H., Walker, W. L., Hughes, T. R., Tyers, M., Boone, C., and Friend, S. H. Signaling and circuitry of multiple MAPK pathways revealed by a matrix of global gene expression profiles. *Science 287*, 5454 (2000), 873–880.

[188] Ross, D. T., Scherf, U., Eisen, M. B., Perou, C. M., Rees, C., Spellman, P., Iyer, V., Jeffrey, S. S., Van de Rijn, M., Waltham, M., Pergamenschikov, A., Lee, J. C. F., Lashkari, D., Shalon, D., Myers, T. G., Weinstein, J. N., Botstein, D., and Brown, P. O. Systematic variation in gene expression patterns in human cancer cell lines. *Nature Genetics 24*, 3 (March 2000), 227–235.

[189] Roth, F. P., Hughes, J. D., Estep, P. W., and Church, G. M. Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation. *Nature Biotechnology 16*, 10 (October 1998), 939–945.

[190] Rudge, J. S., Mather, P. E., Pasnikowski, E. M., Cai, N., Corcoran, T., Acheson, A., Anderson, K., Lindsay, R. M., and Wiegand, S. J. Endogenous BDNF protein is increased in adult rat hippocampus after a kainic acid induced excitotoxic insult but exogenous BDNF is not neuroprotective. *Exp. Neurol. 149*, 2 (February 1998), 398–410.

[191] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1. MIT Press, Cambridge, MA, 1986, pp. 318–362.

[192] Sander, T., Bockenkamp, B., Hildmann, T., Blasczyk, R., Kretz, R., Wienker, T. F., Schmitz, B., Beck-Mannagetta, G., Riess, O., Epplen, J. T., Janz, D., and Ziegler, A. Refined mapping of the epilepsy susceptibility locus EJM1 on chromosome 6. *Neurology 49*, 3 (1997), 842–847.

[193] Sarle, W. S. Neural network FAQ. Periodic posting to the Usenet newsgroup comp.ai.neural-nets, available via ftp from ftp.sas.com in /pub/neural/FAQ.html, 1997.

[194] Savageau, M. A. Design of molecular control mechanisms and the demand for gene expression. *Proc. Natl. Acad. Sci. USA 74* (1977), 5647–5651.

[195] Savageau, M. A. Power-law formalism: a canonical nonlinear approach to modeling and analysis. In *Proceedings of the World Congress of Nonlinear Analysts '92* (1995), V. Lakshmikantham, Ed., vol. 4, pp. 3323–3334.

[196] Savageau, M. A. Rules for the evolution of gene circuitry. In Altman et al. [15], pp. 54–65.

[197] Schena, M., Shalon, D., Davis, R. W., and Brown, P. O. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science 270*, 5235 (1995), 467–470.

[198] Scherf, U., Ross, D. T., Waltham, M., Smith, L. H., Lee, J. K., Tanabe, L., Kohn, K. W., Reinhold, W. C., Myers, T. G., Andrews, D. T., Scudiero, D. A., Eisen, M. B., Sausville, E. A., Pommier, Y., Botstein, D., Brown, P. O., and Weinstein, J. N. A gene expression database for the molecular pharmacology of cancer. *Nature Genetics 24*, 3 (March 2000), 236–244.

[199] Schultz, C., and Tautz, D. Autonomous concentration-dependent activation and repression of kruppel by hunchback in the drosophila embryo. *Development 120*, 10 (October 1994), 3043–3049.

[200] Schwartzer, C., and Sperk, G. Hippocampal granule cells express glutamic acid decarboxylase-67 after limbic seizures in the rat. *Neuroscience 69*, 3 (December 1995), 705–709.

[201] Shannon, C. E. A mathematical theory of communication. *Bell Sys. Tech. J. 27* (1948), 379–423,623–656.

[202] Shapire, R. E. The strength of weak learnability. *Machine Learning 5*, 2 (1990), 197–227.

[203] Sharan, R., and Shamir, R. CLICK: A clustering algorithm with applications to gene expression analysis. In Bourne et al. [36], pp. 307–316.

[204] Sharp, D. H., and Reinitz, J. Prediction of mutant expression patterns using gene circuits. *Biosystems 47*, 1–2 (1998), 79–90.

[205] Siegelman, H. T., and Sontag, E. D. Turing computability with a neural network. *Applied Mathematics Letters 4*, 6 (1991), 77–80.

[206] Siegelman, H. T., and Sontag, E. D. Analog computation, neural networks, and circuits. *Theor. Comp. Sci. 131* (1993), 331–360.

[207] Singh-Gasson, S., Green, R. D., Yue, Y., Nelson, C., Blattner, F., Sussman, M. R., and Cerrina, F. Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array. *Nature Biotechnology 17*, 10 (October 1999), 974–978.

[208] SINHA, S., AND TOMPA, M. A statistical method for finding transcription factor binding sites. In Bourne et al. [36], pp. 344–354.

[209] SLONIM, D. K., TAMAYO, P., MESIROV, J. P., GOLUB, T. R., AND LANDER, E. S. Class prediction and discovery using gene expression data. In Istrail et al. [121], pp. 263–272.

[210] SNEATH, P. H. A., AND SOKAL, R. R. *Numerical Taxonomy*. W.H. Freeman, San Francisco, 1973.

[211] SOKAL, R. R., AND MICHENER, C. D. A statistical method for evaluating systematic relationships. *Univ. Kansas Sci. Bull. 38* (1958), 1409–1438.

[212] SOMOGYI, R. Unpublished data, January 1998.

[213] SOMOGYI, R., FUHRMAN, S., ASKENAZI, M., AND WUENSCHE, A. The gene expression matrix: towards the extraction of genetic network architectures. In *Proceedings of the Second World Congress of Nonlinear Analysts (WCNA96)* (1996), vol. 30 of *Nonlinear Analysis*, Pergamon Press.

[214] SOMOGYI, R., AND SNIEGOSKI, C. Modeling the complexity of genetic networks: understanding multigenic and pleiotropic regulation. *Complexity 1*, 6 (1996), 45–63.

[215] SOMOGYI, R., WEN, X., MA, W., AND BARKER, J. L. Developmental kinetics of GAD family mRNAs parallel neurogenesis in the rat spinal cord. *J. Neurosci. 15*, 4 (April 1995), 2575–2591.

[216] SONTAG, E. D. Shattering all sets of k points in general position requires (k-1)/2 parameters. *Neural Computation 9* (1997), 337–348.

[217] SPELLMAN, P. T., SHERLOCK, G., ZHANG, M. Q., IYER, V. R., ANDERS, K., EISEN, M. B., BROWN, P. O., BOTSTEIN, D., AND FUTCHER, B. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Mol. Biol. Cell 9*, 12 (December 1998), 3273–3297.

[218] STEEMERS, F. J., FERGUSON, J. A., AND WALT, D. R. Screening unlabeled DNA targets with randomly ordered fiber-optic gene arrays. *Nature Biotechnology 18*, 1 (January 2000), 91–94.

[219] STOLLBERG, J., URSCHITZ, J., URBAN, Z., AND BOYD, C. D. A quantitative evaluation of SAGE. *Genome Research 10*, 8 (August 2000), 1241–1248.

[220] STRANG, G. *Linear Algebra and Its Applications*, 3rd ed. Harcourt College Publishers, San Diego, CA, 1988.

[221] SZABO, G., KATAROVA, Z., AND GREENSPAN, R. Distinct protein forms are produced from alternatively spliced bicistronic glutamic acid decarboxylase mRNAs during development. *Mol. Biol. Cell 14*, 11 (November 1994), 7535–7545.

[222] SZALLASI, Z. Genetic network analysis in light of massively parallel biological data acquisition. In Altman et al. [16], pp. 5–16.

[223] SZALLASI, Z., AND LIANG, S. Modeling the normal and neoplastic cell cycle with "realistic Boolean genetic networks": their application for understanding carcinogenesis and assessing therapeutic strategies. In Altman et al. [15], pp. 66–76.

[224] TAMAYO, P., SLONIM, D., MESIROV, J., ZHU, Q., KITAREEWAN, S., DMITROVSKY, E., LANDER, E. S., AND GOLUB, T. R. GENECLUSTER. Available at http://www.genome.wi.mit.edu/MPR/software.html.

[225] TAMAYO, P., SLONIM, D., MESIROV, J., ZHU, Q., KITAREEWAN, S., DMITROVSKY, E., LANDER, E. S., AND GOLUB, T. R. Interpreting patterns of gene expression with self-organizing maps: methods and application to hematopoietic differentiation. *Proc. Natl. Acad. Sci. USA 96*, 6 (1999), 2907–2912.

[226] TANDON, P., YANG, Y., DAS, K., HOLMES, G. L., AND STAFSTROM, C. E. Neuroprotective effects of brain-derived neurotrophic factor in seizures during development. *Neuroscience 91*, 1 (1999), 293–303.

[227] TAO, R., MA, Z., AND AUERBACH, S. B. Influence of AMPA/kainate receptors on extracellular 5-hydroxytryptamine in rat midbrain raphe and forebrain. *Br. J. Pharmacol. 121*, 8 (August 1997), 1707–1715.

[228] TAVAZOIE, S., HUGHES, J. D., CAMPBELL, M. J., CHO, R. J., AND CHURCH, G. M. Systematic determination of genetic network architecture. *Nature Genetics 22*, 3 (July 1999), 281–285.

[229] THIEFFRY, D., AND THOMAS, R. Qualitative analysis of gene networks. In Altman et al. [15], pp. 77–88.

[230] THOMAS, R. Regulatory networks seen as asynchronous automata: a logical description. *J. Theor. Biol. 153* (1991), 1–23.

[231] Tibshirani, T. J. H. R. J. *Generalized Additive Models*. Chapman $ Hall, London, UK, 1990.

[232] Uetz, P., Giot, L., Cagney, G., Mansfield, T. A., Judson, R. S., Knight, J. R., Lockshon, D., Narayan, V., Srinivasan, M., Pochart, P., Qureshi-Emili, A., Li, Y., Godwin, B., Conover, D., Kalbfleisch, T., Vijayadamodar, G., Yang, M., Johnston, M., Fields, S., and Rothberg, J. M. A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae. *Nature 403*, 6770 (2000), 623–627.

[233] van Helden, J., Andre, B., and Collado-Vides, J. Extracting regulatory sites from the upstream region of yeast genes by computational analysis of oligonucleotide frequencies. *J. Mol. Biol. 281*, 5 (1998), 827–842.

[234] van Someren, E. P., Wessels, L. F. A., and Reinders, M. J. T. Linear modeling of genetic networks from experimental data. In Bourne et al. [36], pp. 355–366.

[235] Velculescu, V. E., Madden, S. L., Zhang, L., Lash, A. E., Yu, J., Rago, C., Lal, A., Wang, C. J., Beaudry, G. A., Ciriello, K. M., Cook, B. P., Dufault, M. R., Ferguson, A. T., Gao, Y., He, T.-C., Hermeking, H., Hiraldo, S. K., Hwang, P. M., Lopez, M. A., Luderer, H. F., Mathews, B., Petroziello, J. M., Polyak, K., Zawel, L., Zhang, W., Zhang, X., Zhou, W., Haluska, F. G., Jen, J., Sukumar, S., Landes, G. M., Riggins, G. J., Vogelstein, B., and Kinzler, K. W. Analysis of human transcriptomes. *Nature Genetics 23*, 4 (April 1999), 387–388.

[236] Velculescu, V. E., Zhang, L., Vogelstein, B., and Kinzler, K. W. Serial analysis of gene expression. *Science 270*, 5235 (1995), 484–487.

[237] Velculescu, V. E., Zhang, L., Zhou, W., Vogelstein, J., Basrai, M. A., Bassett Jr, D. E., Hieter, P., Vogelstein, B., and Kinzler, K. W. Characterization of the yeast transcriptome. *Cell 88* (1997), 243–251.

[238] von Dassow, G., Meir, E., Munro, E. M., and Odell, G. M. The segment polarity network is a robust developmental module. *Nature 406*, 6792 (2000), 188–192.

[239] Wahde, M., and Hertz, J. Course-grained reverse engineering of genetic regulatory networks. *Biosystems 55*, 1–3 (2000), 129–136.

[240] Walker, M. G., Volkmuth, W., Sprinzak, E., Hodgson, D., and Klingler, T. Prediction of gene function by genome-scale expression analysis: prostate cancer-associated genes. *Genome Research 9*, 12 (December 1999), 1198–1203.

[241] Weaver, D. C., Workman, C. T., and Stormo, G. D. Modeling regulatory networks with weight matrices. In Altman et al. [16], pp. 112–123.

[242] Weigend, A. S., Rumelhart, D. E., and Huberman, B. A. Generalization by weight-elimination with application to forecasting. In *Advances in Neural Information Processing Systems 3* (Redwood City, CA, 1991), R. P. Lippmann, J. Moody, and D. S. Touretzky, Eds., Morgan Kaufmann, pp. 875–882.

[243] Weinstein, J. N., Myers, T. G., O'Connor, P. M., Friend, S. H., Fornace Jr, A. J., Kohn, K. W., Fojo, T., Bates, S. E., Rubinstein, L. V., Anderson, N. L., Buolamwini, J. K., van Osdol, W. W., Monks, A. P., Scudiero, D. A., Sausville, E. A., Zaharevitz, D. W., Bunow, B., Viswanadhan, V. N., Johnson, G. S., Wittes, R. E., and Paull, K. D. An information-intensive approach to the molecular pharmacology of cancer. *Science 275*, 5298 (1997), 343–349.

[244] Wen, X. personal communication, March 1998.

[245] Wen, X., Fuhrman, S., Michaels, G. S., Carr, D. B., Smith, S., Barker, J. L., and Somogyi, R. Large-scale temporal gene expression mapping of CNS development. *Proc. Natl. Acad. Sci. USA 95*, 1 (1998), 334–339.

[246] Werbos, P. J. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Ph.d. thesis, Harvard University, 1974.

[247] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (October 1990), 1550–1560.

[248] Werbos, P. J. *The Roots of Backpropagation : From Ordered Derivatives to Neural Networks and Political Forecasting*. John Wiley & Sons, Inc., New York, NY, 1994.

[249] Williams, R. J., and Zipser, D. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation 1* (1989), 270–280.

[250] Winzeler, E. A., Shoemaker, D. D., Astromoff, A., Liang, H., Anderson, K., Andre, B., Bangham, R., Benito, R., Boeke, J. D., Bussey, H., Chu, A. M., Connelly, C., Davis, K., Dietrich, F., Dow, S. W., Bakkoury, M. E., Foury, F., Friend, S. H., Gentalen, E., Giaever, G., Hegemann, J. H., Jones, T., Laub, M., Liao, H., Liebundguth, N., Lockhart, D. J., Lucau-Danila, A., Lussier, M., M'Rabet, N., Menard, P., Mittmann, M., Pai, C., Rebischung, C., Revuelta, J. L., Riles, L., Roberts, C. J., Ross-MacDonald, P., Scherens, B., Snyder, M., Sookhai-Mahadeo, S., Storms, R. K., Véronneau, S., Voet, M., Volckaert, G., Ward, T. R., Wysocki, R., Yen, G. S., Yu, K., Zimmermann, K., Philippsen, P., Johnston, M., and Davis, R. W. Functional characterization of the S. cerevisiae genome by gene deletion and parallel analysis. *Science 285*, 5429 (1999), 901–906.

[251] Wodicka, L., Dong, H., Mittmann, M., Ho, M.-H., and Lockhart, D. J. Genome-wide expression monitoring in Saccharomyces cerevisiae. *Nature Biotechnology 15*, 12 (December 1997), 1359–1367.

[252] Wolfsberg, T. G., Gabrielian, A. E., Campbell, M. J., Cho, R. J., Spouge, J. L., and Landsman, D. Candidate regulatory sequence elements for cell cycle-dependent transcription in Saccharomyces cerevisiae. *Genome Research 9*, 8 (August 1999), 775–792.

[253] Yang, H. Y., Lieska, N., Kriho, V., Wu, C. M., and Pappas, G. D. A subpopulation of reactive astrocytes at the immediate site of cerebral cortical injury. *Exp. Neurol. 146*, 1 (July 1997), 199–205.

[254] Yeung, K. Y., Haynor, D. R., and Ruzzo, W. L. Validating clustering for gene expression data. Tech. Rep. UW-CSE-00-01-01, Department of Computer Science & Engineering, University of Washington, 2000.

[255] Yin, H. S., Chou, H. C., and Chiu, M. M. Changes in the microtubule proteins in the developing and transected spinal cords of the bullfrog tadpole: induction of microtubule-associated protein 2c and enhanced levels of Tau and tubulin in regenerating central axons. *Neuroscience 67*, 3 (August 1995), 763–775.

[256] Zhang, L., Zhou, W., Velculescu, V. E., Kerm, S. E., Hruban, R. H., Hamilton, S. R., Vogelstein, B., and Kinzler, K. W. Gene expression profiles in normal and cancer cells. *Science 276* (1997), 1286–1272.

[257] Zhu, J., and Zhang, M. Q. Cluster, function and promoter: Analysis of yeast expression array. In Altman et al. [17], pp. 476–487.

[258] Zien, A., Kuffner, R., Zimmer, R., and Lengauer, T. Analysis of gene expression data with pathway scores. In Bourne et al. [36], pp. 407–417.